Der Schutz der Privatsphäre bei der Anfragebearbeitung in Datenbanksystemen

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Dipl.-Inf. Lukas Dölle

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät:
Prof. Dr. Elmar Kulke

Gutachter:
1. Prof. Johann-Christoph Freytag, Ph.D.
2. Prof. Dr. Nicole Schweikardt
3. Prof. Dr. Hannes Federrath

eingereicht am: 24. März 2015
Tag der mündlichen Prüfung: 27. Mai 2016

Abstract

Over the last ten years many techniques for privacy-preserving data publishing have been proposed. Most of them anonymize a complete data table such that sensitive values cannot clearly be assigned to individuals. Their privacy is considered to be adequately protected, if an adversary cannot discover the actual value from a given set of at least k values.

For this thesis we assume that users interact with a data base by issuing a sequence of queries against one table. The system returns a sequence of results that contains sensitive values. The goal of this thesis is to check if adversaries are able to link uniquely sensitive values to individuals despite anonymized result sets. So far, there exist algorithms to prevent deanonymization for aggregate queries. Our novel approach prevents deanonymization for arbitrary queries. We show that our approach can be transformed to matching problems in special graphs. However, finding maximum matchings in these graphs is NP-complete. Therefore, we develop several approximation algorithms, which compute specific matchings in polynomial time, that still maintaining privacy.

Zusammenfassung

In den letzten Jahren wurden viele Methoden entwickelt, um den Schutz der Privatsphäre bei der Veröffentlichung von Daten zu gewährleisten. Die meisten Verfahren anonymisieren eine gesamte Datentabelle, sodass sensible Werte einzelnen Individuen nicht mehr eindeutig zugeordnet werden können. Deren Privatsphäre gilt als ausreichend geschützt, wenn eine Menge von mindestens k sensiblen Werten existiert, aus der potentielle Angreifer den tatsächlichen Wert nicht herausfinden können.

Ausgangspunkt für die vorliegende Arbeit ist eine Sequenz von Anfragen auf personenbezogene Daten, die durch ein Datenbankmanagementsystem mit der Rückgabe einer Menge von Tupeln beantwortet werden. Das Ziel besteht darin herauszufinden, ob Angreifer durch die Kenntnis aller Ergebnisse in der Lage sind, Individuen eindeutig ihre sensiblen Werte zuzuordnen, selbst wenn alle Ergebnismengen anonymisiert sind. Bisher sind Verfahren nur für aggregierte Anfragen wie Summen- oder Durchschnittsbildung bekannt. Daher werden in dieser Arbeit Ansätze entwickelt, die den Schutz auch für beliebige Anfragen gewährleisten. Es wird gezeigt, dass die Lösungsansätze auf Matchingprobleme in speziellen Graphen zurückgeführt werden können. Allerdings ist das Bestimmen größter Matchings in diesen Graphen NP-vollständig. Aus diesem Grund werden Approximationsalgorithmen vorgestellt, die in Polynomialzeit eine Teilmenge aller Matchings konstruieren, ohne die Privatsphäre zu kompromittieren.

Inhaltsverzeichnis

Teil I

Einführung

1 Einleitung und Grundlagen

1.1 Motivation

Durch die rasante Entwicklung immer leistungsfähigerer Computer und kostengünstigerer Speichermedien ist es heutzutage möglich, Daten im großen Ausmaß effizient zu speichern. Beim Surfen im Internet, Bezahlen im Supermarkt, Buchen einer Reise oder Abschließen einer Versicherung werden immer öfter Daten erhoben, die potentiell Rückschlüsse auf das Verhalten oder die persönlichen Umstände der betreffenden Personen zulassen. Die Intentionen können dabei zum Beispiel in wirtschaftlichen Interessen oder staatlichen Kontrollen liegen. Daneben entstehen auch große Mengen an personenbezogenen Daten durch die zunehmende Bereitschaft vieler Menschen, einen Teil ihrer Privatsphäre freiwillig preiszugeben. Solche Angaben können den Gesundheitszustand, Bankdaten, die Höhe des Ersparten, die politische Einstellung oder das Familienleben betreffen. Problematisch wird dieser Umstand vor allem dann, wenn die Kontrolle über jene Daten vermehrt nicht mehr denen obliegt, über die die Daten erfasst wurden, sondern denen, die die Daten verwalten. Insbesondere das Herausgeben oder Veröffentlichen der Daten kann die Privatsphäre der Personen beeinträchtigen. Aus diesem Grund wird das Vorbeugen vor missbräuchlicher Datenverarbeitung, was mit dem Begriff Datenschutz bezeichnet wird, immer notwendiger.

In Deutschland gründet der Datenschutz auf dem Recht auf informationelle Selbstbestimmung, welches allen Bürgern[1] vom Bundesverfassungsgericht im sogenannten Volkszählungsurteil von 1983 zugesprochen wurde [140]. Darin wird jedem das Recht attestiert, „grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen“. Werden Daten von Dritten gesammelt und verarbeitet, muss der Datenschutz beachtet werden. Insbesondere beim Herausgeben von Daten muss gewährleistet werden, dass jeder Bürger sein Recht auf informationelle Selbstbestimmung ausüben kann. In Deutschland regeln das Bundesdatenschutzgesetz (BDSG) und Datenschutzbestimmungen der einzelnen Länder den Umgang mit personenbezogenen Daten. Beispielsweise wird darin festgesetzt, dass Daten für bestimmte Zwecke nur anonymisiert verwendet werden dürfen [12]. Dazu müssen die Daten so verändert werden, dass Einzelangaben nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand einer Person zugeordnet werden können (§ 3 Abs. 6 BDSG [12]). In diesem Fall wird die Privatsphäre von natürlichen Personen als geschützt erachtet, worauf im Kapitel 2 konkreter eingegangen wird.

In den letzten Jahren wurden mehrere Modelle entwickelt, Daten so zu anonymisieren, dass ihre Veröffentlichung die Privatsphäre der betroffenen Personen nicht verletzen sollte. Viele dieser Ansätze basieren auf dem von Samarati und Sweeney vorgestellten Prinzip der k-Anonymität [123, 135]. Es besagt, dass sich die Informationen für jedes Individuum in den veröffentlichten Daten von mindestens $k-1$ anderen Individuen nicht unterscheiden dürfen. Damit kann jeder veröffentlichte sensible Wert, wie zum Beispiel eine Krankheit,

[1] Im Sinne einer besseren Lesbarkeit wird in dieser Arbeit auf die Nennung der jeweils weiblichen und männlichen Form zugunsten der männlichen Schreibweise verzichtet, womit keine Diskriminierung beabsichtigt ist.

mindestens k verschiedenen Individuen zugeordnet werden. Eine Verschärfung dieses Modells findet sich in der von Machanavajjhala et al. [101] eingeführten ℓ-Diversität. Sie dreht die vorige Aussage um und fordert, dass durch die veröffentlichten Daten jedem Individuum mindestens ℓ unterschiedliche sensible Werte zugeordnet werden können. Ein Angreifer, der Individuen mit ihren sensiblen Werten verknüpfen möchte, hat damit für jedes mindestens ℓ verschiedene mögliche Werte.

Während sich Modelle wie ℓ-Diversität beim einmaligen Veröffentlichen gesamter Datenbestände bewährt haben, existieren nur wenige Ansätze, die den Schutz der Privatsphäre bei der Anfragebearbeitung gewährleisten. In diesem Fall können Nutzer mehrere Anfragen an ein Datenbankmanagementsystem (DBMS) stellen und erhalten als Antwort jeweils eine Menge von Tupeln inklusive sensibler Werte. Die Schwierigkeit liegt hierbei darin zu entscheiden, ob durch das Herausgeben der Ergebnisse die Privatsphäre von Personen verletzt wird. Dazu kann einerseits jedes dieser Ergebnisse separat anonymisiert werden. Andererseits ist es eventuell möglich, dass die Kombination aller Ergebnisse Rückschlüsse auf sensible Werte bestimmter Individuen zulässt. Seien beispielsweise die Ergebnisse zweier Anfragen gegeben, bei denen das erste Ergebnis die Krankheiten Asthma, Bronchitis und Erkältung sowie das zweite die Werte Asthma, Diabetes und Grippe enthalte. Jedes Ergebnis erfüllt unabhängig vom anderen das Schutzkriterium ℓ-Diversität mit $\ell = 3$, denn es kommen jeweils drei verschiedene sensible Werte vor. Findet allerdings ein Angreifer durch externes Wissen heraus, dass in beiden Ergebnissen Angaben zu einer Person namens Alina enthalten sind, kann er ihre Krankheit schlussfolgern. Da Asthma der einzige Wert ist, der in beiden Ergebnissen vorkommt, muss Alina Asthma haben.

1.2 Problemstellung und Zielsetzung

Die vorliegende Arbeit integriert die Methoden der Datenveröffentlichung in das sogenannte interaktive Anfragemodell. Dabei sind Daten gegeben, die Informationen über Personen inklusive sensibler Werte enthalten. Nutzer können Anfragen an diese Daten stellen und erhalten vom System anonymisierte Ergebnisse. Somit ist die Privatsphäre der Personen in jedem einzelnen Ergebnis geschützt. Zu untersuchen bleibt jedoch, ob durch die Kombination aller Ergebnisse die Privatsphäre der betroffenen Personen verletzt ist oder nicht.

Da sich Modelle wie k-Anonymität oder ℓ-Diversität für eine Sequenz von Anfragen als nicht ausreichend erweisen, wird zunächst ein geeignetes Modell für den Schutz der Privatsphäre bei der Anfragebearbeitung erstellt. Das grundlegende Konzept der genannten Modelle soll jedoch erhalten bleiben. Die Privatsphäre gilt dementsprechend als geschützt, wenn ein Angreifer aus einer Menge potentieller sensibler Werte den tatsächlichen Wert eines Individuums nicht bestimmen kann. Eine größere Menge gewährt dabei einen stärkeren Schutz als eine kleinere.

Mithilfe des definierten Modells soll entschieden werden, ob eine Sequenz von Ergebnissen zu Anfragen die Privatsphäre von Individuen verletzt oder nicht. Dazu werden Anfragen $Q_1, Q_2, \ldots, Q_n$ und dazugehörige Ergebnisse $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_n$ betrachtet. Zunächst muss aus jedem $\hat{R}_i$, $1 \leq i \leq n$, ein anonymisiertes R_i erstellt werden, damit das Schutzkriterium für jedes einzelne Ergebnis erfüllt ist. Dafür können bereits existierende Verfahren verwendet werden. Die entscheidende Frage lautet danach, ob das Wissen, das aus der Kombination aller anonymisierten Ergebnisse $R_1, R_2, \ldots, R_n$ gewonnen werden kann, zu einer Verletzung der Privatsphäre entsprechend dem zuvor definierten Modell führt (s.

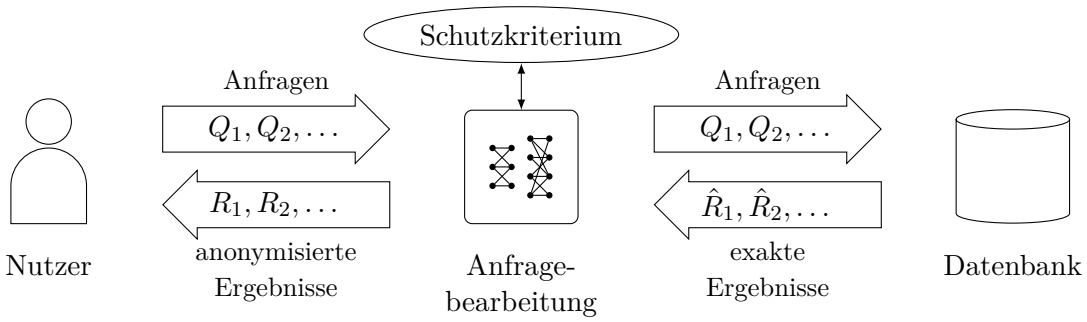


Abbildung 1.1: Architektur des vorgestellten Anfragesystems

Abbildung 1.1). Es wird gezeigt, dass dieses Problem NP-vollständig ist. Somit ergibt sich als weiteres Ziel die Angabe eines Approximationsalgorithmus, der in polynomieller Laufzeit zumindest sicherstellt, dass keine Verletzung der Privatsphäre vorliegt, und damit praktischen Anforderungen genügt.

1.3 Überblick über die Arbeit

In den weiteren Abschnitten des Kapitels 1 werden grundlegende Begriffe und Algorithmen eingeführt, die in der vorliegenden Arbeit verwendet werden. Kapitel 1.4 stellt kurz das relationale Datenmodell vor, das in der vorliegenden Arbeit Grundlage aller Datenrepräsentationen ist. In Kapitel 1.5 werden Optimierungsprobleme und Komplexitätsklassen definiert, ehe Kapitel 1.6 einen ausführlichen Einblick in die Graphentheorie gibt, die die Basis für den vorgestellten Lösungsansatz darstellt. Eine weitere Voraussetzung sind Kenntnisse im Bereich der linearen Programmierung, die in Kapitel 1.7 zusammengefasst werden.

Ein Überblick über den aktuellen Stand der Forschung im Bereich des Schutzes der Privatsphäre in Datenbanksystemen wird in Kapitel 2 gegeben. Dazu werden wichtige Begriffe eingeführt und Schutzkriterien vorgestellt. Details zu den einzelnen Unterkapiteln werden jeweils am Anfang des entsprechenden Kapitels ausgeführt.

In Kapitel 3 wird die Zielsetzung der vorliegenden Arbeit konkretisiert. Es wird ein Modell für den Schutz der Privatsphäre bei der Anfragebearbeitung entwickelt und ein Algorithmus vorgestellt, der die Einhaltung dieses Schutzkriteriums garantiert. Grundlage dafür bilden bipartite Graphen, in denen Matchings gesucht werden. Allerdings wird auch gezeigt, dass die Berechnung solcher Matchings und damit die Überprüfung des Schutzkriteriums NP-vollständige Probleme sind.

Das Ziel der darauffolgenden Kapitel liegt darin, einen Approximationsalgorithmus zu entwerfen, der den Schutz der Privatsphäre sicherstellt und nur polynomielle Laufzeit benötigt. In Kapitel 4 werden dafür Anforderungen definiert, die zu einer Verfeinerung des zuvor eingeführten Graphenmodells führen. Insbesondere werden die Matchings so eingeschränkt, dass deren effiziente Berechnung möglich ist. Die Approximation besteht darin, nur eine Teilmenge aller existierenden Matchings zu bestimmen, wodurch das Schutzkriterium nur näherungsweise angegeben werden kann.

Der Approximationsalgorithmus selbst wird in Kapitel 5 vorgestellt. Er basiert auf den zuvor festgelegten Vereinfachungen und besitzt trotz der Näherung die positive Eigen-

schaft, jede Verletzung der Privatsphäre zu erkennen. In diesem Fall kann entweder die verursachende Anfrage verweigert oder ein Ergebnis zurückgegeben werden, welches neben den angefragten sensiblen Werten noch zusätzliche künstliche Werte enthält.

Kapitel 6 widmet sich dem wichtigsten Schritt des vorgestellten Algorithmus, in dem neue Matchings dynamisch aus bereits zuvor ermittelten Matchings konstruiert werden. Es werden mehrere Ansätze vorgeschlagen, wie diese Berechnung durchgeführt werden kann.

Für einen Spezialfall von Ergebnissen zu Anfragen, in denen kein sensibler Wert mehrfach vorkommt, wird in Kapitel 7 ein vereinfachter Approximationsalgorithmus entworfen. Es wird gezeigt, wie dieser Ansatz auf beliebige Anfragen beziehungsweise Ergebnisse erweitert werden kann und damit eine Alternative zum Algorithmus aus Kapitel 5 darstellt.

In Kapitel 8 werden die entwickelten Algorithmen evaluiert. Es wird die Güte der Approximationen experimentell abgeschätzt und untersucht, wie sich die Laufzeit auch bei großen Datenmengen mit unterschiedlichen Verteilungen der sensiblen Attributwerte verhält. Tests mit realen Daten zeigen die Anwendbarkeit der Approximationsalgorithmen.

Im letzten Teil der vorliegenden Arbeit werden in Kapitel 9 die wichtigsten Ergebnisse zusammengefasst und diskutiert. Es werden Angriffsszenarien auf das eingeführte Modell für den Schutz der Privatsphäre erläutert und mögliche zukünftige Forschungsthemen angerissen.

1.4 Relationales Datenmodell

1.4.1 Relationen

Ein *Datenmodell* ist die theoretische Grundlage für eine Datenbank und bestimmt, auf welche Art und Weise Daten in einem Datenbanksystem gespeichert und verarbeitet werden können. Das bekannteste Beispiel ist das *relationale Datenmodell*, welches 1970 von Edgar F. Codd [25] erstmals vorgeschlagen wurde und bis heute trotz einiger Kritikpunkte einen etablierten Standard für Datenbanken bildet. Grundlage dieses Konzeptes ist die *Relation*, ein im mathematischen Sinn wohldefinierter Begriff. Sie stellt eine mathematische Beschreibung einer Tabelle dar.

Ein *Wertebereich* (oder eine *Domäne*) ist eine Menge unteilbarer Werte. Er kann logisch (z. B. Menge aller Postleitzahlen) oder durch einen Datentyp (z. B. Menge reeller Zahlen) spezifiziert werden. Ein *Attribut* ist der Name einer *Rolle*, die ein Wertebereich spielen kann. Es werden *kontinuierliche* oder *numerische* (z. B. Menge ganzer Zahlen) von *diskreten* Attributwerten (z. B. Menge von Krankheiten) unterschieden. Seien $A_1, A_2, \ldots, A_n$ Attribute, denen die Wertebereiche $\mathrm{dom}(A_1), \mathrm{dom}(A_2), \ldots, \mathrm{dom}(A_n)$ zugeordnet sind. Dann ist eine Relation R eine Teilmenge des kartesischen Produkts der Domänen:

$$R \subseteq \mathrm{dom}(A_1) \times \mathrm{dom}(A_2) \times \cdots \times \mathrm{dom}(A_n).$$

Ein Element $t \in R$ heißt *n-Tupel* oder kurz *Tupel*. Es besteht aus einer geordneten Liste von n Werten $t = [v_1, v_2, \ldots, v_n]$, wobei jeder Wert v_i, $1 \leq i \leq n$, ein Element von $\mathrm{dom}(A_i)$ oder ein spezieller *Nullwert*[2] ist. Der i-te Wert im Tupel t wird mit $t[A_i]$ angegeben.

Relationen können als Tabellen dargestellt werden, wobei die Spaltenüberschriften den Attributen und die übrigen Zeilen den Tupeln entsprechen. Im Rahmen der vorliegen-

[2] Nullwerte stellen Attribute dar, deren Werte unbekannt sind oder für bestimmte Tupel nicht existieren.

den Arbeit liegen Daten immer in relationaler Form vor, wobei die Begriffe Tabelle und Relation synonym verwendet werden.

1.4.2 Anfragen auf Relationen

Ein Datenmodell enthält in der Regel neben der Definition der Datenrepräsentation eine Menge von Operationen für die Manipulation der Daten. Die Operationen des relationalen Modells lassen sich in *Veränderungs-* und *Suchoperationen* unterteilen. Das Verändern umfasst das Einfügen und Löschen von Tupeln in Relationen sowie das Modifizieren von Attributwerten existierender Tupel. Eine Basismenge von Suchoperationen bildet die *Relationale Algebra*. Das Ergebnis einer Suche ist dabei eine neue Relation, die aus einer oder mehreren Relationen erzeugt wurde. Die Algebra-Operatoren produzieren also Relationen, die mithilfe von Operationen der gleichen Algebra weiter manipuliert werden können. Beispiele für Operatoren sind Vereinigung, Schnitt, Differenz, kartesisches Produkt, Selektion, Projektion und Join[3]. Auf eine detaillierte Einführung wird an dieser Stelle jedoch verzichtet und auf die einschlägige Datenbankliteratur verwiesen (z. B. auf [139, 44, 58, 130, 79]). Im Rahmen der vorliegenden Arbeit spielen nur die Selektion und Projektion eine wichtige Rolle.

Die Selektionsoperation wird benutzt, um eine Teilmenge der Tupel einer Relation auszuwählen, die eine gegebene Bedingung erfüllen. Diese Operation stellt demnach eine Art Filter dar, der nur bestimmte Tupel aussucht. Demgegenüber wählt die Projektionsoperation nur gewisse Spalten aus der Tabelle aus und verwirft andere. Sie wird benutzt, wenn nicht alle Attribute einer Relation von Interesse sind.

Außerhalb der Relationalen Algebra existieren weitere Operationen, die in kommerziellen Anfragesprachen benötigt werden. Ein Beispiel sind die *Aggregatfunktionen*, die einer Menge von Werten einen einzelnen Wert zuordnen. Dazu gehören die Summen- und Durchschnittsbildung, die Berechnung des Maximums und Minimums sowie die Zählfunktion, die die Mächtigkeit einer Menge bestimmt.

Als Standardanfragesprache relationaler Datenbanken hat sich in den letzten Jahrzehnten *SQL* (engl. *Structured Query Language*) durchgesetzt. Sie beinhaltet einige Merkmale der Relationalen Algebra, basiert aber größtenteils auf dem *relationalen Tupelkalkül*, einer weiteren formalen Anfragesprache. Beispiele von Anfragen an Relationen werden in dieser Arbeit jeweils in SQL notiert, wobei auf eine Einführung der Syntax erneut verzichtet und auf die angegebene Literatur verwiesen wird.

1.5 Komplexitätstheorie

1.5.1 Probleme und Laufzeit

Die Komplexitätstheorie beschäftigt sich mit dem Aufwand, der nötig ist, um gegebene algorithmische Probleme zu lösen. Dabei wird ein *Problem* formal durch eine Menge I von Instanzen, eine Menge S von zulässigen Lösungen und eine bestimmte Fragestellung charakterisiert. Die wichtigsten Fragen sind dabei, ob bestimmte Lösungen bestmöglich

[3] In der deutschen Literatur ist anstelle von „Join“ auch der Begriff „Verbund“ geläufig. Da aber auch im Deutschen häufiger von einem „Join“ gesprochen wird, wird in dieser Arbeit der englische Begriff verwendet.

sind oder eine gegebene Eigenschaft erfüllen. Dadurch entstehen Optimierungs- oder Entscheidungsprobleme.[4]

Optimierungsprobleme sind Probleme, die im Allgemeinen viele zulässige Lösungen besitzen. Jeder Lösung wird ein bestimmter Wert (Zielfunktionswert oder Kosten) zugeordnet. Es ist in der Menge aller zulässigen Lösungen nach derjenigen gefragt, die den besten, das heißt optimalen, Wert besitzt.

Definition 1.1 (Optimierungsproblem) *Ein* Optimierungsproblem P *wird definiert durch*

- *eine Menge I von Instanzen,*
- *eine Menge $S(x)$ von zulässigen Lösungen für jedes $x \in I$ und*
- *eine Funktion f, die jedem Paar $(x, s) \in I \times S$ einen reellen Wert zuordnet.*

*Das Ziel ist, bei gegebener Instanz eine zulässige Lösung zu finden, sodass f maximiert (*Maximierungsproblem*) beziehungsweise minimiert (*Minimierungsproblem*) wird.*

Für eine Instanz $x \in I$ heißt eine Lösung $s \in S(x)$ *optimal*, falls

$$\begin{aligned} f(x,s) &\geq f(x,s'), \quad \forall s' \in S \qquad \text{(Maximierungsproblem) beziehungsweise} \\ f(x,s) &\leq f(x,s'), \quad \forall s' \in S \qquad \text{(Minimierungproblem).} \end{aligned}$$

Mit $f^*(x)$ wird der Wert einer optimalen Lösung angegeben.

Ein *Entscheidungsproblem* hat als mögliche Antworten nur „Ja“ oder „Nein“. Hier ist zusätzlich ein Wert k für die Funktion f gegeben und es ist gefragt, ob eine Lösung x mit $f(x, s) \geq k$ (Maximierungsproblem) beziehungsweise $f(x, s) \leq k$ (Minimierungsproblem) existiert. Man spricht hierbei auch von der Entscheidungsvariante des dazugehörigen Optimierungsproblems.

Beispiel 1.1 Beim Erfüllbarkeitsproblem aussagenlogischer Formeln Sat (engl. *satisfiability*) ist eine Boolesche Formel in konjunktiver Normalform[5] (CNF für engl. *conjunctive normal form*) gegeben. Bei diesem Entscheidungsproblem ist nach der Existenz einer erfüllenden Belegung gefragt. Ein dazugehöriges Optimierungsproblem könnte zum Beispiel nach der maximalen Anzahl erfüllbarer Klauseln fragen. Dieses Problem heißt Max-Sat. Auch hier existiert eine Entscheidungsvariante, bei der gefragt ist, ob mindestens k Klauseln erfüllbar sind.

Ein *Algorithmus* A für ein Problem P berechnet zu jeder Eingabe x einen Wert $A(x)$. Für Optimierungsprobleme gilt dabei $A(x) \in \mathbb{R}$ und für Entscheidungsprobleme $A(x) \in \{\text{ja}, \text{nein}\}$. Als Rechnermodell werden für Algorithmen (meist deterministische) Turingmaschinen verwendet, auf deren Definition aber an dieser Stelle verzichtet wird. Ist eine Funktion $f\colon X \to Y$ gegeben, so berechnet A die Funktion f, falls $A(x) = f(x)$ für alle $x \in X$ gilt. Die *Laufzeit* eines Algorithmus bei Eingabe x ist die Anzahl der elementaren Operationen, die er (bzw. die Turingmaschine) ausführt. Besonders interessant ist die

[4] Es gibt noch weitere Fragestellungen und Probleme, die in dieser Arbeit jedoch eine untergeordnete Rolle spielen.

[5] Eine Boolesche Formel ist in konjunktiver Normalform, wenn sie eine Konjunktion von Klauseln darstellt. Eine Klausel wiederum ist eine Disjunktion von Literalen und ein Literal eine nichtnegierte oder negierte Variable.

Zeitkomplexität $t_A(n)$ oder auch *Worst-case-Laufzeit* von A. Das ist die maximale Laufzeit des Algorithmus über alle Eingaben der Länge n. A hat *polynomielle Laufzeit*, wenn es ein Polynom $p\colon \mathbb{N} \to \mathbb{R}$ gibt mit $t_A(n) \leq p(n)$ für alle $n \in \mathbb{N}$. In diesem Fall spricht man von einem *polynomiellen* oder auch *effizienten* Algorithmus.

1.5.2 Komplexitätsklassen

Ein Problem heißt *polynomiell* oder *in Polynomialzeit lösbar*, falls es einen polynomiellen Algorithmus zur Lösung des Problems gibt. Mit P wird die Menge aller polynomiell lösbaren Entscheidungsprobleme bezeichnet. Ein Entscheidungsproblem ist nichtdeterministisch polynomiell lösbar, wenn für jede „Ja"-Instanz[6] x ein Lösungsstring (das *Zertifikat*) existiert, mit dem die Korrektheit der „Ja"-Antwort in polynomieller Laufzeit nachgewiesen werden kann.[7] Die Klasse NP umfasst alle Entscheidungsprobleme, die nichtdeterministisch polynomiell lösbar sind. Es gilt offensichtlich $\mathsf{P} \subseteq \mathsf{NP}$. Bis heute ist die Frage, ob $\mathsf{P} = \mathsf{NP}$ gilt, ungelöst. Sie ist eine der bedeutendsten offenen Fragen der theoretischen Informatik.

Ein Entscheidungsproblem P_1 ist *polynomiell reduzierbar* auf ein Entscheidungsproblem P_2 ($P_1 \leq_p P_2$), wenn es einen Algorithmus gibt, der jede Instanz x_1 von P_1 in polynomieller Laufzeit in eine Instanz x_2 von P_2 transformiert, sodass x_2 genau dann „Ja"-Instanz von P_2 ist, wenn x_1 „Ja"-Instanz von P_1 ist. In diesem Fall löst ein effizienter Algorithmus für P_2 auch P_1 und P_2 ist mindestens so schwer wie P_1. Ein Entscheidungsproblem Q heißt *NP-hart*, falls sich jedes andere Problem aus NP polynomiell auf Q reduzieren lässt. Liegt Q zudem auch in NP, heißt Q *NP-vollständig*. Kann ein NP-vollständiges Problem P_2 polynomial auf ein Problem $P_1 \in \mathsf{NP}$ transformiert werden, ist auch P_1 NP-vollständig. Gibt es einen effizienten Algorithmus für eines der NP-vollständigen Probleme, sind alle effizient lösbar. Daher wird seit Jahren $\mathsf{P} \neq \mathsf{NP}$ vermutet.

Beispiel 1.2 SAT ist das erste Problem, von dem bewiesen wurde, dass es NP-vollständig ist [26]. Darauf aufbauend wurde die NP-Vollständigkeit vieler weiterer Probleme gezeigt [77]. Beispielsweise ist 3-SAT, das heißt SAT eingeschränkt auf Klauseln mit höchstens drei Literalen, ebenfalls NP-vollständig. Auch das oben erwähnte MAX-SAT fällt in diese Komplexitätsklasse.

Polynomielle Algorithmen werden für gewöhnlich als für die Praxis geeignete Algorithmen betrachtet. Für Probleme, die nicht in P liegen, werden daher sogenannte *Approximationsalgorithmen* angegeben, die die Probleme in polynomieller Laufzeit näherungsweise lösen. Sei A ein Algorithmus für ein Optimierungsproblem Q. Für eine Instanz $x \in I$ sei $OPT(x)$ der Wert einer optimalen Lösung von x. Weiter sei $A(x)$ der Wert der vom Algorithmus A gefundenen zulässigen Lösung. Dann ist die *(relative) Approximationsgüte* oder der *Approximationsfaktor* von A bei Eingabe x entweder $\frac{A(x)}{OPT(x)}$, wenn Q ein Minimierungsproblem, oder $\frac{OPT(x)}{A(x)}$, wenn Q ein Maximierungsproblem ist.

[6] Eine „Ja"-Instanz eines Problems hat als Antwort „ja".

[7] Alternativ kann auch definiert werden, dass ein Problem genau dann nichtdeterministisch polynomiell lösbar ist, wenn es von einer nichtdeterministischen Turingmaschine in polynomialer Zeit gelöst werden kann. Das ist äquivalent zu der Aussage, dass die Lösung eines Problems in polynomieller Zeit von einer deterministischen Turingmaschine verifiziert werden kann.

1.5.3 Asymptotische Laufzeit und Landausche Symbole

Landau-Symbole werden in der Informatik und Mathematik benutzt, um das asymptotische Wachstumsverhalten von Funktionen und Folgen zu beschreiben. In der Informatik werden sie bei der Analyse von Algorithmen verwendet und geben ein Maß für die Anzahl der Elementarschritte in Abhängigkeit von der Größe der Eingabevariablen an.

Der Großbuchstabe $\mathcal{O}$[8] (damals eigentlich ein großes Omikron) als Symbol für Ordnung wurde erstmals 1894 vom deutschen Zahlentheoretiker Paul Bachmann (1837–1920) [5] vorgeschlagen. Bekannt gemacht wurde diese Notation jedoch erst 1909 durch den ebenfalls deutschen Zahlentheoretiker Edmund Landau (1877–1938) [88], mit dessen Namen sie insbesondere im deutschen Sprachraum heute in Verbindung gebracht wird.

Definition 1.2 *Seien $f, g\colon \mathbb{N} \to \mathbb{R}_0^+$ zwei reellwertige Funktionen.[9] Man schreibt $f(n) = \mathcal{O}(g(n))$, falls es Zahlen $c > 0$ und $n_0 \in \mathbb{N}$ gibt mit*

$$\forall n \geq n_0\colon f(n) \leq c \cdot g(n).$$

Die Bedeutung der Aussage $f(n) = \mathcal{O}(g(n))$ ist, dass „f nicht wesentlich schneller“ wächst als g. Formal bezeichnet der Term $\mathcal{O}(g(n))$ die Klasse aller Funktionen f, die die obige Bedingung erfüllen, das heißt

$$\mathcal{O}(g(n)) = \{f\colon \mathbb{N} \to \mathbb{R}^+ \cup \{0\} \mid \exists c \in \mathbb{R}^+\, \exists n_0 \in \mathbb{N}\, \forall n \geq n_0\colon f(n) \leq c \cdot g(n)\}.$$

Die Gleichung $f(n) = \mathcal{O}(g(n))$ drückt also in Wahrheit die Element-Beziehung $f \in \mathcal{O}(g(n))$ aus. $\mathcal{O}$-Terme können auch auf der linken Seite einer Gleichung vorkommen. In diesem Fall wird eine Inklusionsbeziehung ausgedrückt. So steht beispielsweise $n^2 + \mathcal{O}(n) = \mathcal{O}(n^2)$ für die Aussage $\{n^2 + f \mid f \in \mathcal{O}(n)\} \subseteq \mathcal{O}(n^2)$.

Es gibt noch weitere nützliche Größenvergleiche von Funktionen. Seien erneut $f, g\colon \mathbb{N} \to \mathbb{R}_0^+$ zwei reellwertige Funktionen. Dann heißt

$$\begin{aligned}
f(n) = o(g(n)) &\Leftrightarrow \forall c \in \mathbb{R}^+\, \exists n_0 \in \mathbb{N}\colon && f(n) \leq c \cdot g(n), \quad \forall n \geq n_0,\\
f(n) = \Omega(g(n)) &\Leftrightarrow \exists c \in \mathbb{R}^+\, \exists n_0 \in \mathbb{N}\colon && f(n) \geq c \cdot g(n), \quad \forall n \geq n_0,\\
f(n) = \omega(g(n)) &\Leftrightarrow \forall c \in \mathbb{R}^+\, \exists n_0 \in \mathbb{N}\colon && f(n) \geq c \cdot g(n), \quad \forall n \geq n_0,\\
f(n) = \Theta(g(n)) &\Leftrightarrow \exists c_1, c_2 \in \mathbb{R}^+\, \exists n_0 \in \mathbb{N}\colon && c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \quad \forall n \geq n_0.
\end{aligned}$$

Diese Aussagen bedeuten in der angegebenen Reihenfolge, dass f „wesentlich langsamer als“, „mindestens so schnell wie“, „wesentlich schneller als“ beziehungsweise „ungefähr genauso schnell wie“ g wächst.

1.6 Graphentheorie

Eine Vielzahl von Problemen lassen sich mithilfe der Graphentheorie modellieren und lösen. In diesem Abschnitt werden grundlegende Begriffe und Zusammenhänge, die in der vorliegenden Arbeit verwendet werden, in kompakter Form eingeführt.

[8] gesprochen „groß O“

[9] Wenn der Wertebereich von f und g die gesamten reellen Zahlen $\mathbb{R}$ sein sollen, müssen in den folgenden Ungleichungen die Beträge $|f(n)|$ und $|g(n)|$ anstelle der Funktionswerte eingesetzt werden.

1.6.1 Einführung in die Graphentheorie

Grundlegende Begriffe

Definition 1.3 (Graph) *Ein* (ungerichteter) Graph *ist ein Tupel* $G = (V, E)$*, wobei* V *eine Menge und* $E \subseteq \binom{V}{2}$ *eine Teilmenge der 2-elementigen Teilmengen von* V *sind. Die Elemente* $v \in V$ *heißen* Knoten *(auch* Ecken*, engl. nodes oder vertices), die Elemente* $e \in E$ Kanten *(engl. edges) von* G*.*

Graphen nach Definition 1.3 heißen auch *einfach*, da sie keine Mehrfachkanten[10] oder Schlingen[11] besitzen. Um die Knoten- und Kantenmenge eines Graphen G von anderen zu unterscheiden, schreibt man auch $V(G)$ und $E(G)$. Je nach Zusammenhang wird gelegentlich G mit V oder mit E identifiziert. So schreibt man statt $v \in V(G)$ oder $e \in E(G)$ auch kurz $v \in G$ oder $e \in G$. Die Anzahl der Elemente in V beziehungsweise E wird mit $n(G)$ beziehungsweise $m(G)$ bezeichnet. Bei gegebenem Graphen wird meist nur n beziehungsweise m geschrieben. Ein Graph G heißt *endlich* beziehungsweise *unendlich* je nachdem, ob V endlich oder unendlich ist. Alle in dieser Arbeit betrachteten Graphen sind, soweit nicht anders angegeben, endlich.

Beispiel 1.3 Graphen werden häufig durch Diagramme veranschaulicht, in denen Knoten durch Punkte und Kanten durch Linien repräsentiert werden. So ist zum Beispiel der Graph $G = (\{1,2,3,4,5,6,7,8\}, \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{2,5\},\{3,4\},\{3,6\},\{4,5\},\{4,7\},\{5,8\},\{7,8\}\})$ in Abbildung 1.2a dargestellt.

Sei $G = (V, E)$ ein Graph. Zwei Knoten v und w aus V heißen *adjazent* (oder *benachbart*), falls $\{v, w\} \in E$. Man sagt auch, v und w sind durch eine Kante *verbunden*. Ein Knoten $v \in V$ und eine Kante $e \in E$ heißen *inzident* (oder *überdeckt*), falls $v \in e$ gilt. Die Knoten v und w werden *Endknoten* der Kante $\{v, w\}$ genannt. Zwei Kanten e und f heißen *inzident*, falls $e \cap f \neq \emptyset$ gilt. Die beiden Kanten haben also einen gemeinsamen Knoten. Im Fall $e \cap f = \emptyset$ nennt man e und f auch *unabhängig*.

Die *Nachbarschaft* $N_G(v)$ oder kurz $N(v)$[12] eines Knotens $v \in V$ ist die Menge $N(v) \coloneqq \{w \in V \mid \{v, w\} \in E\}$. Die Elemente in $N(v)$ heißen *Nachbarn* von v. Der *Grad* (oder die *Valenz*) $d(v)$ eines Knotens v ist die Anzahl der mit v inzidenten Kanten $|E(v)|$. Für einfache Graphen ist dies gerade die Anzahl der Nachbarn von v, das heißt, es gilt $d(v) = |N(v)|$.[13] Weiterhin ist hier für eine Teilmenge $S \subseteq V$ die Nachbarschaft von S als $N(S) \coloneqq \bigcup_{v \in S} N(v)$ definiert. Ein Knoten vom Grad 0 heißt *isoliert*. Die Zahl $\delta(G) \coloneqq \min\{d(v) \mid v \in V\}$ heißt *Minimalgrad* und $\Delta(G) \coloneqq \max\{d(v) \mid v \in V\}$ *Maximalgrad* von G. Hat jeder Knoten von G den gleichen Grad k, so heißt G *regulär* beziehungsweise *k-regulär*. Einen 3-regulären Graphen nennt man auch *kubisch*.

Ein Graph $G = (V, E)$ heißt *vollständig* oder *Clique*, falls $E = \binom{V}{2}$ gilt. Er enthält also jede mögliche Kante. Ein vollständiger Graph auf n Knoten wird mit K_n bezeichnet. So ist der K_3 beispielsweise ein *Dreieck*. G heißt *leerer* Graph oder *stabile Menge*, falls $E = \emptyset$ gilt.

[10] mehrfache (parallele) Kanten zwischen zwei Knoten
[11] Kanten mit zwei identischen Eckpunkten
[12] Auch bei anderen Bezeichnungen, die den Bezugsgraphen als Index angeben, wird der Index häufig weggelassen, wenn der Bezug klar ist.
[13] Das gilt zum Beispiel nicht für Multigraphen, die Mehrfachkanten enthalten.

Teilgraphen

Ein besonderes Augenmerk liegt in dieser Arbeit auf Graphen, die in anderen enthalten sind. Sie heißen *Teilgraphen*.

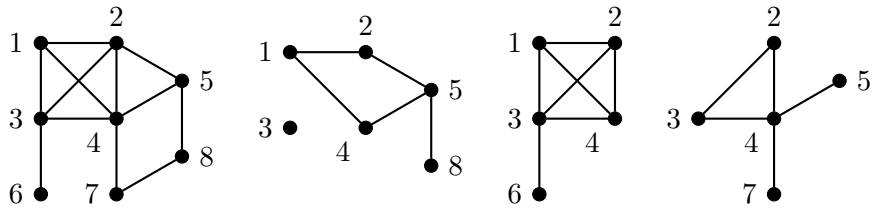
Definition 1.4 (Teilgraph) *Sei $G = (V, E)$ ein Graph und $V' \subseteq V$ sowie $E' \subseteq \binom{V'}{2} \cap E$ Teilmengen seiner Knoten und Kanten. Dann heißt $H = (V', E')$* Teilgraph *(oder* Subgraph *bzw.* Untergraph*) von G. Man sagt, G enthält H und notiert dies mit $H \subseteq G$. Falls $V' = V$ gilt, so nennt man H einen* (auf)spannenden Teilgraphen.

Teilgraphen haben viele interessante Eigenschaften, wenn sie sich aus einer gegebenen Knoten- oder Kantenmenge ableiten lassen. In diesem Fall spricht man auch von *induzierten* Teilgraphen. Seien $G = (V, E)$ ein Graph und $V' \subseteq V$ sowie $E' \subseteq E$ zwei Teilmengen der Knoten- beziehungsweise Kantenmenge. Aus V' wird ein sogenannter *knoteninduzierter* Teilgraph erstellt, indem zu V' alle Kanten aus $E(G)$ hinzugefügt werden, die beide Endknoten in V' haben. Ein *kanteninduzierter* Teilgraph wird aus E' erstellt, wenn alle Knoten ausgewählt werden, die in mindestens einer Kante in E' vorkommen.

Definition 1.5 (induzierter Teilgraph) *Seien $G = (V, E)$ ein Graph und $V' \subseteq V$ sowie $E' \subseteq E$ Teilmengen seiner Knoten und Kanten. Dann heißt $H = (V', E')$* (knoten-)induzierter Teilgraph *von G, falls $E' = E \cap \binom{V'}{2}$. $H = (V', E')$ heißt* kanteninduzierter Teilgraph *von G, falls $V' = \bigcup_{e \in E'} e$.*[14]

Ein induzierter Teilgraph H eines Graphen $G = (V, E)$ ist bereits durch die Angabe einer Knotenmenge $V' \subseteq V$ vollständig beschrieben. Man nennt H daher auch *den von V' induzierten Teilgraphen* und notiert diesen Graphen als $G[V']$. Analog nennt man $G[E'] = (\bigcup_{e \in E'} e, E')$ den von E' kanteninduzierten Teilgraphen. Zur Unterscheidung von Teilgraphen und induzierten Teilgraphen werden erstere bisweilen auch *schwache Teilgraphen* genannt.

Für eine Knotenmenge $X \subseteq V$ definiere $G - X := G[V \setminus X]$. Für eine Kantenmenge $F \subseteq E$ definiere $G - F := (V, E \setminus F)$. Für einzelne Elemente $v \in V$ beziehungsweise $e \in E$ existieren auch abkürzende Schreibweisen $G - v$ und $G - e$ anstelle von $G - \{v\}$ und $G - \{e\}$.



(a) G (b) H (c) H_V (d) H_E

Abbildung 1.2: Graphen und Teilgraphen

Beispiel 1.4 In Abbildung 1.2 ist in (a) ein Graph G dargestellt. Der Graph H aus (b) ist ein schwacher Teilgraph von G, der aus einer Teilmenge aller Knoten und Kanten besteht. H_V aus (c) ist ein durch die Knotenmenge $V' = \{1, 2, 3, 4, 6\}$ induzierter Teilgraph von G, da er alle Kanten aus G enthält, die beide Eckknoten in V' haben. Analog ist H_E aus (d) ein kanteninduzierter Teilgraph. Die entsprechende Kantenmenge ist $E' = \{\{2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}, \{4, 7\}\}$.

[14] V' ist die Menge aller Knoten, die in mindestens einer Kante $e \in E'$ vorkommen.

Wege, Pfade und Kreise

Ein *Kantenzug* in einem Graphen $G = (V, E)$ besteht aus einer Folge $v_0, v_1, \ldots, v_l$ von Knoten und einer Folge $e_1, \ldots, e_l$ von Kanten, wobei $l \geq 0$ und $e_i = \{v_{i-1}, v_i\} \in E$ für $i = 1, \ldots, l$ gilt. Falls der Kantenzug keine Kante enthält, so handelt es sich um einen *leeren* Kantenzug. Ein *Weg* ist ein Kantenzug, in dem keine Kante mehrfach vorkommt. Oft wird ein Weg der Einfachheit halber nur durch die Folge seiner Knoten $v_0, v_1, \ldots, v_l$ angegeben. Die Knoten v_0 und v_l werden als die *Endknoten* des Weges bezeichnet. Man sagt auch, dass der Weg die Knoten v_0 und v_l *verbindet*, beziehungsweise spricht von einem v_0-v_l-Weg. Der Wert l gibt die *Länge* des Weges an. Ein *Pfad* ist ein Weg, in dem kein Knoten mehrfach vorkommt. Mit $E(P)$ wird die Menge der Kanten des Pfades P bezeichnet. Ein Graph, der nur aus einem Pfad der Länge $n-1$ besteht, wird auch als P_n notiert.[15] Bei einem Pfad $v_0, v_1, \ldots, v_l$ werden die Knoten $v_1, \ldots, v_{l-1}$ *innere Knoten* des Pfades genannt. Der *Abstand* $dist(v, w)$ zweier Knoten v und w in einem Graphen G ist als die Länge eines kürzesten v-w-Pfades definiert. Falls ein solcher nicht existiert, wird der Abstand auf ∞ gesetzt. Der größte Abstand zweier Knoten in einem Graphen G ist der *Durchmesser* $diam(G)$ des Graphen. Knoten heißen *zentral*, wenn ihr größter Abstand zu anderen Knoten minimal ist. Dieser Abstand heißt *Radius* $rad(G)$ von G und ist formal als $rad(G) \coloneqq \min_{v \in V(G)} \max_{w \in V(G)} dist(v, w)$ definiert.

Ein *geschlossener Kantenzug* in einem Graphen $G = (V, E)$ besteht aus einer Folge $v_1, \ldots, v_l$ von Knoten und einer Folge $e_1, \ldots, e_l$ von Kanten, wobei $l \geq 2$ und $e_i = \{v_i, v_{i+1}\} \in E$ für $i = 1, \ldots, l-1$ und $e_l = \{v_l, v_1\} \in E$ gilt. Der Wert l gibt die *Länge* des geschlossenen Kantenzugs an. Ein *Zykel* ist ein geschlossener Kantenzug, in dem keine Kante mehrfach vorkommt. Ein *Kreis* ist ein Zykel, in dem kein Knoten mehrfach vorkommt. Ein Kreis muss somit mindestens drei Knoten enthalten. Mit C_n bezeichnet man einen Kreis der Länge n, das heißt einen Kreis mit n Kanten und n Knoten. Ein Kreis heißt *gerade* oder *ungerade*, je nachdem ob seine Länge gerade oder ungerade ist. Die *Taillenweite* (engl. *girth*) $g(G)$ ist die Länge eines kürzesten Kreises in G. Falls G keinen Kreis besitzt, so wird $g(G) = \infty$ gesetzt.

Ein Graph $G = (V, E)$ heißt *zusammenhängend*, falls es für je zwei Knoten $v, w \in V$ einen v-w-Pfad gibt. Ein inklusionsmaximaler zusammenhängender Teilgraph von G heißt *Zusammenhangskomponente*. Die Anzahl der Zusammenhangskomponenten eines Graphen G wird mit $c(G)$ notiert.

Bäume und Wälder

Viele Algorithmen haben auf Graphen, die keine Kreise enthalten, besonders schnelle Laufzeit. Daher wird diese Gruppe von Graphen auch unter dem Begriff Bäume beziehungsweise Wälder zusammengefasst.

Definition 1.6 (Baum, Wald) *Ein Graph wird* Wald *genannt, falls er keinen Kreis enthält. Ein zusammenhängender und kreisfreier Graph heißt* Baum.

Ein Wald ist somit ein Graph, dessen Zusammenhangskomponenten Bäume sind. Ist ein Baum T Teilgraph eines Graphen G und gilt $V(T) = V(G)$, so wird T *spannender Baum* genannt. Ein Knoten eines Baumes heißt *Blatt*, falls er Grad höchstens 1 hat.

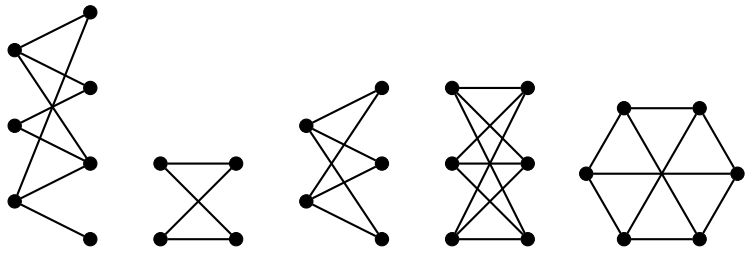
[15] Dieser Graph enthält genau n Knoten.

1.6.2 Bipartite Graphen

Eine spezielle Klasse von Graphen stellen die bipartiten beziehungsweise allgemein r-partiten Graphen dar. Ein Graph kann dabei in Partitionen geteilt werden, sodass Kanten nur zwischen den Partitionen verlaufen.

Definition 1.7 (bipartiter Graph) *Ein Graph $G = (V, E)$ heißt genau dann* bipartit[16]*, wenn sich seine Knotenmenge V in zwei Teile A und B partitionieren lässt, sodass Kanten nur zwischen A und B verlaufen.*

Bipartite Graphen werden oft auch als Tripel $G = (A, B, E)$[17] mit $E \subseteq A \times B = \{\{a, b\} \mid a \in A, b \in B\}$ geschrieben. $A \dot\cup B$ heißt *Bipartition* von G und die Mengen A sowie B heißen *Bipartitionsmengen* oder auch einfach *Teile*. Der *vollständige bipartite* Graph mit Partitionen der Größe $|A| = r$ und $|B| = s$ wird mit $K_{r,s}$ notiert. Er hat $r \cdot s$ viele Kanten.



(a) G (b) $K_{2,2}$ (c) $K_{2,3}$ (d) $K_{3,3}$ (e) $K_{3,3}$

Abbildung 1.3: Bipartite Graphen

Beispiel 1.5 Abbildung 1.3 zeigt mehrere bipartite Graphen, wobei der $K_{3,3}$ sowohl in (d) als auch in (e) dargestellt ist.

Das Partitionieren von Graphen kann auch auf eine größere Anzahl von Teilen angewandt werden. Sei $r \geq 2$ eine natürliche Zahl. Ein Graph $G = (V, E)$ heißt r-partit, wenn eine Partition von V in r Teile existiert, sodass die Endknoten einer jeden Kante von G in verschiedenen Partitionsklassen liegen. Ein bipartiter Graph ist somit 2-partit.

1.6.3 Matchings

Von besonderem Interesse in der vorliegenden Arbeit sind spezielle Kantenmengen, in denen Kanten keine gemeinsame Knoten haben. Viele theoretischen Probleme lassen sich auf das Suchen solcher Mengen reduzieren.

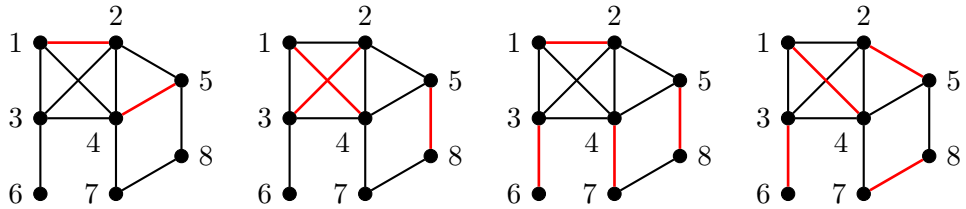
Definition 1.8 (Matching) *Sei $G = (V, E)$ ein Graph. Eine Menge $M \subseteq E$ heißt* Matching*, falls keine zwei Kanten aus M einen gemeinsamen Knoten haben.*

Im Deutschen wird ein Matching auch *Paarung* genannt, jedoch hat sich auch hier der englische Begriff durchgesetzt.

Sei $G = (V, E)$ ein Graph. Eine Menge M heißt *maximales Matching*, falls $M \cup \{e\}$ für alle $e \in E \setminus M$ kein Matching ist. Ein Matching M heißt *größtes Matching* (oder auch

[16] In der deutschen Literatur manchmal auch *paar* genannt.
[17] oder als Tupel $G = (A \dot\cup B, E)$

Maximummatching), falls $|M| \geq |M'|$ für alle Matchings M' von G gilt. Die Kardinalität eines größten Matchings in G wird mit $\nu(G)$ bezeichnet. Ein Matching M heißt *perfekt*, falls $2 \cdot |M| = |V|$ gilt. Ein Knoten $v \in V$ heißt *überdeckt bezüglich eines Matchings* M, falls es eine Kante $e \in M$ gibt mit $v \in e$.



(a) M_1 (b) M_2 (c) M_3 (d) M_4

Abbildung 1.4: Verschiedene Matchings für den Graphen G aus Abbildung 1.2a. Die Mengen der farbigen Kanten bilden jeweils ein Matching.

Beispiel 1.6 Abbildung 1.4 zeigt vier verschiedene Matchings ((a) bis (d)) im Graphen G aus Abbildung 1.2a. M_2 ist ein maximales Matching, denn das Hinzufügen einer beliebigen Kante würde die Matchingsbedingung verletzen. Es ist aber kein größtes Matching, da es nur aus drei Kanten besteht. M_3 und M_4 sind mit jeweils vier Kanten perfekte und demzufolge auch größte Matchings.

Größte Matchings in bipartiten Graphen

Für das Berechnen größter Matchings in bipartiten Graphen existieren mehrere Algorithmen, die jeweils polynomielle Laufzeit haben. Im Folgenden werden zwei Verfahren vorgestellt, die existierende Matchings sukzessive mithilfe sogenannter *augmentierender* Pfade vergrößern, bis das Maximum erreicht ist.

Ein Pfad in einem Graphen $G = (V, E)$ heißt *alternierend* bezüglich eines Matchings M (kurz M-alternierend), falls er abwechselnd Kanten aus M und aus $E(G) \setminus M$ benutzt. Ein Pfad der Länge mindestens 1 heißt *M-augmentierend* (oder *M-verbessernd*), falls er alternierend ist und Anfangs- und Endknoten nicht von M überdeckt sind.

Beispiel 1.7 Für das Matching M_2 aus Abbildung 1.4b ist der Pfad $P = 6, 3, 2, 5, 8, 7$ ein augmentierender Pfad, da er abwechselnd Matchingkanten (rot) und Nicht-Matchingkanten (schwarz) benutzt und Anfangs- (6) sowie Endknoten (7) nicht überdeckt sind. Auch $P' = 6, 3, 2, 1, 4, 7$ ist ein M_2-augmentierender Pfad. Zu den (jeweils größten) Matchings M_3 und M_4 in Abbildung 1.4c beziehungsweise 1.4d existieren keine augmentierende, sondern nur alternierende Pfade.

Zwischen augmentierenden Pfaden und größten Matchings besteht folgender einfacher Zusammenhang.

Satz 1.1 (Berge [8]) *Ein Matching M ist genau dann ein größtes Matching, wenn kein M-augmentierender Pfad existiert.*

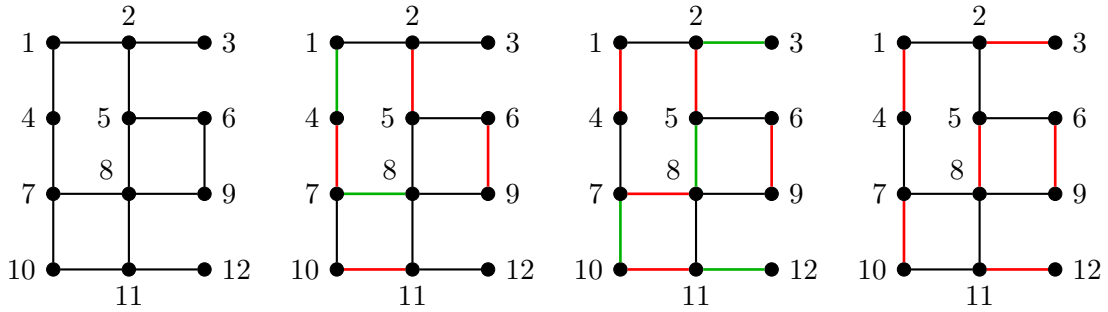
Beweis: Falls es einen M-augmentierenden Pfad gibt, so ist M offensichtlich kein größtes Matching, denn durch Vertauschen von Matchingkanten und Nicht-Matchingkanten entlang des M-augmentierenden Pfades wird ein größeres Matching erzeugt. Somit gibt es bezüglich eines größten Matchings M keine M-augmentierenden Pfade.

Angenommen, M ist kein größtes Matching. Sei M' ein Matching mit $|M'| > |M|$. Betrachte die symmetrische Differenz $M \Delta M'$[18]. Diese induziert einen Graphen mit Maximalgrad 2 und besteht somit aus der Vereinigung von geraden Kreisen und Pfaden. Da $|M'| > |M|$ muss es insbesondere auch mindestens einen Pfad geben, der mit jeweils einer Kante aus M' beginnt und endet. Dieser ist M-augmentierend. □

Der Beweis von Satz 1.1 zeigt sogar eine etwas stärkere Aussage, die später für einen vorgestellten Algorithmus benötigt wird.

Korollar 1.2 *Seien M_1 und M_2 Matchings in einem Graphen G mit $|M_1| > |M_2|$. Dann existieren mindestens $|M_1| - |M_2|$ viele (knoten-)disjunkte M_2-augmentierende Pfade in G, die nur Kanten aus $M_1 \Delta M_2$ enthalten.*

Beweis: Der Beweis folgt direkt aus dem Beweis von Satz 1.1. □



(a) G (b) M_1 (c) M_2 (d) M_3

Abbildung 1.5: Alternierende und augmentierende Pfade: Die roten Kanten stellen jeweils das Matching dar, die grünen Kanten veranschaulichen Nicht-Matchingkanten augmentierender Pfade.

Beispiel 1.8 Abbildung 1.5 zeigt augmentierende Pfade bezüglich verschiedener Matchings. Zum bipartiten Graphen $G = (V, E)$ aus (a) bilden die roten Kanten in (b) das Matching M_1. Ein M_1-augmentierender Pfad ist zum Beispiel $P_1 = 1, 4, 7, 8$, da er abwechselnd Kanten aus M_1 (rot) und $E \setminus M_1$ (grün)[19] benutzt und Anfangs- (1) sowie Endknoten (8) nicht überdeckt sind. M_2 in (c) entsteht durch das Vertauschen von Matchingkanten und Nicht-Matchingkanten entlang von P_1 und ist offensichtlich um 1 größer als M_1. M_3 (siehe (d)) entsteht aus M_2, indem anhand des augmentierenden Pfades $P_2 = 3, 2, 5, 8, 7, 10, 11, 12$ Kanten vertauscht werden. In M_3 existieren keine augmentierenden Pfade und es ist ein größtes Matching in G.

Ein einfacher Algorithmus, der ein größtes Matching mithilfe von Satz 1.1 konstruiert, erzeugt in einem Graphen ein beliebiges (oder auch leeres Matching), berechnet sukzessive M-augmentierende Pfade und vergrößert jeweils das Matching, indem Kanten entlang der Pfade vertauscht werden. Wenn kein augmentierender Pfad mehr existiert, ist das erhaltene Matching ein größtes. Seien n die Anzahl der Knoten und m die Anzahl der Kanten des

[18] $M \Delta M' \coloneqq (M \setminus M') \cup (M' \setminus M)$

[19] Die Nicht-Matchingkanten von P_1 sind grün hervorgehoben. Die restlichen Kanten aus $E \setminus M_1$ sind schwarz gezeichnet.

Graphen. Der Standardalgorithmus für bipartite Graphen findet einen augmentierenden Pfad durch eine modifizierte Breitensuche. Da höchstens $n/2$ Augmentierungen[20] nötig sind, die jeweils in Zeit $\mathcal{O}(m)$ bewerkstelligt werden können, hat dieser Algorithmus Laufzeit $\mathcal{O}(nm)$. Der Algorithmus von Hopcroft und Karp [67] dagegen arbeitet in Phasen und erreicht durch quasi simultanes Augmentieren mehrerer Pfade eine Laufzeit von $\mathcal{O}(\sqrt{n}m)$. Beide Verfahren werden im Folgenden vorgestellt.

Seien ein Graph G und ein Matching M gegeben. Ein Baum in G mit einem als Wurzel ausgezeichneten Knoten heißt *M-alternierend*, falls die Wurzel nicht von M überdeckt ist und alle von der Wurzel zu den Blättern gehenden Pfade alternierend sind sowie gerade Länge haben. Ein *M-alternierender Wald* ist die disjunkte Vereinigung M-alternierender Bäume.

Ein alternierender Baum mit Wurzel x enthält alle potenziellen Anfangsstücke von augmentierenden Pfaden, die in x beginnen. Die Idee bei der Suche nach augmentierenden Pfaden in bipartiten Graphen ist daher, alternierende Bäume so lange zu erweitern, bis eine Nicht-Matchingkante gefunden wird, die zwei Knoten verschiedener alternierender Bäume verbindet. Haben beide Knoten geraden Abstand zu ihrer jeweiligen Wurzel, hat man in diesem Fall einen augmentierenden Pfad von der Wurzel des einen alternierenden Baumes zur Wurzel des anderen alternierenden Baumes gefunden. Falls eine solche Kante nicht existiert und kein alternierender Baum erweitert werden kann, so nennt man den erhaltenen Graphen einen *ungarischen Wald*[21].

Der Algorithmus *Bipartites Matching* bestimmt nach genau dieser Idee augmentierende Pfade in bipartiten Graphen und ist in Algorithmus 1.1 gegeben.

Satz 1.3 *Der Algorithmus Bipartites Matching bestimmt ein größtes Matching im bipartiten Graphen G und hat Laufzeit $\mathcal{O}(nm)$.*

Beweis: Zum Nachweis der Korrektheit reicht es offensichtlich zu zeigen, dass M größtes Matching ist, wenn die While-Schleife ohne augmentierenden Pfad beendet wird.[22] Hierbei ist F ein ungarischer Wald.

Nach dem Verlassen der While-Schleife sind gemäß der Bedingung die Knoten in V_{even} nur mit Knoten aus V_{odd} adjazent. Jedes Matching lässt somit mindestens $|V_{\text{even}}| - |V_{\text{odd}}|$ viele Knoten in G unüberdeckt. Andererseits enthält jede Komponente des ungarischen Waldes F genau einen von M nicht überdeckten Knoten (Wurzel). Auf der Menge $V \setminus (V_{\text{even}} \cup V_{\text{odd}})$ bildet M ein perfektes Matching, da alle ungematchten Knoten in V_{even} liegen und es keine Matchingkante gibt, von der nur ein Endknoten im ungarischen Wald liegt. Die Anzahl der unüberdeckten Knoten ist demnach gleich der Anzahl der Komponenten des ungarischen Waldes $|V_{\text{even}}| - |V_{\text{odd}}|$, das heißt, M ist größtes Matching.

Ohne Einschränkung kann $n = \mathcal{O}(m)$ angenommen werden. Die Repeat-Until-Schleife hat Laufzeit $\mathcal{O}(n+m)$ und wird maximal $n/2$-mal durchlaufen, womit sich die angegebene Gesamtlaufzeit ergibt. □

[20] Das größte Matching kann bei n Knoten maximal Größe $n/2$ haben und in jedem Augmentierungsschritt vergrößert sich das Matching um mindestens 1.

[21] Die Bezeichnung „ungarisch“ bezieht sich auf die beiden ungarischen Mathematiker Dénes Kőnig (1884–1944) und Jenő Egerváry (1891–1958) und wurde vom US-amerikanischen Mathematiker Harold William Kuhn (1925–2014) [86] eingeführt. Ein ungarischer Wald muss nicht notwendigerweise alle Knoten des Graphen enthalten. Als Beispiel betrachte man einen Pfad der Länge 3, an den an einen der beiden mittleren Knoten eine Kante angehängt wird.

[22] In diesem Fall ist P undefiniert, die Repeat-Until-Schleife wird verlassen und der Algorithmus terminiert mit der Rückgabe von M.

Algorithmus 1.1 Bipartites Matching

Eingabe: $G = (V, E)$: bipartiter Graph
Ausgabe: M: größtes Matching in G

```
1:  M ← ∅
2:  repeat
3:    F ← Teilgraph von G, der aus allen von M nicht überdeckten Knoten besteht
4:    V_even ← Menge aller Knoten in F
5:    V_odd ← ∅
6:    P ← undefined                                   # augmentierender Pfad
7:    while ∃{x, y} ∈ E mit x ∈ V_even, y ∉ V_odd do
8:      if y ∉ V_even then
9:        Erweitere F um {x, y} und Matchingkante {y, z} ∈ M
10:       Füge y in V_odd und z in V_even ein
11:     else
12:       P ← augmentierender Pfad von Wurzel des Baumes, der x enthält, zu Wurzel
          des Baumes, der y enthält
13:       Augmentiere M entlang P
14:       break                                       # beendet While-Schleife
15:     end if
16:   end while
17: until P = undefined                               # F ist ungarischer Wald
18: return M
```

Mit nur wenig mehr Aufwand gewinnt man einen $\mathcal{O}(\sqrt{n}m)$-Algorithmus. Der Algorithmus von Hopcroft und Karp [67] arbeitet in Phasen, während derer das bestehende Matching jeweils entlang mehrerer disjunkter augmentierender Pfade „gleichzeitig“ vergrößert wird. Bezeichne M das bereits konstruierte Matching (zu Beginn gilt also $M = \emptyset$). In jeder Phase wird eine maximale Menge knotendisjunkter kürzester M-augmentierender Pfade bestimmt und das bestehende Matching M entlang dieser Pfade augmentiert. Die erste Phase berechnet also schon ein maximales Matching im Graphen G. Es zeigt sich, dass einerseits jede Phase nur lineare Laufzeit erfordert und dass andererseits nach maximal $\mathcal{O}(\sqrt{n})$[23] vielen Phasen keine M-augmentierende Pfade mehr vorhanden sind. In diesem Fall ist M ein größtes Matching.

Für ein Matching M bezeichne $l(M)$ die Länge eines kürzesten M-augmentierenden Pfades. Falls M größtes Matching ist und somit kein augmentierender Pfad existiert, so setze $l(M) = \infty$. Das folgende Lemma 1.4 besagt, dass sich die Länge eines kürzesten augmentierenden Pfades immer vergrößert, wenn ein Matching mithilfe einer (inklusions-)maximalen Menge knotendisjunkter augmentierender Pfade[24] vergrößert wird. Genau auf dieser Idee basiert der Algorithmus von Hopcroft und Karp.

Lemma 1.4 *Seien M ein Matching und $P_1, \ldots, P_t$ eine (inklusions-)maximale Menge knotendisjunkter M-augmentierender Pfade der Länge $l(M)$. Dann gilt*

$$l(M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)) > l(M).$$

[23] Genauer gesagt werden maximal $\frac{3}{2}\sqrt{n}$ viele Phasen benötigt.

[24] Eine Menge knotendisjunkter augmentierender Pfade heißt *(inklusions-)maximal*, wenn kein weiterer Pfad hinzugefügt werden kann, ohne die Knotendisjunktheit zu verletzen.

Beweis: Sei Q ein augmentierender Pfad in $M' \coloneqq M \Delta E(P_1) \Delta \cdots \Delta E(P_t)$.[25] Dann gilt $|M' \Delta E(Q)| = |M| + t + 1$.[26] Nach Korollar 1.2 gibt es somit $t+1$ knotendisjunkte M-augmentierende Pfade $Q_1, \ldots, Q_{t+1}$ in $M \Delta (M' \Delta E(Q))$, die alle Länge mindestens $l(M)$ haben. Man erhält:

$$(t+1) \cdot l(M) \leq |E(Q_1)| + \cdots + |E(Q_{t+1})| \leq |M \Delta (M' \Delta E(Q))|.$$

Letztere Ungleichung ergibt sich dabei wegen $E(Q_i) \subseteq M \Delta (M' \Delta E(Q))$. Aus der Definition von M' und der Eigenschaft, dass die P_i knotendisjunkt sind, folgt[27]

$$\begin{aligned} |M \Delta (M' \Delta E(Q))| &= |E(P_1) \Delta \cdots \Delta E(P_t) \Delta E(Q)| \\ &= |(E(P_1) \cup \cdots \cup E(P_t)) \Delta E(Q)| \\ &= |E(P_1) \cup \cdots \cup E(P_t)| + |E(Q)| - 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| \\ &= t \cdot l(M) + |E(Q)| - 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)|. \end{aligned}$$

Somit gilt also

$$|E(Q)| \geq l(M) + 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)|.$$

Falls Q knotendisjunkt zu $P_1 \cup \cdots \cup P_t$ ist, so gilt $|E(Q)| > l(M)$, denn andernfalls wären die P_i nicht inklusionsmaximal. Falls Q nicht knotendisjunkt zu $P_1 \cup \cdots \cup P_t$ ist, so ist Q auch nicht kantendisjunkt zu $P_1 \cup \cdots \cup P_t$ (da in M' alle Knoten der P_i überdeckt sind) und es gilt daher $|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| > 0$. Somit erhält man in diesem Fall

$$|E(Q)| \geq l(M) + 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| > l(M),$$

was den Beweis vervollständigt. □

Beispiel 1.9 In Abbildung 1.5 auf Seite 16 sind ein Graph G und drei Matchings M_1, M_2, M_3 gegeben. Kürzeste M_1-augmentierende Pfade sind $P_1 = 1, 4, 7, 8$ (siehe (b)) und $P_1' = 3, 2, 5, 8$. Beide haben die Länge 3, das heißt, $l(M_1) = 3$. Da P_1 und P_1' aber nicht knotendisjunkt sind (beide beinhalten Knoten 8), kann M_1 nur mithilfe eines der beiden Pfade augmentiert werden. Wird dafür P_1 ausgewählt, so entsteht als vergrößertes Matching M_2 (siehe (c)). Nach Lemma 1.4 muss sich die Länge eines kürzesten augmentierenden Pfades vergrößert haben. Da $P_2 = 3, 2, 5, 8, 7, 10, 11, 12$ der einzige augmentierende Pfad ist und Länge 7 hat, folgt $l(M_2) = 7$. Das größte Matching M_3 in (d) entsteht durch das Augmentieren von M_2 mithilfe von P_2 und hat keine augmentierende Pfade mehr (d. h. $l(M_3) = \infty$).

Satz 1.5 *Der Algorithmus von Hopcroft und Karp findet ein größtes Matching in bipartiten Graphen in $\mathcal{O}(\sqrt{n}m)$.*

Beweis: Zunächst wird gezeigt, dass die While-Schleife im Algorithmus von Hopcroft und Karp nur $\mathcal{O}(\sqrt{n})$-mal durchlaufen wird. Nach jedem Durchlauf erhöht sich die Länge eines

[25] $E(P)$ bezeichnet die Menge der Kanten des Pfades P.
[26] Denn $|M'| = |M \Delta E(P_1) \Delta \cdots \Delta E(P_t)| = |M| + t$ und Q ist M'-augmentierend.
[27] Man beachte hierbei, dass für zwei Mengen A und B gilt: $|A \Delta B| = |A \setminus B| + |B \setminus A| = |A| - |A \cap B| + |B| - |A \cap B|$.

Algorithmus 1.2 Hopcroft-Karp

Eingabe: $G = (V, E) = (A \dot\cup B, E)$: bipartiter Graph
Ausgabe: M: größtes Matching in G

1: $M \leftarrow \emptyset$
2: **while** $l(M) < \infty$ **do**
3: $S \leftarrow$ Menge nicht überdeckter Knoten in A
4: $T \leftarrow$ Menge über alternierende Pfade der Länge $l(M)$ von S aus erreichbarer Knoten
5: Bestimme mittels Tiefensuche von den Knoten in T eine maximale Menge augmentierender Pfade der Länge $l(M)$ und augmentiere diese
6: **end while**
7: **return** M

kürzesten augmentierenden Pfades nach Lemma 1.4 um mindestens 1. Sei M^* ein größtes Matching von G und M das gefundene Matching nach $\sqrt{n}$ Durchläufen der While-Schleife. Dann beträgt die Länge eines kürzesten M-augmentierenden Pfades mindestens $\sqrt{n}+1$ und nach Korollar 1.2 existieren mindestens $|M^*| - |M| = \nu(G) - |M|$ M-augmentierende Pfade. Da diese knotendisjunkt sind und alle mindestens Länge $\sqrt{n}+1 \geq \sqrt{n}$ haben, folgt

$$\begin{aligned} n &\geq (\text{Anzahl der Pfade}) \cdot (\text{minimale Anzahl Knoten eines Pfades}) \\ &\geq (\nu(G) - |M|) \cdot \sqrt{n} \end{aligned}$$

und daraus

$$\sqrt{n} \geq \nu(G) - |M|.$$

Somit kann M noch höchstens $\sqrt{n}$-mal augmentiert werden, weil durch jedes Augmentieren $|M|$ um mindestens 1 erhöht wird und $|M| \leq \nu(G)$ gilt. Insgesamt gibt es also höchstens $\sqrt{n} + \sqrt{n} = \mathcal{O}(\sqrt{n})$ viele Durchläufe der While-Schleife.

Es bleibt zu zeigen, dass jeder Durchlauf der While-Schleife eine Laufzeit von $\mathcal{O}(n+m)$ benötigt. Die Länge $l(M)$ kann mittels einer modifizierten Version der Breitensuche, die von allen nicht überdeckten Knoten in A ausgeht, in linearer Zeit gefunden werden: in jedem Schritt werden ausgehend von einem Knoten in A alle dessen Nachbarn besucht, diese aber nicht in die Warteschlange aufgenommen, sondern alle die Knoten (in A), die mit diesen Nachbarn über eine Matchingkante verbunden sind. Die Menge T bestimmt man dadurch, dass man die Breitensuche nur bis zu einer Tiefe von $l(M)$ laufen lässt. Zur Bestimmung einer maximalen Menge knotendisjunkter augmentierender Pfade startet man in den Knoten von T jeweils eine Tiefensuche zu Knoten in S, wobei einmal besuchte Knoten für folgende Tiefensuchen verboten werden und nur Kanten benutzt werden, die den Abstand entlang eines alternierenden Pfades zu einem Knoten in S verringern[28]. Sobald ein Knoten in S erreicht wird, wird die Tiefensuche beendet, ein augmentierender Pfad ist gefunden und eine Tiefensuche vom nächsten Knoten in T wird gestartet. Es ist damit sichergestellt, dass jeder Knoten maximal einmal besucht wird und jede Kante somit maximal zweimal abgesucht wird. Damit ist die Gesamtlaufzeit eines Durchlaufes

[28] Durch die vorherige Breitensuche wurde berechnet, welchen Abstand jeder Knoten zu einem Knoten in S hat. Dadurch ist für jeden Knoten klar, welche seiner Nachbarknoten den Abstand zu Knoten in S verringern und welche nicht.

der While-Schleife $\mathcal{O}(m)$, falls der Graph zusammenhängend ist. Ansonsten liefert eine Betrachtung der Komponenten des Graphen ebenfalls Laufzeit $\mathcal{O}(m)$. □

1.6.4 Repräsentation von Graphen

Um Berechnungen auf Graphen effizient durchzuführen, wurden für Graphen verschiedene Datenstrukturen entwickelt. Die wichtigsten sind die Adjazenzmatrix, die Adjazenzliste und die Inzidenzmatrix.

Sei $G = (V, E)$ ein Graph mit $V = \{v_1, \ldots, v_n\}$. Dann heißt die $(n \times n)$-Matrix $A = (a_{ij})$ mit den Einträgen

$$a_{ij} = \begin{cases} 1, & \text{falls } \{v_i, v_j\} \in E \\ 0, & \text{sonst} \end{cases}$$

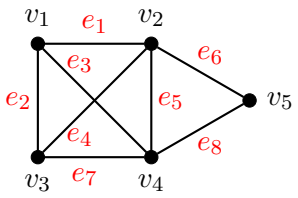
die *Adjazenzmatrix* von G. Sie speichert für alle Paare von Knoten, ob sie durch eine Kante verbunden sind.

Eine *Adjazenzliste* verwaltet für jeden Knoten v_i eine Liste mit seinen Nachbarn. Falls die Anzahl der Knoten konstant bleibt, können die einzelnen Listen in einem Feld verwaltet werden, wobei das Feldelement mit Index i auf die Adjazenzliste von Knoten v_i verweist. Falls sich die Anzahl der Knoten dynamisch ändert, werden die Adjazenzlisten typischerweise ebenfalls in einer doppelt verketteten Liste verwaltet.

Eine weitere Möglichkeit, um Graphen zu repräsentieren, ist, die Endknoten für jede Kante des Graphen zu speichern. Sei $E = \{e_1, \ldots, e_m\}$, dann heißt die $(n \times m)$-Matrix $B = (b_{ij})$ mit den Einträgen

$$b_{ij} = \begin{cases} 1, & \text{falls } v_i \in e_j \\ 0, & \text{sonst} \end{cases}$$

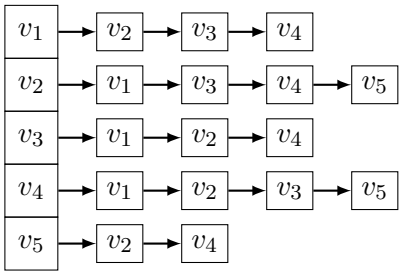
Inzidenzmatrix von G.



(a) G

	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	1	0
v_2	1	0	1	1	1
v_3	1	1	0	1	0
v_4	1	1	1	0	1
v_5	0	1	0	1	0

(b) Adjazenzmatrix A



(c) Adjazenzliste

	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
v_1	1	1	1	0	0	0	0	0
v_2	1	0	0	1	1	1	0	0
v_3	0	1	0	1	0	0	1	0
v_4	0	0	1	0	1	0	1	1
v_5	0	0	0	0	0	1	0	1

(d) Inzidenzmatrix B

Abbildung 1.6: Repräsentation von Graphen. Kanten sind in (a) farbig nummeriert.

Beispiel 1.10 Abbildung 1.6 zeigt für den Graphen $G = (V, E)$ aus (a) in (b) die Adjazenzmatrix, in (c) die Adjazenzliste und in (d) die Inzidenzmatrix.

Je nach gewählter Repräsentation hat das Initialisieren der Datenstruktur und das Einfügen oder Löschen von Knoten beziehungsweise Kanten unterschiedliche Laufzeit. An dieser Stelle wird aber auf eine detaillierte Untersuchung der Komplexität verzichtet. Eine Übersicht dazu findet man zum Beispiel bei Turau [138].

1.6.5 Hypergraphen

Hypergraphen können als Verallgemeinerungen von Graphen angesehen werden. Dabei wird auf die Forderung, dass jede Kante genau zwei Knoten enthält, verzichtet.

Definition 1.9 (Hypergraph) *Ein* Hypergraph *ist ein Tupel $\mathcal{H} = (V, \mathcal{E})$, wobei V eine Menge und $\mathcal{E} \in \mathcal{P}(V)$ eine Menge von nichtleeren Teilmengen von V ist.*

Die Elemente von $\mathcal{E}$ werden häufig auch *Hyperkanten* genannt. Hypergraphen sind meist nur schwer darstellbar. In der Regel werden die Knoten als Punkte und die Hyperkanten als geschlossene Linien gezeichnet (siehe Abbildung 1.7). Besteht der Hypergraph aus vielen Kanten, ist eine übersichtliche Darstellung aber meist kaum möglich.

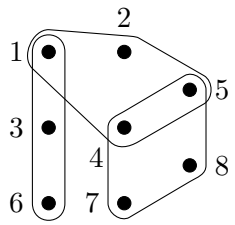


Abbildung 1.7: Hypergraph mit acht Knoten und den Kanten $\{1, 3, 6\}$, $\{1, 2, 4, 5\}$ sowie $\{4, 5, 7, 8\}$

Ein wichtiges Problem in dieser Arbeit ist das Bestimmen größter Matchings. Dafür existiert auch für Hypergraphen eine Verallgemeinerung.

Definition 1.10 (k-Matching) *Sei $\mathcal{H} = (V, \mathcal{E})$ ein Hypergraph. Eine Teilmenge der Kantenmenge $\mathcal{M} \subseteq \mathcal{E}$ heißt* k-Matching *in $\mathcal{H}$, falls jeder Knoten $v \in V$ mit höchstens k Kanten aus $\mathcal{M}$ inzident ist.*

1-Matchings heißen auch kurz Matchings. Im Gegensatz zu einfachen Graphen ist das Matchingproblem in Hypergraphen wesentlich schwerer zu lösen. Es ist äquivalent zum Set Packing-Problem[29], welches eins der 21 Probleme ist, für die Karp [77] bereits 1972 die NP-Vollständigkeit zeigte. Auch für den Spezialfall, dass alle Kanten aus genau drei Knoten aus jeweils verschiedenen Teilmengen von V bestehen, ist das Bestimmen eines größtes Matchings NP-vollständig [77, 74]. Dieses Problem wird 3-dimensionales Matching genannt.

[29] Im Deutschen wird auch vom Mengenpackungsproblem gesprochen.

1.7 Lineare Programmierung

Sei folgendes einfaches Beispiel gegeben. Ein Unternehmen stellt zwei Produkte P_1 und P_2 her, für deren Fertigung die drei Ausgangsstoffe A_1, A_2 und A_3 verwendet werden. Für 1 Mengeneinheit (ME) von P_1 wird je 1 ME A_1 und A_2 sowie 3 ME A_3 benötigt. P_2 lässt sich durch 4 ME A_1 sowie je 1 ME A_2 und A_3 herstellen. Der Verkauf von P_1 bringt einen Gewinn von 1 Geldeinheit (GE) und der von P_2 2 GE. Das Unternehmen möchte seinen Gewinn maximieren und sucht für beide Produkte die optimale Herstellungsmenge, wenn insgesamt 12 ME von A_1, 4 ME von A_2 und 9 ME von A_3 vorhanden sind.

Bezeichne x_1 die Herstellungsmenge von P_1 und x_2 die von P_2. Damit die Einschränkungen für die drei Ausgangsstoffe A_1, A_2 und A_3 erfüllt sind, müssen für alle möglichen Werte von x_1 und x_2 folgende drei Ungleichungen gelten:

$$A_1\colon \quad x_1 + 4x_2 \leq 12 \tag{1.1}$$

$$A_2\colon \quad x_1 + \quad x_2 \leq 4 \tag{1.2}$$

$$A_3\colon 3x_1 + \quad x_2 \leq 9 \tag{1.3}$$

Der Gesamtgewinn G des Unternehmens ist

$$G = x_1 + 2x_2. \tag{1.4}$$

Gesucht sind nun Werte für x_1 und x_2, bei denen die Ungleichungen 1.1 bis 1.3 erfüllt sind und G maximiert wird. An dieser Stelle ist zunächst noch unklar, wie diese optimale Lösung berechnet wird. Die gegebene Aufgabe ist allerdings ein Beispiel für ein sogenanntes *lineares Programm*, welches im Folgenden vorgestellt wird und für das Lösungsalgorithmen existieren.

Die Grundidee der linearen Programmierung ist, eine lineare Funktion zu optimieren, die aus n Freiheitsgraden besteht und durch lineare Gleichungen oder Ungleichungen eingeschränkt ist. Sie stellt damit eine sehr allgemeine Methode zur Optimierung von Problemen dar, für die keine speziell entwickelten Algorithmen existieren. Dementsprechend spricht man auch von der *linearen Optimierung*.

Ein lineares Optimierungsproblem wird auch als lineares Programm (engl. *linear program*, LP) bezeichnet und ist wie folgt definiert:

Definition 1.11 (lineares Programm (LP)) *Seien $A \in \mathbb{R}^{m\times n}$ eine Matrix sowie $b \in \mathbb{R}^m$ und $c \in \mathbb{R}^n$ zwei Vektoren. Eine zulässige Lösung ist ein Vektor $x \in \mathbb{R}^n$, für den $Ax \leq b$ und $x \geq 0$ gilt. Gesucht ist eine Lösung, die das Produkt $c^T x$ maximiert:*

$$\max\{c^T x \mid Ax \leq b, x \geq 0\}. \tag{1.5}$$

$c^T x$ steht hier für das Standardskalarprodukt von c und x, wobei c^T den transponierten Vektor zu c bezeichnet.[30] Die Vergleichsoperatoren in Definition 1.11 sind komponenten-

[30] Das Standardskalarprodukt zweier reeller Vektoren $x, y \in \mathbb{R}^n$ mit $x = (x_1, \ldots, x_n)^T$ und $y = (y_1, \ldots, y_n)^T$ ist definiert als $\langle x, y\rangle := x_1y_1 + \cdots + x_ny_n = \sum_{i=1}^n x_iy_i = x^Ty$, wobei x^T den transponierten Vektor zu x bezeichnet und das Ergebnis eine reelle Zahl ist.

weise zu verstehen, zum Beispiel ist für $x = (x_1, \ldots, x_n)^T$ gefordert, dass $x_1 \geq 0, \ldots, x_n \geq 0$ gilt.[31]

Die Menge der zulässigen Lösungen eines linearen Programms wird als *zulässiger Bereich* bezeichnet. Eine zulässige Lösung, für welche das Maximum angenommen wird, heißt *optimale Lösung*. Ein lineares Programm muss nicht immer eine optimale Lösung besitzen. Insgesamt werden dabei drei verschiedene Fälle unterschieden.

1. Das LP hat keine Lösung, da sich die Ungleichungen widersprechen. Ein solches LP heißt *unzulässig*.

2. Das LP ist *unbeschränkt*, das heißt, es gibt Lösungen mit beliebig hohem Zielfunktionswert.

3. Das LP besitzt mindestens eine optimale Lösung.

Ein LP ist demnach *zulässig*, wenn es unbeschränkt ist oder mindestens eine optimale Lösung besitzt.

Beispiel 1.11 Für das anfangs eingeführte Beispiel können die Matrix A und der Vektor b sehr einfach abgeleitet werden. Die Ungleichungen 1.1, 1.2 und 1.3 sind äquivalent zu

$$\underbrace{\begin{pmatrix} 1 & 4 \\ 1 & 1 \\ 3 & 1 \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x} \leq \underbrace{\begin{pmatrix} 12 \\ 4 \\ 9 \end{pmatrix}}_{b}. \tag{1.6}$$

Gesucht ist eine zulässige Lösung x mit maximalem Wert für

$$\underbrace{(1; 2)}_{c^T} \cdot \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{x}. \tag{1.7}$$

Folglich lässt sich das Problem als lineares Programm $\max\{c^T x \mid Ax \leq b, x \geq 0\}$ darstellen. Die optimale Lösung wird in Kapitel 1.7.1 graphisch hergeleitet.

Für lineare Programme gibt es mehrere äquivalente Formulierungen, wobei Definition 1.11 die sogenannte *Standardform* eines LP darstellt.[32] Andere Formen entstehen durch folgende Veränderungen in der Gleichung 1.5:

1. Minimierungs- statt Maximierungsproblem

2. Größer-gleich- statt Kleiner-gleich-Bedingungen

3. Gleichheits- statt Ungleichheitsbedingungen

4. Variablen ohne Nichtnegativitätsbedingung

[31] Die $\leq$- und $\geq$-Relationen können sehr einfach auf Vektoren definiert werden. Seien $x, y \in \mathbb{R}^n$, dann gilt $x \geq y$ genau dann, wenn $x_1 \geq y_1 \wedge \cdots \wedge x_n \geq y_n$. $\leq$ ist analog definiert.

[32] Neben der Standardform gibt es noch eine zweite Normalform der Darstellung von linearen Programmen, die *Slackform* heißt und bei der die Bedingungen als Gleichungen notiert sind.

Jedes LP lässt sich durch geeignete Umformungen in die oben angegebene Standardform bringen. Für die hier aufgelisteten Veränderungen sind das zum Beispiel:

1. Multipliziere den Zielfunktionsvektor c mit -1.
2. Multipliziere die entsprechende Ungleichung mit -1.
3. Ersetze $a_i x = b_i$ durch $a_i x \leq b_i$ und $-a_i x \leq -b_i$.
4. Ersetze x_i durch $x_i' - x_i''$ mit $x_i', x_i'' \geq 0$.

1.7.1 Geometrische Interpretation

Lineare Optimierungsprobleme können sehr intuitiv geometrisch interpretiert werden. Sei wie zuvor $A \in \mathbb{R}^{m \times n}$ eine Matrix sowie $b \in \mathbb{R}^m$ und $x \in \mathbb{R}^n$ zwei Vektoren. Eine Bedingung $Ax = b$ besteht aus m Gleichungen, die jeweils n Variablen $x_1, \ldots, x_n$ enthalten. Jede dieser Gleichungen beschreibt eine Hyperebene[33] im n-dimensionalen Raum. Ein Punkt $x = (x_1, \ldots, n_n)^T$, der eine dieser Gleichungen erfüllt, liegt in der entsprechenden Hyperebene. Die dazugehörige lineare Ungleichung beschreibt nun einen sogenannten Halbraum, der aus allen Punkten besteht, die auf der einen Seite der Hyperebene liegen (inkl. der Hyperebene selbst). Dieser Halbraum stellt die Menge aller gültigen Lösungen für die gegebene Ungleichung dar.

Definition 1.12 (Halbraum) *Eine Menge der Form* $\{x \in \mathbb{R}^n \mid a^T x \leq b, a \in \mathbb{R}^n, b \in \mathbb{R}\}$ *heißt* Halbraum.

Werden mehrere lineare Ungleichungen zu einem Ungleichungssystem kombiniert, besteht die Lösungsmenge gerade aus den Punkten, die alle Ungleichungen erfüllen, also den Schnitt der Halbräume. Dieser Schnitt bildet immer eine *konvexe Menge*, ein sogenanntes *Polyeder* P. Anschaulich ist das ein n-dimensionales Vieleck, bei dem die Verbindungslinie zwischen zwei beliebigen Punkten von P vollständig in P enthalten ist.

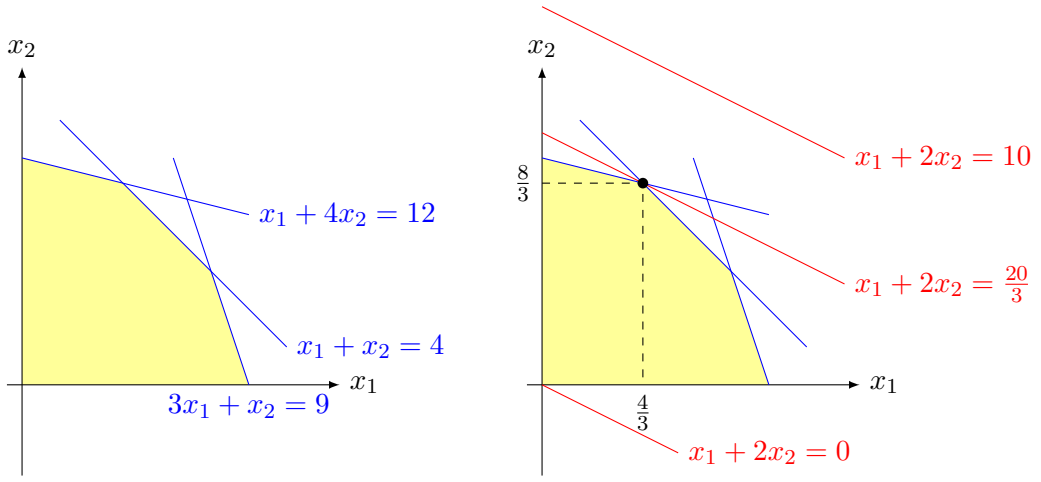
Definition 1.13 (konvexe Menge) *Eine Menge* $X \subseteq \mathbb{R}^n$ *heißt* konvex*, wenn für alle* $x, y \in X$ *und* $0 \leq \lambda \leq 1$ *auch* $z := \lambda x + (1 - \lambda) y \in X$ *gilt.* z *heißt* Konvexkombination *von* x *und* y*.*

Definition 1.14 (Polyeder, Polytop) *Seien* $A \in \mathbb{R}^{m \times n}$ *eine Matrix und* $b \in \mathbb{R}^m$ *ein Vektor. Dann heißt eine Menge* $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ *ein* Polyeder *im* $\mathbb{R}^n$*. Ein beschränktes Polyeder wird auch* Polytop *genannt.*

Die zu maximierende Zielfunktion $c^T x$ stellt ebenfalls eine Hyperebene dar, allerdings mit noch unbekanntem Abstand vom Ursprung und damit unbekanntem Zielwert. Wird die Hyperebene $\{x \mid c^T x = 0\}$ vom Unendlichen entgegen der Richtung des Vektors c verschoben, so ist die Lösung erreicht, sobald die Ebene das Polyeder zum ersten Mal berührt. Die Menge aller Berührungspunkte ist genau die Menge der optimalen Lösungen des linearen Programms. Es kann bewiesen werden, dass immer mindestens ein Extremalpunkt des Polyeders, eine sogenannte *Ecke*[34], in der Menge der optimalen Lösungen liegt.

[33] Eine Hyperebene in einem n-dimensionalen Raum ist ein $(n-1)$-dimensionaler Teilraum. Im 2-dimensionalen Raum ist das zum Beispiel eine Gerade, im 3-dimensionalen eine Ebene.

[34] Auf eine formale Definition einer Ecke eines Polyeders wird an dieser Stelle verzichtet und stattdessen auf die Literatur verwiesen (z. B. [28, 23, 128]).



(a) Ohne Zielfunktion (b) Mit Zielfunktion

Abbildung 1.8: Graphische Veranschaulichung eines LP. Die blauen Geraden beschränken den gelb gefärbten Bereich der zulässigen Lösungen. Die zu maximierende Zielfunktion ist rot dargestellt.

Hierbei erklärt sich auch, warum bei linearen Programmen keine strikten Ungleichungen wie $Ax < b$ zugelassen sind. Lösungen eines linearen Optimierungsproblems sind Punkte auf den durch die Ungleichungen definierten Hyperebenen. Bei strikten Ungleichungen könnte man die Punkte, die dann neben den Hyperebenen liegen, nicht exakt bestimmen.

Beispiel 1.12 Abbildung 1.8 veranschaulicht das bereits anfangs des Kapitels eingeführte Beispiel. Aus der Bedingung $Ax \leq b$ aus Ungleichung 1.6 werden zunächst drei Geraden für $Ax = b$ konstruiert. In diesem Fall sind das die Gleichungen

$$\begin{aligned} x_1 + 4x_2 &= 12, \\ x_1 + \ \ x_2 &= 4, \\ 3x_1 + \ \ x_2 &= 9, \end{aligned}$$

die in (a) dargestellt sind. Zulässige Lösungen für $x = (x_1, x_2)^T$, die die entsprechenden Ungleichungen erfüllen, müssen jeweils unterhalb der Geraden liegen, also im gelb markierten Bereich.

Aus der zu maximierenden Funktion $x_1 + 2x_2$ sind in (b) für die möglichen Zielwerte 0, $\frac{20}{3}$ und 10 ebenfalls Geraden erstellt worden. Die optimale Lösung ist dabei $x = (\frac{4}{3}, \frac{8}{3})^T$ mit dem Wert $\frac{20}{3}$, denn alle Geraden mit höherem Zielwert liegen komplett außerhalb des zulässigen Bereichs.

Die optimale Lösung eines linearen Programms kann auch mathematisch berechnet werden. Entsprechende Lösungsalgorithmen werden kurz in Kapitel 1.7.2 vorgestellt, aber in dieser Arbeit nicht weiter vertieft.

1.7.2 Lösungsalgorithmen

Der erste in der Praxis verwendbare Algorithmus zur Lösung linearer Optimierungsprobleme wurde 1947 vom US-amerikanischen Mathematiker George Dantzig [28] vorgestellt.

Sein sogenannter Simplex-Algorithmus ist auch heute noch eines der meist genutzten Verfahren zur Lösung linearer Programme. Die Grundidee des Algorithmus besteht darin, von einer beliebigen Ecke des Polyeders, das durch die lineare Optimierungsaufgabe definiert wird, entlang seiner Kanten zu einer optimalen Ecke zu laufen. Dabei springt der Algorithmus jeweils zu einer benachbarten Ecke, deren Zielfunktionswert nicht kleiner als der aktuelle Wert ist. Wenn ein lokales Maximum gefunden ist, bei dem keine benachbarte Ecke einen höheren Zielfunktionswert hat, terminiert das Verfahren. Es kann bewiesen werden, dass dieses lokale Maximum auch ein globales ist, wodurch die Korrektheit folgt.

Die Anzahl Ecken des Polyeders kann allerdings exponentiell in der Anzahl der Variablen und Ungleichungen sein. Dazu gibt es Eingaben, bei denen das Simplex-Verfahren tatsächlich alle Ecken besucht und somit aus theoretischer Sicht exponentielle Laufzeit hat. In der Praxis sind solche Eingaben jedoch sehr selten und der Algorithmus terminiert meist nach wesentlich weniger Schritten. Es ist aber bis heute unklar, ob es eine Variante mit polynomieller Laufzeit für alle Instanzen gibt.

Der sowjetische Mathematiker Leonid Khachiyan [81, 82] konnte 1979 mit einem anderen Ansatz zeigen, dass lineare Optimierungsprobleme tatsächlich in polynomieller Zeit gelöst werden können. Seine vorgestellte *Ellipsoidmethode* hat allerdings eine so große Laufzeit, dass sie für praktische Anwendungen nicht geeignet ist.[35] Der Algorithmus testet, ob der Mittelpunkt eines Ellipsoids innerhalb eines speziellen Polyeders liegt oder nicht. Es kann gezeigt werden, dass das Lösen eines LPs äquivalent zum Finden eines Punktes in einem geeignet definierten Hilfspolyeder ist.

Einen weiteren Algorithmus stellte der indische Mathematiker Narendra Karmarkar [76] vor. Sein *Innere-Punkte-Verfahren* war der erste Algorithmus, der polynomielle Laufzeit garantierte und auch in der Praxis mit der Geschwindigkeit des Simplex-Algorithmus mithalten konnte.[36] Die Idee dabei ist, sich dem Optimum des LP durch das Innere des Polyeders zu nähern.

Für weitere Einzelheiten der hier aufgeführten Algorithmen sei auf die Fachliteratur verwiesen (z. B. auf [28, 23, 128]).

1.7.3 Ganzzahlige lineare Programmierung

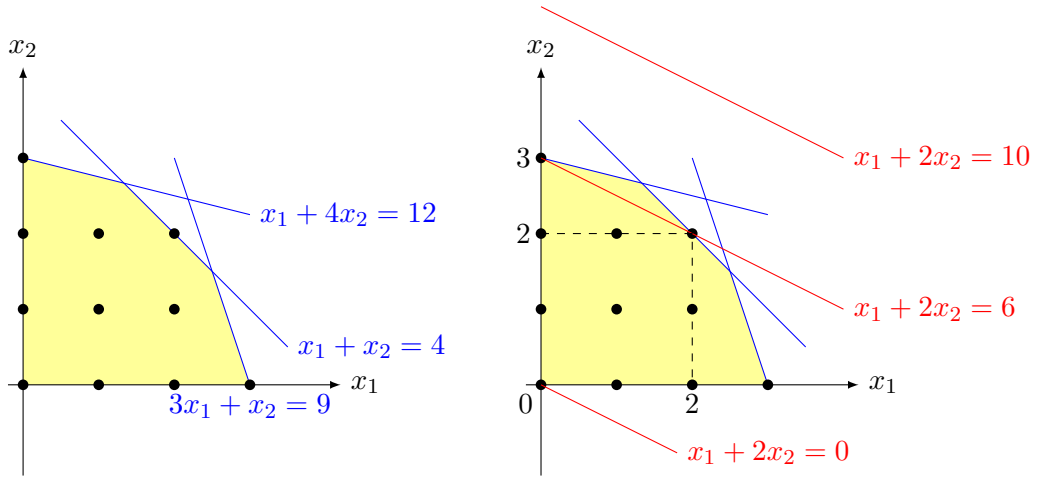
Ein Spezialfall der linearen Programmierung ist die sogenannte *diskrete lineare Programmierung* (*ILP* für engl. *integer linear programming*). Dabei soll für alle Lösungen zusätzlich $x \in \mathbb{Z}^n$ gelten, das heißt, alle Variablen x_i dürfen nur ganzzahlige Werte annehmen. Diese Form heißt daher auch *ganzzahlige lineare Programmierung* und hat die Standardform

$$\max\{c^T x \mid Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\}. \tag{1.8}$$

Ist ein Teil der Variablen ganzzahlig und der andere Teil reellwertig, spricht man auch von der *gemischt-ganzzahligen linearen Programmierung* (*MIP* für engl. *mixed-integer programming*). Durch die Beschränkung auf ganzzahlige Werte wird der Raum der Lösungen weiter eingeschränkt.

Beispiel 1.13 Der farbig markierte Bereich in Abbildung 1.9 stellt analog zu Abbildung 1.8 die Menge der zulässigen Lösungen für das bereits eingeführte lineare Programm

[35] Die Laufzeit beträgt $\mathcal{O}(n^6 L^2)$, wobei n die Dimensionen und L die Anzahl der Bits in der Eingabe sind.
[36] Die Laufzeit beträgt $\mathcal{O}(n^{3,5} L^2)$, wobei n die Dimensionen und L die Anzahl der Bits in der Eingabe sind.



(a) Ohne Zielfunktion (b) Mit Zielfunktion

Abbildung 1.9: Graphische Veranschaulichung eines ILP. Die blauen Geraden beschränken den gelb gefärbten Bereich der zulässigen Lösungen, wobei nur die angegebenen Punkte Lösungen sind. Die zu maximierende Zielfunktion ist rot dargestellt.

dar. Gelte zusätzlich die Bedingung $x \in \mathbb{Z}^2$, so sind genau elf Punkte enthalten. Für das ursprüngliche Problem der Gewinnmaximierung eines Unternehmens bedeutet die Beschränkung auf ganzzahlige Lösungen, dass Produkte nur in ganzzahliger Anzahl hergestellt werden können. Der optimalen Lösung für (x_1, x_2) entsprechen jeweils die Punkte $(0, 3)$ und $(2, 2)$. Der dazugehörige Zielwert ist 6 und damit kleiner als der maximale Wert des LP ($\frac{20}{3} \approx 6{,}67$).

Im Unterschied zur linearen Programmierung, die in polynomieller Zeit gelöst werden kann, ist bereits das Finden einer beliebigen Lösung eines ILP ein NP-schweres Problem.

Satz 1.6 *Das Problem* ILP[37] *ist NP-schwer.*

Beweis: Es genügt, 3-SAT $\leq_p$ ILP zu zeigen. Sei $F = C_1 \wedge \cdots \wedge C_m$ eine 3-SAT-Formel über den Variablen $x_1, \ldots, x_n$. Für den Lösungsvektor $x = (x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n)^T$ wird das Ungleichungssystem $Ax \geq b$ erstellt, das folgende Ungleichungen repräsentiert:

- Für jedes $i = 1, \ldots, n$ werden die vier Ungleichungen

$$\begin{aligned} x_i + \bar{x}_i &\geq 1, & x_i &\geq 0, \\ -x_i - \bar{x}_i &\geq 1, & \bar{x}_i &\geq 0 \end{aligned}$$

  erstellt.

- Für jede Klausel $C_j = l_{j_1} \vee l_{j_2} \vee l_{j_3}$ mit $l_{j_k} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ für $j = 1, \ldots, m$ und $k = 1, 2, 3$ entsteht die Ungleichung

$$l_{j_1} + l_{j_2} + l_{j_3} \geq 1.$$

[37] Der Einfachheit halber bezeichne ILP auch das Problem, eine beliebige Lösung zu einem ganzzahligen linearen Programm zu berechnen.

Die Ungleichungen für die Variablen sind für ganzzahlige $x_i, \bar{x}_i$ genau dann erfüllt, wenn eine Variable den Wert 0 und die andere den Wert 1 hat. Die Klauselungleichungen stellen sicher, dass mindestens ein Literal in jeder Klausel C_j wahr wird. Nun ist leicht zu sehen, dass jede Lösung x von $Ax \geq b$ einer erfüllenden Belegung von F entspricht und umgekehrt. □

Es ist nicht leicht zu sehen, dass das Problem ILP in NP entscheidbar ist. Ein nichtdeterministischer Algorithmus kann zwar eine Lösung raten, aber a priori ist nicht klar, ob eine solche Lösung polynomiell in der Länge der Eingabe (A, b) ist. Mit Methoden der linearen Algebra lässt sich jedoch zeigen, dass jede lösbare ILP-Instanz (A, b) auch eine Lösung x hat, deren Kodierung polynomiell in der Länge von (A, b) ist. Mit dem obigen Satz folgt, dass das Lösen eines ILP NP-vollständig ist.

Ganzzahlige lineare Optimierungsprobleme können mithilfe von Schnittebenenverfahren oder Branch-and-Bound exakt gelöst werden. Beim Erstgenannten wird ein analoges lineares Programm gelöst, bei dem auf die Ganzzahligkeitsbedingungen verzichtet wird.[38] Die dabei entstehenden Ecken des Polyeders, die nicht ganzzahlig sind, werden durch das Hinzufügen weiterer geeigneter Ungleichungen „weggeschnitten“. Die Menge der zulässigen Lösungen wird dadurch so lange verkleinert, bis sämtliche Ecken ganzzahlig sind. Beim Branch-and-Bound wird die Menge zulässiger Lösungen möglichst effizient durchsucht, wobei im ungünstigsten Fall alle in Frage kommenden Werte getestet werden müssen. Da Algorithmen für ILP exponentielle Laufzeit haben, existiert eine Vielzahl von Heuristiken. Oftmals werden auch mehrere Lösungsverfahren miteinander kombiniert.

Eine weitere Spezialisierung dieser Art von Problemen ist die *binäre (diskrete) lineare Programmierung* (0/1-ILP für engl. *0-1 integer linear programming*), bei der $x \in \{0, 1\}^n$ gilt. Die Variablen x_i können also nur die beiden Werte 0 oder 1 annehmen. Auch 0/1-ILP ist nach dem Beweis von Satz 1.6 NP-schwer.

$$\max\{c^T x \mid Ax \leq b, x \in \{0, 1\}^n\} \tag{1.9}$$

Viele Probleme insbesondere aus der Graphentheorie lassen sich als binäre lineare Programme darstellen. Ein Beispiel hierfür ist das bekannte Traveling-Salesman-Problem (TSP), bei dem eine kürzeste Route gesucht ist, die eine gegebene Anzahl von Punkten verbindet. In dieser Arbeit wird vor allem das Matchingproblem untersucht, auf das im nächsten Abschnitt eingegangen wird.

1.7.4 Matchingproblem als ganzzahliges lineares Programm

Viele kombinatorische Optimierungsprobleme können als lineares Programm formuliert werden. Um dies zu erreichen, werden zulässige Lösungen als Vektoren im $\mathbb{R}^n$ für ein passendes n kodiert. Ein Beispiel dafür ist das bereits in Kapitel 1.6.3 eingeführte Matchingproblem, das heißt die Suche nach größten Matchings in Graphen.

[38] Das ist die sogenannte *LP-Relaxation*.

Sei $G = (V, E)$ ein einfacher Graph, zu dem ein größtes Matching $M \subseteq E$ bestimmt werden soll. Für jede Kante $e \in E$ wird eine Variable $x_e \in \{0, 1\}$ erzeugt. Entsprechend der Zugehörigkeit zu M soll dabei gelten:

$$x_e = \begin{cases} 1, & \text{falls } e \in M \\ 0, & \text{sonst.} \end{cases} \tag{1.10}$$

Alle Variablen können in einem Vektor $x \in \{0, 1\}^E$ zusammengefasst werden. x heißt der *Inzidenz-* oder auch *charakteristische Vektor* von M bezüglich E.

Die Matchingbedingung besagt, dass jeder Knoten in höchstens einer Matchingkante enthalten ist. Folglich kann für jeden Knoten v maximal eine Kante x_e mit $v \in e$ den Wert 1 haben. Definiere $\delta(v) := \{e \in E \mid v \in e\}$, dann ist die entsprechende Nebenbedingung des ILP für jeden Knoten v

$$\sum_{e \in \delta(v)} x_e \leq 1. \tag{1.11}$$

Für eine Formulierung in Standardform werden eine Matrix A und ein Vektor b benötigt. Sei $A \in \{0, 1\}^{V \times E}$ die Inzidenzmatrix von G mit

$$A_{v,e} = \begin{cases} 1, & \text{falls } v \in e \\ 0, & \text{sonst.} \end{cases}$$

Dann lautet die Nebenbedingung des linearen Programms

$$Ax \leq \mathbb{1},$$

wobei $\mathbb{1}$ ein Vektor der Dimension $|V|$ ist, der in jeder Komponente eine 1 hat. Da der Wert der Variablen für jede Matchingkante 1 ist, muss die Summe aller Variablen in einem größten Matching maximal sein. Das Optimierungsproblem des Bestimmens größter Matchings kann somit als binäres lineares Programm

$$\max\{\mathbb{1}^T x \mid Ax \leq \mathbb{1}, x \in \{0, 1\}^E\} \tag{1.12}$$

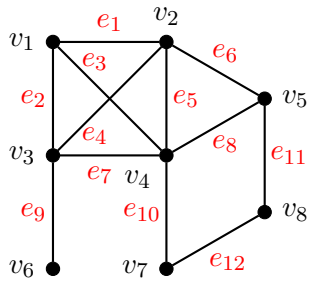
zusammengefasst werden. Wird auf die Verwendung der Matrizen und Vektoren verzichtet, ist

$$\max\{\sum_{e \in E} x_e \mid \sum_{e \in \delta(v)} x_e \leq 1 \text{ für alle } v \in V, x_e \in \{0, 1\} \text{ für alle } e \in E\}$$

eine äquivalente Formulierung des Problems.

Beispiel 1.14 In Abbildung 1.10 ist in (a) ein Graph dargestellt, zu dem ein größtes Matching berechnet werden soll. Die Nebenbedingungen eines dazugehörigen ganzzahligen linearen Programms sind für alle Knoten v_i in (b) angegeben. Die äquivalente Formulierung in Matrixschreibweise ist in Beispiel A.1 im Anhang auf Seite 269 dargestellt. Die Zielfunktion des ILP ist $\max \sum_{i=1}^{12} x_{e_i}$ und eine mögliche optimale Lösung

$$\begin{aligned} x_{e_3} = x_{e_6} = x_{e_9} \ &= x_{e_{12}} = 1, \\ x_{e_1} = x_{e_2} = x_{e_4} = x_{e_5} = x_{e_7} = x_{e_8} = x_{e_{10}} &= x_{e_{11}} = 0. \end{aligned}$$



(a) Graph G

$$
\begin{aligned}
v_1 &: \quad x_{e_1} + x_{e_2} + x_{e_3} \leq 1 \\
v_2 &: \quad x_{e_1} + x_{e_4} + x_{e_5} + x_{e_6} \leq 1 \\
v_3 &: \quad x_{e_2} + x_{e_4} + x_{e_7} + x_{e_9} \leq 1 \\
v_4 &: \quad x_{e_3} + x_{e_5} + x_{e_7} + x_{e_8} + x_{e_{10}} \leq 1 \\
v_5 &: \quad x_{e_6} + x_{e_8} + x_{e_{11}} \leq 1 \\
v_6 &: \quad x_{e_9} \leq 1 \\
v_7 &: \quad x_{e_{10}} + x_{e_{12}} \leq 1 \\
v_8 &: \quad x_{e_{11}} + x_{e_{12}} \leq 1
\end{aligned}
$$

(b) Nebenbedingungen des ILP

Abbildung 1.10: Matchingproblem als ganzzahliges lineares Programm: Für den Graphen G in (a) sind in (b) die Nebenbedingungen des ILP angegeben, wobei zusätzlich $x_{e_i} \in \{0,1\}$ für alle $1 \leq i \leq 12$ gilt.

Die Lösung hat den Wert 4 und entspricht Matching M_4 aus Abbildung 1.4d auf Seite 15.

Perfektes Matching

Ein ähnliches Problem wie die Suche nach größten Matchings ist das Bestimmen von perfekten Matchings. Häufig werden dabei Kantengewichte c_e beachtet und ein maximal (oder minimal) gewichtetes Matching gesucht. Sei $c_f \colon E \to \mathbb{R}$ eine Gewichtsfunktion, die jeder Kante $e \in E$ ein Gewicht c_e zuordnet. Das Gewicht eines Matchings M ist dann $c(M) \coloneqq \sum_{e \in M} c_e$. Da in perfekten Matchings jeder Knoten von genau einer Kante überdeckt wird, ist die Nebenbedingung des ILP für jeden Knoten v

$$\sum_{e \in \delta(v)} x_e = 1.$$

Die Formulierung in Matrixform lautet

$$\max\{c^T x \mid Ax = \mathbb{1}, x \in \{0,1\}^E\} \tag{1.13}$$

für ein $c \in \mathbb{R}^E$. Wird auf Kantengewichte verzichtet, ist das Problem

$$\max\{\mathbb{1}^T x \mid Ax = \mathbb{1}, x \in \{0,1\}^E\} \tag{1.14}$$

zu lösen. In dieser Arbeit werden nur Graphen mit ungewichteten Kanten betrachtet. Für diese gilt, dass jede zulässige Lösung bereits optimal ist.

Beispiel 1.15 Um ein perfektes Matching im Graphen G aus Abbildung 1.10a zu berechnen, werden in den Nebenbedingungen alle Ungleich- durch Gleichheitszeichen ersetzt. Das im Beispiel 1.14 angegebene größte Matching ist gleichzeitig auch perfekt und erfüllt alle Gleichungen.

1.7.5 Relaxation

Eine Möglichkeit, um ganzzahlige lineare Programme effizient zu lösen, ist, einige Nebenbedingungen wegzulassen oder zu lockern. Dadurch können zwar Lösungen entstehen, die keine Lösungen des ursprünglichen Problems sind. In einigen Fällen kann aber auch gezeigt werden, dass durch diese sogenannte *Relaxation* keine weitere Lösung hinzukommt.

Definition 1.15 (Relaxation) *Seien $X, X' \subseteq \mathbb{R}^n$ zwei Mengen, $f, f' \colon \mathbb{R}^n \to \mathbb{R}$ zwei Funktionen und P, P' zwei Optimierungsprobleme mit*

$$\begin{aligned} P\colon &\quad \max\{f(x) \mid x \in X\} \\ P'\colon &\quad \max\{f'(x) \mid x \in X'\} \end{aligned}$$

P' heißt Relaxation *von P, wenn*

1. *$X \subseteq X'$ und*
2. *$f(x) \leq f'(x)$ für alle $x \in X$.*

Offensichtlich ist das Optimum $v(P')$ der Relaxation mindestens so groß wie der optimale Wert $v(P)$ des ursprünglichen Problems. Eine Relaxation heißt *exakt*, wenn $v(P') = v(P)$ gilt. Diese Fälle sind bei vielen Problemen besonders interessant.

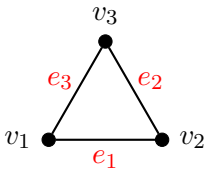
Für ein ganzzahliges lineares Programm entsteht die *LP-Relaxation* durch das Weglassen der Ganzzahligkeitsbedingung, das heißt aus

$$\max\{c^T x \mid Ax \leq b, x \geq 0, x \in \mathbb{Z}^n\}$$

wird

$$\max\{c^T x \mid Ax \leq b, x \geq 0\}.$$

Dementsprechend wird für ein binäres Programm $x \in [0,1]^n$ anstelle von $x \in \{0,1\}^n$ angenommen, das heißt, es sind reelle Zahlen von 0 bis 1 zugelassen. Das entstehende Problem ist nun ein lineares Programm, welches effizient gelöst werden kann.



(a) Graph K_3

$$\begin{aligned} v_1\colon &\quad x_{e_1} + x_{e_3} = 1 \\ v_2\colon &\quad x_{e_1} + x_{e_2} = 1 \\ v_3\colon &\quad x_{e_2} + x_{e_3} = 1 \end{aligned}$$

(b) Nebenbedingungen des LP

$$x_{e_1} = x_{e_2} = x_{e_3} = 0{,}5$$

$$\max\{\sum_{i=1}^{3} x_{e_i}\} = 1{,}5$$

(c) Lösung der LP-Relaxation

Abbildung 1.11: LP-Relaxation des perfekten Matchingproblems

Offensichtlich ist jede zulässige Lösung des ganzzahligen linearen Programms auch eine Lösung des relaxierten Problems. Umgekehrt trifft das im Allgemeinen nicht zu. Beispielsweise besitzt der Graph K_3 aus Abbildung 1.11 kein perfektes Matching, aber $x_{e_1} = x_{e_2} = x_{e_3} = 0{,}5$ ist eine optimale Lösung für das entsprechende relaxierte LP. Es gibt aber Fälle, in denen sogar das Optimum der Relaxation ein Optimum des ursprünglichen Problems darstellt. Dazu betrachte man das Polyeder $P \coloneqq \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$. Besitzt P nur

ganzzahlige Ecken aus $\{0,1\}^m$,[39] ist eine Ecke sowohl eine optimale Lösung des relaxierten Problems als auch des ganzzahligen Problems. In diesem Fall kann eine Lösung mithilfe eines polynomiellen Algorithmus für LP gefunden werden (vgl. Kapitel 1.7.2).

Von besonderer Bedeutung sind Matrizen $A \in \mathbb{Z}^{m\times n}$, für die P für jedes $b \in \mathbb{Z}^m$ nur ganzzahlige Ecken hat. Solche Matrizen werden *total unimodular* oder auch *vollständig unimodular* genannt.

Definition 1.16 ((total) unimodular) *Eine Matrix A heißt* unimodular*, falls sie ganzzahlig und ihre Determinante gleich 1 oder -1 ist. A heißt* total unimodular*, falls die Determinante jeder quadratischen Untermatrix von A den Wert 0, 1 oder -1 annimmt.*

Offensichtlich ist ein notwendiges Kriterium für eine total unimodulare Matrix A, dass jeder Eintrag 0, 1 oder -1 ist, das heißt $A \in \{0,1,-1\}^{m\times n}$. Der entscheidende Zusammenhang zwischen total unimodularen Matrizen und den entstehenden Polyedern liefert der Satz von Hoffmann und Kruskal.

Satz 1.7 (Hoffman und Kruskal [66]) *$A \in \mathbb{Z}^{m\times n}$ ist genau dann total unimodular, wenn für jedes $b \in \mathbb{Z}^m$ alle Ecken des Polyeders $P := \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ ganzzahlig sind.*

Damit können ganzzahlige lineare Programme, bei denen die Matrix A total unimodular ist, mithilfe von relaxierten LPs gelöst werden. Ein Beispiel für diesen Fall ist das Matchingproblem in bipartiten Graphen. Der folgende Satz liefert dafür die Grundlage.

Satz 1.8 (Heller und Tompkins [64]) *Sei $A \in \{0,1,-1\}^{m\times n}$ eine Matrix mit höchstens zwei Einträgen pro Spalte. Dann ist A genau dann total unimodular, wenn die Zeilen von A in zwei Klassen eingeteilt werden können, sodass folgende Bedingungen erfüllt sind:*

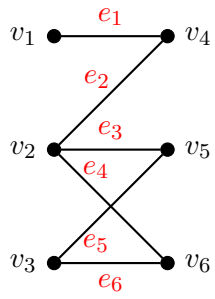
1. *Zeilen mit einem Eintrag 1 und -1 in derselben Spalte sind in derselben Klasse und*
2. *Zeilen mit zwei vorzeichengleichen Einträgen in derselben Spalte sind in unterschiedlichen Klassen.*

Sei $G = (V, E)$ ein bipartiter Graph. Nach Definition 1.7 können die Knoten aus V in zwei Klassen aufgeteilt werden, sodass Kanten nur zwischen den Klassen verlaufen. In der Inzidenzmatrix von G liegen daher alle Einträge 1 in derselben Spalte in unterschiedlichen Klassen. Da es keine Einträge -1 gibt, ist die Inzidenzmatrix eines Graphen nach Satz 1.8 genau dann total unimodular, wenn G bipartit ist.

Korollar 1.9 *Die Inzidenzmatrix eines Graphen G ist genau dann total unimodular, wenn G bipartit ist.*

Beispiel 1.16 In Abbildung 1.12 ist in (a) ein bipartiter Graph G und in (b) die dazugehörige Inzidenzmatrix A dargestellt. Werden die Zeilen von A zwischen v_3 und v_4 in zwei Klassen aufgeteilt, haben alle Spalten eine 1 in der oberen und in der unteren Klasse. Die entstehende Klasseneinteilung entspricht der Bipartition. Nach Satz 1.8 ist A somit total unimodular.

[39] Ein solches Polyeder heißt *ganzzahlig*.



(a) Graph G

$$
\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array}
\begin{array}{c}
\begin{array}{cccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \end{array} \\
\left(\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1
\end{array} \right)
\end{array}
$$

(b) Inzidenzmatrix A

Abbildung 1.12: Bipartite Graphen haben total unimodulare Inzidenzmatrizen

Zusammenfassend gilt, dass größte Matchings in bipartiten Graphen in polynomieller Zeit gefunden werden können. Dazu wird aus dem ganzzahligen linearen Programm, welches größte Matchings beschreibt, die LP-Relaxation gebildet. Da der Graph bipartit ist, ist seine Inzidenzmatrix total unimodular und nach dem Satz von Hoffman und Kruskal sind alle Lösungen des relaxierten LPs, die Ecken des dazugehörigen Polyeders beschreiben, ganzzahlig. Die Relaxation ist somit exakt und eine optimale Lösung kann mithilfe polynomieller Algorithmen für lineare Programme berechnet werden. Dieses Verfahren ist eine Alternative zu den in Kapitel 1.6.3 vorgestellten Matchingalgorithmen.

2 Schutz der Privatsphäre bei der Veröffentlichung von Daten

Eine Herausforderung beim Veröffentlichen personenbezogener Daten ist, die Privatsphäre der entsprechenden Individuen zu schützen. Dieses Kapitel gibt dazu einen Überblick über den aktuellen Stand der Forschung. Zunächst werden einige gesetzliche Bestimmungen vorgestellt (Kapitel 2.1). Danach wird in das Gebiet des Veröffentlichens sogenannter Mikrodaten eingeführt (Kapitel 2.2) und es werden mehrere Modelle für den Schutz der Privatsphäre erläutert (Kapitel 2.3). Nach einer kurzen Zusammenfassung einiger Algorithmen (Kapitel 2.4) wird das Prinzip des dynamischen Veröffentlichens besprochen (Kapitel 2.5). Am Ende dieses Kapitels werden Methoden im Bereich der Anfrageauditierung präsentiert (Kapitel 2.6).

2.1 Privatsphäre und gesetzliche Bestimmungen

Der Begriff Privatsphäre gehört zu den am meisten diskutierten Termini der letzten Jahre. Immer wieder berichten Medien von Fällen, in denen die Privatsphäre von Personen verletzt wurde [136, 131, 109, 132]. Eine Ursache dafür liegt darin, dass die Privatsphäre nicht immer klar definiert ist und einen subjektiven Charakter besitzt. Hinzu kommt, dass es in verschiedenen Staaten unterschiedliche Regelungen gibt.

Das Bundesverfassungsgericht bezeichnet mit *Privatsphäre* den nichtöffentlichen Bereich, in dem ein Mensch unbehelligt von äußeren Einflüssen sein Recht auf freie Entfaltung der Persönlichkeit wahrnimmt [141]. Das Recht auf Privatsphäre gilt als Menschenrecht und ist in allen modernen Demokratien verankert. Der *Schutz der Privatsphäre* ist im deutschen Grundgesetz (GG) aus dem allgemeinen Persönlichkeitsrecht (Art. 2 Abs. 1 in Verbindung mit Art. 1 Abs. 1 GG [13]) abzuleiten. So heißt es in Art. 2 Abs. 1 GG: „Jeder hat das Recht auf die freie Entfaltung seiner Persönlichkeit, soweit er nicht die Rechte anderer verletzt und nicht gegen die verfassungsmäßige Ordnung oder das Sittengesetz verstößt.“ Art. 1 Abs. 1 GG wiederum lautet: „Die Würde des Menschen ist unantastbar. Sie zu achten und zu schützen ist Verpflichtung aller staatlichen Gewalt.“ Dadurch soll Menschen ein persönlicher Bereich zugesichert werden, in welchem sie sich frei und ungezwungen verhalten können, ohne dass sie befürchten müssen, von Dritten beobachtet oder abgehört zu werden. Der Schutzbereich wird durch die Unverletzlichkeit der Wohnung (Art. 13 GG [13]) und das Post- und Fernmeldegeheimnis (Art. 10 GG [13]) konkretisiert.[1]

Einen dritten Bereich stellt der Schutz personenbezogener Daten dar, der sich vom Recht auf informationelle Selbstbestimmung ableitet [140]. Die Rechtsprechung des Bundesverfassungsgerichts sieht darin das Recht des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner personenbezogenen Daten zu bestimmen. Es handelt

[1] Eine Zusammenfassung der wichtigsten durch die deutsche Rechtsprechung definierten Begriffe zum Thema Schutz der Privatsphäre gibt Löser [97].

sich dabei um ein Datenschutzgrundrecht, das im Grundgesetz für die Bundesrepublik Deutschland nicht ausdrücklich erwähnt wird. Daher wurde in die meisten Landesverfassungen eine Datenschutzregelung aufgenommen.

Auf Bundesebene regelt das Bundesdatenschutzgesetz (BDSG) [12] den Datenschutz für die Bundesbehörden und den privaten Bereich. Daneben regeln die Datenschutzgesetze der Länder den Datenschutz in Landes- und Kommunalbehörden. In § 1 Abs. 1 BDSG heißt es zum Beispiel: „Zweck dieses Gesetzes ist es, den Einzelnen davor zu schützen, dass er durch den Umgang mit seinen personenbezogenen Daten in seinem Persönlichkeitsrecht beeinträchtigt wird." Schutzbedürftig sind demnach spezielle Daten, die als personenbezogen bezeichnet werden.

Im Allgemeinen sind Daten *personenbezogen*, wenn sie eindeutig einer bestimmten natürlichen Person zugeordnet sind oder diese Zuordnung zumindest mittelbar erfolgen kann. Im zweiten Fall spricht man auch von personenbeziehbaren Daten [154]. Das deutsche Bundesrecht definiert in § 3 Abs. 1 BDSG [12] personenbezogene Daten als „Einzelangaben über persönliche oder sachliche Verhältnisse einer bestimmten oder bestimmbaren natürlichen Person (Betroffener)". Dazu gehören insbesondere nach § 3 Abs. 9 BDSG „Angaben über die rassische und ethnische Herkunft, politische Meinungen, religiöse oder philosophische Überzeugungen, Gewerkschaftszugehörigkeit, Gesundheit oder Sexualleben".

Auch die Europäische Union definiert für ihre Mitgliedstaaten in Art. 2 lit. a der Richtlinie 95/46/EG (Datenschutzrichtlinie) [46] den Begriff der personenbezogenen Daten als „alle Informationen über eine bestimmte oder bestimmbare natürliche Person (‚betroffene Person'); als bestimmbar wird eine Person angesehen, die direkt oder indirekt identifiziert werden kann, insbesondere durch Zuordnung zu einer Kennnummer oder zu einem oder mehreren spezifischen Elementen, die Ausdruck ihrer physischen, physiologischen, psychischen, wirtschaftlichen, kulturellen oder sozialen Identität sind".[2]

In den Vereinigten Staaten ist der Datenschutz kaum durch Gesetze oder andere Vorschriften rechtlich verankert. Für einzelne Teilbereiche gibt es zwar Regelungen, wie zum Beispiel für Internetwebseiten den Children's Online Privacy Protection Act (COPPA) [49] oder für Krankenversicherungen den Health Insurance Portability and Accountability Act (HIPAA) [48]. Umfassende Gesetze für den Umgang mit personenbezogenen Daten existieren aber nicht [151].

In der vorliegenden Arbeit ist der Schutz von Daten von zentraler Bedeutung. Im wissenschaftlichen Umfeld wird in diesem Zusammenhang meist von *sensiblen* (engl. *sensitive*) anstelle von personenbezogenen Daten gesprochen. Im Allgemeinen sind Daten sensibel, wenn deren Verlust, Missbrauch, Änderung oder ein unbefugter Zugriff darauf sich nachteilig auf die Privatsphäre eines Individuums auswirken können.[3] Eine weitere viel zitierte Definition von Privatsphäre stammt von Alan Westin, einem emeritierten Professor für öffentliches Recht an der Columbia-Universität: „Privacy is the claim of individuals to determine for themselves when, how and to what extent information about them is communicated to others" [150]. Diese Definition offenbart die subjektive Wahrnehmung des Begriffs der Privatsphäre. Gleichzeitig assoziiert sie Privatsphäre mit Informationen, welche wiederum aus Daten gewonnen werden können. Der Schutz von Daten insbesondere

[2] In der englischen Version des entsprechenden Artikels wird die „betroffene Person" mit „*data subject*", also Datensubjekt, bezeichnet.

[3] Vgl. die Definition von sensiblen Informationen in [3].

bei deren Herausgabe dient somit der Kontrolle von Informationen, die andere über ein Individuum erhalten.

2.2 Veröffentlichung sensibler Daten

Die Mehrzahl der Mechanismen zum Schutz sensibler Daten findet ihren Ursprung in Arbeiten zu statistischen Veröffentlichungen [54]. Diese Konzepte vereinen sich unter dem Begriff des *Privacy-Preserving Data Publishing (PPDP)*. In diesem Zusammenhang wird vornehmlich über die Verarbeitung sogenannter *Mikrodaten* [75] gesprochen.

Mikrodaten sind die Originaldaten[4], die einen hohen Detaillierungsgrad besitzen und aus Datenschutzgründen nicht öffentlich zugänglich sind. Sie entstehen oftmals durch statistische Erhebungen und sind in der Regel personenbezogen, das heißt, sie sind eindeutig einer bestimmten natürlichen Person (*Datensubjekt* oder *Individuum*) zugeordnet. Im Rahmen dieser Arbeit wird stets davon ausgegangen, dass Daten in relationaler Form, das heißt insbesondere als Tabellen, vorliegen. Tupel entsprechen Zeilen der Tabelle und Spalten den Attributen. Weiterhin wird davon ausgegangen, dass jedes Tupel zu genau einem Individuum gehört und kein Individuum mehrfach in der Tabelle vorkommt (vgl. [135]).

Name	Alter	PLZ	Geschlecht	Krankheit
Alina	20	10104	w	Asthma
Bill	21	10104	m	Bronchitis
Clive	21	10155	m	Diabetes
Doris	23	21410	w	Grippe
Emily	24	21410	w	Diabetes
Frank	24	21410	m	Grippe
Gill	24	21421	w	Bronchitis
Harry	26	29011	m	Diabetes
Isabel	27	29025	w	Diabetes

Tabelle 2.1: Mikrodaten

Tabelle 2.1 stellt medizinische Mikrodaten von neun Personen dar, die jeweils durch ihren Namen identifiziert werden sollen. Für jede Person sind das Alter, die Postleitzahl (PLZ), das Geschlecht und eine Krankheit aufgelistet. Die Daten sind die Grundlage für alle Beispiele in diesem Kapitel.

Statistische Veröffentlichungen enthalten meist Daten in aufgearbeiteter Form. Beispielsweise werden Daten aggregiert oder unter Beibehaltung der statistischen Eigenschaften derart verändert, dass keine Rückschlüsse auf Individuen möglich sind. Die entstehenden Daten werden *Makrodaten* [24, 75] genannt und haben einen wesentlich geringeren Detaillierungsgrad als die Originaldaten. Für statistische Analysen oder Forschungszwecke sind jedoch Mikrodaten von erheblich größerer Relevanz als Makrodaten. Aus diesem Grund beschäftigen sich die Techniken des PPDP hauptsächlich mit der Veröffentlichung von Mikrodaten.

Soll Tabelle 2.1 veröffentlicht werden, muss die Privatsphäre der neun Personen geschützt werden. Das bedeutet vereinfacht, dass niemand aus den veröffentlichten Daten

[4]Im Rahmen dieser Arbeit werden die Begriffe Originaldaten und Mikrodaten synonym verwendet.

schlussfolgern kann, welche Personen welche Krankheiten haben. Konkret soll eine *Re-Identifikation* vermieden werden. Diese entsteht, wenn Tupel in der veröffentlichten Tabelle eindeutig Individuen in den Mikrodaten zugeordnet werden können [24]. Der Prozess, der eine Re-Identifikation unmöglich macht, heißt *Anonymisierung* oder *De-Identifikation*. Das Bundesdatenschutzgesetz definiert dazu in § 3 Abs. 6 BDSG [12]: „*Anonymisieren* ist das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können."[5]

Ein zweites Ziel des PPDP ist, Schlüsseleigenschaften der Originaldaten zu erhalten. Darunter versteht man, dass eine Analyse der Mikrodaten und der veröffentlichten Daten ähnliche Ergebnisse produzieren soll. In der Literatur wird dazu häufig ein Nutzen (engl. *utility*) der veröffentlichten Daten definiert und gefordert, dass dieser maximal sei. Eine andere Möglichkeit besteht darin, die Kosten der Anonymisierung zu quantifizieren und zu minimieren.

Zusammenfassend ist das Ziel des PPDP, aus einer gegebenen Mikrodatentabelle T eine Tabelle T^* zu erstellen, die die folgenden beiden Eigenschaften erfüllt [24]:

1. Das Risiko, dass ein Angreifer T^* benutzen kann, um vertrauliche Informationen zu erhalten oder ein Individuum zu identifizieren, ist gering.

2. Statistische Analysen von T und T^* produzieren ähnliche Ergebnisse.

2.2.1 Datenverknüpfungsproblem

Ein erster Schritt zur Erstellung veröffentlichbarer Daten ist das Entfernen sämtlicher identifizierender Attribute aus den Originaldaten. Tabelle 2.2 demonstriert diesen Ansatz, wobei die Namen der Individuen zur besseren Übersicht neben den entsprechenden Tupeln notiert sind. Auf den ersten Blick scheint die Tabelle die festgelegten Kriterien zu erfüllen. In einer häufig zitierten Studie allerdings zeigte Sweeney [133], dass dieser Schritt nicht ausreicht, um die Privatsphäre ausreichend zu schützen, weil die Kombination mehrerer nicht-identifizierender Attribute einer Person einzigartig sein kann. Sind diese Merkmale ebenfalls in einer anderen Datenquelle vorhanden, welche nicht anonymisiert wurde, kann die Verknüpfung beider Datenquellen zur Re-Identifikation führen. Im konkreten Fall hatte Sweeney Zugang zu einer öffentlich verfügbaren Wählerliste, in der Personen unter anderem mit ihrem Geburtsdatum, ihrer Postleitzahl und ihrem Geschlecht aufgelistet waren. Sie verknüpfte diese Liste mit medizinischen Daten, bei denen nur identifizierende Attribute fehlten, und konnte Krankheiten und Behandlungsmethoden von etlichen Bürgern ableiten. Laut ihrer Studie sind 87 % der amerikanischen Bevölkerung allein durch die drei Merkmale Geburtsdatum, 5-stellige Postleitzahl und Geschlecht eindeutig gekennzeichnet.

Wird das Attribut `Geburtsdatum` vereinfacht mit `Alter` gleichgesetzt, folgt für Tabelle 2.2 dasselbe Problem, denn die Kombination von `Alter`, `PLZ` und `Geschlecht` ist für jedes Tupel eindeutig. Ein Angreifer, der neben Tabelle 2.2 Zugang zu nicht-anonymisierten Daten hat, die Alter, Postleitzahl und Geschlecht für die gegebenen Individuen enthalten, kann beide Datensätze miteinander verknüpfen und Individuen identifizieren. Dieser Vorgang wird auch *Linken* (engl. *linking*) von Daten genannt.

[5] Das BDSG fordert speziell im Bereich der Markt- und Meinungsforschung (§ 30, § 30a) sowie der wissenschaftlichen Forschung (§ 40) ein Anonymisieren personenbezogener Daten.

	Alter	PLZ	Geschlecht	Krankheit
Alina	20	10104	w	Asthma
Bill	21	10104	m	Bronchitis
Clive	21	10155	m	Diabetes
Doris	23	21410	w	Grippe
Emily	24	21410	w	Diabetes
Frank	24	21410	m	Grippe
Gill	24	21421	w	Bronchitis
Harry	26	29011	m	Diabetes
Isabel	27	29025	w	Diabetes

Tabelle 2.2: Daten ohne identifizierende Attribute

Nach diesen Erkenntnissen stellt sich die Frage, mit welchen anderen Datensätzen veröffentlichte Daten verknüpft werden können. Eine einfache Lösung dieses sogenannten *Datenverknüpfungsproblems* [75] besteht in weitgehenden Beschränkungen von Veröffentlichungen. Im extremen Fall könnte zum Beispiel nur die Spalte `Krankheit` aus Tabelle 2.1 herausgegeben werden. Als Nachteil erweist sich dann jedoch, dass die Daten auf diese Weise wertvolle Informationen für Forschung und statistische Zwecke verlieren. Insbesondere gehen alle Zusammenhänge zwischen den Attributen `Alter`, `PLZ` und `Geschlecht` zum Attribut `Krankheit` verloren. Das Veröffentlichen von Daten stellt daher eine große Herausforderung an die Datenbesitzer dar, denn einerseits soll die Privatsphäre der Individuen geschützt werden, andererseits sollen die Daten nutzbar bleiben.

2.2.2 Kategorisierung der Attribute

Um personenbezogene Daten zu anonymisieren, reicht es nicht aus, nur identifizierende Attribute zu entfernen. Vielmehr müssen die Daten derart verändert werden, dass sie nicht mehr eindeutig einer Person zugeordnet werden können. Dazu werden die Attribute der Mikrodaten in drei Kategorien unterteilt.

Ein *Identifikator* (*ID*) ist eine Menge von Attributen, welche jede Person eindeutig identifiziert, wie es beispielsweise bei Ausweis- oder Sozialversicherungsnummern der Fall ist. In abgegrenzten Datensammlungen können auch vollständige Namen, Konto- oder Immatrikulationsnummern eindeutig einer Person zugeordnet werden. In veröffentlichten Tabellen müssen Identifikatoren entfernt werden, denn durch sie können Daten direkt mit einzelnen Individuen in Verbindung gebracht werden.

Ein *Quasi-Identifikator* (*QI*) ist eine Menge von Attributen, welche einzeln nicht identifizierend sind, jedoch in Kombination eine Verknüpfung von Daten ermöglichen. Beispiele dafür sind Alter, Geschlecht, Geburtsdatum oder Postleitzahl. Attributwerte von Quasi-Identifikatoren werden in der veröffentlichten Tabelle meist leicht verändert, damit Rückschlüsse auf Individuen nicht mehr eindeutig möglich sind.

Ein *sensibles Attribut*[6] (*SA*) enthält Werte, die nicht mit Individuen in Verbindung gebracht werden sollen. Vorstellbar sind Beispiele wie Krankheit, Gehalt oder politische Haltung. Werte sensibler Attribute (kurz *sensible Werte* oder *SA-Werte*) werden in der Regel direkt veröffentlicht.

[6] Bei Karjoth [75] wird dieses Merkmal *sensitives Attribut* genannt.

Im Rahmen der vorliegenden Arbeit wird davon ausgegangen, dass jedes Attribut in eine der genannten Kategorien fällt. Es wird ebenfalls vorausgesetzt, dass diese Kategorien disjunkt sind, das heißt insbesondere sensible Attribute nicht gleichzeitig quasi-identifizierend sind.[7]

Beispiel 2.1 In Tabelle 2.1 sind Mikrodaten von neun verschiedenen Personen aufgelistet. `Name` sei hier der Identifikator, `Krankheit` das sensible Attribut und der Quasi-Identifikator bestehe aus `Alter`, `PLZ` und `Geschlecht`.

Der Quasi-Identifikator hängt von externen zur Verfügung stehenden Daten ab. Nicht jeder Angreifer hat Zugang zu denselben Daten und Attribute verschiedener externer Tabellen können sich überlappen. Dadurch kann es für eine Tabelle mehrere Quasi-Identifikatoren geben. Damit der Datenbesitzer nicht wissen muss, auf welche externen Quellen Angreifer zugreifen können, definiert er der Einfachheit halber nur einen Quasi-Identifikator, welcher aus allen Attributen besteht, die weder sensibel noch identifizierend sind. Hierin liegt auch der Grund, warum auf eine vierte Kategorie der „sonstigen Attribute“ verzichtet wird.

2.2.3 Anonymisierungsoperationen

Um Mikrodaten so zu verändern, dass eine Re-Identifikation unmöglich ist, gibt es in der Literatur mehrere Ansätze. Es können Werte einzelner Attribute ersetzt oder Verbindungen zwischen mehreren Attributen aufgebrochen werden. Im Folgenden werden die wichtigsten dieser sogenannten *Anonymisierungsoperationen* vorgestellt.

Generalisierung und Unterdrückung

Die beiden gängigsten Methoden, um Daten für das Veröffentlichen zu anonymisieren, sind *Generalisierung* und *Unterdrückung* [124, 134]. Unter Generalisierung wird das Ersetzen eines Wertes durch einen allgemeineren Begriff verstanden. Bei Unterdrückung hingegen wird ein Attributwert vollständig entfernt und durch ein Platzhaltersymbol ersetzt. Unterdrückung kann als extreme Form der Generalisierung betrachtet werden, bei der jegliche Information verloren gegangen ist. Eine verwandte Form ist die *Tupelunterdrückung*, bei der ganze Tupel unveröffentlicht bleiben.

Tabelle 2.3 zeigt eine Möglichkeit, um die Mikrodaten aus Tabelle 2.1 zu anonymisieren. Die Namen der einzelnen Individuen sind erneut der Übersicht halber aufgelistet, ohne jedoch veröffentlicht zu werden. Während die sensiblen Werte unverändert übernommen wurden, sind die Werte aller quasi-identifizierenden Attribute ersetzt worden. Anstelle genauer Altersangaben sind in den Daten Intervalle enthalten, aus denen die jeweiligen Werte stammen. Beispielsweise wurde Alinas Alter 20 durch das Intervall 20–22 generalisiert. Bei den Postleitzahlen wurden die letzten beiden Ziffern durch „*“ maskiert, sodass 101** für jede Zahl zwischen 10100 und 10199 stehen kann. Das Geschlecht wiederum ist komplett durch den Stern ersetzt worden, wodurch diese Information komplett verschwunden und damit unterdrückt worden ist.

[7] Wang et al. [147] untersuchten den Fall, bei dem Attribute sowohl sensible als auch quasi-identifizierende Werte enthalten.

	Alter	PLZ	Geschlecht	Krankheit
Alina	20–22	101**	*	Asthma
Bill	20–22	101**	*	Bronchitis
Clive	20–22	101**	*	Diabetes
Doris	23–25	214**	*	Grippe
Emily	23–25	214**	*	Diabetes
Frank	23–25	214**	*	Grippe
Gill	23–25	214**	*	Bronchitis
Harry	26–28	290**	*	Diabetes
Isabel	26–28	290**	*	Diabetes

Tabelle 2.3: Anonymisierte Daten T^*

Um eine generalisierte Tabelle zu erzeugen, werden meist sogenannte Generalisierungshierarchien verwendet.[8] Dabei wird die Domäne eines Attributs in mehrere, meist gleich große Bereiche unterteilt, welche jeweils durch einen neuen Bezeichner referenziert werden. Im Fall von numerischen Attributwerten werden dazu in der Regel Intervalle verwendet, bei diskreten Werten müssen allgemeinere Begriffe abgeleitet werden. In der generalisierten Tabelle wird nun anstatt eines Attributwertes der Bereich veröffentlicht, der den Wert enthält. Diese Generalisierung kann sich hierarchisch fortsetzen, das heißt, bereits verallgemeinerte Werte können erneut generalisiert werden.

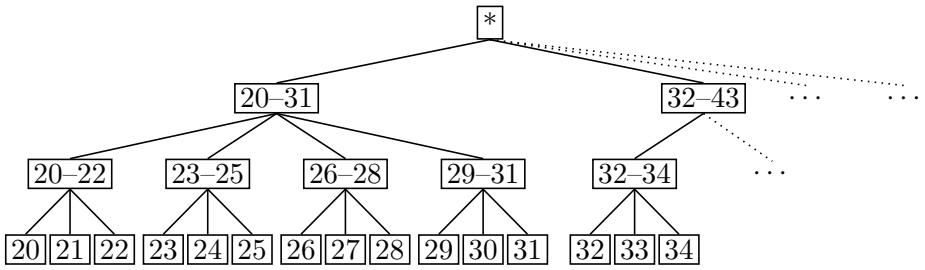


Abbildung 2.1: Generalisierungshierarchie für das Attribut `Alter`

Für das Attribut `Alter` stellt Abbildung 2.1 eine mögliche Generalisierungshierarchie dar. In der ersten Stufe werden jeweils drei Werte zu einem Intervall zusammengefasst. In den weiteren Stufen werden auch diese Intervalle zu größeren Intervallen vereinigt. Abbildung 2.2 gibt ein ähnliches Konzept für das Attribut `PLZ` wieder, wobei die Generalisierung durch das Entfernen der jeweils letzten Ziffer der Postleitzahl entsteht.

Beispiel 2.2 In Tabelle 2.3 wurden alle Altersangaben einmal und alle Postleitzahlen zweimal generalisiert. Die Werte für das Geschlecht sind unterdrückt worden.

Beim Generalisieren von Mikrodaten wird die *globale* (engl. *global recoding*) von der *lokalen Umcodierung* (engl. *local recoding*) unterschieden [89]. Beim globalen Umcodieren wird jedem Wert eines Attributs derselbe Eintrag in der dazugehörigen Hierarchie zugewiesen. Tritt demzufolge ein Wert mehrfach in den Originaldaten auf, wird er jeweils auf

[8] In der Literatur werden Domänen- und Wertegeneralisierungshierarchien [134] unterschieden. Die hier vorgestellten Hierarchien entsprechen den Wertegeneralisierungshierarchien.

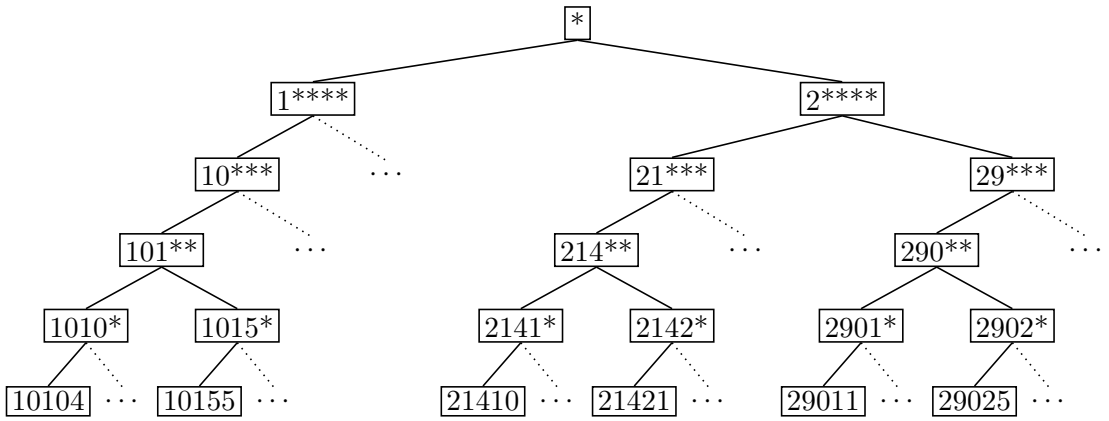


Abbildung 2.2: Generalisierungshierarchie für das Attribut `PLZ`

denselben Wert generalisiert. Ein Spezialfall der globalen Umcodierung ist die sogenannte *Domänen-Vollgeneralisierung* (engl. *full-domain generalization*), bei der alle Werte eines Attributs zusätzlich auf dieselbe Stufe in der Hierarchie generalisiert werden. Tabelle 2.3 ist ein Beispiel für eine Domänen-Vollgeneralisierung.

	Alter	PLZ	Geschlecht	Krankheit
Alina	20–21	101**	*	Asthma
Bill	20–21	101**	*	Bronchitis
Clive	20–21	101**	*	Diabetes
Doris	23–24	214**	*	Grippe
Emily	23–24	214**	*	Diabetes
Frank	23–24	214**	*	Grippe
Gill	24–27	2****	*	Bronchitis
Harry	24–27	2****	*	Diabetes
Isabel	24–27	2****	*	Diabetes

Tabelle 2.4: Anonymisierte Daten T^{**}

Bei der lokalen Umcodierung werden Attributwerte einzeln generalisiert, je nachdem, zu welchem Tupel sie gehören. Dadurch ist es möglich, denselben Wert in verschiedenen Tupeln unterschiedlich zu generalisieren. Wird auf Generalisierungshierarchien verzichtet, können sich die generelleren Werte an den tatsächlich vorkommenden Werten einzelner Tupel orientieren. Tabelle 2.4 stellt eine lokale Umcodierung der bereits eingeführten Mikrodaten (s. Tabelle 2.1) dar. Erkennbar ist die Lokalität am Alterswert 24, der bei Emily und Frank zu 23–24, aber bei Gill zu 24–27 generalisiert wurde. Die Intervalle der Altersangaben umfassen jeweils drei Tupel und sind so klein wie möglich gewählt worden. Die Angaben zur Postleitzahl entsprechen denen der Hierarchie aus Abbildung 2.2.

Generalisierung und Unterdrückung führen immer zu einem Informationsverlust. Für statistische Zwecke kann es jedoch wichtig sein, dass anstatt des korrekten Wertes für das Alter 20 der Wertebereich 20–22, 20–30 oder sogar nur 0–100 veröffentlicht wird. Im letzten Fall werden so wenig Informationen herausgegeben, dass dieser Wert fast mit der Unterdrückung gleichgesetzt werden kann. Daher besteht das Ziel beim Veröffentlichen

von Daten darin, immer möglichst wenig Daten zu generalisieren und trotzdem die Privatsphäre der Individuen ausreichend zu schützen.

Anatomie und Bucketisierung

Xiao und Tao [160] stellten einen alternativen Ansatz zur Veröffentlichung von Mikrodaten vor. Sie benannten ihr Modell *Anatomie*, allerdings wurde es von weiteren Autoren verfeinert und ist auch unter dem Namen *Bucketisierung* [104, 39] bekannt. Im Gegensatz zur Generalisierung und Unterdrückung werden dabei keine Werte modifiziert. Um eine Re-Identifizierung zu verhindern, wird die Beziehung zwischen den quasi-identifizierenden und den sensiblen Attributwerten getrennt. Ermöglicht wird dies durch die Veröffentlichung zweier separater Tabellen.

Die Tupel einer gegebenen Mikrodatentabelle werden zunächst in Gruppen bestimmter Größe aufgeteilt. Jede Gruppe bekommt einen eindeutigen Identifikator. Danach werden die Attribute der Tupel in quasi-identifizierende und sensible Attribute getrennt und zwei Tabellen erstellt. In die erste Tabelle (QI-Tabelle) werden für jedes Tupel die Werte der Quasi-Identifikatoren und die zuvor bestimmte Gruppen-ID eingefügt. Die zweite Tabelle (sensible Tabelle) enthält für jede Gruppe alle sensiblen Attributwerte, die in Tupeln dieser Gruppe vorkommen.

	Alter	PLZ	Geschlecht	Gruppen-ID
Alina	20	10104	w	1
Bill	21	10104	m	1
Clive	21	10155	m	1
Doris	23	21410	w	2
Emily	24	21410	w	2
Frank	24	21410	m	2
Gill	24	21421	w	2
Harry	26	29011	m	3
Isabel	27	29025	w	3

(a) QI-Tabelle

Gruppen-ID	Krankheit	Anzahl
1	Asthma	1
1	Bronchitis	1
1	Diabetes	1
2	Grippe	2
2	Diabetes	1
2	Bronchitis	1
3	Diabetes	2

(b) Sensible Tabelle

Tabelle 2.5: Anonymisierte Tabelle für Anatomie bzw. Bucketisierung

Beispiel 2.3 Die Mikrodaten aus Tabelle 2.1 werden zunächst in drei Gruppen eingeteilt. Die erste Gruppe enthält die Tupel von Alina, Bill und Clive, die zweite die von Doris, Emily, Frank und Gill sowie die dritte die von Harry und Isabel. Tabelle 2.5a zeigt die entstehende QI-Tabelle. Die sensiblen Attributwerte der Tupel sind in der sensiblen Tabelle 2.5b abgebildet. Für jeden Wert ist die Anzahl angegeben, wie häufig er in der entsprechenden Gruppe vorkommt.

Ein äquivalenter Ansatz mit dem Namen *Permutation* wurde von Zhang et al. [166] eingeführt. Der einzige Unterschied zur Bucketisierung besteht in der Zusammenfassung der QI- und sensiblen Tabelle. Das entspricht der Veröffentlichung von Mikrodaten ohne identifizierende Attribute, in der die sensiblen Werte beliebig vertauscht (permutiert) wurden. Tabelle 2.6 stellt den Permutationsansatz für das bereits eingeführte Beispiel dar. Für eine bessere Unterscheidung dieses Modells stehen die Krankheiten nicht untereinander, sondern sind als Listen zusammengefasst worden.

	Alter	PLZ	Geschlecht	Krankheit
Alina	20	10104	w	Asthma, Bronchitis, Cholera
Bill	21	10104	m	
Clive	21	10155	m	
Doris	23	21410	w	Bronchitis, Diabetes, Grippe, Grippe
Emily	24	21410	w	
Frank	24	21410	m	
Gill	24	21421	w	
Harry	26	29011	m	Diabetes, Diabetes
Isabel	27	29025	w	

Tabelle 2.6: Permutation der Mikrodaten aus Tabelle 2.1 analog zu Tabelle 2.5

Gegenüber der Generalisierung hat die Bucketisierung den Vorteil, dass die Werte der Quasi-Identifikatoren unverändert bleiben. Damit können mitunter genauere Analysen der Daten durchgeführt werden [160, 166].

Perturbation

Perturbation[9] hat eine lange Tradition im Bereich der statistischen Veröffentlichungen [1]. Dabei werden Originalwerte durch synthetische Werte ersetzt, sodass die statistischen Eigenschaften der veränderten Daten denen der Originaldaten entsprechen. Der Vorteil dieses Ansatzes zeichnet sich darin aus, dass die so veröffentlichten Tupel zu keinen realen Personen gehören und damit Verletzungen der Privatsphäre verhindert werden können. Gegenüber den anderen hier vorgestellten Methoden ist Perturbation aber mit dem Nachteil behaftet, dass die veröffentlichten Daten synthetisch und damit für viele Forschungszwecke und reale Anwendungsgebiete ungeeignet sind. Im Kontext dieser Arbeit spielen die Methoden der Perturbation daher keine Rolle und werden dementsprechend auch nicht weiter erläutert. Eine Übersicht in dieses Gebiet geben zum Beispiel Ciriani [24] und Fung et al. [54].

2.2.4 Informationspreisgabe

Schutz der Privatsphäre bedeutet zugleich Schutz vor dem Herausgeben von Informationen. Insbesondere sollen die sensiblen Attributwerte von Individuen geschützt und damit nicht preisgegeben werden. Der überwiegende Teil der Literatur des PPDP beschreibt eine recht einfache und praktische Vorstellung, wann Informationen als geeignet geschützt angesehen werden. Dabei werden drei Arten der Informationspreisgabe unterschieden [54].

Bei der *Identitätspreisgabe* (engl. *identity disclosure*) wird ein Individuum mit einem Tupel der veröffentlichten Daten verknüpft, sodass ein Angreifer ein bestimmtes Tupel eindeutig einem Individuum zuordnen kann (Re-Identifikation). Eine *Attributpreisgabe* (engl. *attribute disclosure*) tritt ein, wenn ein Angreifer den sensiblen Wert eines Individuums herausfindet. Mitunter wird bereits die Tatsache, dass ein bestimmter sensibler Wert mit hoher Wahrscheinlichkeit zu einem Individuum gehört, als Attributpreisgabe angesehen. Aus Identitätspreisgabe folgt in der Regel eine Attributpreisgabe, andersherum muss das aber nicht der Fall sein. In beiden Fällen wird davon ausgegangen, dass der

[9] Perturbation von lat. *perturbare* bedeutet Verwirrung oder Störung.

Angreifer bereits weiß, dass ein Individuum durch ein Tupel in der veröffentlichten Tabelle beschrieben wird. Außerdem kennt er sämtliche Werte der Quasi-Identifikatoren durch Hintergrundwissen oder andere öffentliche Datenquellen.

Die dritte Art der Informationspreisgabe tritt ein, wenn ein Angreifer schlussfolgern kann, dass irgendein oder gar kein Tupel in der veröffentlichten Tabelle zu einem Individuum gehört. Dabei muss er das entsprechende Tupel oder den sensiblen Wert selbst nicht herausfinden. Bei dieser sogenannten *Tabellenpreisgabe* (engl. *table disclosure*) weiß der Angreifer im Vorfeld nicht, ob ein Individuum überhaupt durch ein Tupel in den veröffentlichten Daten beschrieben wird. Ein Beispiel hierfür ist eine anonymisierte und veröffentlichte Tabelle von Patienten, die schwere Herzkrankheiten haben. Allein die Information darüber, ob eine Person in dieser Tabelle erfasst ist, kann bereits eine Verletzung ihrer Privatsphäre darstellen.

Eine veröffentlichte Tabelle schützt die Privatsphäre, wenn sie eine der drei Informationspreisgaben verhindert. Dabei ist zu bemerken, dass das Unterbinden von Tabellenpreisgabe einen stärkeren Schutz der Privatsphäre bietet als das Vermeiden von Attributpreisgabe. Letztgenanntes ist wiederum restriktiver als Identitätspreisgabe.

Eine andere Formulierung des Schutzes der Privatsphäre stellt das *uninformative Prinzip* [101] dar. Es besagt, dass eine veröffentlichte Tabelle Angreifern wenig zusätzliche Informationen zur Verfügung stellen soll. Der Grundgedanke hierbei besteht darin, dass ein Angreifer durch Hintergrundwissen und andere Quellen bestimmte Annahmen über sensible Werte von Individuen hat (apriorisches Wissen). Durch Kenntnisse aus einer veröffentlichten Tabelle ändern sich seine Annahmen (aposteriorisches Wissen). Der Unterschied zwischen beiden Kenntnisständen kann als *Informationsgewinn* bezeichnet werden. Ziel ist die Beschränkung des Informationsgewinns durch die veröffentlichte Tabelle. Es gibt Modelle, die das uninformative Prinzip beachten und gleichzeitig die Informationspreisgabe nach einer der drei eingeführten Varianten verhindern.

2.3 Modelle für den Schutz der Privatsphäre

Um die Privatsphäre beim Veröffentlichen von Daten zu schützen, wurden in den letzten Jahren viele Modelle entwickelt. Alle beschreiben in unterschiedlicher Form, wie anonymisierte Daten aussehen sollen. Tabelle 2.7 gibt einen Überblick über die im Rahmen dieser Arbeit wichtigsten Modelle. Für jeden Ansatz ist markiert, welche Informationspreisgabe er verhindert.

2.3.1 Identitätspreisgabe

Um vor der Identitätspreisgabe zu schützen, müssen die Daten in einer Tabelle so anonymisiert werden, dass ein Verknüpfen von externen Quellen mit Tupeln nicht eindeutig möglich ist. Tupel müssen also in einer gewissen Form ähnlich oder identisch zueinander sein. Das Grundprinzip, auf dem die meisten hier aufgeführten Modelle basieren, ist die k-Anonymität.

k-Anonymität

Das erste und gleichzeitig bekannteste Modell wurde von Samarati und Sweeney [123, 135] eingeführt und *k-Anonymität* (engl. *k-anonymity*) genannt. Es stellt sicher, dass die Daten

Modell	Identitätspreisgabe	Attributpreisgabe	Tabellenpreisgabe
k-Anonymität	×		
(X, Y)-Anonymität	×		
MultiR k-Anonymität	×		
(c, t)-Isolation	×		
p-sensible k-Anonymität	×	×	
ℓ-Diversität	×	×	
(α, k)-Anonymität	×	×	
Confidence-Bounding	×	×	
(X, Y)-Privatsphäre	×	×	
Personalisierte Anonymität	×	×	
t-closeness	×	×	
(α_i, β_i)-closeness	×	×	
quadrierte Fehlerdiversität	×	×	
(k, e)-Anonymität	×	×	
(ε, m)-Anonymität	×	×	
δ-Präsenz	×	×	×
Differentielle Privatsphäre			×

Tabelle 2.7: Modelle für den Schutz der Privatsphäre und verhinderte Informationspreisgabe

jedes Individuums in einer veröffentlichten Tabelle von $k - 1$ anderen Individuen nicht unterschieden werden können. Ein Verknüpfen mit anderen Datenquellen ist dadurch nicht eindeutig möglich.

Um eine k-anonyme Tabelle zu erstellen, werden zunächst alle identifizierenden Attribute aus den Mikrodaten entfernt. Danach werden die quasi-identifizierenden Attribute so weit anonymisiert, dass immer mindestens k Tupel bezüglich der Quasi-Identifikatoren nicht unterscheidbar sind. Eine Tabelle erfüllt k-Anonymität, wenn für jedes Tupel $k - 1$ andere Tupel existieren, die dieselben Werte bezüglich der QI besitzen.

	Alter	PLZ	Geschlecht	Krankheit	
Alina	20–22	101**	*	Asthma	QI-Gruppe 1
Bill	20–22	101**	*	Bronchitis	
Clive	20–22	101**	*	Diabetes	
Doris	23–25	214**	*	Grippe	QI-Gruppe 2
Emily	23–25	214**	*	Diabetes	
Frank	23–25	214**	*	Grippe	
Gill	23–25	214**	*	Bronchitis	
Harry	26–28	290**	*	Diabetes	QI-Gruppe 3
Isabel	26–28	290**	*	Diabetes	

Tabelle 2.8: Anonymisierte Daten T^*

In der bereits eingeführten Tabelle 2.8 haben die Tupel von Alina, Bill und Clive dieselben Werte für `Alter`, `PLZ` und `Geschlecht`. Ebenso sind die Tupel von Doris, Emily, Frank

und Gill sowie von Harry und Isabel nicht zu unterscheiden. Tabelle 2.8 erfüllt demnach k-Anonymität mit $k = 2$.

Die Definition der k-Anonymität induziert eine Klassenbildung bezüglich der QI. Die Menge aller Tupel, die dieselbe Kombination quasi-identifizierender Attributwerte besitzen, wird *QI-Gruppe*[10] genannt. Tabelle 2.8 besteht beispielsweise aus drei QI-Gruppen.

Die Idee hinter k-Anonymität ist, dass Tupel nicht eindeutig mit anderen Datenquellen verknüpft werden können. Kennt ein Angreifer beispielsweise Alinas Alter, Postleitzahl und Geschlecht, kann er zwar schlussfolgern, dass eines der ersten drei Tupel in Tabelle 2.8 zu Alina gehört. Das tatsächliche Tupel kann er aber nicht herausfinden. Im Allgemeinen kommen dafür immer mindestens k Tupel in Frage, das heißt, k-Anonymität schützt vor Identitätspreisgabe.

Allerdings reicht dieses Modell für gewöhnlich nicht aus, um auch Attributpreisgabe zu verhindern. Der Grund dafür ist, dass auf die sensiblen Attributwerte beim Erzeugen der QI-Gruppen nicht geachtet wird. Möchte ein Angreifer beispielsweise das Tupel von Harry mithilfe dessen QI-Werten herausfinden, kann er nur ableiten, dass es das letzte oder vorletzte Tupel in Tabelle 2.8 sein muss. Da in beiden Tupeln aber die Krankheit Diabetes vorkommt, kann der Angreifer jedoch den sensiblen Wert von Harry schlussfolgern. In der Literatur wird dieser Angriff als *Homogenitätsattacke* [101] bezeichnet.

	Alter	PLZ	Geschlecht	Krankheit	
Alina	20–21	101**	*	Asthma	} QI-Gruppe 1
Bill	20–21	101**	*	Bronchitis	
Clive	20–21	101**	*	Diabetes	
Doris	23–24	214**	*	Grippe	} QI-Gruppe 2
Emily	23–24	214**	*	Diabetes	
Frank	23–24	214**	*	Grippe	
Gill	24–27	2****	*	Bronchitis	} QI-Gruppe 3
Harry	24–27	2****	*	Diabetes	
Isabel	24–27	2****	*	Diabetes	

Tabelle 2.9: Anonymisierte Daten T^{**}

Tabelle 2.9 stellt eine andere Anonymisierung der Mikrodaten dar. Sie erfüllt k-Anonymität mit $k = 3$, denn alle QI-Gruppen enthalten drei Tupel. Im Gegensatz zur Tabelle 2.8 kommen in jeder QI-Gruppe mindestens zwei verschiedene sensible Attributwerte vor, das heißt, in Tabelle 2.9 ist die Homogenitätsattacke nicht möglich.

Viele Ansätze erweitern k-Anonymität, indem sie Restriktionen für die sensiblen Attributwerte in den einzelnen QI-Gruppen hinzufügen. Sie schützen dadurch vor Attributpreisgabe und werden im Folgenden vorgestellt. Zuvor allerdings werden Modelle eingeführt, die einige andere Voraussetzungen als der Ansatz von k-Anonymität haben.

(X, Y)-Anonymität

Eine wichtige Voraussetzung beim Ansatz von k-Anonymität besteht darin, dass jedes Tupel in der Datentabelle genau ein anderes Individuum repräsentiert. Ist das nicht der

[10] In der Literatur gibt es neben dem Begriff der QI-Gruppe [161, 162, 92, 53] noch die äquivalenten Ausdrücke q^*-Block [101], QI-Cluster [17, 16] und Äquivalenzklasse [90, 159, 93].

Fall, kann die Privatsphäre trotz k-Anonymität verletzt werden. Beispielsweise können in medizinischen Daten Personen mehrere Krankheiten haben, die jeweils durch ein anderes Tupel ausgedrückt werden. Damit kann es passieren, dass in einer QI-Gruppe von k Tupeln mehrere oder sogar alle Tupel zur selben Person gehören, deren Privatsphäre dadurch nicht mehr ausreichend geschützt ist.

Um dieses Problem zu vermeiden, haben Wang und Fung [143] als Verallgemeinerung von k-Anonymität (X, Y)-*Anonymität* vorgeschlagen. Seien X und Y zwei disjunkte Mengen von Attributen und x ein Wert aus X. Dann ist die Anonymität $a_Y(x)$ von x bezüglich Y die Anzahl verschiedener Werte aus Y, die in einem Tupel vorkommen, das den Wert x enthält: $a_Y(x) = |\pi_Y(\sigma_x(T))|$, wenn T die Datentabelle ist.[11] Sei $A_Y(X) = min\{a_Y(x) \mid x \in X\}$. Eine Tabelle erfüllt (X, Y)-Anonymität für eine gegebene natürliche Zahl k, wenn $A_Y(X) \geq k$.

(X, Y)-Anonymität verlangt, dass jeder Wert in X mit mindestens k verschiedenen Werten in Y verbunden werden kann. Somit ist k-Anonymität der Spezialfall, bei dem X die Menge der Quasi-Identifikatoren und Y ein ID-Attribut ist. Andererseits können durch (X, Y)-Anonymität verschiedene Anforderungen an die Privatsphäre sehr flexibel spezifiziert werden. Wird Y beispielsweise auf die Menge der sensiblen Werte gesetzt, dann fordert (X, Y)-Anonymität, dass in jeder QI-Gruppe mindestens k verschiedene SA-Werte vorkommen.

MultiRelationale k-Anonymität

Die meisten Beiträge in der Literatur zur k-Anonymität richten ihr Augenmerk auf das Anonymisieren einer einzelnen Datentabelle. Viele reale Datenbanksysteme enthalten allerdings mehrere relationale Tabellen. Daher führten Nergiz et al. [113] ein Modell namens *MultiR (MultiRelationale) k-Anonymität* ein, um k-Anonymität auch auf mehreren Relationen sicherzustellen. In ihrem Modell besteht ein relationales Datenbanksystem aus einer Tabelle PT, die personenbezogene Daten enthält, und einer Menge weiterer Tabellen $T_1, \ldots, T_n$. Während PT einen Personenidentifikator Pid und einige sensible Attribute enthält, kommen in T_i, $1 \leq i \leq n$, Fremdschlüssel, quasi-identifizierende und ebenfalls sensible Attribute vor. Eine MultiR k-Anonymität liegt vor, wenn der Join aller Tabellen $JT \coloneqq PT \bowtie T_1 \bowtie \cdots \bowtie T_n$ k-anonym bezüglich Pid ist. Es wird gefordert, dass für jede durch einen Pid-Wert referenzierte Person, zu der ein Tupel in JT gehört, mindestens $k-1$ weitere Personen existieren, die Tupel mit denselben QI-Werten besitzen. Damit wird k-Anonymität auf Personenebene und nicht wie traditionell auf Tupelebene angewendet. Dieser Ansatz ist analog zu (X, Y)-Anonymität, wobei X die Menge der QI und Y die Menge mit Pid ist.

Neben dem hier vorgestellten Modell gibt es noch weitere Ansätze, k-Anonymität auf verteilten Daten sicherzustellen [144, 167, 72].

(c, t)-Isolation

Ein sehr abstraktes Modell, um die Privatsphäre von Individuen zu schützen, stellten Chawla et al. [19] vor. Ihr Ziel ist es zu verhindern, dass ein Angreifer Tupel aus der veröffentlichten Tabelle Individuen eindeutig zuordnen kann. Sei jedes Tupel einer Datentabelle

[11] Aus T werden alle Tupel mit Wert x selektiert und dann auf die Attribute von Y projiziert. Die Anzahl der verbleibenden Werte ist $a_Y(x)$.

durch einen Punkt in einem hochdimensionalen Raum repräsentiert und sei q der Punkt, den ein Angreifer durch Hintergrundwissen und Kenntnis der veröffentlichten Daten für ein Individuum erhält. Weiterhin sei p ein Punkt der Originaldatentabelle, der den kleinsten Abstand δ_p zu q hat. Bezeichne $B(q, c\delta_p)$ eine Kugel im hochdimensionalen Raum mit Mittelpunkt q und Radius $c\delta_p$. Dann (c,t)-*isoliert* q den Punkt p, wenn $B(q, c\delta_p)$ weniger als t Punkte der Originaldatentabelle enthält. Dahinter steht die Idee, dass alle t Punkte ähnlich zu q sind und der Angreifer q potentiell t Individuen zuordnen kann. Dieser Ansatz ist damit ähnlich zu k-Anonymität mit $k = t$ und verhindert Identitätspreisgabe, schützt aber nicht vor Attributpreisgabe.

2.3.2 Attributpreisgabe

Da k-Anonymität zwar vor Identitäts-, aber nicht vor der Attributpreisgabe schützt, wurden in den letzten Jahren viele Erweiterungen dieser Idee entwickelt. Die meisten Ansätze beschränken die sensiblen Attributwerte in den einzelnen QI-Gruppen und fordern eine gewisse Heterogenität unter ihnen. Andere Modelle beschäftigen sich mit Spezialfällen wie zum Beispiel mit numerischen Attributwerten. Meistens liegt der jeweiligen Idee eine bestimmte Art von Angriff zugrunde, vor dem zusätzlich geschützt werden soll.

Angriffe

Der erste Angriff auf Daten, die k-anonym sind, stellt die bereits vorgestellte Homogenitätsattacke dar, bei der eine QI-Gruppe nur einen einzigen sensiblen Wert enthält. Sie ist die Ursache dafür, dass k-Anonymität im Allgemeinen nicht vor Attributpreisgabe schützt. Daneben existiert noch eine Vielzahl weiterer Angriffe, auf die in diesem Abschnitt kurz eingegangen wird.

Die *Attacke durch Hintergrundwissen* (engl. *background knowledge attack*) [101] setzt zusätzliche Kenntnisse des Angreifers voraus. Kann er beispielsweise bestimmte Werte für ein Individuum von vornherein ausschließen, ist es ihm eventuell möglich, den sensiblen Wert des Individuums durch die veröffentlichte Tabelle abzuleiten. Falls ein Angreifer weiß, dass Emily keine Erkrankung des Atmungssystems[12] hat, kann er leicht aus Tabelle 2.8 schlussfolgern, dass sie Diabetes haben muss.

Eine *probabilistische Attacke* (engl. *probabilistic inference attack*) [93] ist möglich, wenn ein sensibler Wert in einer QI-Gruppe wesentlich häufiger auftritt als die anderen Werte. Ohne Hintergrundwissen kann ein Angreifer die Wahrscheinlichkeit, dass ein Individuum einen bestimmten sensiblen Wert besitzt, mit der relativen Häufigkeit des Wertes innerhalb der entsprechenden QI-Gruppe abschätzen. In Tabelle 2.9 kann er zum Beispiel für jedes Individuum der zweiten QI-Gruppe mit einer Wahrscheinlichkeit von $\frac{2}{3}$ annehmen, dass es Grippe hat, und mit $\frac{1}{3}$, dass es unter Diabetes leidet. Wenn nun ein sensibler Wert besonders häufig vorkommen würde, wäre auch die Wahrscheinlichkeit, dass ein Individuum diesen Wert hat, besonders groß.

Ein weiterer Angriff ist die *Ähnlichkeitsattacke* (engl. *similarity attack*) [93]. Sie tritt ein, wenn sich die sensiblen Werte innerhalb einer QI-Gruppe semantisch wenig unterscheiden.

[12] wie z. B. Grippe oder Bronchitis [33]

Enthält eine Gruppe zum Beispiel die Werte Gastritis[13], Dyspepsie[14] und Duodenitis[15], so leiden alle dazugehörigen Individuen an Krankheiten des Verdauungssystems [34].

Ein dazu verwandter Angriff, der bei numerischen Attributwerten auftritt, ist die *Näheattacke* (engl. *proximity attack*) [92]. Kommen in einer QI-Gruppe neun Werte zwischen 1 und 10 und ein Wert von 1000 vor, kann ein Angreifer mit hoher Wahrscheinlichkeit schlussfolgern, dass die sensiblen Werte der Individuen im Intervall [1–10] liegen. Da auch größere Werte wie 1000 möglich sind, kann diese Vermutung bereits eine Verletzung der Privatsphäre darstellen.

Ein Informationsgewinn des Angreifers durch eine bestimmte Verteilung der Daten stellt ebenfalls einen Angriff dar. Unterscheidet sich die Verteilung der sensiblen Attributwerte in einer QI-Gruppe stark von der Verteilung unter allen Individuen, gelangt ein Angreifer zu neuem Wissen. Ein Beispiel dafür ist eine QI-Gruppe, in der ein sensibler Wert zu 50 % vorkommt, aber insgesamt nur 1 % aller Individuen in der Datentabelle den besagten Wert haben. Diese Form von Angriff wird *Schiefeattacke* (engl. *skewness attack*) [93] genannt.

p-sensible k-Anonymität

Von Truta und Vinay [137] stammt die erste Erweiterung in dieser Auflistung. Sie fügten dem k-Anonymitätsansatz eine einfache Restriktion der sensiblen Werte hinzu. Eine Tabelle erfüllt *p-sensible k-Anonymität*, wenn in jeder QI-Gruppe mindestens p verschiedene sensible Werte vorkommen.

Beispiel 2.4 Tabelle 2.8 erfüllt nur p-sensible k-Anonymität mit $p = 1$ und $k = 2$, denn in der letzten QI-Gruppe gibt es nur einen sensiblen Wert. Tabelle 2.9 ist hingegen 2-sensibel 3-anonym.

Campan et al. [16] entwickelten dieses Modell weiter und fügten ihm Generalisierungsbeschränkungen hinzu. Damit ist es möglich, für jeden Wert eines Quasi-Identifikators anzugeben, bis zu welcher Hierarchiestufe er höchstens generalisiert werden darf.

ℓ-Diversität

Ein wesentlich allgemeineres Prinzip von Heterogenität unter sensiblen Werten stellten Machanavajjhala et al. [101] vor. Sie forderten, dass in jeder QI-Gruppe mindestens ℓ „wohl-repräsentierte“ sensible Werte vorkommen sollen. Dieses Prinzip heißt *ℓ-Diversität* (engl. *ℓ-diversity*) und hat mehrere Instanziierungen, die sich alle in der Definition von „wohl-repräsentiert“ unterscheiden.

Der einfachste Ansatz ist, dass in jeder QI-Gruppe mindestens ℓ verschiedene sensible Werte vorkommen sollen, was *eindeutige ℓ-Diversität* (engl. *distinct ℓ-diversity*) genannt wird. Diese Idee entspricht p-sensibler k-Anonymität mit $p = k = \ell$. Da es allerdings Werte geben kann, die wesentlich häufiger in Daten vorkommen als andere,[16] haben Machanavajjhala et al. zwei weitere Instanziierungen eingeführt. Die erste heißt *Entropie-ℓ-Diversität* (engl. *entropy ℓ-diversity*) und fordert, dass in jeder QI-Gruppe q

[13] Gastritis ist eine Entzündung der Magenschleimhaut.
[14] Dyspepsie heißt übersetzt Verdauungsstörung.
[15] Duodenitis ist eine akute oder chronische Entzündung der Schleimhaut des Zwölffingerdarms (Duodenum).
[16] Vgl. probabilistische Attacke.

$-\sum_{s \in S} p_{q,s} \cdot \log(p_{q,s}) \geq \log(\ell)$ gelten soll, wobei S die Menge sensibler Werte und $p_{q,s}$ der Anteil von Tupeln in q ist, die den sensiblen Wert s haben. Die linke Seite der Ungleichung wird Entropie des sensiblen Attributs genannt und hat die Eigenschaft, dass gleichmäßige Verteilungen einen größeren Wert besitzen. Es kann gezeigt werden, dass ℓ-Diversität immer auch k-Anonymität mit $k = \ell$ impliziert.

Beispiel 2.5 Tabelle 2.3 erfüllt nur eindeutige 1-Diversität, denn in der letzten QI-Gruppe kommt nur der sensible Wert Diabetes vor. Die anderen beiden QI-Gruppen sind jeweils eindeutig 3-divers. Die Entropie der einzelnen Gruppen sind 1,585, 1,5 und 0. Somit ist T^* Entropie-1-divers.

Tabelle 2.9 ist eindeutig 2-divers, denn in jeder QI-Gruppe kommen mindestens zwei verschiedene sensible Werte vor. Darüber hinaus ist sie auch Entropie-1,889-divers, denn die Entropie der letzten beiden QI-Gruppen ist jeweils 0,918.

Die letzte hier vorgestellte Instanziierung wird *rekursive* (c, ℓ)*-Diversität* (engl. *recursive* (c, ℓ)*-diversity*) genannt und stellt sicher, dass der häufigste Wert nicht zu häufig vorkommt. Sei m die Anzahl der sensiblen Werte, die in einer QI-Gruppe q vorkommen, und seien diese absteigend nach ihrer Häufigkeit sortiert. Weiterhin bezeichne r_i für $1 \leq i \leq m$ die Häufigkeit des i-ten Elements (d. h., r_1 ist die Häufigkeit des häufigsten Elements, r_2 die des zweithäufigsten Elements usw.). Dann erfüllt eine Tabelle rekursive (c, ℓ)-Diversität, wenn für jede QI-Gruppe $r_1 < c \cdot (r_\ell + r_{\ell+1} + \cdots + r_m)$ gilt, wobei c eine Konstante ist. Dahinter steht die Intuition, dass selbst wenn ein Angreifer bestimmte sensible Werte durch Hintergrundwissen für ein bestimmtes Individuum ausschließen kann, weiterhin viele andere mögliche sensible Werte existieren.

(α, k)-Anonymität

Von Wong et al. [159] stammt ein Modell, das sehr ähnlich zur ℓ-Diversität ist. Eine Tabelle erfüllt (α, k)*-Anonymität*, wenn sie k-Anonymität erfüllt und in jeder QI-Gruppe die relative Häufigkeit jedes sensiblen Wertes höchstens α ist. Damit ist für potentielle Angreifer die Wahrscheinlichkeit, dass ein Individuum einen bestimmten sensiblen Wert hat, durch α beschränkt.

Beispiel 2.6 Tabelle 2.8 erfüllt nur $(1, 2)$-Anonymität, denn in der letzten QI-Gruppe kommt nur der sensible Wert Diabetes vor. Tabelle 2.9 ist $(\frac{2}{3}, 3)$-anonym, denn sie erfüllt 3-Anonymität und in zwei QI-Gruppen (der Größe 3) kommt jeweils ein sensibler Wert zweimal vor.

Confidence-Bounding

Eine ähnliche Idee setzten Wang et al. [145, 146] mit einem viel allgemeineren Modell um. Sie beschränkten die Wahrscheinlichkeit, dass ein Angreifer einen sensiblen Wert eines Individuums schlussfolgern kann, mit sogenannten *Privatsphärenvorlagen* (engl. *privacy templates*) der Form $\langle QID \to s, p \rangle$. Dabei ist QID ein Quasi-Identifikator, s ein sensibler Wert und p eine Wahrscheinlichkeit. Sei $conf(q \to s)$ die relative Häufigkeit von Tupeln in der QI-Gruppe q, die den sensiblen Wert s haben. Dann erfüllt eine Tabelle eine Vorlage $\langle QID \to s, p \rangle$, wenn $\max_q \{conf(q \to s)\} \leq p$ gilt, wobei q über alle QI-Gruppen bezüglich

QID berechnet wird. Mit anderen Worten beschränkt $\langle QID \to s, p\rangle$ die Wahrscheinlichkeit, dass ein Angreifer den sensiblen Wert s in irgendeiner QI-Gruppe bezüglich QID schlussfolgern kann, auf höchstens p.

Beispiel 2.7 Sei $QID = \{\texttt{Alter}, \texttt{PLZ}, \texttt{Geschlecht}\}$ ein Quasi-Identifikator. Dann erfüllt Tabelle 2.8 die Vorlagen $\langle QID \to \text{Bronchitis}, \frac{1}{3}\rangle$ und $\langle QID \to \text{Grippe}, \frac{1}{2}\rangle$.

Dieses Modell mit dem Namen *Confidence-Bounding*[17] ist flexibler als die bisherigen Ansätze, da es möglich ist, für unterschiedliche Quasi-Identifikatoren und sensible Werte verschiedene Schranken für die Wahrscheinlichkeiten festzulegen. Ähnlich wie (α, k)-Anonymität ist es zudem sehr intuitiv, da das Risiko einer Verletzung der Privatsphäre klar angegeben werden kann.

(X, Y)-Privatsphäre

Eine Kombination von (X, Y)-Anonymität und Confidence-Bounding stellt das von Wang und Fung [143] eingeführte Konzept der (X, Y)-*Privatsphäre* (engl. (X, Y)-*privacy*) dar. Dazu müssen einerseits in jeder Gruppe bezüglich X mindestens k verschiedene Werte aus Y vorkommen ((X, Y)-Anonymität) und andererseits muss die Vorlage $\langle X \to y, p\rangle$ für jeden Wert $y \in Y$ und ein gegebenes p erfüllt sein.

Personalisierte Anonymität

Einen Ansatz, der nicht auf k-Anonymität basiert, stellten Xiao und Tao [161] vor. Dabei dürfen die Individuen, zu denen Tupel in einer Datentabelle gehören, selbst ein Maß für den Schutz ihrer Privatsphäre wählen. Die Voraussetzung dazu ist, dass für das sensible Attribut eine Taxonomie existiert, die es ermöglicht, sensible Werte analog zu den quasi-identifizierenden Werten zu generalisieren. Individuen können einen Knoten innerhalb dieser Taxonomie auswählen, der ihr gewünschtes Maß an Schutz angibt. Die Privatsphäre ist verletzt, wenn es Angreifern möglich ist, einem Individuum einen sensiblen Wert unterhalb des gewählten Knotens mit einer bestimmten Wahrscheinlichkeit zuzuordnen.

Beispiel 2.8 In einer möglichen Taxonomie von Krankheiten seien Asthma und Bronchitis Kindknoten von Atemwegserkrankung.[18] Für Alina aus Tabelle 2.1 ist es damit möglich, Atemwegserkrankung als ihren Schutzknoten anzugeben. Somit darf ein Angreifer aus einer Veröffentlichung von Tabelle 2.1 zwar schlussfolgern, dass Alina eine, aber nicht welche Atemwegserkrankung hat. Das kann erreicht werden, indem ihr sensibler Wert Asthma zu Atemwegserkrankung generalisiert wird oder ihr entsprechendes Tupel zu einer QI-Gruppe hinzugefügt wird, in der weitere Krankheiten vorkommen, die keine Atemwegserkrankungen sind.

Andererseits ist es für Alina auch möglich, veröffentlichen zu lassen, dass sie den sensiblen Wert Asthma hat. Sofern sie ihre Privatsphäre dadurch als nicht verletzt ansieht, wählt sie einfach keinen Schutzknoten aus.

Der Vorteil dieser sogenannten *personalisierten Anonymität* besteht darin, dass jedes Individuum das Maß an Schutz seiner Privatsphäre selbst angeben kann. In der Praxis kann

[17] Ein deutscher Begriff für Confidence-Bounding ist nicht geläufig.

[18] Vgl. Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme (ICD-10-GM), Kapitel X [33].

sich dies aber auch leicht als Nachteil erweisen. Um geeignete Schutzknoten auszuwählen, müssen nämlich Individuen Kenntnisse über andere sensible Werte in der Datentabelle haben, was im Allgemeinen nicht der Fall ist.

t-closeness

Erhält ein Angreifer veröffentlichte Daten, hat er in der Regel einen Informationsgewinn. Beispielsweise kann er zwar vermuten, dass Doris krank ist, aber erst durch die Kenntnis von Tabelle 2.9 kann er schlussfolgern, dass sie Grippe oder Diabetes hat. Ohne weiteres Hintergrundwissen kann er sogar mit einer Wahrscheinlichkeit von $\frac{2}{3}$ annehmen, dass sie Grippe hat, da dieser Wert zweimal in der QI-Gruppe von Doris vorkommt.

Li et al. [93] untersuchten den Informationsgewinn durch veröffentlichte Daten. Sie setzten dabei voraus, dass ein Angreifer jeweils die Verteilung aller sensiblen Werte in einer Tabelle kennt, und beschränkten den Informationsgewinn, der durch die Identifizierung einzelner QI-Gruppen entsteht. In Tabelle 2.9 kommt unter allen neun Tupeln der Wert Grippe zweimal vor. Ein Angreifer, der nur dieses Wissen hat, würde die Wahrscheinlichkeit, dass eine beliebige Person Grippe hat, mit $\frac{2}{9}$ abschätzen. Kann der Angreifer einzelne QI-Gruppen identifizieren, wächst diese Wahrscheinlichkeit auf $\frac{2}{3}$ für alle Personen der zweiten QI-Gruppe oder fällt auf 0 für alle anderen Personen. Um diesen Informationsgewinn zu beschränken, forderten Li et al., dass die Verteilung der sensiblen Werte in jeder QI-Gruppe ähnlich zu der Verteilung in der gesamten Tabelle ist. Um Verteilungen miteinander zu vergleichen, wählten sie die Earth Mover Distance [121] und legten fest, dass sich Verteilungen nur um einen maximalen Wert von t unterscheiden dürfen. Dieses Prinzip wird *t-closeness*[19] genannt.

Ein Vorteil dieses Modells ist, dass die Privatsphäre sehr gut geschützt wird und Angriffe wie zum Beispiel die Schiefeattacke verhindert werden. Für Auswertungen der veröffentlichten Daten entstehen aber auch große Nachteile, denn Korrelationen zwischen Quasi-Identifikatoren und dem sensiblen Attribut werden zerstört. Der Nutzen der veröffentlichten Tabelle für statistische Zwecke ist daher stark beschränkt.

(α_i, β_i)-closeness

Als Kombination von (α, k)-Anonymität und t-closeness stellten Frikken und Zhang [53] *(α_i, β_i)-closeness* vor. Dabei kann für jeden sensiblen Wert s_i ein Intervall $[\alpha_i, \beta_i]$ angegeben werden, das die relative Häufigkeit von s_i in jeder QI-Gruppe beschränkt. Eine Tabelle erfüllt (α_i, β_i)-closeness, wenn in jeder QI-Gruppe die relative Häufigkeit jedes sensiblen Wertes s_i mindestens α_i und höchstens β_i beträgt. (α_i, β_i)-closeness hat den Vorteil, dass die Werte für α_i und β_i wesentlich intuitiver sind als die Schranke t für t-closeness. Dahingegen erlangt ein Angreifer bei diesem Modell und ungenauer Wahl von α_i und β_i mehr Wissenszuwachs.

Beispiel 2.9 Der sensible Wert Diabetes kommt in den ersten beiden QI-Gruppen von Tabelle 2.9 je einmal und in der dritten zweimal vor. Damit die Tabelle (α_i, β_i)-closeness erfüllt, sind die Werte $\alpha_{\text{Diabetes}} = \frac{1}{3}$ und $\beta_{\text{Diabetes}} = \frac{2}{3}$ möglich. Da Asthma in zwei Gruppen gar nicht und in einer einmal vorkommt, kann $\alpha_{\text{Asthma}} = 0$ und $\beta_{\text{Asthma}} = \frac{1}{3}$ gewählt werden.

[19] Ein deutscher Begriff für t-closeness ist nicht geläufig.

Quadrierte Fehlerdiversität

Für sensible numerische Attributwerte existieren weitere Angriffe. Wenn beispielsweise in einer QI-Gruppe zwar viele verschiedene sensible Werte vorkommen, diese aber in einem nur sehr kleinen Intervall liegen, kann ein Angreifer zumindest auf die Größenordnung der Werte schließen.[20] Um das zu verhindern, führten LeFevre et al. [91] die *quadrierte Fehlerdiversität* ein. Seien $\bar{s}(q)$ der Mittelwert der sensiblen Werte in der QI-Gruppe q und S_q die Menge aller sensiblen Werte, die in q vorkommen. Dann erfüllt eine Tabelle quadrierte Fehlerdiversität, wenn für jede QI-Gruppe q $\sum_{s \in S_q} (s - \bar{s}(q))^2 \geq error$ gilt, wobei *error* eine vorgegebene Konstante ist. Die Privatsphäre ist demnach ausreichend geschützt, wenn in jeder Gruppe Werte vorkommen, die sich deutlich vom Mittelwert unterscheiden. Ein Nachteil dieser Definition ist, dass selbst ein hoher Wert für *error* nur eindeutige 2-Diversität sicherstellt.

	Alter	PLZ	Geschlecht	Gehalt
Alina	20–21	101**	*	2490
Bill	20–21	101**	*	2500
Clive	20–21	101**	*	2510
Doris	23–24	214**	*	2000
Emily	23–24	214**	*	2200
Frank	23–24	214**	*	9000
Gill	24–27	2****	*	2000
Harry	24–27	2****	*	5000
Isabel	24–27	2****	*	8000

Tabelle 2.10: Anonymisierte Daten T^{***}

Beispiel 2.10 Tabelle 2.10 enthält einige Daten von neun Personen, insbesondere Werte des sensiblen Attributs `Gehalt`. Der Mittelwert der ersten QI-Gruppe ist 2500, der der zweiten 4400 und der der dritten 5000. Als Fehlertoleranzen für quadrierte Fehlerdiversität ergeben sich 200 für die erste, 31,76 Mio für die zweite und 18 Mio für die dritte QI-Gruppe. Die Tabelle erfüllt demnach quadrierte Fehlerdiversität mit einem Fehlerwert von $error = 200$.

(k, e)-Anonymität

Ein weiteres Anonymitätsprinzip für numerische Werte wurde von Zhang et al. [166] eingeführt. Sie forderten, dass das Intervall, aus dem sensible Werte stammen, eine gewisse Größe haben soll. Eine Tabelle erfüllt demnach (k, e)-*Anonymität*, wenn jede QI-Gruppe mindestens k verschiedene sensible Werte hat und der Bereich, aus dem diese Werte kommen, nicht kleiner als e ist. Jede (k, e)-anonyme Tabelle erfüllt damit automatisch eindeutige ℓ-Diversität mit $\ell = k$. Nachteilig an dieser Definition ist der Umstand, dass die Verteilung der sensiblen Werte ignoriert wird und das Modell damit nicht vor der Näheattacke schützt.

Beispiel 2.11 Jede QI-Gruppe in Tabelle 2.10 enthält genau drei verschiedene Werte für das sensible Attribut `Gehalt`. Die Differenz zwischen dem größten und kleinsten Wert

[20] Vgl. Näheattacke.

beträgt in der ersten QI-Gruppe 20, in der zweiten 7000 und in der dritten 6000. Demnach erfüllt die Tabelle $(3, 20)$-Anonymität.

(ε, m)-Anonymität

Li et al. [92] schlugen ein alternatives Modell zum Schutz der Privatsphäre vor, das sie (ε, m)-*Anonymität* nannten. Das Ziel dabei ist, in jeder QI-Gruppe die Anzahl der sensiblen Werte, die „ähnlich" zu einem vorkommenden Wert sind, zu beschränken. Dafür wird ein Parameter ε eingeführt und zwei Interpretationen dieser Ähnlichkeit angegeben. Bei *absoluter* (ε, m)-*Anonymität* sind zwei Werte x und y ähnlich, wenn ihre absolute Differenz höchstens ε ist, das heißt wenn $|y - x| \leq \varepsilon$ gilt. Bei *relativer* (ε, m)-*Anonymität* wird der Abstand von y zu x mit $\varepsilon \cdot x$ verglichen, das heißt, y ist ähnlich zu x, wenn $|y - x| \leq \varepsilon \cdot x$ gilt. Eine Tabelle erfüllt (ε, m)-Anonymität, wenn für jedes Tupel t in einer QI-Gruppe q das Verhältnis von sensiblen Werten in q, die ähnlich zum sensiblen Wert in t sind, und der Größe von q höchstens $\frac{1}{m}$ beträgt. Sowohl absolute als auch relative (ε, m)-Anonymität haben die Eigenschaft, dass ein höherer Schutz mit einem höheren Wert für ε oder m erreicht wird.

Beispiel 2.12 Tabelle 2.10 enthält drei verschiedene QI-Gruppen jeweils der Größe 3. Während die sensiblen Werte in der ersten Gruppe sehr ähnlich sind, sind sie in der zweiten und dritten zum Teil sehr verschieden. Sei absolute (ε, m)-Anonymität das Modell für den Schutz der Privatsphäre. Für $\varepsilon \geq 10$ sind die sensiblen Werte von Alina (2490) und Clive (2510) jeweils ähnlich zum Wert von Bill (2500). (ε, m)-Anonymität gilt demnach nur für $m = 1$. Für $\varepsilon < 10$ ist jeder der drei Werte nur zu sich selbst ähnlich und es folgt (ε, m)-Anonymität mit $m = 3$. Ein analoges Ergebnis folgt für die dritte QI-Gruppe und $\varepsilon = 3000$. In der zweiten Gruppe sind die sensiblen Werte von Doris (2000) und Emily (2200) ab $\varepsilon \geq 200$ ähnlich zueinander. In diesem Fall gilt (ε, m)-Anonymität mit $m = \frac{3}{2}$.

2.3.3 Tabellenpreisgabe

Es gibt Fälle, bei denen die Privatsphäre bereits dann verletzt ist, wenn ein Angreifer schlussfolgern kann, dass irgendein Tupel in der veröffentlichten Tabelle zu einem bestimmten Individuum gehört. Enthält beispielsweise die Tabelle nur bestimmte Krankheiten, kann die Kenntnis vom Vorkommen eines Tupels bereits die Privatsphäre des dazugehörigen Individuums verletzen. Eine Tabellenpreisgabe entsteht, wenn ein Angreifer die Präsenz oder Abwesenheit des Tupels von einem bestimmten Individuum in der veröffentlichten Tabelle herausfinden kann.

δ-Präsenz

Um eine Tabellenpreisgabe zu verhindern, schlugen Nergiz et al. [112] δ-*Präsenz* vor. Seien T die private Tabelle, die veröffentlicht werden soll, und E eine externe öffentliche Tabelle mit $T \subseteq E$. Eine generalisierte Tabelle T^* erfüllt $(\delta_{\min}, \delta_{\max})$-Präsenz, wenn $\delta_{\min} \leq P(t \in T \mid T^*) \leq \delta_{\max}$ für alle $t \in E$ gilt. Somit ist die Wahrscheinlichkeit, dass ein Tupel aus T zu einem Individuum aus E gehört, auf das Intervall $\delta = [\delta_{\min}, \delta_{\max}]$ beschränkt.

Indirekt wird dadurch auch vor Identitäts- und Attributpreisgabe geschützt, denn ein Angreifer, der nur mit einer Wahrscheinlichkeit von δ weiß, dass irgendein Tupel zu einem Individuum gehört, kann das tatsächliche Tupel oder den tatsächlichen sensiblen Wert

auch nur mit einer Wahrscheinlichkeit von δ herausfinden. Ein Nachteil von δ-Präsenz ist allerdings die Annahme, dass sowohl derjenige, der die Daten veröffentlicht, als auch der Angreifer Zugriff auf dieselbe externe Tabelle E haben. In der Praxis ist dieser Fall oft so nicht gegeben.

Beispiel 2.13 Seien Jake ein 27 Jahre alter Mann, der die Postleitzahl 29025 hat, und E eine öffentliche Tabelle, die das Alter, die Postleitzahl und das Geschlecht von Gill, Harry, Isabel und Jake enthält. Ein Angreifer kann nun für jede der vier Personen mit $\frac{3}{4}$ Wahrscheinlichkeit schlussfolgern, dass ein Tupel in der letzten QI-Gruppe von Tabelle 2.9 zu dieser Person gehört.

Differentielle Privatsphäre

Die *differentielle Privatsphäre* (engl. *differential privacy* [41]) legt ihren Fokus darauf, dass das Hinzufügen oder Entfernen eines Tupels in einer Tabelle jede mögliche Art von Analyse der Daten nicht substantiell beeinflussen soll. Das Risiko einer Verletzung der Privatsphäre soll für ein Individuum nicht entscheidend zunehmen, wenn ein dazugehöriges Tupel in der Tabelle vorkommt. Dadurch wird ein sehr starker Schutz der Privatsphäre kreiert, denn Angreifer können nur schwer herausfinden, ob überhaupt ein Tupel eines Individuums in den Daten vorhanden ist.

Sei A ein randomisierter Algorithmus, der auf die Daten angewandt wird, wenn Informationen herausgegeben werden. Mit A könnte beispielsweise das Anonymisieren und Veröffentlichen von Mikrodaten beschrieben werden. Weiterhin seien D_1 und D_2 zwei Datentabellen, die sich in höchstens einem Tupel unterscheiden. Dann erfüllt A ε-differentielle Privatsphäre, wenn für jede Teilmenge S aller möglichen Ausgaben von A

$$P(A(D_1) \in S) \leq \mathrm{e}^{\varepsilon} \cdot P(A(D_2) \in S)$$

gilt. Mit anderen Worten ist die Wahrscheinlichkeit, dass der Algorithmus bei Eingabe D_1 eine Ausgabe aus S erzeugt, nur höchstens um den Term e^{ε} verschieden von der gleichen Wahrscheinlichkeit, wenn D_2 eingegeben wird. Kleine Werte für ε erzeugen einen höheren Schutz als große Werte. Dwork empfiehlt selbst, Werte in der Größenordnung von 0,01 bis 0,1 zu verwenden [43].

Differentielle Privatsphäre trifft keine Annahmen über den Angreifer und ist unabhängig von zusätzlichen Informationen. Sie kann sowohl für das nicht-interaktive Szenario (d. h. das einmalige Veröffentlichen sensibler Daten) als auch für das interaktive Szenario (d. h. die Auswertung mehrerer Anfragen auf sensible Daten) verwendet werden [142]. Die Techniken, die ε-differentielle Privatsphäre garantieren sollen, fügen den Daten in der Regel Rauschen hinzu. Dabei werden neue Einträge generiert, die zusätzlich zu den tatsächlich vorkommenden Werten ausgegeben werden.

Weitere Details zur differentiellen Privatsphäre werden bei Dwork [42] ausgeführt.

2.3.4 Weitere Modelle

Neben den hier aufgelisteten Prinzipien zum Schutz der Privatsphäre existieren noch weitere, die meist kleinere Erweiterungen der vorgestellten Ansätze sind. Beispiele dafür sind (k, p, q, r)-*Anonymität* [38], l^+-*Diversität* [95], (n, t)-*closeness* [94] oder β-*likeness* [18].

Ersteres ist eine Mischung aus p-sensibler k-Anonymität und einem Ansatz, der relative Häufigkeiten sensibler Werte in QI-Gruppen mithilfe ihrer Varianz beschränkt. l^+-Diversität entspricht in etwa (α_i, β_i)-closeness mit $\alpha_i = 0$. Das bedeutet, die relativen Häufigkeiten von sensiblen Werten werden nur nach oben beschränkt. (n, t)-closeness ist wiederum eine Erweiterung von t-closeness. Dabei wird der Unterschied zwischen der Verteilung der sensiblen Werte in einer einzelnen QI-Gruppe Q nicht zur Gesamtverteilung gemessen, sondern zu einer „natürlichen Obermenge“ von Q, die mindestens die Größe n hat. Solche Obermengen sind dadurch gekennzeichnet, dass sie aus Q durch weitere Generalisierungen von Quasi-Identifikatoren entstehen. Das zuletzt genannte Beispiel beschränkt für jeden sensiblen Wert den Informationsgewinn, den ein Angreifer durch die Kenntnis der veröffentlichten Tabelle erhält. Hierfür wird gefordert, dass die Distanz zwischen der relativen Häufigkeit eines sensiblen Wertes in einer QI-Gruppe und der relativen Häufigkeit in der gesamten Datentabelle nicht größer als ein fest vorgegebener Wert ist. Im Unterschied zu anderen Modellen wird bei β-likeness diese Distanz jedoch nicht absolut, sondern ebenfalls relativ zu den entsprechenden Häufigkeiten gemessen.

Eine andere Klasse von Modellen bezieht Hintergrundwissen des Angreifers ein. Dabei werden neben den Kenntnissen aus der veröffentlichten Tabelle weitere Zusammenhänge modelliert, die eventuell zu einer Verletzung der Privatsphäre führen könnten. Beispielsweise könnte ein Angreifer wissen, dass ein bestimmtes Individuum einen sensiblen Wert garantiert nicht oder mit einer gegebenen Wahrscheinlichkeit hat. Beispiele dafür sind *m-confidentiality* [158], (c, k)-*safety* [104], *skyline privacy* [20], (τ, λ)-*uniqueness* [149] und *Privacy-MaxEnt* [39].

2.4 Anonymisierungsalgorithmen

In der Literatur sind das k-Anonymitäts- und ℓ-Diversitätsprinzip die bekanntesten Methoden, um die Privatsphäre beim Veröffentlichen sensibler Daten zu schützen. Für beide gibt es eine Vielzahl von Algorithmen, deren Ziel es ist, *optimal* anonymisierte Tabellen zu erstellen. Diese besitzen unter allen Tabellen, die ein gefordertes Schutzkriterium erfüllen, den größten Informationsgehalt.

In den letzten Jahren wurden viele Metriken vorgeschlagen, um den Informationsgehalt von Daten zu bestimmen [123, 134, 70, 7, 56, 90, 101, 161, 14, 158, 51, 61, 16]. Es wurde auch gezeigt, dass das *k-Anonymitätsproblem*, also das Bestimmen einer optimalen k-anonymen Tabelle, NP-hart ist [106, 2]. Neben theoretisch fundierten Approximationsalgorithmen mit beweisbarer Güte [106, 2, 116] existiert eine Vielzahl eher praktisch veranlagter Heuristiken [56, 90, 60, 69], die ein Optimum näherungsweise bestimmen. Andere Algorithmen führen starke Restriktionen ein, wodurch ein Optimum in Polynomialzeit gefunden werden kann [123, 134, 7, 89]. Da in dieser Arbeit vielmehr die Konzepte des Schutzes der Privatsphäre und weniger die Algorithmen zum Anonymisieren im Vordergrund stehen, werden an dieser Stelle nur zwei Verfahren kurz vorgestellt.

Datafly [135] ist ein Beispiel für einen praktisch nutzbaren Algorithmus, der eine k-anonyme Tabelle erstellt. Von gegebenen Mikrodaten anonymisiert er mithilfe von Generalisierungshierarchien jeweils das Attribut, welches die meisten verschiedenen Werte enthält. In Tabelle 2.1 wäre das zum Beispiel `Alter` oder `PLZ`, denn beide Attribute enthalten jeweils sechs verschiedene Werte. Der Generalisierungsschritt wird iterativ mit dem Attribut fortgesetzt, das nun die meisten verschiedenen Werte enthält. Der Algorithmus

terminiert, wenn die Anzahl der Tupel, die noch nicht in einer QI-Gruppe der Größe k enthalten sind, auf weniger als k gesunken ist. Im letzten Schritt werden die Werte der verbliebenen und einiger anderer Tupel komplett unterdrückt, sodass aus ihnen ebenfalls eine QI-Gruppe der Größe mindestens k entsteht.

Incognito [89] testet alle möglichen Generalisierungen der Attribute nach einem speziellen System. Um die Komplexität zu beschränken, werden nur Domänen-Vollgeneralisierungen entsprechend gegebener Hierarchien betrachtet. Die Besonderheit ist, dass alle Lösungen berechnet werden, die ein gegebenes Schutzkriterium erfüllen. Einerseits ist es dadurch leicht möglich, die optimale Anonymisierung zu bestimmen. Andererseits können neben k-Anonymität auch andere Prinzipien wie ℓ-Diversität [101] oder t-closeness [93] getestet werden.

2.5 Dynamische Veröffentlichung von Daten

Die bisher vorgestellten Modelle legen ihren Fokus auf einmalige Publikationen. Ein anderes Szenario liegt vor, wenn sensible Daten mehrfach hintereinander veröffentlicht werden sollen. Dabei ist es möglich, dass zwischen den einzelnen Publikationen Tupel entfernt oder hinzugefügt wurden.

	Alter	PLZ	Geschlecht	Krankheit
Jill	20–21	101**	*	Erkältung
Ken	20–21	101**	*	Fieber
Clive	20–21	101**	*	Diabetes
Doris	23–24	214**	*	Grippe
Emily	23–24	214**	*	Diabetes
Liam	23–24	214**	*	Grippe
Mike	24–27	2****	*	Erkältung
Harry	24–27	2****	*	Diabetes
Isabel	24–27	2****	*	Diabetes

Tabelle 2.11: Anonymisierte Daten T_2^{**}

Seien erneut die Mikrodaten aus Tabelle 2.1 gegeben, die zu einem Zeitpunkt t_1 wie in Tabelle 2.9 veröffentlicht werden. Zu einem späteren Zeitpunkt t_2 sollen die Mikrodaten erneut publiziert werden. In der Zwischenzeit wurden die Tupel von Alina, Bill, Frank und Gill gelöscht sowie Tupel für Jill, Ken, Liam und Mike hinzugefügt. Tabelle 2.11 stellt die zweite Veröffentlichung T_2^{**} dar. Offensichtlich erfüllen beide Tabellen eindeutige 2-Diversität, sodass sie einzeln vor Attributpreisgabe schützen. Ein Angreifer kann jedoch durch geschicktes Kombinieren sensible Werte schlussfolgern. Beispielsweise weiß er aus T^{**}, dass ein Tupel von Clive in der ersten QI-Gruppe vorkommt und dieser somit Asthma, Bronchitis oder Diabetes haben muss. Analog folgt aus T_2^{**}, dass Clive Erkältung, Fieber oder Diabetes hat. Da Diabetes der einzige Wert ist, der in beiden Aufzählungen vorkommt, muss es die gesuchte Krankheit von Clive sein. Für die zweite Tabelle T_2^{**} gilt insgesamt, dass die sensiblen Werte Asthma und Bronchitis nirgends vorkommen. Daraus folgt, dass ein Angreifer unabhängig von der Generalisierung immer diese beiden Werte ausschließen kann.[21]

[21] Xiao und Tao [162] nannten dieses Phänomen *kritische Abwesenheit* (engl. *critical absence*).

Für das mehrfache Veröffentlichen von Tabellen schlugen Xiao und Tao [162] ein Modell mit dem Namen *m-Invarianz* (engl. *m-invariance*) vor. Im Gegensatz zu anderen Ansätzen auf diesem Gebiet [15, 117, 55] können zwischen zwei Publikationen Tupel nicht nur hinzugefügt, sondern auch entfernt werden. Eine Folge von Tabellen erfüllt m-Invarianz, wenn drei Eigenschaften gelten. Erstens muss jede vorkommende QI-Gruppe aus genau m Tupeln bestehen. Als Zweites darf in keiner QI-Gruppe ein sensibler Wert mehrfach vorkommen. Die dritte Eigenschaft betrifft Tupel, die in mehreren Tabellen vorkommen. Für sie muss gelten, dass die Menge aller sensiblen Werte, die in der QI-Gruppe vorkommen, in der auch das Tupel vorkommt, in allen Tabellen identisch ist.

Beispiel 2.14 Die zweite und dritte QI-Gruppe von Tabelle 2.11 verletzen m-Invarianz, da jeweils sensible Werte mehrfach vorkommen. Die erste Gruppe enthält zwar genau drei verschiedene Werte, für Clive ist aber das letzte Kriterium nicht erfüllt. Während die sensiblen Werte seines Tupels der QI-Gruppe in T_2^{**} Erkältung, Fieber und Diabetes sind, kommt sein Tupel in T^{**} zusammen mit Asthma, Bronchitis und Diabetes vor.

Xiao und Tao zeigten, dass in m-invarianten Tabellen ein Angreifer immer genau m potentielle sensible Werte mit einem Individuum verknüpfen kann. Der tatsächliche Wert kann aber nicht herausgefunden werden. Sie führten einen Algorithmus ein, der Tabellen so generalisiert, dass diese m-Invarianz erfüllen. Im oben beschriebenen Fall, in dem bestimmte sensible Werte nicht vorhanden sind, werden zusätzliche künstliche Werte[22] in die veröffentlichte Tabelle eingefügt. Somit kann in den herausgegebenen Daten immer auch ein Rauschen vorhanden sein.

2.6 Anfrageauditierung

Dicht verwandt, aber doch orthogonal zu PPDP ist die Frage, ob die Beantwortung mehrerer Datenbankanfragen zu einer Verletzung der Privatsphäre führen kann. Dieses Problem wird in der Literatur allgemein mit *Anfrageauditierung*[23] bezeichnet. Es wird eine proaktive Online- von einer retroaktiven Offlineauditierung unterschieden.

Mithilfe der *Onlineauditierung* soll entschieden werden, ob das Ergebnis einer gegebenen Anfrage ein Privatsphärenkriterium verletzt. Dabei werden Informationen aus bereits gestellten und beantworteten Anfragen mit berücksichtigt. Es ist möglich, die Antwort zu verweigern, wenn ihre Herausgabe zu einem Bruch der Privatsphäre führen würde. Andernfalls wird das korrekte Ergebnis ausgegeben.

Bei der *Offlineauditierung* stellen Nutzer ihre Anfragen und erhalten die entsprechenden korrekten Ergebnisse. Erst später wird überprüft, ob etwas „Verbotenes“ herausgegeben wurde. Brüche der Privatsphäre werden mit diesem Ansatz erst nach der Verletzung festgestellt. Im Unterschied zum Online- soll beim Offlineauditieren nur die Einhaltung des Schutzkriteriums überprüft und nicht verhindert werden, dass ein Angreifer sensible Daten erhält.

Weitere Unterscheidungsmerkmale der Ansätze zur Anfrageauditierung sind die Typen der zugelassenen Anfragen und die verwendeten Schutzmodelle.

[22] Xiao und Tao [162] bezeichneten diese Werte als *Fälschungen* (engl. *counterfeits*).
[23] Im Datenbankumfeld versteht man unter *Auditieren* das Aufzeichnen von Benutzeraktivitäten.

2.6.1 Statistische Anfragen

Der Großteil der Arbeiten betrachtet sogenannte statistische Anfragen. Diese berechnen eine Aggregation numerischer Werte wie Summen- oder Durchschnittsbildung. Erste Arbeiten stammen von Schlörer [125], Denning et al. [29, 30], Dobkins et al. [37] sowie Reiss [120] und zeigen, wie sensible Werte auf verschiedenste Weise durch gezielte Anfragen herausgefunden werden können.

Name	Alter	PLZ	Geschlecht	Gehalt
Emily	24	21410	w	2200
Frank	24	21410	m	9000
Gill	24	21421	w	2000

Tabelle 2.12: Mikrodaten

Seien die Daten aus Tabelle 2.12 gegeben, wobei die Werte der Spalte `Gehalt` nicht herausgegeben werden dürfen, aggregierte Anfragen jedoch zugelassen sind. Ein Angreifer kann nun mittels weniger Anfragen sämtliche sensible Werte (d. h. Gehälter) schlussfolgern. Dazu fragt er zuerst nach der Anzahl der Individuen, die 24 Jahre alt sind, und danach nach denen, die zusätzlich noch weiblich sind. Er stellt die Anfragen

Q_1: `SELECT COUNT(Gehalt) FROM T WHERE Alter = 24`,
Q_2: `SELECT COUNT(Gehalt) FROM T WHERE Alter = 24 AND Geschlecht = 'w'`

und erhält als Antworten die Zahlen 3 und 2. Daraus kann er folgern, dass es nur ein Individuum gibt (Frank), das 24 Jahre alt und männlich ist. Stellt der Angreifer die gleichen Anfragen mit der Summenfunktion, kann er leicht das Gehalt des männlichen Individuums herausfinden.

Q_3: `SELECT SUM(Gehalt) FROM T WHERE Alter = 24`
Q_4: `SELECT SUM(Gehalt) FROM T WHERE Alter = 24 AND Geschlecht = 'w'`

Die Antwort auf Q_3 ist 13200 und Q_4 hat das Ergebnis 4200. Die Differenz der beiden Ergebnisse $13200 - 4200 = 9000$ ist der sensible Wert von Frank. Die Werte von Emily und Gill kann der Angreifer auf analoge Weise herausfinden, indem er zum Beispiel anstatt des Geschlechts die Postleitzahl 21410 selektiert.

Als Modell für den Schutz der Privatsphäre ist im obigen Beispiel *volle Preisgabe* (engl. *full disclosure*) [110] verwendet worden. In diesem Fall liegt eine Verletzung nur dann vor, wenn der Angreifer einem Individuum dessen sensiblen Wert exakt zuordnen kann. Demgegenüber steht der Begriff der *partiellen Preisgabe* (engl. *partial disclosure*) [110]. Hier liegt eine Verletzung bereits vor, wenn sich die Vermutung des Angreifers, in welchem Bereich ein bestimmter sensibler Wert liegt, nach dem Erhalt des Ergebnisses zu einer Anfrage signifikant ändert.

Sind als Aggregatfunktion nur Summe oder nur Maximum zugelassen, existieren polynomielle Algorithmen, die volle Preisgabe kontrollieren [22]. Sind hingegen beide Funktionen zusammen erlaubt, ist das Problem NP-hart [21]. Neben numerischen Attributwerten gibt es auch Untersuchungen, bei denen das sensible Attribut Boolesche Werte (d. h. 0 oder 1) annehmen kann. Interessanterweise ist die Auditierung co-NP-hart, wenn als Anfragen nur Summenbildungen über 0/1-Variablen zugelassen sind [83].

Ein Problem dieser Schutzmodelle bei der Onlineauditierung ist, dass bereits durch das Verweigern einer Anfrage sensible Informationen herausgegeben werden können. Seien Q_1'

und Q'_2 zwei Anfragen, die nach der Summe beziehungsweise dem Maximum der reellen Zahlen x_1, x_2 und x_3 fragen. Diese Variablen könnten beispielsweise erneut Gehaltsangaben dreier Individuen darstellen.

$$Q'_1\colon\ \texttt{SUM}(x_1, x_2, x_3)$$
$$Q'_2\colon\ \texttt{MAX}(x_1, x_2, x_3)$$

Sei ein Angreifer gegeben, der die Anfrage Q'_1 stellt und als Ergebnis 15 erhält. Danach stellt er die Anfrage Q'_2, deren Antwort vom System verweigert wird. Nun kann er schlussfolgern, dass das Herausgeben des Ergebnisses zu Q'_2 zu einer Verletzung der Privatsphäre geführt hätte. Wenn das Schutzkriterium darin besteht, die konkreten Werte der Variablen zu verheimlichen, kann der Angreifer Folgendes ableiten: Das Maximum der drei Zahlen kann nicht kleiner als 5 sein, da sonst die Summe nicht 15 wäre. Ist es größer als 5, kann dieser Wert nicht eindeutig einer der Variablen zugeordnet werden und die Privatsphäre wäre ausreichend geschützt. Damit eine Verletzung entsteht, muss daher $x_1 = x_2 = x_3 = 5$ gelten. In diesem Fall sind trotz der Verweigerung eines Ergebnisses sensible Werte preisgegeben worden. Aus dieser Betrachtung folgt, dass Algorithmen für die Offline- nicht automatisch auch für die Onlineauditierung verwendet werden können.

Als Lösung für das aufgezeigte Problem schlugen Kenthapadi et al. [80] ein Modell mit dem Namen *simulierbare Auditierung* (engl. *simulatable auditing*) vor, welches das Ziel verfolgt, Anfragen unabhängig von den angefragten sensiblen Werten zu verweigern. Ob überhaupt das korrekte Ergebnis einer Anfrage ausgegeben oder die Anfrage verweigert wird, soll vom DBMS wie auch vom Angreifer gleichermaßen entschieden werden können. Durch das Verweigern einer Anfrage hat der Angreifer somit keinen zusätzlichen Wissenszuwachs. Für das obige Beispiel folgt, dass die Anfrage Q'_2 in jedem Fall nicht beantwortet wird, da es Fälle gibt, wo Q'_2 beantwortet werden kann, und Fälle, wo das nicht möglich ist.

2.6.2 Selektions- und Projektionsanfragen

Für Anfragen, die Selektions-, Projektions- oder auch Joinoperationen beinhalten, existieren andere Modelle. Die zu schützenden Informationen werden meist als Sicht (View) in der Datenbank modelliert, die durch eine Anfrage spezifiziert werden kann. Beispielsweise können die sensiblen Werte aus Tabelle 2.12 mit einer View angegeben werden, die alle Tupel auf die Attribute `Name` und `Gehalt` projiziert. Das Schützen dieser sogenannten geheimen View bedeutet, dass Angreifer den Individuen (bzw. den Namen) ihre Gehälter nicht eindeutig zuordnen können. Ergebnisse von Anfragen werden ebenfalls als Views modelliert.

Modelle für den Schutz der Privatsphäre bei Selektions-, Projektions- oder Joinanfragen unterscheiden sich stark von denen beim Veröffentlichen von Daten. Ein immer noch weit verbreitetes Prinzip wurde bereits 1949 von Shannon [129] unter dem Namen *perfekte Geheimhaltung* (engl. *perfect secrecy*) vorgestellt. Shannon repräsentierte das Wissen des Angreifers über bestimmte Werte durch eine Wahrscheinlichkeitsverteilung. Perfekte Geheimhaltung liegt genau dann vor, wenn die Wahrscheinlichkeitsverteilung des Angreifers, bevor er (z. B. anonyme) Daten erhält, identisch ist mit der Verteilung, nachdem er die Daten erhalten hat.

Diese generelle Idee wurde später auf viele Probleme angepasst und erweitert. Im Rahmen der Anfrageauditierung wird das entsprechende Modell mit *perfekter Privatsphäre*

(engl. *perfect privacy*) [100] bezeichnet. Im Allgemeinen ist das Testen, ob durch eine (konjunktive) Anfrage Informationen preisgegeben werden, Π_2^P-vollständig, was im Wesentlichen schwerer als co-NP bedeutet [108]. Von Miklau and Suciu [108] sowie Deutsch und Papakonstantinou [31] existieren Ansätze, die die Informationspreisgabe bestimmter Views in Bezug auf die geheime View beschränken. Evfimievski et al. [47] stellten ein Verfahren vor, das Veränderungen der Wahrscheinlichkeiten nur nach oben beschränkt, deren Verminderung aber zulässt.

Das Kriterium der perfekten Geheimhaltung ist in vielen Fällen zu streng für den Schutz der Privatsphäre, da es nur wenige Anfragen zulässt, die Daten aber zum Teil zu stark schützt. Die Modelle des PPDP, die auf k-Anonymität aufbauen, lockern diese Restriktion, da sie in der Regel eine Änderung der Wahrscheinlichkeitsverteilung des Angreifers zulassen.

Teil II

Anfragebearbeitung

3 Modellierung des Angreiferwissens mithilfe valider Matchings

In diesem Kapitel wird die zentrale Problemstellung und Zielsetzung der vorliegenden Arbeit eingeführt (Kapitel 3.1). Anhand eines einfachen Beispiels wird erläutert, was unter dem Schutz der Privatsphäre bei der Anfragebearbeitung zu verstehen ist (Kapitel 3.2). Es wird ein Modell entwickelt, das Individuen ein bestimmtes Maß an Anonymität garantiert (Kapitel 3.3). Um es als Schutzkriterium zu verwenden, werden Ergebnisse von Anfragen als Graphen repräsentiert (Kapitel 3.4), in denen Matchings gesucht werden (Kapitel 3.5).

Jedoch stellt sich heraus, dass das entstehende Matchingproblem NP-vollständig ist (Kapitel 3.6), woraus auch folgt, dass die Einhaltung des Schutzkriteriums nicht in Polynomialzeit überprüft werden kann.[1] Da bekannte Matchingalgorithmen mit den erstellten Graphen nicht korrekt funktionieren (Kapitel 3.7), wird ein Ansatz präsentiert, der auf ganzzahliger linearer Programmierung beruht (Kapitel 3.8). Mit seiner Hilfe kann ein Verfahren angegeben werden, das den Schutz der Privatsphäre bei der Anfragebearbeitung sicherstellt (Kapitel 3.9). Am Ende dieses Kapitels werden kurz alternative Darstellungsformen besprochen (Kapitel 3.10).

3.1 Zielsetzung und Annahmen

Das Ziel der vorliegenden Arbeit liegt in der Verbindung der Konzepte zum Schutz der Privatsphäre beim Veröffentlichen von Mikrodaten und den Methoden der Anfrageauditierung. Dabei sollen eine Reihe von Schwachstellen bisheriger Ansätze vermieden werden. Diese beziehen sich im Wesentlichen auf die verwendeten Schutzmodelle und Typen von Anfragen.

Viele bisherigen Arbeiten beschäftigen sich ausschließlich mit Anfragen, die eine Aggregatfunktion wie Summen- oder Durchschnittsbildung enthalten.[2] Dadurch verlieren die zurückgegebenen Daten an Informationsgehalt, denn ein aggregierter Wert ist weniger aussagekräftig als eine Menge exakter Werte. Für statistische Analysen und Forschungszwecke sind die realen Mikrodaten von erheblich größerer Relevanz. Des Weiteren können mit Aggregatfunktionen nur numerische Attribute angefragt werden. Sensible Daten sind in der Praxis aber häufig diskreter Art, wie zum Beispiel Angaben über Krankheiten oder die politische Haltung von Individuen. Aus den genannten Gründen werden in der vorliegenden Arbeit Anfragen zugelassen, die aus beliebigen Selektionen und Projektionen bestehen und insbesondere auch die tatsächlichen Daten in nicht-aggregierter Form zurückgeben.

Ein zweiter Nachteil bisheriger Ansätze ist die starke Beschränkung auf ein festes Modell zum Schutz der Privatsphäre. Dabei kommen Ansätze zum einen aus dem Bereich der Sicherheit und fordern, dass durch die Kenntnis der Ergebnisse von Anfragen das Wissen

[1] wenn $\mathsf{P} \neq \mathsf{NP}$

[2] Vgl. Kapitel 2.6.1.

eines Angreifers nicht steigen darf.[3] Dieses Modell mag für Sicherheitsaspekte ideal sein, für den Schutz der Privatsphäre ist es jedoch nicht flexibel genug. Oftmals stellt es keine Verletzung der Privatsphäre dar, wenn ein Angreifer zwar nicht den exakten sensiblen Wert eines Individuums erfährt, bestimmte Werte jedoch ausschließen kann. Dadurch hätte er einen Wissenszuwachs, der nicht zum Bruch der Privatsphäre führen würde. Auf der anderen Seite existieren Modelle, die es als ausreichend erachten, dass nur der exakte sensible Wert verschleiert werden muss.[4] Somit kann es für Angreifer möglich sein, zu 99 % einen Wert vorherzusagen, was wiederum zu einer Verletzung führen kann. k-Anonymität und deren Erweiterung stellen flexible Modelle zum Schutz der Privatsphäre dar, die sich beim PPDP bewährt haben und daher in der vorliegenden Arbeit verwendet werden. Insbesondere lässt sich der gewünschte Schutz durch einen Parameter variabel erhöhen oder abschwächen.

Neuere Arbeiten benutzen häufig die differentielle Privatsphäre als Schutzkriterium. Damit erreichen sie zwar ein ausreichendes Maß an Privatsphäre, haben aber oftmals den Nachteil, dass sie die Anzahl an Anfragen beschränken müssen. Der Grund hierfür liegt im Erzeugen von Rauschen innerhalb der Ergebnisse, was bei zu vielen Anfragen zur Verletzung der differentiellen Privatsphäre führen kann [105, 118]. Der Ansatz, der in der vorliegenden Arbeit präferiert wird, lässt hingegen beliebig viele Anfragen zu.

Für die weiteren Betrachtungen gelten folgende Annahmen. Es seien Mikrodaten in Form einer Tabelle gegeben, für die dieselben Eigenschaften gelten wie beim PPDP (vgl. Kapitel 2.2). Das bedeutet insbesondere, dass jedes Tupel der Tabelle eindeutig einem Individuum zugeordnet werden kann und dass zu jedem Individuum höchstens ein Tupel gehört. Nutzer können Anfragen an die Daten mithilfe einfacher SQL-Ausdrücke stellen.[5] Dabei sind nur Selektions- und Projektionsoperationen erlaubt. Im Verlauf dieses Prozesses finden keine Veränderungen der Daten statt. Somit liefern zwei identische Anfragen, die an zwei verschiedenen Zeitpunkten gestellt wurden, dasselbe Ergebnis. Insbesondere ändert sich kein sensibler Wert eines Tupels.

Ein Nutzer des Systems kennt alle Ergebnisse der Anfragen, die er gestellt hat. Weiterhin kann er auf externe Daten zugreifen und Individuen identifizieren, zu denen Tupel in der Tabelle gehören. Es ist möglich, dass er von mehreren oder sogar allen Individuen sämtliche quasi-identifizierende Attributwerte kennt. In der Literatur existiert dafür der Begriff des *QI-bewussten* Nutzers [163]. *Angreifer* sind Nutzer, die Individuen mit den zugehörigen sensiblen Werten verknüpfen wollen. Eine solche Attributpreisgabe soll durch den hier vorgestellten Ansatz verhindert werden.

3.2 Wissen des Angreifers

Modelle für den Schutz der Privatsphäre orientieren sich meistens am Wissen, das eine Person wie beispielsweise ein Angreifer über ein bestimmtes Individuum erlangt. Übersteigt dieser Informationsgehalt eine vorgegebene Schranke, wird von einer Verletzung der Privatsphäre gesprochen. Demnach ist es essentiell, das Wissen eines Angreifers zu begrenzen. In der vorliegenden Arbeit stellt ein Nutzer Anfragen an ein Datenbankmanagementsystem und erhält entsprechende Ergebnisse. Die Kenntnisse über sensible Werte,

[3] Vgl. Kapitel 2.6.2.
[4] Vgl. Kapitel 2.6.1.
[5] Anfragen in Relationaler Algebra sind ebenfalls möglich.

die er daraus ableiten kann, stellen seinen Informationsgewinn dar. Um ein geeignetes Modell für den Schutz der Privatsphäre zu entwickeln, muss zunächst das Wissen des Angreifers quantifiziert werden.

Name	Alter	PLZ	Geschlecht	Krankheit
Alison	20	10140	w	Asthma
Ben	21	10140	m	Asthma
Clark	21	12555	m	Bronchitis
Debra	22	13410	w	Cholera
Elaine	23	25011	w	Bronchitis
Fiona	24	25011	w	Bronchitis
Gary	25	30125	m	Diabetes
Helen	25	30125	w	Cholera
Ian	28	42011	m	Asthma

Tabelle 3.1: Mikrodaten

Sei eine Datenbankrelation gegeben, die sensible Daten enthält und in Tabelle 3.1 dargestellt ist. Sie besteht aus neun Tupeln, die jeweils einer natürlichen Person zugeordnet werden können, die hier mit ihrem Namen angegeben ist. Als weitere Merkmale werden das Alter, die Postleitzahl (PLZ), das Geschlecht und eine Krankheit aufgeführt. Wie in den Beispielen in Kapitel 2 sei `Krankheit` ein sensibles Attribut, `Name` identifizierend und die weiteren Attribute bilden den Quasi-Identifikator.[6]

Sei Q_1 eine Anfrage, die alle Tupel der Individuen selektiert, deren Alter zwischen 20 und 22 liegt. Der entsprechende Ausdruck in SQL lautet

Q_1: `SELECT * FROM T WHERE Alter BETWEEN 20 AND 22`

und liefert als Ergebnis die ersten vier Tupel der Datentabelle 3.1 (siehe Tabelle 3.2).

Name	Alter	PLZ	Geschlecht	Krankheit
Alison	20	10140	w	Asthma
Ben	21	10140	m	Asthma
Clark	21	12555	m	Bronchitis
Debra	22	13410	w	Cholera

Tabelle 3.2: Ergebnis zu Anfrage Q_1

Um die Privatsphäre der Individuen zu schützen, können die in Kapitel 2.3 vorgestellten Methoden verwendet werden. Beispielsweise kann das Ergebnis so verändert werden, dass es k-Anonymität erfüllt. Abbildung 3.1 stellt dafür zwei mögliche Varianten dar, nämlich in (a) eine Generalisierung und in (b) eine Permutation (vgl. Kapitel 2.2.3). Ohne zusätzliches Wissen kann der Anfragende in beiden Fällen keinen sensiblen Attributwert einer Person eindeutig zuordnen.

Die Tabellen in Abbildung 3.1 sind 4-anonym und 3-divers, denn es kommen vier Tupel mit insgesamt drei verschiedenen sensiblen Werten vor. Es wäre in diesem Beispiel auch möglich, zwei jeweils 2-diverse Teiltabellen zurückzugeben. Dazu könnten Alison und Clark

[6] Vgl. die Kategorisierung der Attribute in Kapitel 2.2.2.

	Alter	PLZ	Geschlecht	Krankheit
Alison	20-22	1****	*	Asthma
Ben	20-22	1****	*	Asthma
Clark	20-22	1****	*	Bronchitis
Debra	20-22	1****	*	Cholera

(a) Anonymisierung mittels Generalisierung

	Alter	PLZ	Geschlecht	Krankheit
Alison	20	10140	w	Asthma, Asthma, Bronchitis, Cholera
Ben	21	10140	m	
Clark	21	12555	m	
Debra	22	13410	w	

(b) Anonymisierung mittels Permutation

Abbildung 3.1: Anonymisiertes Ergebnis zu Anfrage Q_1. Die Namen sind nur zur besseren Übersicht angegeben.

mit den sensiblen Werten Asthma und Bronchitis sowie Ben und Debra mit den Werten Asthma und Cholera zusammengefasst werden. Es wird in den folgenden Kapiteln stets davon ausgegangen, dass die gesamte Ergebnismenge komplett (d. h. als eine einzige QI-Gruppe) anonymisiert wird. Eine Diskussion anderer Möglichkeiten bietet Kapitel 9.2.2.

Im Folgenden wird eine Vereinfachung in der Darstellung von Tabellen und Ergebnissen eingeführt, die nur die für die Privatsphäre relevanten Daten enthält. Dazu wird jeder Relation ein numerisches Attribut `ID` hinzugefügt, es werden sämtliche Quasi-Identifikatoren entfernt und das sensible Attribut wird verkürzt mit `SA` benannt. Ebenfalls werden dessen Werte mit Buchstaben abgekürzt. Der Vorteil dieser verkürzten Schreibweise wird in Form von übersichtlichen Grafiken im Laufe dieses Kapitels sichtbar. Auf die Modellierung von QI-Attributen kann verzichtet werden, wenn beispielsweise der Permutationsansatz zur Anonymisierung verwendet wird. Für das zuvor eingeführte Beispiel ergibt sich als Datenbasis Tabelle 3.3, die die Grundlage weiterer Beispiele in diesem Kapitel darstellt.

ID	Name	SA
1	Alison	A
2	Ben	A
3	Clark	B
4	Debra	C
5	Elaine	B
6	Fiona	B
7	Gary	D
8	Helen	C
9	Ian	A

Tabelle 3.3: Vereinfachte Mikrodaten

Als Anfragen werden zunächst nur jene betrachtet, die bestimmte Tupel selektieren und keine Projektionen enthalten. Dahinter steht die Idee, dass in diesem Fall die meisten In-

formationen herausgegeben werden und der Schutz der Privatsphäre am kritischsten zu beurteilen ist. Wird die Ergebnisrelation auf einzelne Attribute projiziert, ist die Privatsphäre geschützt, wenn sie es vor der Projektion war. Durch Q_1 werden die ersten vier Tupel selektiert und als Ergebnis R_1 zurückgegeben. Eine anonymisierte Form von R_1 ist in Tabelle 3.4 angegeben.

ID	Name	SA
1	Alison	A, A, B, C
2	Ben	
3	Clark	
4	Debra	

Tabelle 3.4: R_1: 3-anonymes Ergebnis zu Anfrage Q_1

Wie zuvor ist das Ergebnis 3-divers. Das bedeutet anschaulich, dass der Anfragende beziehungsweise ein Angreifer den SA-Wert von Alison, Ben, Clark oder Debra nicht eindeutig zuordnen kann. Er hat aber jeweils drei verschiedene Möglichkeiten, das heißt, er weiß, dass zum Beispiel Alison A, B oder C haben muss. Weiterhin kann er schlussfolgern, dass die vier im Ergebnis vorkommenden Personen nicht alle gleichzeitig A, B oder C haben können. Die Verteilung ihrer Werte muss zweimal den Wert A und je einmal die Werte B sowie C enthalten. Dementsprechend kann das Wissen des Angreifers als Menge aller möglicher Verteilungen der SA-Werte auf die vorkommenden Personen abgebildet werden.

Um das zu realisieren, wird der Begriff der *SA-Werteverteilung* eingeführt. Dazu wird eine formale Definition von Ergebnismengen benötigt. Um unabhängig von der verwendeten Anonymisierungsoperation[7] zu sein, werden in Ergebnissen sensible von den übrigen Attributwerten getrennt. Dementsprechend besteht R_1 aus den Tupeln (1, Alison), (2, Ben), (3, Clark), (4, Debra) und den SA-Werten A, A, B sowie C. In der ungekürzten Darstellung können Tupel um QI-Attributwerte erweitert und auch generalisierte Werte entsprechend eingesetzt werden. Für die Zwecke der vorliegenden Arbeit reicht es allerdings aus, dass Ergebnistupel nur aus einer ID bestehen. Das `Name`-Attribut ist aus Gründen der Anschaulichkeit hinzugefügt worden.

Im Allgemeinen hat eine Ergebnismenge R_i die Form $R_i = (R_{T_i}, R_{S_i})$, wobei R_{T_i} die Menge von Tupeln ohne sensible Attributwerte und R_{S_i} die Multimenge von sensiblen Attributwerten bezeichne. Weiterhin sei $R^*_{S_i}$ die Menge aller verschiedener sensibler Werte, das heißt, $R^*_{S_i} = \bigcup_{s \in R_{S_i}} \{s\}$. Für Tupel $t \in R_{T_i}$ beziehungsweise SA-Werte $s \in R_{S_i}$ wird verkürzt $t \in R_i$ beziehungsweise $s \in R_i$ geschrieben.

Beispiel 3.1 Wird das Attribut `Name` weggelassen, gilt für R_1 aus Tabelle 3.4 $R_{T_i} = \{1, 2, 3, 4\}$, $R_{S_i} = \{A, A, B, C\}$ und $R^*_{S_i} = \{A, B, C\}$.

Eine SA-Werteverteilung ist eine Zuordnung der sensiblen Werte zu den einzelnen Tupeln. Dabei kommt jeder SA-Wert genauso häufig vor wie im Ergebnis und jedem Tupel muss genau ein Wert zugeordnet werden. Die Idee dahinter ist, dass jede Person, zu der ein Tupel gehört, genau einen der vorhandenen sensiblen Werte haben muss.

[7] Der hier vorgestellte Ansatz gilt sowohl für die Anonymisierung mittels Generalisierung bzw. Unterdrückung als auch für die Bucketisierung bzw. Permutation (vgl. Kapitel 2.2.3).

Definition 3.1 (SA-Werteverteilung) *Sei Q_i eine Anfrage mit dazugehöriger Ergebnismenge $R_i = (R_{T_i}, R_{S_i})$. Eine* SA-Werteverteilung *von R_i ist eine Abbildung $w\colon R_{T_i} \to R^*_{S_i}$, in der jeder SA-Wert genauso häufig vorkommt, wie er in R_{S_i} vorkommt.*[8]

Für R_1 ergeben sich insgesamt zwölf verschiedene Verteilungen, die in Tabelle 3.5 dargestellt sind.

ID	Name	1	2	3	4	5	6	7	8	9	10	11	12
1	Alison	A	A	A	A	A	A	B	B	B	C	C	C
2	Ben	A	A	B	B	C	C	A	A	C	A	A	B
3	Clark	B	C	A	C	A	B	A	C	A	A	B	A
4	Debra	C	B	C	A	B	A	C	A	A	B	A	A

Tabelle 3.5: K_1: Wissen des Angreifers aus R_1

Ein potentieller Angreifer kann aus dem Ergebnis zu Q_1 schlussfolgern, dass genau eine dieser SA-Werteverteilungen der Realität entspricht. Ohne weitere Kenntnisse kann er aber nicht herausfinden, welche es ist. Damit entspricht sein Wissen genau der Menge der SA-Werteverteilungen.

Definition 3.2 (Wissen des Angreifers) *Sei Q_i eine Anfrage mit dazugehöriger Ergebnismenge $R_i = (R_{T_i}, R_{S_i})$. Das* Wissen des Angreifers *K_i aus R_i ist die Menge aller SA-Werteverteilungen von R_i.*

3.3 k-assign Anonymität

Eine wichtige Erkenntnis ist, dass sich das Wissen des Angreifers verändert, wenn Ergebnisse zu weiteren Anfragen hinzukommen. Sei Q_2 eine weitere Anfrage, die die Tupel von Alison, Elaine, Fiona und Gary selektiert. Analog zu Q_1 beziehungsweise R_1 wird als Ergebnis R_2 aus Tabelle 3.6 zurückgegeben.

ID	Name	SA
1	Alison	A, B, B, D
5	Elaine	
6	Fiona	
7	Gary	

Tabelle 3.6: R_2: 3-anonymes Ergebnis zu Anfrage Q_2

Wird das Ergebnis zur ersten Anfrage nicht beachtet, gibt es zu R_2 wieder zwölf SA-Werteverteilungen, die in Tabelle 3.7 dargestellt sind. Für einen Angreifer, der nur Zugriff auf R_2 hat, gibt es wieder genau drei verschiedene mögliche SA-Werte für jedes Tupel. Werden allerdings die Ergebnisse R_1 und R_2 kombiniert, ergeben sich neue Erkenntnisse. Beispielsweise hat Alison nach R_1 den sensiblen Wert A, B oder C und nach R_2 A, B oder D. Da sich ihr Wert zwischen den Zeitpunkten der beiden Anfragen laut Voraussetzung

[8] Die Forderung nach einer Abbildung gewährleistet, dass auch jedem Tupel genau ein SA-Wert zugeordnet wird.

ID	Name	1	2	3	4	5	6	7	8	9	10	11	12
1	Alison	A	A	A	B	B	B	B	B	B	D	D	D
5	Elaine	B	B	D	A	A	B	B	D	D	A	B	B
6	Fiona	B	D	B	B	D	A	D	A	B	B	A	B
7	Gary	D	B	B	D	B	D	A	B	A	B	B	A

Tabelle 3.7: K_2: Wissen des Angreifers aus R_2

nicht geändert hat, kann Alison weder C noch D haben. Die einzigen beiden Werte, die in beiden Ergebnissen vorkommen, sind A und B.

Diese Schlussfolgerung kann ebenfalls mithilfe von SA-Werteverteilungen gezogen werden. Seien $Q_1, Q_2, \ldots, Q_n$ Anfragen mit dazugehörigen Ergebnismengen $R_1, R_2, \ldots, R_n$. Dann bezeichne $\mathcal{Q}^{(n)} = (Q_1, Q_2, \ldots, Q_n)$ die *Sequenz* beziehungsweise *Folge der Anfragen* und $\mathcal{R}^{(n)} = (R_1, R_2, \ldots, R_n)$ die *Sequenz* beziehungsweise *Folge der Ergebnisse*. Für beliebige, aber feste n seien $\mathcal{Q}$ beziehungsweise $\mathcal{R}$ Abkürzungen für $\mathcal{Q}^{(n)}$ beziehungsweise $\mathcal{R}^{(n)}$. Für eine SA-Werteverteilung für $\mathcal{R}^{(n)}$ wird nun gefordert, dass die Verteilung für jedes einzelne Ergebnis R_i eine SA-Werteverteilung ist. Damit ist die Konsistenz sichergestellt, dass einem Tupel in jedem Ergebnis auch derselbe SA-Wert zugeordnet wird.

Definition 3.3 (SA-Werteverteilung einer Ergebnisfolge) *Sei $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ eine Folge von Anfragen mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$. Sei $w\colon \bigcup_{i=1}^{n} R_{T_i} \to \bigcup_{i=1}^{n} R^*_{S_i}$ eine Abbildung der Menge aller in $\mathcal{R}^{(n)}$ vorkommenden Tupel auf die Menge aller in $\mathcal{R}^{(n)}$ vorkommenden sensiblen Werte und sei $w[R_i]$, $1 \leq i \leq n$, die Teilabbildung aller Tupel und SA-Werte, die in R_i vorkommen. Dann ist w genau dann eine* SA-Werteverteilung *von $\mathcal{R}^{(n)}$, wenn für alle i gilt, dass $w[R_i]$ eine SA-Werteverteilung von R_i ist.*

Auf analoger Weise wird das Wissen des Angreifers für eine Folge von Anfragen beziehungsweise Ergebnissen definiert.

Definition 3.4 (Wissen des Angreifers aus einer Ergebnisfolge) *Sei $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ eine Folge von Anfragen mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$. Das* Wissen des Angreifers $\mathcal{K}^{(n)}$ *aus $\mathcal{R}^{(n)}$ ist die Menge aller SA-Werteverteilungen von $\mathcal{R}^{(n)}$.*

Für die gegebene Folge von Ergebnissen $\mathcal{R}^{(2)} = (R_1, R_2)$ gibt es insgesamt 36 verschiedene SA-Werteverteilungen, die alle in Tabelle 3.8 aufgelistet sind. Die Nummerierung orientiert sich dabei an der Nummerierung von R_1. Das bedeutet, die Verteilungen 1.1, 1.2 und 1.3 stimmen mit der Verteilung 1 des ersten Ergebnisses auf den Tupeln von Alison, Ben, Clark und Debra überein. Daraus ist ersichtlich, dass aus der Verteilung 1 im Prinzip drei Verteilungen entstanden sind. Dahingegen sind die Verteilungen 10, 11 und 12, bei denen Alison ein C zugeordnet wurde, weggefallen. Das deckt sich mit der zuvor durchgeführten Analyse, dass Alison den SA-Wert A oder B haben muss. Interessanterweise können allen anderen Tupeln weiterhin drei verschiedene SA-Werte zugeordnet werden.

Sei Q_3 eine vorerst letzte Anfrage, die die Tupel für Ben, Elaine, Gary und Helen selektiert und deren Ergebnis in Tabelle 3.9 dargestellt ist. Werden erneut die SA-Werteverteilungen von $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$ betrachtet, fallen diesmal viele Zuordnungen weg und es verbleiben nur die 18 Verteilungen aus Tabelle 3.10. Ein Angreifer hat dadurch allerdings wieder einen Wissenszuwachs, denn für Ben fällt der SA-Wert B weg, wodurch er A oder

ID	Name	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2	3.3	4.1	4.2	4.3	5.1	5.2	5.3	6.1	6.2	6.3
1	Alison	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
2	Ben	A	A	A	A	A	A	B	B	B	B	B	B	C	C	C	C	C	C
3	Clark	B	B	B	C	C	C	A	A	A	C	C	C	A	A	A	B	B	B
4	Debra	C	C	C	B	B	B	C	C	C	A	A	A	B	B	B	A	A	A
5	Elaine	B	B	D	B	B	D	B	B	D	B	B	D	B	B	D	B	B	D
6	Fiona	B	D	B	B	D	B	B	D	B	B	D	B	B	D	B	B	D	B
7	Gary	D	B	B	D	B	B	D	B	B	D	B	B	D	B	B	D	B	B

ID	Name	7.1	7.2	7.3	7.4	7.5	7.6	8.1	8.2	8.3	8.4	8.5	8.6	9.1	9.2	9.3	9.4	9.5	9.6
1	Alison	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
2	Ben	A	A	A	A	A	A	A	A	A	A	A	A	C	C	C	C	C	C
3	Clark	A	A	A	A	A	A	C	C	C	C	C	C	A	A	A	A	A	A
4	Debra	C	C	C	C	C	C	A	A	A	A	A	A	A	A	A	A	A	A
5	Elaine	A	A	B	B	D	D	A	A	B	B	D	D	A	A	B	B	D	D
6	Fiona	B	D	A	D	A	B	B	D	A	D	A	B	B	D	A	D	A	B
7	Gary	D	B	D	A	B	A	D	B	D	A	B	A	D	B	D	A	B	A

Tabelle 3.8: $\mathcal{K}^{(2)}$: Wissen des Angreifers aus $\mathcal{R}^{(2)} = (R_1, R_2)$

ID	Name	SA
2	Ben	A, B, C, D
5	Elaine	
7	Gary	
8	Helen	

Tabelle 3.9: R_3: 3-anonymes Ergebnis zu Anfrage Q_3

C haben muss. Diese Erkenntnis kann auch logisch hergeleitet werden. Wenn Ben B hätte, müsste Alison auf jeden Fall A haben (laut R_1 und R_2). In R_2 kommt zweimal der Wert B vor, woraus folgt, dass Elaine oder Gary auch ein B haben müssten. Da beide allerdings auch in R_3 vorkommen, müsste hier auch zweimal der SA-Wert B vorkommen. Das ist aber nicht der Fall.

Eine Übersicht, welchen Tupeln beziehungsweise Personen welche SA-Werte zugeordnet werden können, ist in Tabelle 3.11 gegeben. Diese setzt sich allein aus den SA-Werten zusammen, die in mindestens einer SA-Werteverteilung in Tabelle 3.10 vorkommen. Dabei wird nur die Existenz einer Zuordnung betrachtet und nicht etwa eine bestimmte Wahrscheinlichkeit. Beispielsweise gibt es nur zwei Verteilungen, bei denen Fiona ein D zugeordnet wird ($\frac{2}{18} = 11{,}1\,\%$), aber zehn Fälle, bei denen sie ein B haben kann ($\frac{10}{18} = 55{,}6\,\%$). Falls alle SA-Werteverteilungen gleich wahrscheinlich sind, würde ein Angreifer mit einer höheren Wahrscheinlichkeit vermuten, dass Fiona den SA-Wert B hat. An dieser Stelle ist aber nur die Existenz mindestens einer SA-Werteverteilung mit einer bestimmten Zuordnung von Interesse. Denn nur wenn es keine SA-Werteverteilung gibt, die einem Tupel t einen SA-Wert s zuordnet, hat auch die entsprechende Person nicht den SA-Wert s.

Aus den bisherigen Betrachtungen kann nun ein Modell für den Schutz der Privatsphäre erstellt werden. Die Idee dabei ist, dass ein Angreifer den sensiblen Wert eines Individuums

ID	Name	1.1	1.3	2.1	2.3	5.1	5.3	6.1	6.3	7.3	7.5	8.3	8.5	9.1	9.2	9.3	9.4	9.5	9.6
1	Alison	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	B	B
2	Ben	A	A	A	A	C	C	C	C	A	A	A	A	C	C	C	C	C	C
3	Clark	B	B	C	C	A	A	B	B	A	A	C	C	A	A	A	A	A	A
4	Debra	C	C	B	B	B	B	A	A	C	C	A	A	A	A	A	A	A	A
5	Elaine	B	D	B	D	B	D	B	D	B	D	B	D	A	A	B	B	D	D
6	Fiona	B	B	B	B	B	B	B	B	A	A	A	A	B	D	A	D	A	B
7	Gary	D	B	D	B	D	B	D	B	D	B	D	B	D	B	D	A	B	A
8	Helen	C	C	C	C	A	A	A	A	C	C	C	C	B	D	A	D	A	B

Tabelle 3.10: $\mathcal{K}^{(3)}$: Wissen des Angreifers aus $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$

ID	Name	SA-Werte	Anzahl
1	Alison	A, B	2
2	Ben	A, C	2
3	Clark	A, B, C	3
4	Debra	A, B, C	3
5	Elaine	A, B, D	3
6	Fiona	A, B, D	3
7	Gary	A, B, D	3
8	Helen	A, B, C, D	4

Tabelle 3.11: Mögliche SA-Werte nach $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$

nicht eindeutig zuordnen darf und mindestens eine bestimmte Anzahl von Werten dafür infrage kommen soll. Für jedes Individuum soll es mindestens k verschiedene SA-Werte geben, die das Individuum potentiell haben kann. Für $\mathcal{R}^{(3)}$ gibt es zum Beispiel für Alison und Ben nach Tabelle 3.10 jeweils zwei mögliche SA-Werte, für Helen vier und für alle anderen genau drei. Fällt diese Anzahl für mindestens eine Person unter die Grenze von k, liegt eine Verletzung der Privatsphäre vor. Diese Idee wird im Folgenden formalisiert.

Da es für Angreifer möglich sein kann, bestimmte identifizierende oder quasi-identifizierende Werte Personen zuzuordnen, werden alle Einschränkungen auf Tupelbasis getroffen. Dabei besteht ein Tupel aus einem Identifikator (ID) und eventuell weiterer nicht sensibler Attribute. Die ID referenziert das Tupel und damit auch die Person, zu der es gehört. Die Menge aller SA-Werte, die einem Tupel zugeordnet werden können, wird *Signatur*[9], ihre Größe, also die Anzahl unterschiedlicher sensibler Werte, *Anonymitätsgrad* genannt.

Definition 3.5 (Signatur und Anonymitätsgrad) *Sei Q_i eine Anfrage mit dazugehöriger Ergebnismenge $R_i = (R_{T_i}, R_{S_i})$. Für ein Tupel $t \in R_i$ ist die* Signatur $Sig_{R_i}(t)$ *von t bezüglich R_i die Menge aller SA-Werte $s \in R_i$, für die es eine SA-Werteverteilung w von R_i gibt mit $(t, s) \in w$. Der* Anonymitätsgrad $a_{R_i}(t)$ *von t bezüglich R_i ist die Größe der Signatur: $a_{R_i}(t) = |Sig_{R_i}(t)|$.*

Signaturen und Anonymitätsgrade können recht einfach auf Folgen von Anfragen erweitert werden.

[9]Der Begriff der Signatur ist in ähnlicher Form von Xiao and Tao [162] eingeführt worden.

Definition 3.6 (Signatur und Anonymitätsgrad einer Ergebnisfolge) *Sei $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ eine Folge von Anfragen mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$. Für ein Tupel $t \in \mathcal{R}^{(n)}$ ist die* Signatur $Sig_{\mathcal{R}^{(n)}}(t)$ *von t bezüglich $\mathcal{R}^{(n)}$ die Menge aller SA-Werte $s \in \mathcal{R}^{(n)}$, für die es eine SA-Werteverteilung w von $\mathcal{R}^{(n)}$ gibt mit $(t, s) \in w$. Der* Anonymitätsgrad $a_{\mathcal{R}^{(n)}}(t)$ *von t bezüglich $\mathcal{R}^{(n)}$ ist die Größe der Signatur: $a_{\mathcal{R}^{(n)}}(t) = |Sig_{\mathcal{R}^{(n)}}(t)|$.*

Mithilfe des Anonymitätsgrades wird ein Maß für den Schutz der Privatsphäre bei einer gegebenen Folge von Anfragen definiert.

Definition 3.7 (k-assign Anonymität) *Sei $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ eine Folge von Anfragen mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$. $\mathcal{R}^{(n)}$ erfüllt* k-assign Anonymität*, wenn der Anonymitätsgrad jedes Tupels $t \in \mathcal{R}^{(n)}$ bezüglich $\mathcal{R}^{(n)}$ mindestens k ist.*

Beispiel 3.2 In Tabelle 3.11 sind die Signaturen (Spalte SA-Werte) und Anonymitätsgrade (Spalte Anzahl) aller Tupel bezüglich $\mathcal{R}^{(3)}$ aufgelistet. Da 2 der kleinste vorkommende Wert ist, erfüllt $\mathcal{R}^{(3)}$ 2-assign Anonymität.

Ein Vergleich mit bereits bekannten Modellen wie k-Anonymität oder ℓ-Diversität zeigt, dass k-assign Anonymität eine Art Verallgemeinerung dieser Begriffe darstellt. Wenn eine Tabelle, die das Ergebnis einer Anfrage darstellt, k-assign Anonymität erfüllt, dann ist sie auch k-anonym und ℓ-divers mit $k = \ell$. Werden mithilfe von Anfragen allerdings mehrere Tabellen beziehungsweise Ergebnisse veröffentlicht, die jeweils k-anonym oder ℓ-divers sind, muss die gesamte Folge von Ergebnissen nicht k-assign anonym sein. Beispielsweise ist jedes Ergebnis zu den eingeführten Anfragen Q_1, Q_2 und Q_3 4-anonym und mindestens 3-divers, denn es kommen immer vier Tupel mit mindestens drei verschiedenen SA-Werten vor. Die Ergebnismengenfolge $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$ ist allerdings nur 2-assign anonym. Eine ausführliche Diskussion über die Verwendung dieses Modells im Vergleich zu anderen Modellen erfolgt im Kapitel 9.2.1.

Um festzustellen, ob für ein gegebenes k eine Folge von Ergebnissen $\mathcal{R}^{(n)}$ k-assign anonym ist, muss der minimale Anonymitätsgrad eines Tupels berechnet werden. Dieser kann zum Beispiel aus allen SA-Werteverteilungen abgelesen werden, wobei zunächst unklar ist, wie diese effizient berechnet werden können. Damit k-assign Anonymität erfüllt ist, muss es für jedes Tupel t mindestens k verschiedene SA-Werte s und SA-Werteverteilungen w mit $(t, s) \in w$ geben. Damit ist das Finden zumindest einer SA-Werteverteilung der erste Schritt für einen Test auf k-assign Anonymität. Dieses Problem wird SA-Werteverteilung genannt.

Problem 3.1 (SA-Werteverteilung) *Gegeben seien eine Folge von Anfragen $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$, ein Tupel t und ein SA-Wert s. Gesucht ist eine SA-Werteverteilung w von $\mathcal{R}^{(n)}$ mit $(t, s) \in w$.*

Eine Lösung von SA-Werteverteilung garantiert, dass einem gegebenen Tupel mindestens ein SA-Wert zugeordnet werden kann. Dieser sensible Wert kann dabei der tatsächliche sensible Wert des dazugehörigen Individuums in den Mikrodaten sein oder ein völlig anderer. Für ein Tupel t wird der reale SA-Wert s, der in den Mikrodaten im vollständigen Tupel vorkommt, sein *originaler* SA-Wert genannt. Offensichtlich kommen t und s in jedem Ergebnis zusammen vor, unabhängig davon, welche Anfragen gestellt wurden. Daher ist die Abbildung, die jedem Tupel seinen originalen SA-Wert zuordnet,

immer eine SA-Werteverteilung. Sie heißt *Originalverteilung* w_{orig} und ist in den vorigen Beispielen mit 1 beziehungsweise 1.1 gekennzeichnet worden (vgl. Tabellen 3.5, 3.8 und 3.10). Eine einfache Schlussfolgerung aus der Existenz der Originalverteilung ist, dass der Anonymitätsgrad nie unter 1 sinken kann.

Ein zentrales Ziel der vorliegenden Arbeit ist, einen Algorithmus zur Lösung des Problems SA-Werteverteilung anzugeben, insbesondere wenn die Eingabe s nicht der originale SA-Wert von t ist. Dazu werden Ergebnisse von Anfragen zunächst in Graphen überführt. Die SA-Werteverteilung einer Anfrage entspricht dann einem perfekten Matching in diesem Graphen. Damit verlagert sich das Problem vom Finden einer SA-Werteverteilung zum Bestimmen eines perfekten Matchings. Es müssen allerdings bestimmte zusätzliche Restriktionen eingehalten werden, die dazu führen, dass das Matchingproblem NP-vollständig ist.

3.4 Anfragegraphen

Um SA-Werteverteilungen zu bestimmen, wird eine geeignete Modellierung von Ergebnismengen von Anfragen benötigt. Dafür wird nun ein sogenannter *Anfragegraph* erstellt, der die Beziehungen zwischen Tupeln und SA-Werten innerhalb der Ergebnismengen darstellt. Das Ziel dabei ist, SA-Werteverteilungen aus den Graphen ableiten zu können.

Sei Q eine Anfrage mit dem Ergebnis R. Wie bereits erwähnt, besteht R aus einer Menge von Tupeln und einer Multimenge von SA-Werten. Ein Anfragegraph G enthält für jedes Tupel und jeden SA-Wert jeweils einen Knoten. Kommen SA-Werte im Ergebnis mehrfach vor, werden auch entsprechend viele Knoten erstellt. Jeder Knoten wird mit dem Tupel oder dem SA-Wert gelabelt[10], für das beziehungsweise den er erzeugt wurde. Kanten verlaufen zunächst zwischen allen Knoten für Tupel und allen Knoten für SA-Werte.

Eine Kante im Anfragegraphen symbolisiert, dass ein gegebener SA-Wert einem bestimmten Tupel zugeordnet werden kann. Das heißt im Wesentlichen, dass eine SA-Werteverteilung mit diesem SA-Wert-Tupel-Paar prinzipiell möglich wäre, ohne sie direkt zu berechnen.[11] Im Gegenzug können Kanten aus dem Graphen entfernt werden, wenn es offensichtlich keine solche Zuordnung geben kann. Das ist der Fall, wenn eine Folge von Anfragen gegeben ist und ein Tupel t in verschiedenen Ergebnissen vorkommt. Dann können t nur die SA-Werte s zugeordnet werden, die in allen Ergebnissen vorkommen, in denen auch t vorkommt. Kanten zwischen dem Knoten für t und Knoten für SA-Werte, die diese Bedingung nicht erfüllen, müssen nicht dargestellt werden.

Definition 3.8 (Anfragegraph) *Sei $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$ eine Folge von Anfragen mit zugehörigen Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$. Zu jeder Anfrage Q_i beziehungsweise Ergebnismenge R_i bezeichne $G_i = (V_i, E_i)$ einen* Anfragegraphen*, der wie folgt konstruiert wird:*

- *Für jedes Tupel $t \in R_i$ enthalte V_i einen Knoten, der mit t gelabelt wird und* Tupelknoten *heißt.*

[10] Für den aus dem Englischen stammenden Begriff „Label“ wird im Deutschen manchmal auch die Bezeichnung „Marke“ verwendet. Da die Begriffe „Label“ zusammen mit dem Verb „labeln“ in der Informatik weit verbreitet und bereits seit 2006 im Rechtschreibduden [40] erfasst sind, werden sie in dieser Arbeit anstelle deutscher Ausdrücke verwendet.

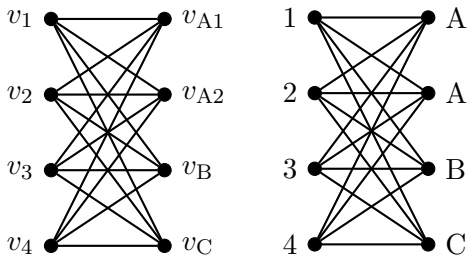
[11] Die SA-Werte der Labels aller Nachbarn eines Knotens mit Label t sind tatsächlich nur eine Obermenge aller SA-Werte, die t durch SA-Werteverteilungen zugeordnet werden können.

- *Für jeden sensiblen Wert $s \in R_i$ enthalte V_i einen Knoten, der mit s gelabelt wird und* SA-Knoten *heißt.*

- *Zwischen jedem Tupelknoten v_t mit Label t und jedem SA-Knoten v_s mit Label s enthalte E_i genau dann eine Kante $\{v_t, v_s\}$, wenn in jedem Ergebnis $R_j \in \mathcal{R}^{(n)}$, in dem Tupel t vorkommt, mindestens einmal der sensible Wert s vorkommt.*

Bei dieser Definition fällt auf, dass die Graphen aus den Ergebnissen von Anfragen erstellt werden und nicht aus den Anfragen direkt. Sie werden trotzdem Anfragegraphen und nicht Ergebnisgraphen genannt, um das verwendete Szenario der Anfragebearbeitung sowie die Abhängigkeit zur gestellten Anfrage und nicht zu einer Anonymisierung der Ergebnismenge zu unterstreichen. Sind die angefragten Daten vorgegeben und ist offensichtlich, wie das Ergebnis R_i zu einer Anfrage Q_i lautet, wird im Folgenden nur noch von Anfragegraphen für Q_i gesprochen. Dadurch kann in vielen Situationen auf die explizite Definition von Ergebnissen verzichtet werden.

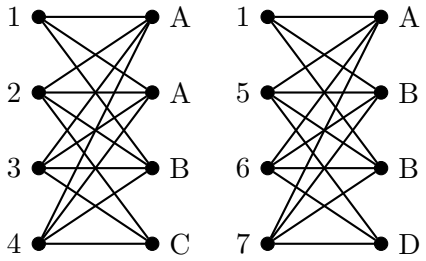
Während Knoten mit Tupeln eindeutig benannt sind, können mehrere SA-Knoten das gleiche Label s besitzen, wenn mehrere sensible Werte s in einem Ergebnis vorkommen. Eine weitere wichtige Eigenschaft von Anfragegraphen ist, dass jeder Graph G_i für eine Anfrage Q_i, $1 \leq i \leq n$, von allen Anfragen $Q_1, \ldots, Q_n$ beziehungsweise Ergebnissen $R_1, \ldots, R_n$ abhängig ist. Entsteht durch das Hinzufügen einer weiteren Anfrage Q_{n+1} eine verlängerte Folge von Anfragen $\mathcal{Q}^{(n+1)} = (Q_1, \ldots, Q_n, Q_{n+1})$, so entsteht für die bereits gestellte Anfrage Q_i ein neuer Anfragegraph G_i'. Dieser ist nun von allen Anfragen $Q_1, \ldots, Q_n, Q_{n+1}$ beziehungsweise Ergebnissen $R_1, \ldots, R_n, R_{n+1}$ abhängig, kann identisch zu G_i sein oder weniger Kanten enthalten. Beispielsweise fehlt in G_i' eine Kante zwischen Knoten für ein Tupel t und einem SA-Wert s, wenn t im Ergebnis zu Q_{n+1} vorkommt, aber nicht der sensible Wert s. Dem Tupel t kann demnach s nicht mehr zugeordnet werden. Das folgende Beispiel erläutert diesen Zusammenhang.



(a) G_1 (b) G_1

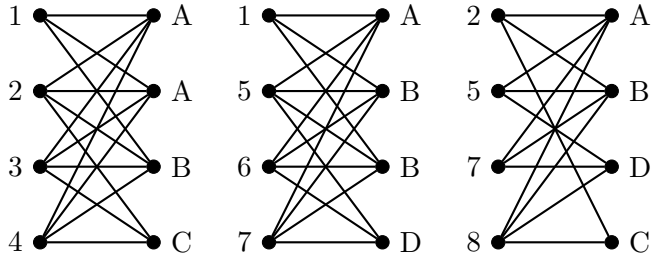
Abbildung 3.2: Anfragegraph G_1 für Q_1 mit Knotennamen (a) bzw. Labels (b)

Beispiel 3.3 Es werden erneut die anfangs dieses Kapitels eingeführten Anfragen Q_1, Q_2 und Q_3 betrachtet. Für $\mathcal{Q}^{(1)} = (Q_1)$ (siehe Tabelle 3.4) ist in Abbildung 3.2 der dazugehörige Anfragegraph G_1 gegeben. Für die Tupel 1, 2, 3 und 4 wird jeweils ein Tupelknoten und für die SA-Werte A, A, B und C jeweils ein SA-Knoten erstellt. In (a) sind die Knoten von G mit einem Namen und in (b) mit ihrem Label beschriften. Da A zweimal im Ergebnis zu Q_1 vorkommt, enthält G_1 auch zwei Knoten mit dem Label A. In allen folgenden Beispielen werden für Knoten nur noch ihre Labels angegeben.



(a) G'_1 (b) G'_2

Abbildung 3.3: Anfragegraphen für $\mathcal{Q}^{(2)} = (Q_1, Q_2)$



(a) G''_1 (b) G''_2 (c) G''_3

Abbildung 3.4: Anfragegraphen für $\mathcal{Q}^{(3)} = (Q_1, Q_2, Q_3)$

Abbildung 3.3 zeigt die Anfragegraphen für $\mathcal{Q}^{(2)} = (Q_1, Q_2)$, das heißt für die Folge, die aus den beiden Anfragen Q_1 und Q_2 (siehe Tabelle 3.6) besteht. Der Graph G'_1 für Q_1 in $\mathcal{Q}^{(2)}$ hat dabei eine Kante weniger als der Graph G_1 für Q_1 in $\mathcal{Q}^{(1)}$. Der Grund dafür ist, dass Alison (ID 1) zwar in beiden Ergebnissen vorkommt, der SA-Wert C aber nur im ersten. Nach der ersten Anfrage konnte Alison noch den sensiblen Wert C haben, nach der zweiten aber nicht mehr.

Zusammen mit der letzten Anfrage Q_3 (siehe Tabelle 3.9) entsteht die Anfragenfolge $\mathcal{Q}^{(3)} = (Q_1, Q_2, Q_3)$. Die dazugehörigen Anfragegraphen sind in Abbildung 3.4 dargestellt. Hier gibt es beispielsweise in G''_3 keine Kante zwischen dem Knoten für Tupel 2 und dem für den SA-Wert D, denn in Q_1 kommt zwar Tupel 2 vor, aber kein sensibler Wert D.

Für eine einheitliche und verkürzte Schreibweise wird folgende Notation vereinbart. Seien Q eine Anfrage mit dem Ergebnis R, $G = (V, E)$ der dazugehörige Anfragegraph und L die Menge aller in R vorkommender Tupel und SA-Werte. Entsprechend Definition 3.8 weise die *Labelfunktion* $l\colon V \to L$ jedem Knoten $v \in V$ ein Label aus L zu. Dabei werden die Labels von Tupelknoten *Tupellabel* und von SA-Knoten *SA-Label* genannt. Die Schreibweise $v =_l v'$ drückt aus, dass zwei Knoten $v, v' \in V$ dasselbe Label besitzen. Sei $e = \{v_t, v_s\} \in E$ eine Kante zwischen einem Tupelknoten v_t mit Label t und einem SA-Knoten v_s mit Label s. Dann heißt $e_l = (t, s)$ *Labelkante* von e. Analog zu den Knoten gilt $e =_l e'$, wenn die Kanten $e, e' \in E$ dieselben Labelkanten haben. Im Folgenden stehen die Variablen v_t und v_s jeweils für Tupel- beziehungsweise SA-Knoten sowie t und s für die Labels dieser Knoten. Verkürzt wird auch $t \in G$ und $s \in G$ geschrieben, wenn entsprechende Knoten $v_t \in V$ und $v_s \in V$ existieren.

Beispiel 3.4 In G_1 aus Abbildung 3.2 gibt es mit $\{v_1, v_{\text{A1}}\}$ und $\{v_1, v_{\text{A2}}\}$ zwar zwei Kanten, die dieselbe Labelkante $(1, \text{A})$ besitzen, aber nur eine Kante, zu der die Labelkante $(1, \text{B})$ gehört.

Analog zu Folgen von Anfragen können auch Anfragegraphen zu einem Graphen zusammengefasst werden. $\mathcal{G}^{(n)} = (G_1, G_2, \ldots, G_n)$ bezeichnet den Anfragegraphen beziehungsweise die Folge von Anfragegraphen, die zu der Folge von Anfragen $\mathcal{Q}^{(n)} = (Q_1, Q_2, \ldots, Q_n)$ erstellt wurden. Für eine bessere Unterscheidung werden die einzelnen G_i auch *Teilanfragegraphen* von $\mathcal{G}^{(n)}$ genannt. Für beliebige, aber feste n existiert die vereinfachte Schreibweise $\mathcal{G} = (G_1, G_2, \ldots, G_n)$.
Anfragegraphen haben eine offensichtliche, aber sehr wichtige strukturelle Eigenschaft.

Korollar 3.1 *Anfragegraphen sind bipartit.*

Beweis: Die Behauptung folgt direkt aus Definition 3.8 zusammen mit Definition 1.7, da es keine Kante zwischen zwei Tupelknoten und keine Kante zwischen zwei SA-Knoten gibt. □

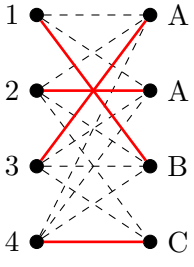
Bisher ist noch nicht klar erkennbar, wie SA-Werteverteilungen aus Anfragegraphen abgeleitet werden können. Beispielsweise enthält der Anfragegraph aus Abbildung 3.4 die Labelkante $(2, \text{B})$, laut Tabelle 3.10 existiert für die entsprechenden Anfragen jedoch keine SA-Werteverteilung, bei der dem Tupel 2 der SA-Wert B zugeordnet wird. Das bedeutet, aus den Kanten an sich lassen sich noch keine Zuordnungen ablesen. Vielmehr müssen noch bestimmte Konsistenzbedingungen überprüft werden, um festzustellen, ob zu einer gegebenen Labelkante (t, s) auch eine SA-Werteverteilung existiert, die das Paar (t, s) enthält. Dafür wird ein Zuordnungsproblem untersucht, welches in der Graphentheorie für gewöhnlich durch ein Matching ausgedrückt wird.

3.5 Matchings in Anfragegraphen

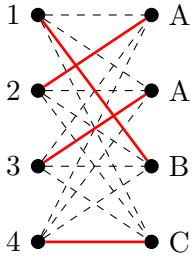
Eine Zuordnung eines SA-Wertes s zu einem Tupel t wird im Anfragegraphen durch die Labelkante (t, s) ausgedrückt. Bei einer SA-Werteverteilung wird jeder SA-Wert genau einem Tupel zugeordnet, sodass jedes Tupel genau einen SA-Wert hat. Die Abbildung ist demnach bijektiv. Dieser Zusammenhang kann in Anfragegraphen durch eine Menge von Kanten zwischen Tupel- und SA-Knoten modelliert werden. Da dabei jeder Knoten genau einmal vorkommt, stellt die Kantenmenge ein perfektes Matching (vgl. Definition 1.8) dar.

ID	Name	7
1	Alison	B
2	Ben	A
3	Clark	A
4	Debra	C

(a) SA-Werteverteilung



(b) M_7



(c) M_7'

Abbildung 3.5: Zwei Matchings in G_1

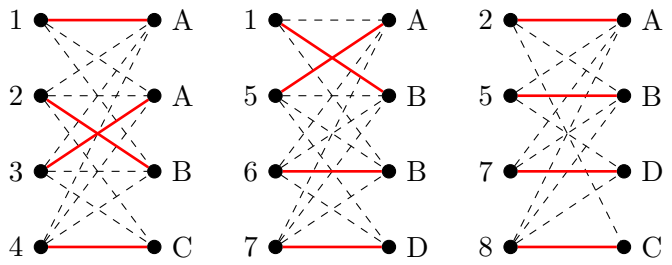
Sei erneut Anfrage Q_1 aus Tabelle 3.4 gegeben. In Abbildung 3.5 ist in (a) eine mögliche SA-Werteverteilung dargestellt, die zuvor mit 7 nummeriert wurde (vgl. Tabelle 3.5). Die farbigen Kanten in (b) zeigen ein entsprechendes Matching im Anfragegraphen G_1. Dabei bedeutet zum Beispiel die Labelkante $(1, \mathrm{B})$, dass Alison mit ID 1 der SA-Wert B zugeordnet wird. Analog bekommen Ben und Clark jeweils ein A und Debra ein C.

Aus jedem perfekten Matching in einem Anfragegraphen kann offensichtlich eine SA-Werteverteilung abgelesen werden. Andersherum existiert zu jeder SA-Werteverteilung auch ein entsprechendes Matching. In Abbildung (c) ist mit M_7' sogar ein zweites Matching angegeben, das zur selben SA-Werteverteilung gehört wie M_7. Somit kann auch aus verschiedenen Matchings dieselbe SA-Werteverteilung abgeleitet werden.

Bevor die getroffenen Beobachtungen formalisiert werden, wird der Fall betrachtet, in dem eine Folge von Anfragegraphen gegeben ist. Um Aussagen einfacher über Tupel und SA-Werte und nicht über die gelabelten Knoten zu treffen, sei vereinbart, dass ein Tupel t mit SA-Wert s genau dann *matcht*, wenn eine Labelkante $e_l = (t, s)$ existiert. Das heißt, es gibt einen Tupelknoten v_t mit Label t, einen SA-Knoten v_s mit Label s und eine Kante $e = \{v_t, v_s\}$ zwischen beiden Knoten. Zu einer Folge von Anfragegraphen $\mathcal{G} = (G_1, \ldots, G_n)$ kann in jedem einzelnen Graphen G_i ein Matching M_i berechnet werden. Das Gesamtmatching $\mathcal{M}$ besteht dann aus der Vereinigung aller Teilmatchings $\mathcal{M} = (M_1, \ldots, M_n)$, was ebenfalls als Folge dargestellt wird. Ein Matching $M_i \in \mathcal{M}$ in einem Teilgraphen G_i wird auch mit $\mathcal{M}[G_i]$ notiert.

ID	Name	SA
1	Alison	A, B
2	Ben	A, B
3	Clark	A
4	Debra	C
5	Elaine	A, B
6	Fiona	B
7	Gary	D
8	Helen	C

(a) Zuordnung von SA-Werten



(b) M_1 (c) M_2 (d) M_3

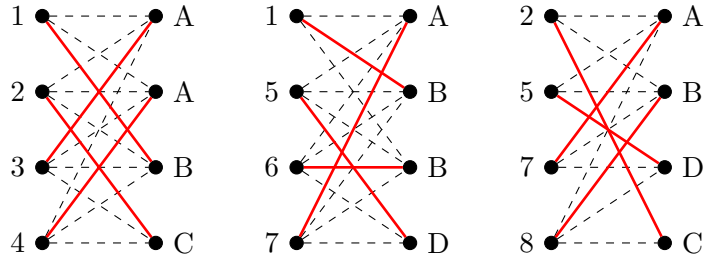
Abbildung 3.6: Ein nicht-valides Matching für $\mathcal{G} = (G_1, G_2, G_3)$

Sei die bereits vorgestellte Folge von Anfragen $\mathcal{Q} = (Q_1, Q_2, Q_3)$ und die dazugehörige Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3)$ in Abbildung 3.6 gegeben. Zunächst wird in jedem Teilanfragegraphen G_i ein perfektes Matching M_i berechnet, was in (b) bis (d) dargestellt ist. Die Vereinigung der Matchings sei $\mathcal{M}_{\text{invalid}} = (M_1, M_2, M_3)$. Offensichtlich gibt es Tupel, die in verschiedenen Graphen mit verschiedenen SA-Werten matchen. Beispielsweise matcht 1 in G_1 mit A und in G_2 mit B. In der Tabelle in (a) ist für jedes Tupel angegeben, mit welchen SA-Werten es in allen Teilgraphen mindestens einmal matcht. Es ist leicht zu erkennen, dass das Gesamtmatching $\mathcal{M}_{\text{invalid}}$ keiner SA-Werteverteilung entspricht, da es mit Alison, Ben und Elaine Tupel gibt, denen verschiedene SA-Werte zugeordnet werden.

Für dieselbe Folge von Anfragegraphen ist in Abbildung 3.7 ein anderes Matching $\mathcal{M}_{9.6} = (M_1, M_2, M_3)$ berechnet worden. In diesem Fall gilt, dass alle Tupel in allen Graphen, in denen sie vorkommen, jeweils mit demselben SA-Wert matchen. In der Ta-

ID	Name	9.6
1	Alison	B
2	Ben	C
3	Clark	A
4	Debra	A
5	Elaine	D
6	Fiona	B
7	Gary	A
8	Helen	B

(a) SA-Werteverteilung



(b) M_1 (c) M_2 (d) M_3

Abbildung 3.7: Ein valides Matching für $\mathcal{G} = (G_1, G_2, G_3)$

belle in (a) ist für jedes Tupel angegeben, mit welchem SA-Wert es in den entsprechenden Teilgraphen matcht. Die entstehende Verteilung ist in Tabelle 3.10 auf Seite 73 in der Spalte mit der Nummer 9.6 zu finden.

Für das Bestimmen von perfekten Matchings in Anfragegraphen ist es entscheidend, dass jedem Tupel in jedem Anfragegraphen derselbe SA-Wert zugeordnet wird. Alle Matchings, die diese Eigenschaft haben, heißen *valid*.

Definition 3.9 (valides Matching) *Sei $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen. Ein Matching $\mathcal{M} = (M_1, \ldots, M_n)$ in $\mathcal{G}$ heißt* valid*, wenn es alle Tupelknoten überdeckt und für alle Labelkanten $(t_i, s_i), (t_j, s_j) \in \mathcal{M}$ gilt: $t_i = t_j \Rightarrow s_i = s_j$.*

Da Anfragegraphen aus derselben Anzahl von Tupel- wie SA-Knoten bestehen, ist jedes valide Matching per Definition perfekt. Der Zusammenhang zwischen validen Matchings und SA-Werteverteilungen wird in der folgenden Proposition festgehalten.

Proposition 3.2 (SA-Werteverteilung eines Matchings) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen zu einer Folge von Anfragen $\mathcal{Q} = (Q_1, \ldots, Q_n)$ mit entsprechenden Ergebnissen $\mathcal{R} = (R_1, \ldots, R_n)$ und $\mathcal{M}$ ein valides Matching in $\mathcal{G}$. Weiterhin sei $w_\mathcal{M}$ eine Menge von Paaren, die für jeden Tupelknoten v_t mit Kante $\{v_t, v_s\} \in \mathcal{M}$ das Paar (t, s) enthält, wobei Tupel t beziehungsweise SA-Wert s die Labels von v_t beziehungsweise v_s sind. Dann ist $w_\mathcal{M}$ eine SA-Werteverteilung von $\mathcal{R}$.*

Beweis: Der Beweis folgt offensichtlich aus den bisher betrachteten Zusammenhängen und Definitionen. □

Für den Zusammenhang zwischen Matchings und SA-Werteverteilungen wird der Begriff der *Äquivalenz* verwendet. Ein valides Matching $\mathcal{M}$ ist *äquivalent* zu einer SA-Werteverteilung w, wenn $w = w_\mathcal{M}$ gilt.

Wichtig ist hierbei festzuhalten, dass zu jedem validen Matching genau eine äquivalente SA-Werteverteilung existiert, die wie in Proposition 3.2 angegeben berechnet werden kann. Andererseits kann auch zu einer gegebenen SA-Werteverteilung w ein äquivalentes valides Matching $\mathcal{M}$ konstruiert werden. Dazu wird für jedes Paar $(t, s) \in w$ in jedem Teilgraphen, in dem ein Tupelknoten v_t vorkommt, der mit t gelabelt ist, eine Kante $\{v_t, v_s\}$ in das Matching aufgenommen, wobei der SA-Knoten v_s das Label s besitzt und in keiner anderen Matchingkante vorkommt. $\mathcal{M}$ ist dabei nicht eindeutig bestimmt, da es

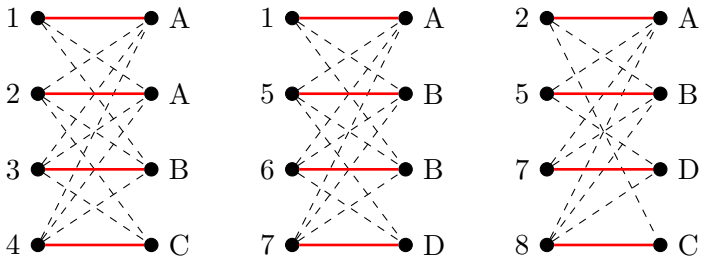
im Graphen mehrere SA-Knoten mit Label s geben kann. Folglich können zu einer SA-Werteverteilung mehrere äquivalente valide Matchings existieren.

Analog zur Originalverteilung wird der Begriff des *Originalmatchings* eingeführt. Dieses Matching besteht aus Kanten zwischen Tupeln und den SA-Werten, die auch tatsächlich in der Datentabelle in den vollständigen Tupeln enthalten sind. Offensichtlich ist dieses Matching stets valid.

Definition 3.10 (Originalmatching) *Sei $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen. Ein Matching $\mathcal{M}_{orig} = (M_1, \ldots, M_n)$ in $\mathcal{G}$ heißt* Originalmatching, *wenn für alle Labelkanten $e_l = (t, s) \in \mathcal{M}$ gilt, dass s der originale SA-Wert von t ist.*

ID	Name	SA
1	Alison	A
2	Ben	A
3	Clark	B
4	Debra	C
5	Elaine	B
6	Fiona	B
7	Gary	D
8	Helen	C

(a) SA-Werteverteilung



(b) $\mathcal{M}_{\text{orig}}[G_1]$ (c) $\mathcal{M}_{\text{orig}}[G_2]$ (d) $\mathcal{M}_{\text{orig}}[G_3]$

Abbildung 3.8: Originalmatching $\mathcal{M}_{\text{orig}}$ für $\mathcal{G} = (G_1, G_2, G_3)$

In Abbildung 3.8 ist für den Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3)$ der anfangs eingeführten drei Anfragen das Originalmatching gegeben und entspricht anschaulich jeweils der Menge der farbig markierten Kanten. Vertauscht man die Matchingpartner der Tupelknoten mit den Labels 1 und 2 in G_1, erhält man ein anderes Matching, das aber ebenfalls der originalen SA-Werteverteilung entspricht. In einem gegebenen Anfragegraphen können demzufolge auch mehrere Originalmatchings existieren, wenn SA-Labels mehrfach vorkommen. Für weitere Betrachtungen wird daher vereinbart, dass mit *dem* Originalmatching immer ein beliebiges, aber festes Matching aus der Menge aller Originalmatchings gemeint ist. Rein optisch wird es immer wie in Abbildung 3.8 durch horizontale Kanten dargestellt.

Als Pendant zum Problem SA-Werteverteilung wird als zweites wichtiges Problem das Bestimmen eines validen Matchings in einer Folge von Anfragegraphen mit QGM (engl. *query graph matching*) bezeichnet.

Problem 3.2 (Query Graph Matching (QGM)) *Gegeben sei eine Folge von Anfragegraphen $\mathcal{G}^{(n)}$. Gesucht ist ein valides Matching $\mathcal{M}^{(n)}$ in $\mathcal{G}^{(n)}$.*

Wird ein Anfragegraph für eine Folge von Anfragen erstellt und ist nach einem beliebigen validen Matching gefragt, ist das Originalmatching immer eine zulässige Lösung. Die Eingabe des eigentlichen Problems SA-Werteverteilung ist allerdings ein konkretes Tupel t und ein fester SA-Wert s. Für Anfragegraphen ist diese Frage äquivalent zur Aufgabenstellung, ein valides Matching zu finden, welches eine fest vorgegebene Kante (t, s) enthält. Dieses Problem kann leicht auf QGM reduziert werden. Dabei wird aus allen Anfragegraphen eine Kante (t, s) inklusive der beiden Endknoten entfernt. Falls im

Restgraphen ein beliebiges valides Matching existiert, kann dieses in allen modifizierten Graphen um die Kanten (t, s) zu einem validen Matching erweitert werden, das die gegebene Kante enthält. Als Eingabe für Problem 3.2 sind somit auch Anfragegraphen zugelassen, bei denen Kanten oder Knoten entfernt wurden. Das ist auch der Grund, warum in den folgenden Kapiteln nur noch Aussagen über Graphen und nicht mehr über Anfragen und Ergebnisse getroffen werden. Es folgt, dass ein Algorithmus für QGM auch zum Lösen von SA-Werteverteilung verwendet werden kann.

In den weiteren Teilen dieses Kapitels werden die Probleme SA-Werteverteilung und QGM untersucht und bewiesen, dass beide NP-vollständig sind. Sie lassen sich mithilfe eines auf ganzzahliger linearer Programmierung basierenden Algorithmus in exponentieller Zeit lösen. In den darauffolgenden Kapiteln werden Heuristiken vorgestellt, die zwar polynomielle Laufzeit haben, dafür aber nur in bestimmten Fällen Matchings beziehungsweise SA-Werteverteilungen berechnen können.

3.6 NP-Vollständigkeit

Die nächsten Sätze besagen, dass die eingeführten Probleme SA-Werteverteilung und QGM NP-vollständig sind.

Theorem 3.3 (NP-Vollständigkeit) *Das Problem* QGM *ist NP-vollständig.*

Beweis: Offensichtlich kann in polynomieller Zeit getestet werden, ob ein gegebenes Matching valid ist. Daraus folgt, dass QGM in NP liegt und es genügt, 3-Sat $\leq_p$ QGM zu zeigen. Das heißt, es muss ein polynomielles Verfahren angegeben werden, das beliebige Boolesche Formeln F in Folgen von Anfragegraphen $\mathcal{G}$ umformt, sodass gilt:

$$F \text{ ist erfüllbar} \Leftrightarrow \mathcal{G} \text{ hat ein valides Matching.}$$

Sei $F = C_1 \wedge \cdots \wedge C_m$ eine 3-Sat-Formel über den Variablen $x_1, \ldots, x_n$. Für jede Variable x_i wird ein Anfragegraph G_{x_i} konstruiert, der aus je zwei Tupel- und SA-Knoten besteht, die untereinander verbunden sind.[12] Die Tupelknoten werden mit x_i und $\bar{x}_i$ sowie die SA-Knoten mit 0 und 1 gelabelt.

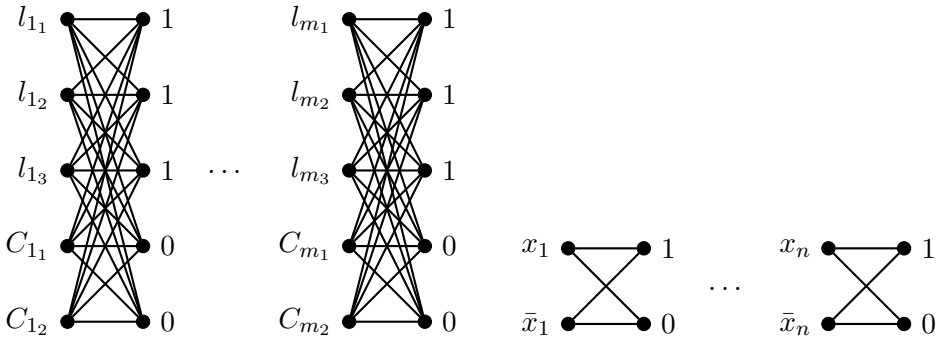
Für jede Klausel $C_j = l_{j_1} \vee l_{j_2} \vee l_{j_3}$ mit $l_{j_k} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ für $j = 1, \ldots, m$ und $k = 1, 2, 3$ wird ein Anfragegraph G_{C_j} erstellt, der je fünf Tupel- und SA-Knoten enthält, die erneut alle untereinander verbunden sind.[13] Die Tupelknoten werden diesmal mit l_{j_1}, l_{j_2}, l_{j_3}, C_{j_1} und C_{j_2} sowie drei SA-Knoten mit 1 und zwei mit 0 gelabelt. Die gesuchte Folge von Anfragegraphen ist $\mathcal{G} = (G_{C_1}, \ldots, G_{C_m}, G_{x_1}, \ldots, G_{x_n})$. In Abbildung 3.9 ist die entsprechende Umformung dargestellt.

Aus jeder Belegung B der Variablen von F kann eine Kantenmenge $\mathcal{M}_B$ im Graphen $\mathcal{G}$ konstruiert werden. Für alle $1 \leq i \leq n$ bezeichne $b_i \in \{0, 1\}$ den Wert, der der Variablen x_i durch B zugewiesen wird. Dann enthält $\mathcal{M}_B$ für jede Variable x_i in allen Graphen, in denen ein Tupelknoten mit Label x_i vorkommt, eine Kante von x_i zu einem SA-Knoten, der mit b_i gelabelt ist. In allen Graphen, in denen ein Tupelknoten mit Label $\bar{x}_i$ vorkommt, enthält $\mathcal{M}_B$ eine Kante von $\bar{x}_i$ zu einem SA-Knoten, der nicht mit b_i gelabelt ist. Offensichtlich ist jede dieser Mengen $\mathcal{M}_B$ ein perfektes Matching in allen Graphen $G_{x_1}, \ldots, G_{x_n}$.

[12] Dieser Graph entspricht einem $K_{2,2}$.
[13] Dieser Graph entspricht einem $K_{5,5}$.

$$F = \underbrace{(l_{1_1} \vee l_{1_2} \vee l_{1_3})}_{C_1} \wedge \cdots \wedge \underbrace{(l_{m_1} \vee l_{m_2} \vee l_{m_3})}_{C_m}$$

(a) 3-SAT-Formel F



(b) G_{C_1} (c) G_{C_m} (d) G_{x_1} (e) G_{x_n}

Abbildung 3.9: Transformation einer 3-SAT-Formel F in eine Folge von Anfragegraphen $\mathcal{G}$, wobei $l_{j_k} \in \{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ gilt.

Wenn B eine erfüllende Belegung ist, muss in jeder Klausel C_j mindestens ein Literal existieren, das unter B den Wert 1 (wahr) erhält. Das bedeutet, die Kanten der Literale in $\mathcal{M}_B$ können in jedem Graphen G_{C_j} so gewählt werden, dass sie sich in keinem SA-Knoten schneiden. Wird für die verbliebenen Tupelknoten C_{j_1} und C_{j_2} jeweils eine Kante zu einem SA-Knoten, der noch nicht überdeckt ist, zu $\mathcal{M}_B$ hinzugefügt, entsteht ein perfektes Matching in allen G_{C_j}. Da B eine Abbildung ist, ist $\mathcal{M}_B$ per Konstruktion ein valides Matching in $\mathcal{G}$.

Mit der umgekehrten Transformation kann aus einem validen Matching $\mathcal{M}$ eine Belegung $B_{\mathcal{M}}$ erstellt werden, die F erfüllt. Dazu wird jeder Variablen x_i, für die eine Kante (x_i, b_i) existiert, der Wert b_i zugewiesen. Da jeder Graph G_{C_j} nur zwei SA-Knoten mit Label 0 enthält, muss mindestens ein Literal mit 1 matchen. Die dazugehörige Klausel C_j wird durch dieses Literal erfüllt.

Da für jede Klausel ein Graph mit zehn und für jede Variable ein Graph mit vier Knoten erstellt wird, ist die Abbildung $F \to \mathcal{G}$ offensichtlich in Polynomialzeit berechenbar. □

Beispiel 3.5 Sei eine Boolesche Formel F gegeben, die in Abbildung 3.10 in (a) dargestellt ist. Die Folge von Anfragegraphen $\mathcal{G}$, die nach der Konstruktion im Beweis von Satz 3.3 erstellt wird, ist in (b) bis (j) angegeben. Hierbei gehören alle gezeichneten Kanten zu den Anfragegraphen. Eine erfüllende Belegung B von F ist beispielsweise

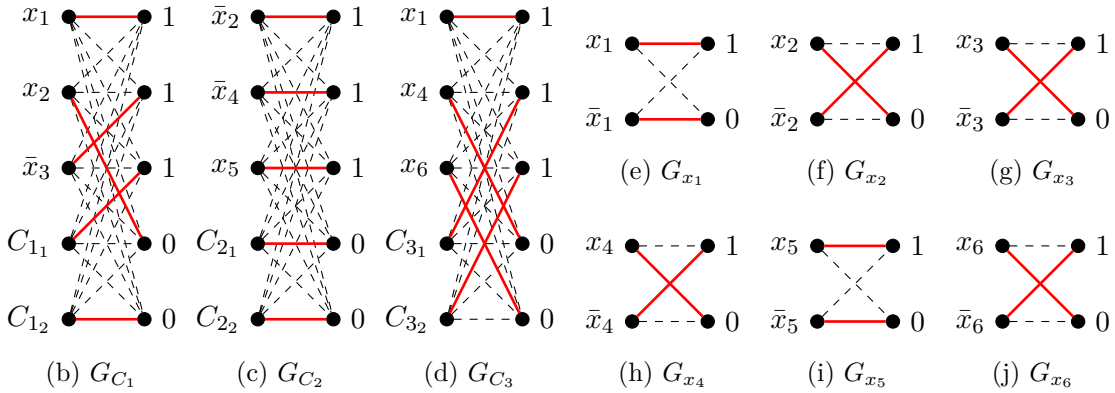
$$\begin{aligned} &B(x_1) = 1, && B(x_3) = 0, && B(x_5) = 1, \\ &B(x_2) = 0, && B(x_4) = 0, && B(x_6) = 0, \end{aligned}$$

denn

$$B(F) = (1 \vee 0 \vee 1) \wedge (1 \vee 1 \vee 1) \wedge (1 \vee 0 \vee 0) = 1.$$

$$F = \underbrace{(x_1 \vee x_2 \vee \bar{x}_3)}_{C_1} \wedge \underbrace{(\bar{x}_2 \vee \bar{x}_4 \vee x_5)}_{C_2} \wedge \underbrace{(x_1 \vee x_4 \vee x_6)}_{C_3}$$

(a) 3-SAT-Formel F



(b) G_{C_1} (c) G_{C_2} (d) G_{C_3} (e) G_{x_1} (f) G_{x_2} (g) G_{x_3} (h) G_{x_4} (i) G_{x_5} (j) G_{x_6}

Abbildung 3.10: Transformation der 3-SAT-Formel F in die Folge von Anfragegraphen $\mathcal{G}$. Die farbigen Kanten bilden ein valides Matching.

Das aus B konstruierte Matching $\mathcal{M}$ besteht aus den farbig markierten Kanten in Abbildung 3.10 und ist offensichtlich valid.

Theorem 3.3 besagt, dass das Finden von validen Matchings in Anfragegraphen nicht effizient durchgeführt werden kann.[14] Es bleibt zu zeigen, dass auch das ursprüngliche Problem, nämlich zu bestimmen, ob einem Tupel ein bestimmter SA-Wert zugeordnet werden kann, ebenso schwer ist. Das bedeutet konkret, dass bewiesen werden soll, dass das Problem SA-Werteverteilung (vgl. Problem 3.1) NP-vollständig ist. Der wesentliche Unterschied zu QGM ist, dass bei SA-Werteverteilung nach speziellen Zuordnungen von SA-Werten zu Tupeln gefragt ist, während QGM nur nach einem beliebigen Matching, das heißt einer beliebigen SA-Werteverteilung fragt. Dazu wird zunächst ein Spezialfall von QGM betrachtet, bei dem nur zwei verschiedene SA-Werte vorkommen. Dieses sogenannte 2-SA-QGM kann auf das Problem SA-Werteverteilung reduziert werden.

Problem 3.3 (2-SA Query Graph Matching (2-SA-QGM)) *Gegeben sei eine Folge von Anfragegraphen $\mathcal{G}^{(n)}$, in der genau zwei verschiedene SA-Werte vorkommen. Gesucht ist ein valides Matching $\mathcal{M}^{(n)}$ in $\mathcal{G}^{(n)}$.*

Satz 3.4 *Das Problem* 2-SA-QGM *ist NP-vollständig.*

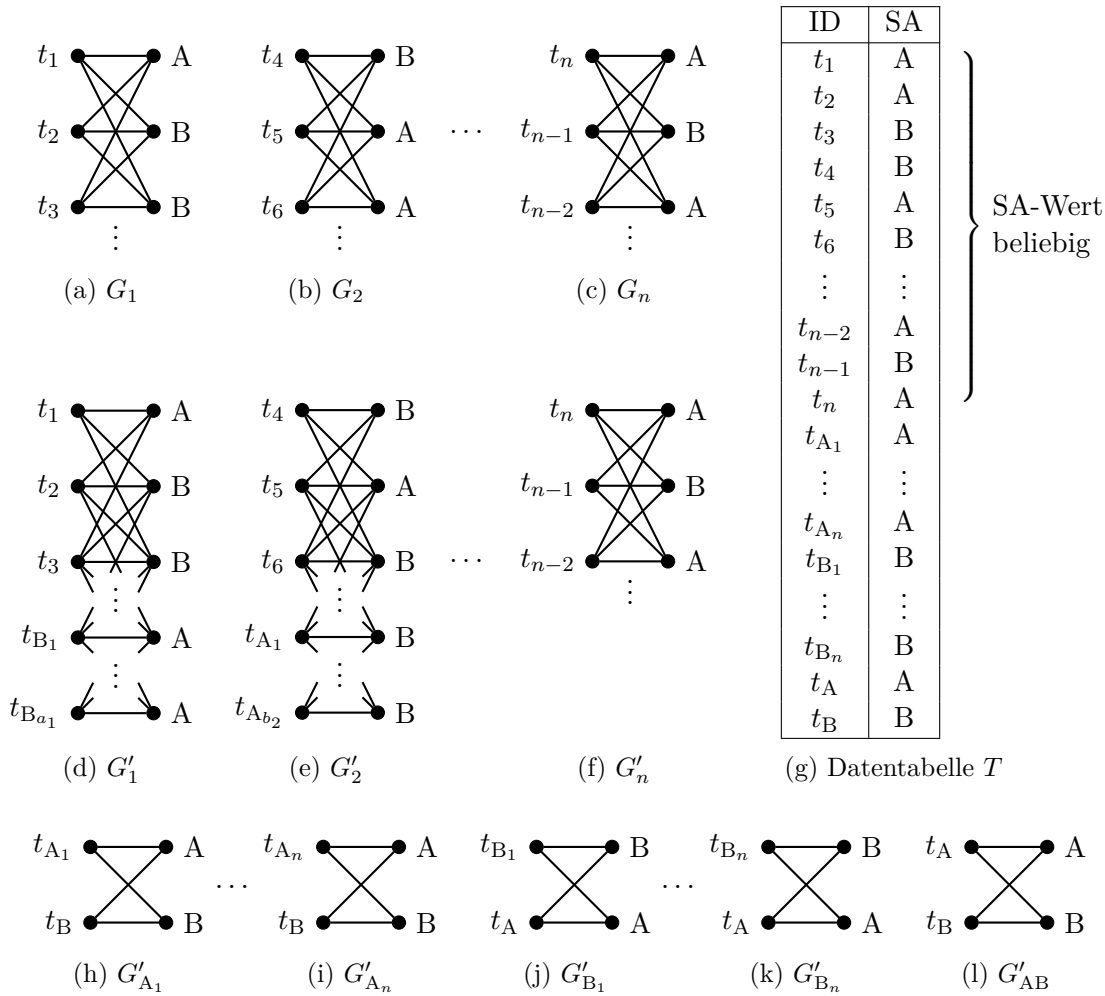
Beweis: Der Beweis folgt direkt aus dem Beweis von Theorem 3.3, da die dort konstruierte Folge von Anfragegraphen genau zwei verschiedene SA-Werte enthält. □

Theorem 3.5 *Das Problem* SA-Werteverteilung *ist NP-vollständig.*

Beweis: Offensichtlich ist SA-Werteverteilung in NP, da die Eigenschaften einer SA-Werteverteilung in Polynomialzeit überprüft werden können. Somit genügt, 2-SA-QGM

[14] wenn P ≠ NP

$\leq_p$ SA-Werteverteilung zu zeigen. Sei $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen, die aus den Tupeln $t_1, \ldots, t_m$ sowie den SA-Werten A und B besteht. Aus $\mathcal{G}^{(n)}$ wird eine andere Folge von Anfragengraphen $\mathcal{G}'^{(n')}$ konstruiert, aus der die Anfragen beziehungsweise Ergebnismengen für SA-Werteverteilung einfach abgeleitet werden können. Dazu wird jedem Tupel t_i zunächst beliebig A oder B als originaler SA-Wert zugewiesen.[15] Seien ferner $t'_{\mathrm{A}}, t'_{\mathrm{A}_1}, \ldots, t'_{\mathrm{A}_m}$ Tupel mit SA-Wert A und $t'_{\mathrm{B}}, t'_{\mathrm{B}_1}, \ldots, t'_{\mathrm{B}_m}$ Tupel mit SA-Wert B, die nicht in $\mathcal{G}^{(n)}$ vorkommen. Dann besteht eine Datentabelle T für mögliche Anfragen aus allen Tupeln t_i, t'_{A}, t'_{A_i}, t'_{B} und t'_{B_i} mit $1 \leq i \leq m$. In Abbildung 3.11 ist in (a) bis (c) ein Beispiel einer Anfragenfolge dargestellt, zu der in (g) eine mögliche Datentabelle angegeben ist.



ID	SA
t_1	A
t_2	A
t_3	B
t_4	B
t_5	A
t_6	B
$\vdots$	$\vdots$
t_{n-2}	A
t_{n-1}	B
t_n	A
t_{A_1}	A
$\vdots$	$\vdots$
t_{A_n}	A
t_{B_1}	B
$\vdots$	$\vdots$
t_{B_n}	B
t_{A}	A
t_{B}	B

Abbildung 3.11: Transformation der Folge von Anfragegraphen $\mathcal{G}^{(n)}$ für 2-SA-QGM in die Folge $\mathcal{G}'^{(n')}$, aus der die Eingabe für SA-Werteverteilung abgelesen werden kann. Die Graphen $G_1, \ldots, G_n$ enthalten beispielhaft die Tupel $t_1, \ldots, t_6, t_{n-2}, \ldots, t_n$ und die angegebenen SA-Werte.

[15] Das bedeutet, dass dieser SA-Wert später in der Datentabelle tatsächlich zu diesem Tupel gehören wird.

Für jeden Graphen G_i bezeichne $n_i(s)$ mit $s \in \{\mathrm{A}, \mathrm{B}\}$ die Anzahl der SA-Knoten mit Label s und $n_i(s_{\mathrm{orig}})$ die Anzahl der vorkommenden Tupel, die den originalen SA-Wert s haben. Wenn $n_i(\mathrm{A}_{\mathrm{orig}}) > n_i(\mathrm{A})$ gilt, werden in G_i je $a_i = n_i(\mathrm{A}_{\mathrm{orig}}) - n_i(\mathrm{A})$ Tupel- und SA-Knoten eingefügt, von denen die Tupelknoten mit $t'_{\mathrm{B}_1}, \ldots, t'_{\mathrm{B}_{a_i}}$ und die SA-Knoten mit A gelabelt sind. Analog werden je $b_i = n_i(\mathrm{B}_{\mathrm{orig}}) - n_i(\mathrm{B})$ Tupel- und SA-Knoten eingefügt, die mit $t'_{\mathrm{A}_1}, \ldots, t'_{\mathrm{A}_{b_i}}$ beziehungsweise B gelabelt sind, wenn $n_i(\mathrm{B}_{\mathrm{orig}}) > n_i(\mathrm{B})$. Offensichtlich gibt es keinen Graphen G_i, bei dem sowohl $n_i(\mathrm{A}_{\mathrm{orig}}) > n_i(\mathrm{A})$ als auch $n_i(\mathrm{B}_{\mathrm{orig}}) > n_i(\mathrm{B})$ gilt. Die entstehenden Anfragegraphen werden mit G'_i bezeichnet und haben aus Paritätsgründen ein perfektes Matching, bei dem jedes Tupel mit seinem originalen SA-Wert matcht.

Seien weiterhin G'_{A_i} und G'_{B_i} mit $1 \leq i \leq n$ Anfragegraphen, die aus je zwei Tupel- und SA-Knoten bestehen. Die Tupelknoten von G'_{A_i} werden mit t'_{B} und t'_{A_i} sowie die Tupelknoten von G'_{B_i} mit t'_{A} und t'_{B_i} gelabelt. Die SA-Knoten werden in beiden Graphen mit A und B gelabelt. Ein letzter Anfragegraph G'_{AB} besteht ebenfalls aus zwei Tupel- und SA-Knoten, wobei die Tupelknoten mit t'_{A} und t'_{B} sowie die SA-Knoten erneut mit A und B gelabelt sind. Offensichtlich existiert in allen Graphen G'_{A_i}, G'_{B_i} und G'_{AB} ein perfektes Matching, bei dem jedes Tupel mit seinem originalen SA-Wert matcht. Sei $n' = 3n + 1$, dann hat die Folge von Anfragegraphen $\mathcal{G}'^{(n')} = (G'_1, \ldots, G'_n, G'_{\mathrm{A}_1}, \ldots, G'_{\mathrm{A}_n}, G'_{\mathrm{B}_1}, \ldots, G'_{\mathrm{B}_n}, G'_{\mathrm{AB}})$ ein valides Originalmatching. Eine mit dieser Vorschrift konstruierte Folge von Anfragegraphen $\mathcal{G}'^{(n')}$ ist in Abbildung 3.11 in (d) bis (f) und (h) bis (l) dargestellt.

Im letzten Schritt wird aus jedem Anfragegraphen $G'_j \in \mathcal{G}'^{(n')}$ eine Ergebnismenge $R_j = (R_{T_j}, R_{S_j})$ von Anfragen erstellt. R_{T_j} entspricht dabei der Menge von Tupeln in G'_j und R_{S_j} der Multimenge aller vorkommenden SA-Werte. Q_j sei eine entsprechende Anfrage, die das Ergebnis R_j hat (für alle $1 \leq j \leq n'$). Dann ist die Eingabe für das Problem SA-Werteverteilung die Folge von Anfragen $\mathcal{Q}^{(n')} = (Q_1, \ldots, Q_{n'})$ zusammen mit den Ergebnissen $\mathcal{R}^{(n')} = (R_1, \ldots, R_{n'})$, das Tupel t'_A und der SA-Wert B.

Es bleibt zu zeigen, dass eine SA-Werteverteilung w von $\mathcal{R}^{(n')}$ mit $(t'_{\mathrm{A}}, \mathrm{B}) \in w$ genau dann existiert, wenn $\mathcal{G}^{(n)}$ ein valides Matching besitzt. Offensichtlich entspricht jede SA-Werteverteilung einem validen Matching in $\mathcal{G}'^{(n')}$. Durch die Konstruktion von $\mathcal{R}^{(n')}$ muss $(t'_{\mathrm{B}}, \mathrm{A}), (t'_{\mathrm{B}_i}, \mathrm{A}), (t'_{\mathrm{A}_i}, \mathrm{B}) \in w$ für alle $1 \leq i \leq m$ gelten. Analog müssen in dem entsprechenden Matching zu w alle Tupel t'_{A} und t'_{A_i} mit B sowie t'_{B} und t'_{B_i} mit A matchen. Werden die Knoten für diese Tupel und SA-Werte aus den Graphen aus $\mathcal{G}'^{(n')}$ entfernt, bleibt die Folge von Anfragegraphen $\mathcal{G}^{(n)}$ übrig. Demzufolge ist w genau dann eine SA-Werteverteilung, wenn $\mathcal{G}^{(n)}$ ein valides Matching besitzt.

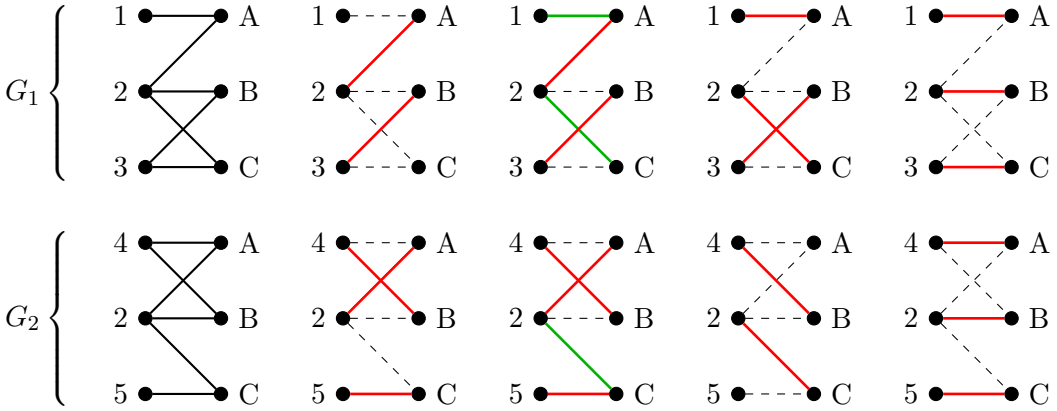
Die vorgestellte Konstruktion ist offensichtlich in Polynomialzeit berechenbar. □

Theorem 3.5 besagt, dass das Finden bestimmter Zuordnungen von SA-Werten zu Tupeln nicht effizient durchgeführt werden kann.[16] Demzufolge gibt es keinen polynomiellen Algorithmus für das eingeführte Problem SA-Werteverteilung. Wie es in exponentieller Zeit gelöst werden kann, zeigen die nächsten Abschnitte.

[16] wenn $\mathsf{P} \neq \mathsf{NP}$

3.7 Einfache Matchingalgorithmen für Anfragegraphen

Im Kapitel 1.6.3 wurden zwei Verfahren vorgestellt, mit denen Matchings in bipartiten Graphen berechnet werden können. Sie basieren beide auf dem sukzessiven Vergrößern von Matchings mithilfe augmentierender Pfade und haben auf einfachen Graphen polynomielle Laufzeit. An dieser Stelle wird gezeigt, dass diese Ansätze erweitert werden müssen, um Matchings in Anfragegraphen zu berechnen, und warum die Laufzeit dadurch nicht mehr polynomiell ist. Im darauffolgenden Abschnitt wird eine elegantere Lösung des Matchingsproblem vorgestellt.



(a) $\mathcal{G}$ (b) $\mathcal{M}_1$ (c) P_1 (d) $\mathcal{M}_2$ (e) $\mathcal{M}_3$

Abbildung 3.12: Verschiedene Matchings für $\mathcal{G} = (G_1, G_2, \dots)$

Abbildung 3.12 zeigt eine Folge von Anfragegraphen $\mathcal{G}$, wobei nur die Graphen G_1 und G_2 abgebildet sind. Es sei vereinbart, dass $\mathcal{G}$ weitere Graphen enthält, sodass nur die hier aufgeführten Kanten zwischen Tupel- und SA-Knoten verlaufen. Beispielsweise geht aus G_1 und G_2 nicht hervor, dass keine Kante $(1, \mathrm{B})$ existiert. Demzufolge muss es in $\mathcal{G}$ mindestens einen weiteren Anfragegraphen geben, in dem ein Knoten für Tupel 1, aber kein Knoten mit SA-Label B vorkommt. Ähnliches folgt für alle weiteren in G_1 und G_2 nicht existierenden Kanten.[17]

Die Algorithmen *Bipartites Matching*[18] und *Hopcroft-Karp*[19] berechnen zunächst ein beliebiges maximales Matching wie zum Beispiel $\mathcal{M}_1$ aus (b). Im Gegensatz zu einfachen Graphen muss bei Anfragegraphen nur beachtet werden, dass das gesuchte größte Matching valid ist. Für das Beispiel bedeutet das, dass die beiden Knoten für Tupel 2 in G_1 und G_2 mit SA-Knoten matchen müssen, die jeweils dasselbe Label (hier A) haben.

Der entscheidende Schritt in beiden Algorithmen ist das Bestimmen eines augmentierenden Pfades. In G_1 existieren mit 1 und C zwei unüberdeckte Knoten, wobei der Einfachheit halber in diesem Beispiel Knoten mithilfe ihrer Labels bezeichnet werden. In G_2 ist jeder Knoten überdeckt, das heißt, $\mathcal{M}_1$ ist perfekt in G_2. Mögliche augmentierende Pfade sind demnach nur $P_1 = 1, \mathrm{A}, 2, \mathrm{C}$ und $P_2 = 1, \mathrm{A}, 2, \mathrm{B}, 3, \mathrm{C}$ in G_1. Während Algorithmus *Bipartites Matching* einen beliebigen Pfad zum Erweitern sucht, wählt Algorithmus

[17] Die besagten zusätzlichen Graphen würden das Beispiel nur unnötig vergrößern und sind für die folgenden Überlegungen nicht relevant. Daher wurde auf ihre Darstellung hier verzichtet.
[18] Vgl. Algorithmus 1.1 auf Seite 18.
[19] Vgl. Algorithmus 1.2 auf Seite 20.

Hopcroft-Karp in jedem Fall den kürzeren, also P_1. Beim Augmentieren (vgl. (c)) wird dabei in G_1 der Knoten 2 mit dem SA-Knoten C verbunden. Aufgrund der Validitätseigenschaft muss auch in G_2 der Knoten von Tupel 2 mit dem SA-Knoten mit Label C verbunden werden. Folglich muss die Kante $(5, \mathrm{C})$ aus dem Matching entfernt werden, damit der SA-Knoten C nicht mehrfach überdeckt wird. Da der Knoten 5 allerdings keinen weiteren Nachbarknoten hat, kann er keinen anderen Matchingpartner bekommen. Das entstehende Matching $\mathcal{M}_2$ ist in (d) dargestellt und besteht wie $\mathcal{M}_1$ aus fünf Kanten. Es gilt $|\mathcal{M}_1[G_1]| < |\mathcal{M}_2[G_1]|$ und $|\mathcal{M}_1[G_2]| > |\mathcal{M}_2[G_2]|$, was bedeutet, dass der ausgewählte Pfad P_1 zwar $\mathcal{M}_1[G_1]$-augmentierend ist, aber zu einer Verkleinerung von $\mathcal{M}_1[G_2]$ führt.

Auch ein erneutes Erweitern anhand des jetzt kürzesten augmentierenden Pfades $5, \mathrm{C}, 2, \mathrm{A}$ in G_2 würde wieder zu Matching $\mathcal{M}_1$ führen, denn in diesem Fall müsste auch 2 in G_1 mit A matchen, sodass 1 keinen Matchingpartner mehr hat. Das Ergebnis ist, dass die Standardvariante des *Hopcroft-Karp*-Algorithmus in diesem Beispiel zu einer Endlosschleife führt.

Wird im ersten Schritt allerdings P_2 zum Erweitern ausgewählt, so entsteht das Matching $\mathcal{M}_3$ aus (e). Hierbei müssen in G_2 nur zwei Kanten umgesetzt werden und $\mathcal{M}_3$ ist perfekt. Der Algorithmus *Bipartites Matching* könnte diese Lösung finden, er kann aber auch wie der *Hopcroft-Karp*-Algorithmus in die Endlosschleife geraten.

Aus diesem Beispiel kann der Schluss gezogen werden, dass beide vorgestellte Matchingalgorithmen nicht zwangsläufig auch valide, das heißt insbesondere perfekte, Matchings in Anfragegraphen finden. Auch ist die Wahl des augmentierenden Pfades von zentraler Bedeutung. Zusätzlich ist an dieser Stelle nicht klar, dass die Grundidee, Matchings mithilfe von augmentierenden Pfaden zu vergrößern, auch in Anfragegraphen möglich ist. Die Existenz solcher Pfade kann zwar aus dem Beweis des Satzes von Berge[20] analog auch für Anfragegraphen gefolgert werden. Dass das Augmentieren anhand von Pfaden auch das Matching vergrößert, ist hingegen nicht offensichtlich und müsste jeweils getestet werden. Das bedeutet, für alle Knoten außerhalb des augmentierenden Pfades, die durch die Validitätseigenschaft ebenfalls „umgematcht“ werden müssen, könnten sämtliche in Frage kommenden Matchingpartner ausprobiert werden. Falls dabei ein größeres Matching gefunden wird, kann der Augmentationsschritt angewendet werden.

Auf eine konkrete Angabe eines entsprechenden Algorithmus wird an dieser Stelle verzichtet, da dieser nur wenig effizienter ist als ein Brute-Force-Ansatz und insbesondere exponentielle Laufzeit hat. Stattdessen wird im nächsten Kapitel ein viel eleganterer Weg gezeigt, mit dem valide Matchings in Anfragegraphen bestimmt werden können.

3.8 Berechnung von Matchings mithilfe linearer Programmierung

3.8.1 Konstruktion des ganzzahligen linearen Programms

Eine andere Möglichkeit, um größte Matchings in bipartiten Graphen zu bestimmen, wurde in Kapitel 1.7.4 vorgestellt. Dabei wird aus dem Matchingproblem ein ganzzahliges lineares Programm (ILP) erstellt. Die optimale Lösung dieses ILP entspricht einem größten Matching in einem gegebenen Graphen. Da Matchings in Anfragegraphen zusätzlich noch valid sein müssen, muss die vorgestellte Transformation angepasst werden.

[20] Vgl. Satz 1.1 auf Seite 15.

Sei $\mathcal{G} = (G_1, \ldots, G_n)$ ein Anfragegraph. Die Grundidee der bereits vorgestellten Modellierung bleibt erhalten. Das bedeutet, es wird für jede Kante eine 0-1-Variable eingeführt, die genau dann 1 ist, wenn die Kante zum Matching gehört. Da Kanten in verschiedenen Teilgraphen G_i und G_j zwischen Tupel- und SA-Knoten, die jeweils dasselbe Label haben, nicht unabhängig voneinander in validen Matchings vorkommen, wird jeweils dieselbe Variable verwendet. Das bedeutet, für jede Kante zwischen einem Tupelknoten, der mit t gelabelt ist, und einem SA-Knoten, der mit s gelabelt ist, wird eine Variable $x_{ts} \in \{0, 1\}$ erzeugt. Sie ist genau dann 1, wenn t in jedem Teilgraphen, in dem ein Tupelknoten mit Label t vorkommt, mit s matcht.

Um die Matchingbedingung zu modellieren, wird zwischen Tupel- und SA-Knoten unterschieden. Jedem Tupel t weise δ_s die Menge aller SA-Labels zu, mit denen t über eine Kante verbunden ist. Analog weise δ_t jedem SA-Wert s in einem Teilgraphen G_i eine Menge von Tupellabels zu.[21] Da jeder Tupelknoten v_t in einem validen Matching von genau einer Kante überdeckt wird, muss für jedes dazugehörige Tupel t

$$\sum_{s \in \delta_s(t)} x_{ts} = 1 \tag{3.1}$$

gelten. Für jedes Tupel wird somit nur eine Gleichung erzeugt, unabhängig davon, in welchen Teilgraphen es vorkommt. Zwischen SA-Knoten mit gleichem Label in verschiedenen Teilgraphen bestehen keine Abhängigkeiten, wodurch sie für jeden Teilgraphen gesondert betrachtet werden. Weiterhin sind die Kanten zwischen einem Tupelknoten v_t und zwei verschiedenen SA-Knoten, die das gleiche Label s besitzen, mit derselben Variable x_{ts} bezeichnet worden. Daher können die Bedingungen für SA-Knoten mit identischem Label s pro Teilgraph G_i zusammengefasst werden. Bezeichne $n_{G_i}(s)$ die Anzahl der SA-Knoten in G_i mit Label s, dann muss die Gleichung

$$\sum_{t \in \delta_t(s)} x_{ts} = n_{G_i}(s) \tag{3.2}$$

für alle $1 \leq i \leq n$ und jeden SA-Wert s erstellt werden. Die Zielfunktion des ganzzahligen linearen Programms misst die Anzahl der Kanten in einem Matching und lautet

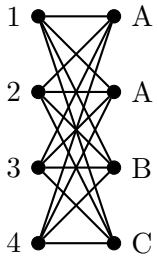
$$\max \sum_{(t,s) \in E(\mathcal{G})} x_{ts}. \tag{3.3}$$

Zusammenfassend entsteht zu einer gegebenen Folge von Anfragegraphen $\mathcal{G} = (G_1, \ldots, G_n)$ das ganzzahlige lineare Programm $P_M(\mathcal{G})$ mit

$$\max \left\{ \sum_{(t,s) \in E(\mathcal{G})} x_{ts} \;\middle|\; \begin{aligned} \sum_{s \in \delta_s(t)} x_{ts} &= 1, && \text{für alle Tupel } t \text{ in } \mathcal{G}, \\ \sum_{t \in \delta_t(G_i, s)} x_{ts} &= n_{G_i}(s), && \text{für alle } G_i \in \mathcal{G}, \text{ SA-Werte } s \text{ in } G_i, \\ x_{ts} &\in \{0, 1\}, && \text{für alle } (t, s) \in E(\mathcal{G}) \end{aligned} \right\}. \tag{3.4}$$

[21] $\delta_s(t) \coloneqq \{s \mid (t, s) \in E(\mathcal{G})\}, \delta_t(G_i, s) \coloneqq \{t \mid (t, s) \in E(G_i), G_i \in \mathcal{G}\}$

Offensichtlich hat jede Lösung von $P_M(\mathcal{G})$ nach Gleichung 3.1 den Wert $|\{t \mid (t,s) \in E(\mathcal{G})\}|$, ist damit optimal und entspricht einem validen Matching in $\mathcal{G}$. Das Erstellen der ganzzahligen linearen Programme wird im Folgenden für die anfangs dieses Kapitels eingeführten Beispielanfragen und dazugehörigen Anfragegraphen erläutert.



(a) G_1

$$
\begin{aligned}
1\colon &\quad x_{1\mathrm{A}} + x_{1\mathrm{B}} + x_{1\mathrm{C}} = 1\\
2\colon &\quad x_{2\mathrm{A}} + x_{2\mathrm{B}} + x_{2\mathrm{C}} = 1\\
3\colon &\quad x_{3\mathrm{A}} + x_{3\mathrm{B}} + x_{3\mathrm{C}} = 1\\
4\colon &\quad x_{4\mathrm{A}} + x_{4\mathrm{B}} + x_{4\mathrm{C}} = 1\\
G_1,\mathrm{A}\colon &\quad x_{1\mathrm{A}} + x_{2\mathrm{A}} + x_{3\mathrm{A}} + x_{4\mathrm{A}} = 2\\
G_1,\mathrm{B}\colon &\quad x_{1\mathrm{B}} + x_{2\mathrm{B}} + x_{3\mathrm{B}} + x_{4\mathrm{B}} = 1\\
G_1,\mathrm{C}\colon &\quad x_{1\mathrm{C}} + x_{2\mathrm{C}} + x_{3\mathrm{C}} + x_{4\mathrm{C}} = 1
\end{aligned}
$$

(b) Nebenbedingungen von $P_M(\mathcal{G}^{(1)})$

Abbildung 3.13: ILP für $\mathcal{G}^{(1)} = (G_1)$

In Abbildung 3.13 ist in (a) der Anfragegraph $\mathcal{G}^{(1)} = (G_1)$ dargestellt.[22] Für die vier Knoten der Tupel 1, 2, 3 und 4 wird jeweils eine Nebenbedingung analog zu Gleichung 3.1 erstellt (siehe (b)). Dabei stehen beispielsweise die Variablen $x_{t\mathrm{A}}$ für alle Kanten, die zwischen dem Knoten für t und einem der beiden SA-Knoten für A verlaufen. Für die SA-Werte A, B und C wird analog zu Gleichung 3.2 jeweils eine Bedingung erstellt. Während in G_1 zwei Knoten mit Label A vorkommen und dementsprechend $n_{G_1}(\mathrm{A}) = 2$ gilt, existiert nur ein Knoten mit SA-Label B beziehungsweise C. Die weiteren Bedingungen $x_{ts} \in \{0,1\}$ für alle t und s sind aus Platzgründen in (b) und in den folgenden Beispielen nicht dargestellt.

Eine mögliche Lösung von $P_M(\mathcal{G}^{(1)})$ ist

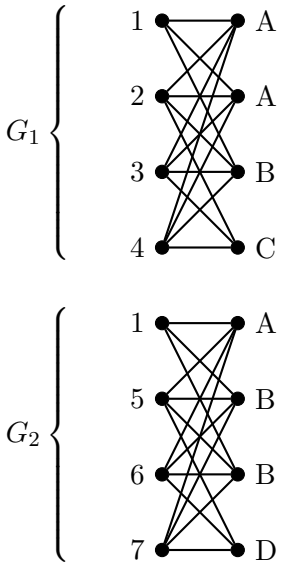
$$x_{1\mathrm{B}} = x_{2\mathrm{A}} = x_{3\mathrm{A}} = x_{4\mathrm{C}} = 1$$

und $x_{ts} = 0$ für alle anderen Variablen. Sie entspricht einem Matching, das aus den Labelkanten $(1,\mathrm{B}), (2,\mathrm{A}), (3,\mathrm{A})$ und $(4,\mathrm{C})$ besteht und bereits in Abbildung 3.5 auf Seite 78 dargestellt wurde. Da im Graphen genau vier Tupel vorkommen, hat jede optimale Lösung den Wert 4.

Für die Folge von Anfragegraphen $\mathcal{G}^{(2)} = (G_1, G_2)$ aus Abbildung 3.14 sind in (b) die entsprechenden Nebenbedingungen des Problems $P_M(\mathcal{G}^{(2)})$ dargestellt. Der Graph G_1 aus $\mathcal{G}^{(2)}$ unterscheidet sich zu dem aus $\mathcal{G}^{(1)}$ durch eine fehlende Kante $(1,\mathrm{C})$. Demzufolge gibt es keine Variable $x_{1\mathrm{C}}$ im linearen Programm. Wie oben beschrieben müssen die Bedingungen für die SA-Werte für jeden Teilgraphen G_1 und G_2 gesondert erstellt werden und führen zu den angegebenen Gleichungen. Eine mögliche Lösung[23] des ILP ist

$$x_{1\mathrm{A}} = x_{2\mathrm{A}} = x_{3\mathrm{B}} = x_{4\mathrm{C}} = x_{5\mathrm{B}} = x_{6\mathrm{B}} = x_{7\mathrm{D}} = 1,$$

[22] Vgl. Abbildung 3.2 auf Seite 76.
[23] Erneut haben alle nicht aufgeführten Variablen den Wert 0.



(a) $\mathcal{G}^{(2)}$

$$
\begin{aligned}
1\colon&\quad x_{1\mathrm{A}} + x_{1\mathrm{B}} = 1\\
2\colon&\quad x_{2\mathrm{A}} + x_{2\mathrm{B}} + x_{2\mathrm{C}} = 1\\
3\colon&\quad x_{3\mathrm{A}} + x_{3\mathrm{B}} + x_{3\mathrm{C}} = 1\\
4\colon&\quad x_{4\mathrm{A}} + x_{4\mathrm{B}} + x_{4\mathrm{C}} = 1\\
5\colon&\quad x_{5\mathrm{A}} + x_{5\mathrm{B}} + x_{5\mathrm{D}} = 1\\
6\colon&\quad x_{6\mathrm{A}} + x_{6\mathrm{B}} + x_{6\mathrm{D}} = 1\\
7\colon&\quad x_{7\mathrm{A}} + x_{7\mathrm{B}} + x_{7\mathrm{D}} = 1\\
G_1,\mathrm{A}\colon&\quad x_{1\mathrm{A}} + x_{2\mathrm{A}} + x_{3\mathrm{A}} + x_{4\mathrm{A}} = 2\\
G_1,\mathrm{B}\colon&\quad x_{1\mathrm{B}} + x_{2\mathrm{B}} + x_{3\mathrm{B}} + x_{4\mathrm{B}} = 1\\
G_1,\mathrm{C}\colon&\quad x_{2\mathrm{C}} + x_{3\mathrm{C}} + x_{4\mathrm{C}} = 1\\
G_2,\mathrm{A}\colon&\quad x_{1\mathrm{A}} + x_{5\mathrm{A}} + x_{6\mathrm{A}} + x_{7\mathrm{A}} = 1\\
G_2,\mathrm{B}\colon&\quad x_{1\mathrm{B}} + x_{5\mathrm{B}} + x_{6\mathrm{B}} + x_{7\mathrm{B}} = 2\\
G_2,\mathrm{D}\colon&\quad x_{5\mathrm{D}} + x_{6\mathrm{D}} + x_{7\mathrm{D}} = 1
\end{aligned}
$$

(b) Nebenbedingungen von $P_M(\mathcal{G}^{(2)})$

Abbildung 3.14: ILP für $\mathcal{G}^{(2)} = (G_1, G_2)$

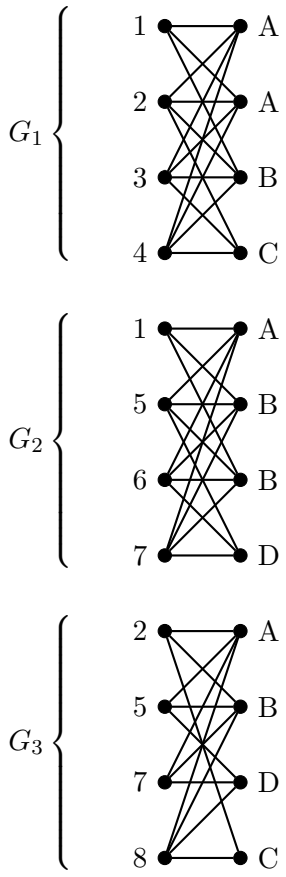
die dem Originalmatching von $\mathcal{G}^{(2)}$ entspricht, das heißt dem Matching, welches aus allen horizontalen Kanten besteht (vgl. Matching $\mathcal{M}_{1.1}$ in Abbildung A.7c auf Seite 276).

In Abbildung 3.15 ist ein letzter Anfragegraph zur Folge hinzugefügt worden. Da die Graphen G_1 und G_2 identisch zu denen aus $\mathcal{G}^{(2)}$ sind, müssen zu $P_M(\mathcal{G}^{(2)})$ nur Gleichungen hinzugefügt werden, um $P_M(\mathcal{G}^{(3)})$ zu erstellen. Es entstehen Gleichungen für das neue Tupel 8 und den neuen Graphen G_3. Auch hier entspricht jede optimale Lösung einem validen Matching in $\mathcal{G}^{(3)}$.

3.8.2 Relaxation der Nebenbedingungen

In Kapitel 1.7.5 wurde gezeigt, dass das Bestimmen von Matchings in bipartiten Graphen mithilfe ganzzahliger linearer Programmierung in Polynomialzeit möglich ist. Der Grund dafür war, dass die Ganzzahligkeitsbedingungen des Programms weggelassen werden konnten, ohne dass sich die Lösungsmenge entscheidend verändert hat. Somit wurde ein sogenanntes relaxiertes lineares Programm erstellt, das im Gegensatz zum ganzzahligen Fall in Polynomialzeit gelöst werden konnte. Für die hier vorgestellte Modellierung von Anfragegraphen ist dieses Vorgehen aber nicht korrekt. Das bedeutet, die Relaxation führt zu optimalen Lösungen, denen kein Matching entspricht und bei denen auch keine ganzzahlige Lösung existiert. Im Folgenden wird diese Behauptung anhand eines Beispiels veranschaulicht.

Abbildung 3.16 zeigt in (a) eine gegebene Folge von Anfragegraphen $\mathcal{G}$. Die dazugehörigen Nebenbedingungen des ganzzahligen linearen Programms $P_M(\mathcal{G})$ sind als Gleichungssystem in (b) angegeben. Die entsprechende Matrixschreibweise ist in (c) aufgeführt. Die Bedingungen $x_{ts} \in \{0, 1\}$ für alle t und s können nach Satz 1.7 auf Seite 33 weggelassen



(a) $\mathcal{G}^{(3)}$

$$
\begin{aligned}
1\colon&\quad x_{1\mathrm{A}} + x_{1\mathrm{B}} = 1\\
2\colon&\quad x_{2\mathrm{A}} + x_{2\mathrm{B}} + x_{2\mathrm{C}} = 1\\
3\colon&\quad x_{3\mathrm{A}} + x_{3\mathrm{B}} + x_{3\mathrm{C}} = 1\\
4\colon&\quad x_{4\mathrm{A}} + x_{4\mathrm{B}} + x_{4\mathrm{C}} = 1\\
5\colon&\quad x_{5\mathrm{A}} + x_{5\mathrm{B}} + x_{5\mathrm{D}} = 1\\
6\colon&\quad x_{6\mathrm{A}} + x_{6\mathrm{B}} + x_{6\mathrm{D}} = 1\\
7\colon&\quad x_{7\mathrm{A}} + x_{7\mathrm{B}} + x_{7\mathrm{D}} = 1\\
8\colon&\quad x_{8\mathrm{A}} + x_{8\mathrm{B}} + x_{8\mathrm{C}} + x_{8\mathrm{D}} = 1\\
G_1,\mathrm{A}\colon&\quad x_{1\mathrm{A}} + x_{2\mathrm{A}} + x_{3\mathrm{A}} + x_{4\mathrm{A}} = 2\\
G_1,\mathrm{B}\colon&\quad x_{1\mathrm{B}} + x_{2\mathrm{B}} + x_{3\mathrm{B}} + x_{4\mathrm{B}} = 1\\
G_1,\mathrm{C}\colon&\quad x_{2\mathrm{C}} + x_{3\mathrm{C}} + x_{4\mathrm{C}} = 1\\
G_2,\mathrm{A}\colon&\quad x_{1\mathrm{A}} + x_{5\mathrm{A}} + x_{6\mathrm{A}} + x_{7\mathrm{A}} = 1\\
G_2,\mathrm{B}\colon&\quad x_{1\mathrm{B}} + x_{5\mathrm{B}} + x_{6\mathrm{B}} + x_{7\mathrm{B}} = 2\\
G_2,\mathrm{D}\colon&\quad x_{5\mathrm{D}} + x_{6\mathrm{D}} + x_{7\mathrm{D}} = 1\\
G_3,\mathrm{A}\colon&\quad x_{2\mathrm{A}} + x_{5\mathrm{A}} + x_{7\mathrm{A}} + x_{8\mathrm{A}} = 1\\
G_3,\mathrm{B}\colon&\quad x_{2\mathrm{B}} + x_{5\mathrm{B}} + x_{7\mathrm{B}} + x_{8\mathrm{B}} = 1\\
G_3,\mathrm{C}\colon&\quad x_{2\mathrm{C}} + x_{8\mathrm{C}} = 1\\
G_3,\mathrm{D}\colon&\quad x_{5\mathrm{D}} + x_{7\mathrm{D}} + x_{8\mathrm{D}} = 1
\end{aligned}
$$

(b) Nebenbedingungen von $P_M(\mathcal{G}^{(3)})$

Abbildung 3.15: ILP für $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$

werden, wenn die Matrix A total unimodular ist. In diesem Fall wäre die Relaxation exakt und das Problem in polynomieller Zeit lösbar.[24] Da aber die Teilmatrix

$$
A_{\mathrm{A}} = \begin{array}{c} \\ G_1,\mathrm{A} \\ G_2,\mathrm{A} \\ G_3,\mathrm{A} \end{array}
\begin{array}{c}
\begin{array}{ccc} x_{1\mathrm{A}} & x_{2\mathrm{A}} & x_{3\mathrm{A}} \end{array} \\
\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}
\end{array}
$$

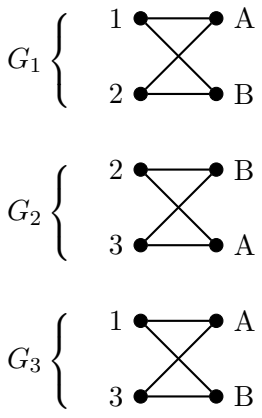
nach dem Laplaceschen Entwicklungssatz [50] (Entwicklung nach der ersten Zeile) die Determinante

$$\det(A_{\mathrm{A}}) = 1 \cdot 1 - 1 \cdot (-1) + 0 = 2$$

hat, kann A nicht total unimodular sein. Weiterhin ist

$$x_{1\mathrm{A}} = x_{1\mathrm{B}} = x_{2\mathrm{A}} = x_{2\mathrm{B}} = x_{3\mathrm{A}} = x_{3\mathrm{B}} = 0{,}5$$

[24] Es ist leicht zu sehen, dass die Inzidenzmatrizen der einzelnen Graphen G_1, G_2 und G_3 zwar in A enthalten sind, A selbst aber nicht die Inzidenzmatrix eines Graphen ist. Demzufolge gilt Korollar 1.9 nicht, das besagt, dass bipartite Graphen eine total unimodulare Inzidenzmatrix haben.



(a) $\mathcal{G}$

$$
\begin{aligned}
1\colon &\quad x_{1\mathrm{A}} + x_{1\mathrm{B}} = 1\\
2\colon &\quad x_{2\mathrm{A}} + x_{2\mathrm{B}} = 1\\
3\colon &\quad x_{3\mathrm{A}} + x_{3\mathrm{B}} = 1\\
G_1,\mathrm{A}\colon &\quad x_{1\mathrm{A}} + x_{2\mathrm{A}} = 1\\
G_1,\mathrm{B}\colon &\quad x_{1\mathrm{B}} + x_{2\mathrm{B}} = 1\\
G_2,\mathrm{A}\colon &\quad x_{2\mathrm{A}} + x_{3\mathrm{A}} = 1\\
G_2,\mathrm{B}\colon &\quad x_{2\mathrm{B}} + x_{3\mathrm{B}} = 1\\
G_3,\mathrm{A}\colon &\quad x_{1\mathrm{A}} + x_{3\mathrm{A}} = 1\\
G_3,\mathrm{B}\colon &\quad x_{1\mathrm{B}} + x_{3\mathrm{B}} = 1
\end{aligned}
$$

(b) Nebenbedingungen des ILP

$$
\underbrace{\begin{array}{c} \\ 1\\2\\3\\G_1,\mathrm{A}\\G_1,\mathrm{B}\\G_2,\mathrm{A}\\G_2,\mathrm{B}\\G_3,\mathrm{A}\\G_3,\mathrm{B}\end{array}\!\!\begin{array}{c}\begin{array}{cccccc} x_{1\mathrm{A}} & x_{1\mathrm{B}} & x_{2\mathrm{A}} & x_{2\mathrm{B}} & x_{3\mathrm{A}} & x_{3\mathrm{B}}\end{array}\\ \begin{pmatrix} 1&1&0&0&0&0\\0&0&1&1&0&0\\0&0&0&0&1&1\\1&0&1&0&0&0\\0&1&0&1&0&0\\0&0&1&0&1&0\\0&0&0&1&0&1\\1&0&0&0&1&0\\0&1&0&0&0&1\end{pmatrix}\end{array}}_{A} \cdot \underbrace{\begin{pmatrix} x_{1\mathrm{A}}\\x_{1\mathrm{B}}\\x_{2\mathrm{A}}\\x_{2\mathrm{B}}\\x_{3\mathrm{A}}\\x_{3\mathrm{B}}\end{pmatrix}}_{x} = \underbrace{\begin{pmatrix}1\\1\\1\\1\\1\\1\\1\\1\\1\end{pmatrix}}_{b}
$$

(c) Nebenbedingungen in Matrixschreibweise $Ax = b$

Abbildung 3.16: Eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3)$ und Nebenbedingungen des dazugehörigen ILP für QGM. Zusätzlich gilt $x_{ts} \in \{0, 1\}$ für alle $t \in \{1, 2, 3\}$ und $s \in \{\mathrm{A}, \mathrm{B}\}$.

eine optimale Lösung der LP-Relaxation mit dem Wert $\sum x_{ts} = 3$. Sie entspricht aber keinem Matching, da die Variablen x_{ts} nur die Werte 0 oder 1 annehmen können. Da es keine weitere Lösung gibt,[25] ist die LP-Relaxation für dieses Beispiel nicht exakt. Die Ganzzahligkeitsbedingungen können folglich nicht weggelassen werden und das Problem ist im Allgemeinen nicht in Polynomialzeit lösbar.

Das Ergebnis dieser Überlegungen ist, dass QGM zwar mithilfe linearer Programmierung gelöst werden kann, dazu aber ein ganzzahliges lineares Programm erstellt werden muss. Die Berechnung einer optimalen Lösung ist dann mit Standardverfahren möglich, hat allerdings exponentielle Laufzeit.

[25] Aus den Nebenbedingungen folgt für jede Lösung $x_{1\mathrm{A}} = x_{1\mathrm{B}} = x_{2\mathrm{A}} = x_{2\mathrm{B}} = x_{3\mathrm{A}} = x_{3\mathrm{B}}$, was genau dann erfüllt ist, wenn alle Variablen den Wert 0,5 haben.

3.8.3 Lösungsalgorithmus

Als Zusammenfassung der vorigen Betrachtungen wird nun ein Algorithmus vorgestellt, der das Problem SA-Werteverteilung löst. Sei dazu eine Folge von Anfragen $\mathcal{Q}^{(n)}$ mit dazugehörigen Ergebnissen $\mathcal{R}^{(n)}$, ein Tupel t und ein SA-Wert s gegeben. Gesucht ist eine SA-Werteverteilung w, die das Paar (t, s) enthält.

Zuerst wird aus den Anfragen beziehungsweise Ergebnissen eine Folge von Anfragegraphen $\mathcal{G}^{(n)}$ erstellt, in denen ein Matching $\mathcal{M}^{(n)}$ bestimmt werden soll. Damit dieses die Kante (t, s) auf jeden Fall enthält, wird sie inklusive beider Endknoten aus allen Teilgraphen entfernt. Für den modifizierten Graphen wird das ganzzahlige lineare Programm $P_M(\mathcal{G})$ konstruiert, welches mit Standardverfahren gelöst werden kann. Dabei entspricht eine Lösung von $P_M(\mathcal{G})$ einem validen Matching in $\mathcal{G}^{(n)}$, welches wiederum in eine SA-Werteverteilung w transformiert werden kann. Für jede Kante (t', s') des Matchings wird das Paar (t', s') in die Verteilung aufgenommen. Algorithmus 3.1 stellt das vorgestellte Verfahren um.

Algorithmus 3.1 Berechne SA-Werteverteilung

Eingabe: $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$: Folge von Anfragen, $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$: Folge von dazugehörigen Ergebnissen, t: Tupel, s: SA-Wert
Ausgabe: w: SA-Werteverteilung von $\mathcal{R}^{(n)}$ mit $(t, s) \in w$, falls vorhanden
1: Erstelle zu $\mathcal{Q}^{(n)}$ die dazugehörige Folge von Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$
2: Entferne aus allen Graphen G_i mit $t \in G_i$ eine Kante (t, s) inkl. beider Endknoten
3: Gib Fehler aus, falls das nicht möglich
4: Erstelle ILP $P_M(\mathcal{G}^{(n)})$ nach Formel 3.4
5: Berechne Lösung von $P_M(\mathcal{G}^{(n)})$ mit beliebigem (exponentiellem) ILP-Algorithmus und dazugehöriges Matching $\mathcal{M}^{(n)}$
6: Gib Fehler aus, falls keine Lösung existiert
7: Erstelle SA-Werteverteilung w mit: $(t', s') \in w :\Leftrightarrow (t', s') \in \mathcal{M}^{(n)}$
8: **return** $w \cup (t, s)$

3.9 Test auf k-assign Anonymität

Mithilfe des Algorithmus *Berechne SA-Werteverteilung* kann überprüft werden, ob eine gegebene Folge von Anfragen $\mathcal{Q}^{(n)}$ mit Ergebnissen $\mathcal{R}^{(n)}$ eine Verletzung der Privatsphäre darstellt oder nicht. Sei k-assign Anonymität das Privatsphärenkriterium, dann muss getestet werden, ob für jedes Tupel t mindestens k SA-Werteverteilungen existieren, die t jeweils einen anderen SA-Wert zuordnen. Für $\mathcal{G}^{(n)}$ bedeutet das, dass für jedes t insgesamt k Matchings existieren, bei denen der Matchingpartner von t jeweils ein anderes SA-Label hat.

Algorithmus 3.2 stellt das entsprechende Vorgehen dar. Als Erstes wird für jedes Tupel $t \in \mathcal{R}^{(n)}$ eine Menge S potentieller SA-Werte berechnet. Das sind alle Werte, die in allen Ergebnissen vorkommen, in denen auch t vorkommt. Diese Menge stellt eine Obermenge aller möglicher sensibler Werte dar, die das Tupel haben kann. Als Zweites wird mithilfe von Algorithmus 3.1 für jedes Element $s \in S$ eine SA-Werteverteilung w bestimmt, die das Paar (t, s) enthält. s gehört genau dann zur Signatur von t, wenn w existiert. Aus

Algorithmus 3.2 Teste k-assign Anonymität

Eingabe: $\mathcal{Q}^{(n)} = (Q_1, \ldots, Q_n)$: Folge von Anfragen, $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$: Folge von dazugehörigen Ergebnissen, k: natürliche Zahl
Ausgabe: true, falls $\mathcal{R}^{(n)}$ k-assign anonym ist; false, andernfalls

```
1: for all Tupel t ∈ R^(n) do
2:   Bestimme Menge S möglicher SA-Werte von t (S := ⋂_{1≤i≤n mit t∈R_{T_i}} R*_{S_i})
3:   for all s ∈ S do
4:     Teste mit Berechne SA-Werteverteilung, ob s in Signatur Sig_{R^(n)}(t) von t
5:   end for
6:   if (a_{R^(n)}(t) ← |Sig_{R^(n)}(t)|) < k then
7:     return false
8:   end if
9: end for
10: return true
```

der Signatur kann der Anonymitätsgrad $a_{\mathcal{R}^{(n)}}(t)$ abgelesen und testet werden, ob er kleiner als das eingegebene k ist. In diesem Fall liegt eine Verletzung der Privatsphäre vor (vgl. Definition 3.7) und der Algorithmus gibt „false“ aus. Wenn alle Anonymitätsgrade mindestens k sind, wird im letzten Schritt „true“ ausgegeben.

3.10 Alternative Darstellungen

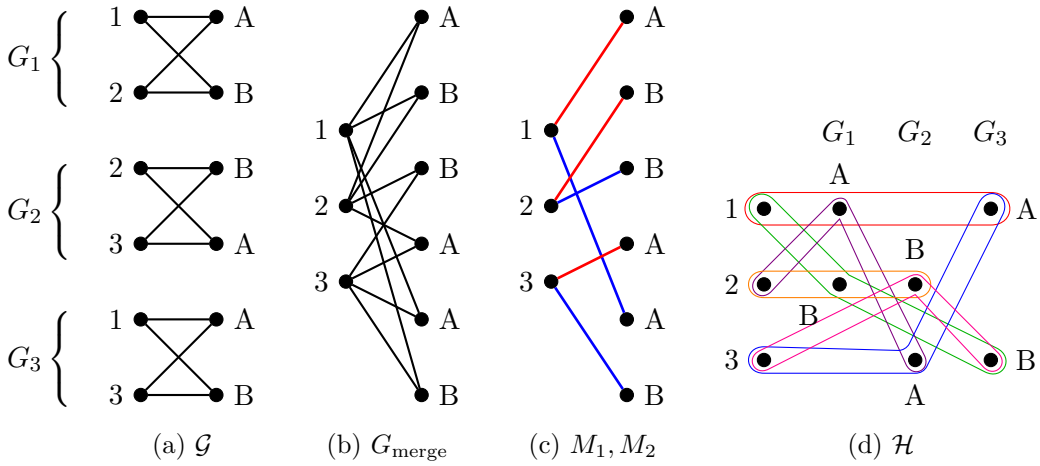
Neben dem bisher beschriebenen Vorgehen, um das Wissen eines Angreifers zu berechnen, existieren noch weitere Möglichkeiten, auf die an dieser Stelle kurz eingegangen wird. Dabei werden die jeweiligen Nachteile erläutert, die dazu geführt haben, die Modellierung mit Anfragegraphen zu wählen.

3.10.1 Modellierung mittels Hypergraphen

Ein Nachteil der Modellierung mit Anfragegraphen ist, dass für jede Anfrage beziehungsweise jedes Ergebnis ein Graph erstellt werden muss. Hinzu kommt, dass Matchings in einzelnen Teilgraphen nicht unabhängig voneinander konstruiert werden können. Genau dieser Zusammenhang führt dazu, dass die Lösung nicht mehr in Polynomialzeit berechnet werden kann. Es wäre womöglich effizienter, wenn nur ein Graph ausreichen würde, aus dem das gesamte Wissen des Angreifers abgeleitet werden kann. Um das Problem zu vereinfachen, sollte zum Beispiel jedes Tupel nur durch einen Knoten repräsentiert werden.

In Abbildung 3.17 ist in (a) eine Folge von Anfragegraphen $\mathcal{G}$ gegeben. Werden diese drei Teilgraphen beispielsweise zu einem Gesamtgraphen vereinigt, in dem es für jedes Tupel genau einen Knoten und für jeden in einem Ergebnis zu einer Anfrage vorkommenden SA-Wert einen Knoten gibt, entsteht der Graph G_{merge} aus (b). Es gilt, dass jedes Matching in einem einzelnen Teilgraphen G_1, G_2 oder G_3 auch ein Matching in G_{merge} darstellt. Daneben existieren mit M_1 und M_2 aus (c) auch Matchings, die jeweils alle Tupelknoten überdecken und zusammen auch alle SA-Knoten.

Nichtsdestotrotz entspricht weder M_1, M_2 noch die Vereinigung beider Matchings einer SA-Werteverteilung. Ein Grund dafür ist, dass in G_{merge} nicht alle Informationen abgebildet werden, die auch in $\mathcal{G}$ enthalten sind. Beispielsweise geht aus G_1 und G_2 hervor, dass



(a) $\mathcal{G}$ (b) G_{merge} (c) M_1, M_2 (d) $\mathcal{H}$

Abbildung 3.17: Alternative Modellierungsmöglichkeiten. Die farbigen Kanten in (c) stellen jeweils ein Matching in G_{merge} dar, wobei Kanten aus M_1 rot und aus M_2 blau gefärbt sind. Das Vereinigen von Graphen führt in diesem Beispiel zu keiner korrekten Modellierung des Angreiferwissens. Die Modellierung mittels Hypergraphen $\mathcal{H}$ in (d) entspricht hingegen der Modellierung mit Anfragegraphen. $\mathcal{H}$ enthält insgesamt sechs Hyperkanten.

Tupel 1 und 3 denselben SA-Wert haben müssen. Diese Bedingung geht durch das Vereinigen aller Teilgraphen verloren und Tupel 1 kann durchaus mit einem anderen SA-Label matchen als Tupel 3 (vgl. Matching M_2).

Eine andere Möglichkeit, Graphen zusammenzufassen, ist in (d) dargestellt. Auch hier wird für jedes Tupel und jeden in einem Ergebnis (bzw. Teilanfragegraphen) vorkommenden SA-Wert ein Knoten erstellt. Im Unterschied zu G_{merge} enthält $\mathcal{H}$ aber mehrdimensionale Kanten und ist somit ein Hypergraph. In der Grafik sind die SA-Knoten so gezeichnet, dass Knoten aus demselben Anfragegraphen G_i in derselben Spalte stehen. Eine Kante enthält immer genau einen Tupelknoten und für jeden Graphen G_i, in dem das dazugehörige Tupel vorkommt, genau einen SA-Wert, der ebenfalls in G_i vorkommt. Die SA-Werte innerhalb einer Kante sind dabei identisch. Beispielsweise steht die oberste Kante $e = \{1, \text{A}, \text{A}\}$ für Tupel 1, das in G_1 und G_3 jeweils zusammen mit einem SA-Knoten mit Label A vorkommt.

Ein Matching in $\mathcal{H}$ ist eine Menge von Hyperkanten, die keine gemeinsamen Knoten besitzen. Da jede Kante genau einen Tupelknoten beinhaltet, besteht ein perfektes Matching aus genau n Kanten, wenn n die Anzahl der Tupel ist. Da jeder Knoten überdeckt ist, kann es leicht in ein valides Matching in $\mathcal{G}$ überführt werden. Dazu wird für jede Hyperkante, die ein Tupel t und mehrere Knoten mit SA-Wert s enthält, in $\mathcal{G}$ eine Kante (t, s) erstellt. Durch die Konstruktion des Hypergraphen ist gewährleistet, dass in jedem Teilgraphen $G_i \in \mathcal{G}$ eine solche Kante existiert. Auf diese Weise kann analog eine SA-Werteverteilung erstellt werden. Im Beispiel in Abbildung 3.17 gibt es allerdings weder ein valides Matching in $\mathcal{G}$ noch ein perfektes Matching in $\mathcal{H}$.

Auf einen Beweis der Äquivalenz der Darstellungen mittels Hypergraphen und Anfragegraphen wird an dieser Stelle verzichtet, ebenso auf eine weitere Betrachtung dieser Variante. An Abbildung 3.17d lässt sich erahnen, dass die Darstellung größerer Hypergra-

phen, die aus Anfragen beziehungsweise Ergebnismengen konstruiert werden, sehr schnell unübersichtlich werden kann. Auch bietet die Theorie der einfachen Graphen viele Definitionen, Theoreme und Vereinfachungen, die das Verwenden von Anfragegraphen entscheidend verbessert. Daher werden in der vorliegenden Arbeit Anfragegraphen für die Modellierung des Angreiferwissens verwendet.

3.10.2 Modellierung als Constraint-Satisfaction-Problem

Viele Aufgabenstellungen in der Informatik, insbesondere im Bereich der künstlichen Intelligenz, lassen sich als *Constraint-Satisfaction-Problem* (*CSP*, zu deutsch in etwa *Bedingungserfüllungsproblem*) beschreiben. Ein CSP besteht im Allgemeinen aus einer Menge von Variablen $\{x_1, \ldots, x_n\}$ und einer Menge von Constraints $\{C_1, \ldots, C_m\}$. Jeder Variablen x_i ist eine Domäne D_i möglicher Werte zugeordnet. Constraints schränken Kombinationen von Werten bestimmter Variablen ein.[26] Eine Lösung eines CSP ist eine Belegung aller Variablen mit Werten aus ihrer jeweiligen Domäne, sodass alle Constraints erfüllt sind. In speziellen Fällen soll die Lösung eine zusätzliche Funktion maximieren.

Das in der vorliegenden Arbeit eingeführte Problem QGM kann als CSP modelliert werden. Das ist offensichtlich, da bereits in Kapitel 3.8 gezeigt wurde, wie QGM als ganzzahliges lineares Programm dargestellt werden kann. Die entstehenden Nebenbedingungen sind nichts anderes als Constraints, die durch eine (zulässige) Lösung erfüllt werden müssen. An dieser Stelle ist erkennbar, dass durch lineare Constraints bereits ein Spezialfall des generelleren Ansatzes als CSP entsteht. Das motiviert die Verwendung der spezialisierten Algorithmen aus dem Bereich der linearen Programmierung anstelle allgemeiner Lösungsverfahren für CSPs.

Einen Überblick über Constraint-Satisfaction-Probleme geben Russell und Norvig [122] sowie Barták [6].

Zusammenfassung

Um den Schutz der Privatsphäre bei der Anfragebearbeitung zu garantieren, werden die Schlussfolgerungen, die ein Angreifer aus den Ergebnissen seiner Anfragen ziehen kann, beschränkt. Das Ziel dabei lautet, dass es nicht möglich sein darf, Individuen ihre sensiblen Werte eindeutig zuzuordnen. Insbesondere müssen für jedes Individuum mindestens k sensible Werte existieren, aus denen der Angreifer den tatsächlichen Wert nicht ableiten kann. Dieses Prinzip wird durch den Begriff der k-assign Anonymität umgesetzt.

Das Wissen eines Angreifers aus den Ergebnissen seiner Anfragen kann als Menge aller validen Matchings in einer Folge von Anfragegraphen dargestellt werden. k-assign Anonymität kann darin überprüft werden, indem Matchings mit bestimmten Kanten berechnet werden. Im Unterschied zu einfachen Graphen ist das Matchingproblem in Anfragegraphen jedoch NP-vollständig und kann beispielsweise mithilfe ganzzahliger linearer Programmierung gelöst werden. Damit existiert zwar ein Algorithmus, der den Schutz der Privatsphäre bei der Anfragebearbeitung garantiert, jedoch hat dieser exponentielle Laufzeit. Hieraus entsteht die Motivation, in den nächsten Kapiteln einen Approximationsalgorithmus vorzustellen, der die Einhaltung des Schutzkriteriums in Polynomialzeit sicherstellt.

[26] Constraints können als Relationen über Teilmengen der Variablenmenge angesehen werden.

4 Verfeinerung der modellierten Matchings

In Kapitel 3 wurde gezeigt, wie der Schutz der Privatsphäre bei der Anfragebearbeitung durch das Bestimmen von validen Matchings in Graphen gewährleistet werden kann. Da das entstehende Matchingproblem NP-vollständig ist, liegt es nahe, Approximationsalgorithmen mit polynomieller Laufzeit anzugeben. In diesem Kapitel werden zunächst Anforderungen definiert, die dabei gelten sollen (Kapitel 4.1), und Möglichkeiten einer Approximation ausgelotet (Kapitel 4.2). Danach werden Matchings in Anfragegraphen klassifiziert, um in den folgenden Kapiteln die Matchings benennen zu können, die für die Approximation verwendet werden (Kapitel 4.3). Insbesondere sind dazu weitere Einschränkungen nötig, die erst beim Vorstellen der Algorithmen in den Kapiteln 5 und 7 vollständig erläutert werden (Kapitel 4.4). Ein weiterer Vorgriff ist die Einführung einer geeigneten Darstellungsform für Matchings, die später für die polynomielle Laufzeit der Verfahren sorgt (Kapitel 4.5).

4.1 Anforderungen an Approximationsalgorithmen

Als Modell für den Schutz der Privatsphäre bei der Bearbeitung von Anfragen wurde in Kapitel 3 k-assign Anonymität eingeführt. Durch die Bestimmung von SA-Werteverteilungen beziehungsweise validen Matchings in einem Anfragegraphen kann berechnet werden, für welches k dieses Kriterium erfüllt ist. Der Einfachheit halber wird in diesem und in den folgenden Kapiteln hauptsächlich nur noch das Berechnen von validen Matchings betrachtet und vorausgesetzt, dass jeder Graph zu einer vorigen Anfrage konstruiert wurde und dass aus den gefundenen Matchings SA-Werteverteilungen bestimmt werden können.

Sei $\mathcal{G}^{(n)}$ eine Folge von Anfragegraphen, in denen valide Matchings berechnet werden sollen. Für einen Approximationsalgorithmus zu diesem Problem können zwei orthogonale Ansätze angegeben werden. Zum einen kann ein Verfahren gesucht werden, das in Polynomialzeit Kantenmengen bestimmt, die Näherungen für valide Matchings darstellen. Beispielsweise könnte eine solche Menge nicht alle Tupelknoten überdecken und damit kein perfektes Matching[1] darstellen. Diese Idee ist aber für das zugrunde liegende Szenario fragwürdig, denn sie stellt den Schutz der Privatsphäre nur zu einem bestimmten Teil oder mit einer gewissen Wahrscheinlichkeit sicher. Ob nämlich tatsächlich ein valides Matching existiert, kann auf diese Weise nicht abschließend geklärt werden, sodass k-assign Anonymität nicht vollkommen garantiert werden kann.

Aus diesem Grund wird in der vorliegenden Arbeit ein anderer Ansatz verfolgt. Das Ziel besteht darin, nicht alle validen Matchings zu berechnen, sondern nur die, die auch in polynomieller Zeit gefunden werden können. Damit kann für jedes berechnete Matching eine SA-Werteverteilung angegeben und k-assign Anonymität abgeschätzt werden. Die Approximation liegt demnach darin, dass nicht alle Matchings beziehungsweise SA-Werteverteilungen gefunden werden. Somit wird auch nicht das komplette Wissen des

[1] Valide Matchings in Anfragegraphen sind per Definition perfekt.

Angreifers repräsentiert, sondern nur ein Teil davon. Gleichzeitig wird der Wert für k nach unten abgeschätzt. Der tatsächliche, das heißt exakt berechnete, Wert für k kann niemals kleiner sein als der durch die Approximation ermittelte Wert. Daraus folgt, dass zwar fälschlicherweise eine Verletzung der Privatsphäre beim Unterschreiten des vorgegebenen k entdeckt werden könnte, obwohl keine Verletzung vorliegt. Jede tatsächliche Verletzung wird aber auch gefunden. Folglich können nur sogenannte „false positives“, aber keine „false negatives“ vorkommen und der Schutz der Privatsphäre ist in jedem Fall gewährleistet. Neben der polynomiellen Laufzeit stellt das das wichtigste Ziel der Approximation dar.

Die entscheidende Frage bei diesem Ansatz lautet, wie Matchings in Polynomialzeit bestimmt werden können. Die Lösung beinhaltet den Schritt, Matchings rekursiv aus bereits bekannten Matchings zu berechnen. Konkret heißt das, dass Matchings für einen Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ aus den Matchings für $\mathcal{G}^{(n-1)} = (G_1, \ldots, G_{n-1})$ konstruiert werden. Wenn also $\mathcal{G}^{(n)}$ gegeben ist, dann werden zunächst nur Matchings in G_1, dann in (G_1, G_2), dann in (G_1, G_2, G_3) und so weiter berechnet. Zum Schluss erst werden die Matchings für $(G_1, \ldots, G_n)$ bestimmt. Wie nun tatsächlich Matchings aus bekannten Matchings konstruiert werden, wird in den Kapiteln 5 und 7 gezeigt. In diesem Kapitel steht die Frage im Vordergrund, welche Matchings für den beschriebenen Ansatz verwendet werden, aus denen später effizient andere Matchings berechnet werden können. Als größtes Problem stellt sich dabei heraus, dass die Anzahl aller in Frage kommenden Matchings in einem Anfragegraphen exponentiell in der Anzahl der Tupel sein kann. Es wird deshalb gezeigt, dass einerseits nicht alle Matchings für diesen Ansatz betrachtet werden müssen. Andererseits wird auch bewusst auf die Konstruktion einiger Matchings verzichtet, um später einen polynomiellen Algorithmus zu erhalten. Dabei spielt ebenfalls eine große Rolle, wie Matchings in einer geeigneten Datenstruktur gespeichert werden, um später effizient darauf zugreifen zu können. In diesem Zusammenhang wird auch von der *Modellierung* von Matchings gesprochen.

4.2 Approximation des Angreiferwissens

Das Wissen des Angreifers aus einer Folge von gegebenen Anfragen beziehungsweise Ergebnissen ist die Menge aller SA-Werteverteilungen. Im Allgemeinen existieren exponentiell viele verschiedene solche Verteilungen. Sind zum Beispiel n Tupel und n verschiedene sensible Attributwerte in einem Ergebnis, so gibt es $n!$ Permutation dieser Werte. Entscheidend für k-assign Anonymität ist allerdings nur, welche verschiedenen SA-Werte welchen Tupeln zugeordnet werden können. Es reicht daher aus, wenn es nur eine SA-Werteverteilung gibt, die einem Tupel t den Wert s zuordnet. Für t spielt es keine Rolle, dass es sogar $(n-1)!$ Permutationen gibt, die genau die gleiche Zuordnung von s zu t gewährleisten. Dementsprechend reichen $n-1$ Permutationen aus, um t jeden sensiblen Wert genau einmal zuzuordnen. Bei n Tupeln sind somit nur $n \cdot (n-1) = \mathcal{O}(n^2)$ verschiedene SA-Werteverteilungen nötig, um die maximale Anzahl von SA-Werten zu bestimmen, die jedem Tupel zugeordnet werden können. Damit kann die Gültigkeit des Schutzkriteriums bestätigt oder, falls einige der Zuordnungen nicht möglich sind, widerlegt werden.

Beispiel 4.1 Sei Q_1 die Anfrage, die bereits in Kapitel 3.2 eingeführt wurde. Das Ergebnis R_1 (siehe Tabelle 3.4 auf Seite 69) besteht aus vier Tupeln für Alison (1), Ben (2), Clark (3) und Debra (4) sowie den SA-Werten A, A, B und C. Insgesamt gibt es $\frac{4!}{2!} = 12$ Permutationen beziehungsweise SA-Werteverteilungen und jeder Wert kann je-

ID	Name	1	4	9	10
1	Alison	A	A	B	C
2	Ben	A	B	C	A
3	Clark	B	C	A	A
4	Debra	C	A	A	B

(a) SA-Werteverteilungen Nr. 1

ID	Name	1	2	3	6	7	11
1	Alison	A	A	A	A	B	C
2	Ben	A	A	B	C	A	A
3	Clark	B	C	A	B	A	B
4	Debra	C	B	C	A	C	A

(b) SA-Werteverteilungen Nr. 2

Tabelle 4.1: Approximation des Angreiferwissens aus R_1

dem Tupel zugeordnet werden (vgl. Tabelle 3.5 auf Seite 70). In Tabelle 4.1 sind in (a) nur vier dieser Verteilungen angegeben, die ebenfalls gewährleisten, dass jeder SA-Wert jedem Tupel mindestens einmal zugeordnet werden kann. Die Nummerierung entspricht dabei derselben Bezeichnung wie in Tabelle 3.5. Ein Angreifer, der nur diese vier Verteilungen kennt, weiß bezüglich der SA-Werte im Prinzip genauso viel wie jemand, der alle zwölf SA-Werteverteilungen kennt. Die daraus abgeleitete k-assign Anonymität ist bei beiden gleich. Es würde daher für die erste Anfrage ausreichen, nur die vier in (a) dargestellten SA-Werteverteilungen beziehungsweise äquivalenten Matchings zu modellieren.

In (b) sind sechs SA-Werteverteilungen angegeben, die zusammen ebenfalls garantieren, dass jeder Wert jedem Tupel zugeordnet werden kann. Es sind zwar insgesamt mehr Verteilungen als in (a), dafür haben diese eine interessante Eigenschaft: In jeder SA-Werteverteilung außer der ersten gibt es nur zwei Tupel, denen ein anderer SA-Wert zugeordnet wurde als ihr originaler. Die Originalverteilung ist dabei in der ersten Spalte dargestellt und mit 1 nummeriert. Beispielsweise haben in Verteilung 2 die Tupel für Clark und Debra einen anderen SA-Wert als in der Originalverteilung 1, während Alison und Ben jeweils ihr originaler SA-Wert zugeordnet wird. Offensichtlich müssen in diesem Fall die SA-Werte der betreffenden Tupel genau getauscht werden, das heißt, Clark hat den originalen SA-Wert von Debra (C) und Debra den von Clark (B). In Verteilung 3 wurden wiederum die SA-Werte von Ben und Clark vertauscht, in 6 die von Ben und Debra und auch in den anderen Verteilungen gibt es genau zwei vertauschte SA-Werte. Diese Eigenschaft wird später eine entscheidende Rolle spielen.

ID	Name	1.1	1.3	2.1	6.1	7.3	9.1	9.4	$Sig_{\mathcal{R}^{(3)}}(t)$
1	Alison	A	A	A	A	B	B	B	A, B
2	Ben	A	A	A	C	A	C	C	A, C
3	Clark	B	B	C	B	A	A	A	A, B, C
4	Debra	C	C	B	A	C	A	A	A, B, C
5	Elaine	B	D	B	B	B	A	B	A, B, D
6	Fiona	B	B	B	B	A	B	D	A, B, D
7	Gary	D	B	D	D	D	D	A	A, B, D
8	Helen	C	C	C	A	C	B	D	A, B, C, D

Tabelle 4.2: Approximation des Angreiferwissens aus $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$

Beispiel 4.2 Sei $\mathcal{R}^{(3)} = (R_1, R_2, R_3)$ die Folge von Ergebnissen zu der bereits eingeführten Folge von Anfragen $\mathcal{Q}^{(3)} = (Q_1, Q_2, Q_3)$, für die es nach Tabelle 3.10 (siehe Seite 73)

18 verschiedene SA-Werteverteilungen gibt. In Tabelle 4.2 sind nur sieben dieser Verteilungen dargestellt, die zusammen ausreichen, um alle Zuordnungen von sensiblen Werten zu Tupeln zu modellieren (vgl. Tabelle 3.11 auf Seite 73).

Das Ziel der Approximation des Angreiferwissens ist, nicht alle möglichen SA-Werteverteilungen zu berechnen und abzubilden, sondern nur eine viel kleinere Teilmenge. Diese soll insgesamt drei wichtige Eigenschaften erfüllen. Sie soll erstens nur polynomielle Größe haben, denn der Algorithmus zur Berechnung der k-assign Anonymität soll nur polynomielle Laufzeit haben. Zweitens sollen möglichst viele verschiedene Zuordnungen von SA-Werten zu Tupeln abgebildet bleiben. Da das Berechnen von SA-Werteverteilungen im Allgemeinen ein NP-vollständiges Problem ist,[2] ist offensichtlich, dass an dieser Stelle nur eine Näherung angegeben werden kann. Das bedeutet, es werden nicht alle Zuordnungen von SA-Werten zu Tupeln gefunden, die tatsächlich möglich sind. Die dritte Eigenschaft betrifft die Konstruktion der Verteilungen. Da die Idee ist, sie aus bereits zuvor berechneten Verteilungen zu bestimmen, dürfen auch nur solche betrachtet werden, für die diese Berechnung in polynomieller Zeit möglich ist. Diese Eigenschaft wird erst in Kapitel 5 ausreichend geklärt.

Da SA-Werteverteilungen mithilfe von validen Matchings in Anfragegraphen berechnet werden können, werden im Folgenden auch nur noch Matchings betrachtet. Der Vorteil ist, dass daraus ein anschauliches Modell entsteht und auf viele Eigenschaften und Begrifflichkeiten von Graphen zurückgegriffen werden kann. Entsprechend den Vorbetrachtungen lautet das Ziel, nicht alle Matchings zu berechnen, sondern nur eine kleinere Teilmenge. Für diese gelten die gleichen Eigenschaften wie für die SA-Werteverteilungen, denen sie entsprechen.

4.3 Klassifikation von Matchings

Für die weiteren Betrachtungen in diesem Kapitel muss die Menge aller validen Matchings in Anfragegraphen so eingeschränkt werden, dass die zuvor definierten Anforderungen erfüllt sind. Das bedeutet insbesondere, dass Matchings in polynomieller Zeit aus bereits bekannten Matchings berechnet werden können. Zunächst wird dazu gezeigt, dass für den Test auf k-assign Anonymität nicht alle Matchings benötigt werden. Es gibt Matchings, auf deren Berechnung explizit verzichtet werden kann, da die daraus konstruierten Zuordnungen von SA-Werten zu Tupeln auch aus anderen Matchings ableitbar sind. Die resultierende Menge von Matchings ist die, die für die Abbildung des Angreiferwissens benötigt wird.[3] Aus ihr wird danach eine Teilmenge gebildet, die die gewünschten Eigenschaften besitzt und insbesondere polynomielle Größe hat. Diese dient als Approximation des Angreiferwissens, das heißt als Näherung für die Menge aller SA-Werteverteilungen beziehungsweise Zuordnungen von SA-Werten zu Tupeln. Abbildung 4.1 zeigt eine Klassifikation, in die im Folgenden noch Kriterien eingefügt werden (vgl. Abbildung 4.16 auf Seite 123).

[2] Vgl. Theorem 3.5 auf Seite 84.

[3] Hierbei ist jeweils vorausgesetzt, dass k-assign Anonymität das Kriterium zum Schutz der Privatsphäre ist.

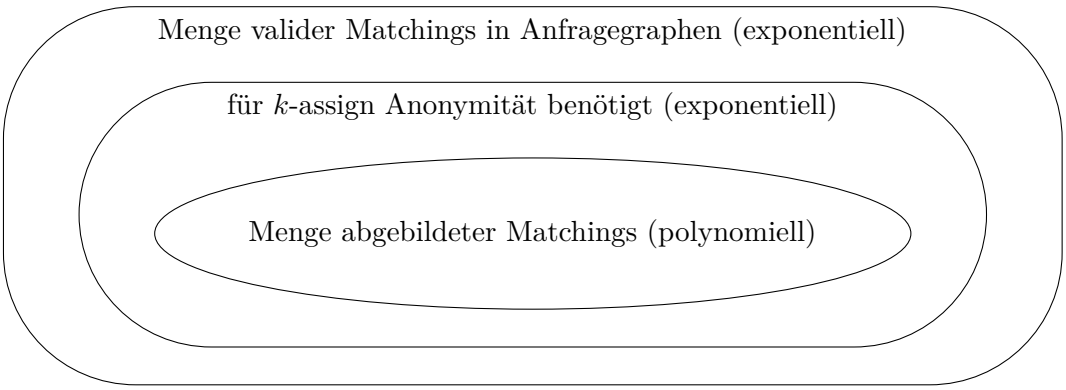


Abbildung 4.1: Modellierte Matchings

4.3.1 SA-Äquivalenz

Aus Kapitel 3 ist bekannt, dass zu jedem validen Matching in einem Anfragegraphen genau eine äquivalente SA-Werteverteilung existiert. Umgekehrt kann es zu Verteilungen auch mehrere äquivalente Matchings geben. Aus offensichtlichen Gründen müssen nicht mehrere Matchings modelliert werden, die dieselbe SA-Werteverteilung erzeugen. In diesem Zusammenhang werden zueinander verschiedene Kanten, die dieselbe Labelkante besitzen, *SA-äquivalent* genannt.

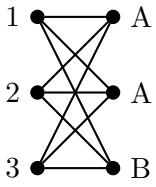
Abbildung 4.2 zeigt in (b) einen Anfragegraphen G, der aus den Tupeln 1, 2 und 3 sowie den SA-Werten A, A und B besteht. Die dazugehörige Datentabelle ist in (a) angegeben (Spalten ID, Name und SA). Da in G ein SA-Label mehrfach vorkommt, existieren auch SA-äquivalente Kanten. Bezeichne v_t den Tupelknoten mit Label t und $v_{\mathrm{A}1}$ den oberen sowie $v_{\mathrm{A}2}$ den mittleren SA-Knoten in (b). Dann sind die Kanten $\{v_1, v_{\mathrm{A}1}\}$ und $\{v_1, v_{\mathrm{A}2}\}$, $\{v_2, v_{\mathrm{A}1}\}$ und $\{v_2, v_{\mathrm{A}2}\}$ sowie $\{v_3, v_{\mathrm{A}1}\}$ und $\{v_3, v_{\mathrm{A}2}\}$ jeweils SA-äquivalent zueinander. In G gibt es insgesamt sechs verschiedene Matchings, die in (c) bis (h) dargestellt sind. Davon erzeugen M_1 und M_2, M_3 und M_4 sowie M_5 und M_6 jeweils dieselbe SA-Werteverteilung, sind aber graphentheoretisch jeweils verschiedene Matchings. In (i) sind für je zwei Matchings die entsprechenden SA-äquivalenten Kanten aufgelistet, wobei hier vereinfacht nur die Labelkanten angegeben sind.

Interessant sind die Beziehungen zwischen den drei Matchings M_1, M_5 und M_6. M_1 und M_6 erzeugen zwar verschiedene SA-Werteverteilungen, enthalten aber SA-äquivalente Kanten. Dahingegen gibt es zwischen M_1 und M_5 keine SA-äquivalenten Kanten, obwohl M_5 und M_6 wiederum dieselbe SA-Werteverteilung haben. Um diesen Zusammenhang näher zu untersuchen, wird der Begriff der SA-Äquivalenz auf Paare von Matchings ausgeweitet. Dabei werden *(vollständig) SA-äquivalente* von *partiell SA-äquivalenten* Matchings unterschieden. Erstere bestehen nur aus identischen oder SA-äquivalenten Kanten, während bei den zuletzt genannten zusätzlich auch völlig verschiedene Kanten vorkommen dürfen.

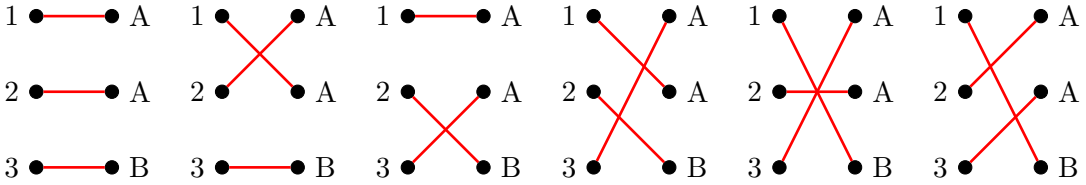
Definition 4.1 (SA-äquivalent) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei verschiedene valide Matchings in $\mathcal{G}$. $\mathcal{M}'$ ist genau dann* partiell SA-äquivalent *bezüglich $\mathcal{M}$, wenn es einen Tupelknoten $v_t \in V(\mathcal{G})$ mit zwei Kanten $e = \{v_t, v_s\} \in \mathcal{M}$ und $e' = \{v_t, v'_s\} \in \mathcal{M}'$ gibt, für die gilt: $v_s \neq v'_s$ und $v_s =_l v'_s$. $\mathcal{M}'$ ist genau dann* (vollständig) SA-äquivalent *bezüglich $\mathcal{M}$, wenn für alle Tupelknoten $v_t \in V(\mathcal{G})$ und*

ID	Name	SA	M_1	M_2	M_3	M_4	M_5	M_6
1	Alison	A	A	A	A	A	B	B
2	Ben	A	A	A	B	B	A	A
3	Clark	B	B	B	A	A	A	A

(a) Datentabelle (Spalten ID, Name, SA) und SA-Werteverteilungen der Matchings M_1 bis M_6



(b) G



(c) M_1 (d) M_2 (e) M_3 (f) M_4 (g) M_5 (h) M_6

M_2	$(1, \mathrm{A}), (2, \mathrm{A})$				
M_3	–	$(1, \mathrm{A})$			
M_4	$(1, \mathrm{A})$	–	$(1, \mathrm{A}), (3, \mathrm{A})$		
M_5	–	$(2, \mathrm{A})$	$(3, \mathrm{A})$	–	
M_6	$(2, \mathrm{A})$	–	–	$(3, \mathrm{A})$	$(2, \mathrm{A}), (3, \mathrm{A})$
Matching	M_1	M_2	M_3	M_4	M_5

(i) SA-äquivalente Kanten zwischen Matchings

Abbildung 4.2: SA-äquivalente Matchings in G_1

je zwei Kanten $e = \{v_t, v_s\} \in \mathcal{M}$ und $e' = \{v_t, v_s'\} \in \mathcal{M}'$ gilt: $v_s =_l v_s'$. Jeweils heißen e und e' SA-äquivalent *zueinander und $\mathcal{M}$ wird das* Bezugsmatching *genannt.*

In der Definition kann auf die Bedingung $v_s \neq v_s'$ bei vollständig SA-äquivalenten Matchings verzichtet werden, da durch die Voraussetzung $\mathcal{M} \neq \mathcal{M}'$ bereits gewährleistet ist, dass es mindestens zwei solche Knoten geben muss. Für identische Knoten $v_s = v_s'$ gilt nämlich immer auch $v_s =_l v_s'$. Aufgrund der Symmetrie folgt, dass ein Matching $\mathcal{M}$ bezüglich eines Matchings $\mathcal{M}'$ genau dann SA-äquivalent ist, wenn auch die umgekehrte Beziehung erfüllt ist. $\mathcal{M}$ und $\mathcal{M}'$ heißen auch SA-äquivalent zueinander.

Beispiel 4.3 Für die Matchings in Abbildung 4.2 gilt, dass M_1 und M_2, M_3 und M_4 sowie M_5 und M_6 jeweils vollständig SA-äquivalent zueinander sind. Alle anderen Kombinationen von Matchings, für die in (i) mindestens eine SA-äquivalente Kante angegeben ist, stellen jeweils partiell SA-äquivalente Matchings dar.

Offensichtlich ist jedes vollständig SA-äquivalente Matching auch partiell SA-äquivalent. Eine weitere Eigenschaft wird in Korollar 4.1 festgehalten.

Korollar 4.1 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei verschiedene valide Matchings in $\mathcal{G}$. Wenn $\mathcal{M}'$ (vollständig) SA-äquivalent zu $\mathcal{M}$ ist, dann sind die SA-Werteverteilungen von $\mathcal{M}$ und $\mathcal{M}'$ identisch.*

Beweis: Die Behauptung folgt direkt aus den Definitionen 3.1 und 4.1. □

Mithilfe der eingeführten Begriffe kann nun die Menge aller betrachteter Matchings beschränkt werden. Offensichtlich ist es nicht erforderlich, alle vollständig SA-äquivalenten Matchings in einem Anfragegraphen zu bestimmen beziehungsweise für weitere Berechnungen zu speichern. Ein Matching, welches allerdings sehr einfach erstellt werden kann, ist das Originalmatching $\mathcal{M}_{\text{orig}}$. Dieses kann direkt aus der Datentabelle abgelesen werden und muss daher nicht explizit gespeichert werden. Somit kann auf die Modellierung von Matchings verzichtet werden, die vollständig SA-äquivalent bezüglich $\mathcal{M}_{\text{orig}}$ sind.

Ein ähnliches Resultat ergibt sich für partiell SA-äquivalente Matchings. Beispielsweise haben M_5 und M_6 in Abbildung 4.2 dieselbe SA-Werteverteilung, aber nur M_5 ist nicht partiell SA-äquivalent bezüglich des Originalmatchings M_1. Der folgende Satz besagt, dass es ausreicht, nur Matchings zu betrachten, die nicht partiell SA-äquivalent sind.

Proposition 4.2 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei verschiedene valide Matchings in $\mathcal{G}$. Wenn $\mathcal{M}'$ partiell, aber nicht vollständig SA-äquivalent bezüglich $\mathcal{M}$ ist, existiert ein valides Matching $\mathcal{M}^* \neq \mathcal{M}'$, das nicht partiell SA-äquivalent bezüglich $\mathcal{M}$ ist und die gleiche SA-Werteverteilung wie $\mathcal{M}'$ besitzt.*

Algorithmus 4.1 Transformiere SA-äquivalentes Matching

Eingabe: $\mathcal{M}, \mathcal{M}'$: valide Matchings, die partiell SA-äquivalent zueinander sind
Ausgabe: $\mathcal{M}^*$: valides Matching, das nicht SA-äquivalent zu $\mathcal{M}$ ist und die gleiche SA-Werteverteilung wie $\mathcal{M}'$ hat

1: $\mathcal{M}^* \leftarrow \mathcal{M}'$
2: **for all** SA-äquivalente Kanten $\{v_t, v_s\} \in \mathcal{M}'$ und $\{v_t, v'_s\} \in \mathcal{M}$ **do**
3: $\quad v'_t \leftarrow$ Matchingpartner von v'_s in $\mathcal{M}'$
4: $\quad \mathcal{M}^* \leftarrow \mathcal{M}^* - \{v'_t, v'_s\} - \{v_t, v_s\} + \{v_t, v'_s\} + \{v'_t, v_s\}$
5: **end for**
6: **return** $\mathcal{M}^*$

Beweis: Algorithmus 4.1 stellt ein iteratives Verfahren dar, um $\mathcal{M}^*$ aus $\mathcal{M}'$ zu konstruieren. Anfangs setze $\mathcal{M}^* = \mathcal{M}'$. Danach werden alle SA-äquivalenten Kanten betrachtet. Das sind alle Kanten $\{v_t, v_s\} \in \mathcal{M}'$, für die eine Kante $\{v_t, v'_s\} \in \mathcal{M}$ existiert mit $v_s \neq v'_s$ und $v_s =_l v'_s$. Sei v'_t der Tupelknoten mit $\{v'_t, v'_s\} \in \mathcal{M}'$, dann setze $\mathcal{M}^* = \mathcal{M}^* - \{v'_t, v'_s\} - \{v_t, v_s\} + \{v_t, v'_s\} + \{v'_t, v_s\}$. Durch jeden dieser Ersetzungsschritte wird die betrachtete SA-äquivalente Kante durch eine Kante aus $\mathcal{M}$ ersetzt, wobei die äquivalente SA-Werteverteilung nicht verändert wird. Die andere gelöschte Kante ist jeweils nur in $\mathcal{M}'$, aber nicht in $\mathcal{M}$ vorhanden, wodurch keine neuen SA-äquivalenten Kanten entstehen können und das Verfahren terminiert. Das auf diese Weise entstehende Matching $\mathcal{M}^*$ ist offensichtlich valid und nicht partiell SA-äquivalent bezüglich $\mathcal{M}$. □

Beispiel 4.4 Die Eingabe für Algorithmus 4.1 seien die Matchings M_1 und M_6 aus Abbildung 4.2, wobei M_1 das Bezugsmatching sei. Da die einzigen SA-äquivalenten Kanten jene sind, die zwischen dem Tupelknoten mit Label 2 und den beiden SA-Knoten mit Label A verlaufen, wird die entsprechende (Label-)Kante (2, A) aus M_6 entfernt. Der Matchingpartner von 2 in M_1 ist der mittlere SA-Knoten, welcher in M_6 mit 3 matcht. Folglich

werden die Matchingpartner von 2 und 3 in M_6 vertauscht und es entsteht das Matching M_5, welches nicht mehr partiell SA-äquivalent bezüglich M_1 ist.

Werden Algorithmus 4.1 zwei vollständig SA-äquivalente Matchings übergeben, überführt das Verfahren das eine Matching in das andere.

Zusammenfassend gilt, dass Matchings, die partiell SA-äquivalent bezüglich des Originalmatchings sind, nicht weiter betrachtet werden. Für das Beispiel in Abbildung 4.2 werden demnach neben M_1 nur M_3 und M_5 modelliert. Proposition 4.2 garantiert, dass jede mögliche SA-Werteverteilung weiterhin erzeugt werden kann.

Abschließend sei erwähnt, dass dieses Kriterium nur für die Wahl von genau einem Bezugsmatching gültig ist. Das bedeutet, alle verbleibenden Matchings sind nur bezüglich des Originalmatchings nicht partiell SA-äquivalent. Untereinander können sie weiterhin partiell SA-äquivalent sein. Zum Beispiel enthalten die modellierten Matchings M_3 und M_5 aus Abbildung 4.2 die SA-äquivalente Kante $(3, \mathrm{A})$. Damit aber Tupel 1 ein B zugeordnet werden kann, muss den Tupeln 2 und 3 jeweils ein A zugeordnet werden. Diese SA-Werteverteilung wird nur durch die Matchings M_5 und M_6 sichergestellt, wobei M_6 nicht modelliert wird, da es partiell SA-äquivalent zu M_1 ist. Ein analoges Argument gilt für M_3 und M_4, sodass auch auf M_3 nicht verzichtet werden kann.

4.3.2 Symmetrische Differenz von Matchings

Die folgenden drei Klassifikationen von Matchings basieren alle auf einer ähnlichen Idee. Das Ziel ist, möglichst „einfache“ Matchings zu konstruieren. Einfach heißt in diesem Zusammenhang, dass sich das Matching möglichst wenig von bereits bekannten anderen Matchings unterscheiden soll. Da das Originalmatching immer leicht bestimmt werden kann, wird es auch hier als Bezugsmatching verwendet. Im Allgemeinen gibt es meistens mehrere Matchings, die eine Kante (t, s) enthalten und damit sicherstellen, dass der SA-Wert s dem Tupel t zugeordnet werden kann. Von all diesen Matchings wird dasjenige bevorzugt, das sich am wenigsten vom Originalmatching unterscheidet. Im Idealfall gibt es in diesem Matching nur zwei Kanten, die keine Originalkanten sind. Das sind die Kanten (t, s) und (t', s'), wobei s' der originale SA-Wert von t ist und t' ein Tupel, das den originalen Wert s hat. Der Vorteil dieses Ansatzes ist, dass die Differenz zwischen solchen Matchings recht klein ist, was später die Grundlage für die polynomielle Laufzeit sein wird.

Um die Begriffe wie „einfaches Matching“ und „geringer Unterschied“ zu quantifizieren, wird die symmetrische Differenz zwischen zwei Matchings M und M' in Graphen betrachtet. Sie ist die Menge aller Kanten, die in genau einem der beiden Matchings vorhanden sind:

$$M \,\Delta\, M' = (M \setminus M') \cup (M' \setminus M). \tag{4.1}$$

$M \Delta M'$ induziert einen Graphen G_Δ, der aus allen Knoten und Kanten der symmetrischen Differenz besteht:

$$\begin{aligned} G_\Delta &= G[M \,\Delta\, M'] = (V_\Delta, E_\Delta) \text{ mit} \\ V_\Delta &= \{v \in e \mid \text{Kante } e \in M \,\Delta\, M'\} \text{ und} \\ E_\Delta &= M \,\Delta\, M'. \end{aligned}$$

Als Vereinfachung wird im Folgenden mit $M \Delta M'$ immer der durch $M \Delta M'$ induzierte Graph $G[M \Delta M']$ bezeichnet. Die symmetrische Differenz von Matchings hat eine sehr nützliche Eigenschaft, die neben Anfragegraphen auch für beliebige einfache Graphen gilt.

Proposition 4.3 *Seien $G = (V, E)$ ein Graph und $M, M' \subseteq E$ zwei perfekte Matchings in G. Dann besteht der durch $M \Delta M'$ induzierte Graph nur aus Kreisen gerader Länge.*

Beweis: Der Beweis erfolgt indirekt. Sei v ein Knoten in $G[M \Delta M']$ mit Grad 1. Dieser Knoten ist entweder von M oder M' überdeckt und dadurch ist eines der beiden Matchings nicht perfekt (Widerspruch zur Annahme). Sei v ein Knoten mit Grad ≥ 3, dann sind in M oder M' mindestens zwei Kanten mit v inzident. Das widerspricht der Matchingdefinition 1.8. Somit müssen alle Knoten Grad 2 haben und $G[M \Delta M']$ besteht nur aus Kreisen. Falls es einen Kreis ungerader Länge gibt, existiert aus Paritätsgründen erneut ein Knoten, der in einem der beiden Matchings mit zwei Kanten inzident ist. □

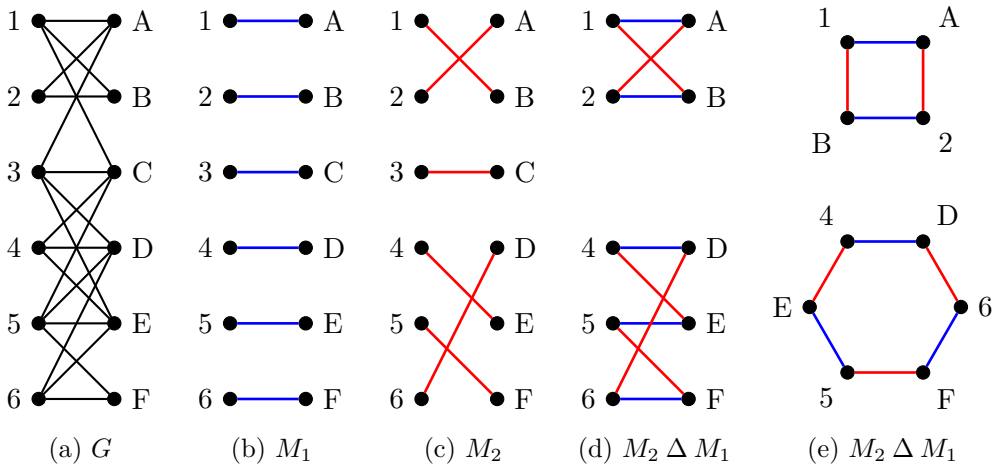


Abbildung 4.3: Symmetrische Differenz von Matchings. Farblich gleiche Kanten gehören zum selben Matching.

Beispiel 4.5 Abbildung 4.3 zeigt einen Graphen G sowie die beiden perfekten Matchings M_1 und M_2. In (d) ist die symmetrische Differenz $M_2 \Delta M_1$ dargestellt, wobei blaue Kanten aus M_1 und rote Kanten aus M_2 stammen. Da die Kante $\{3, \mathrm{C}\}$[4] in beiden Matchings vorkommt, ist sie nicht in der symmetrischen Differenz enthalten. In (e) ist erneut $M_2 \Delta M_1$ dargestellt, wobei die Knoten zur besseren Übersicht anders angeordnet sind. Die entstehenden Kreise haben die Längen 4 und 6.

4.3.3 Δ-minimale Matchings

Mithilfe der symmetrischen Differenz kann jetzt eine Aussage darüber getroffen werden, ob ein Matching sich nur wenig oder sogar minimal von einem anderen Matching unterscheidet. Das Ziel liegt dabei darin, in Anfragegraphen nur sogenannte minimale Matchings zu

[4] In diesem Beispiel wird von einfachen Graphen gesprochen, das heißt, mit 3 beziehungsweise C sind tatsächlich die Knoten gemeint und keine Labels.

berechnen. Dazu wird bewiesen, dass trotz dieser Einschränkung immer noch alle möglichen Zuordnungen von SA-Werten zu Tupeln modelliert werden.

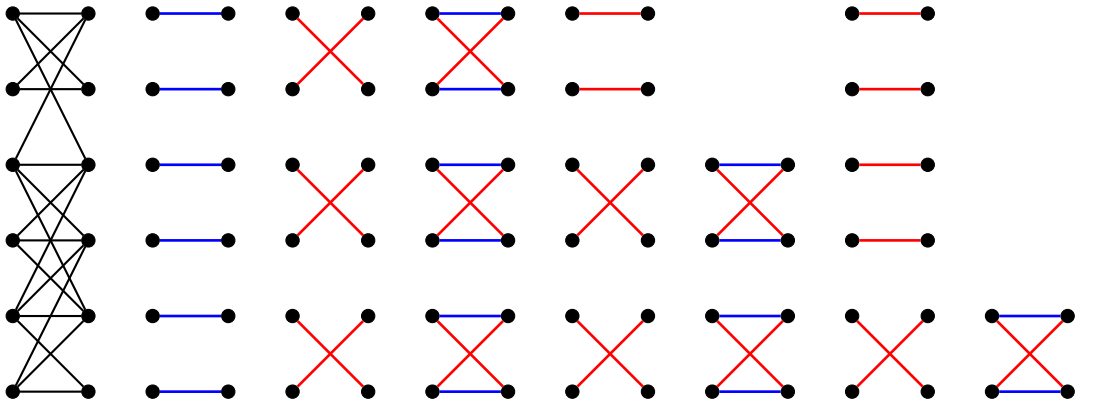
Seien M, M_1 und M_2 drei Matchings in einem einfachen Graphen. M unterscheidet sich weniger von M_1 als von M_2, wenn die symmetrische Differenz $M_1 \Delta M$ eine echte Teilmenge von $M_2 \Delta M$ ist. Demzufolge haben M und M_1 mehr identische Kanten als M und M_2. Für diesen Zusammenhang wird der Begriff Δ-*klein*[5] eingeführt.

Definition 4.2 (Δ-klein) *Seien G ein einfacher Graph und M, M_1, M_2 drei Matchings in G. Dann ist M_1 bezüglich M genau dann* Δ-kleiner *als M_2 (notiert als $M_1 <_{\Delta M} M_2$), wenn $M_1 \Delta M \subset M_2 \Delta M$ gilt.*

Zu einem gegebenen Matching M sind besonders die kleinsten solcher Matchings interessant. Diese werden Δ-*minimal*[6] genannt, während M erneut als Bezugsmatching bezeichnet wird.

Definition 4.3 (Δ-minimales Matching für Graphen) *Seien G ein einfacher Graph und M, M' zwei perfekte Matchings in G. Dann ist M' genau dann ein* Δ-minimales Matching *bezüglich M, wenn für alle zu M verschiedenen perfekten Matchings M^* gilt: $M^* \Delta M \not\subset M' \Delta M$.*

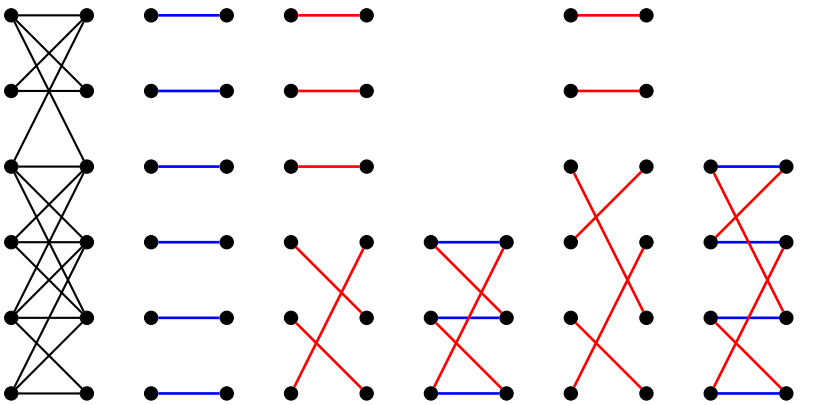
Der Begriff der Δ-Minimalität ist nur für perfekte Matchings sinnvoll. Bei einem nicht perfekten Matching ist jedes andere Matching Δ-minimal, das genau eine Kante mehr oder weniger enthält. In diesem Fall enthält nämlich die symmetrische Differenz genau diese eine Kante. Da valide Matchings in Anfragegraphen jedoch immer auch perfekt sind, ist diese Einschränkung zumindest für sie nicht relevant. Weiterhin sei angemerkt, dass jedes bezüglich M Δ-minimale Matching M' nicht zwangsläufig auch Δ-kleiner als jedes andere Matching sein muss. Es ist nur gewährleistet, dass es kein Δ-kleineres Matching gibt. Somit kann nicht von einem Δ-kleinsten Matching oder Minimum[7] gesprochen werden.



(a) G (b) M (c) M_1 (d) $M_1 \Delta M$ (e) M_2 (f) $M_2 \Delta M$ (g) M_3 (h) $M_3 \Delta M$

Abbildung 4.4: Beispiel 1 für Δ-minimale Matchings

[5] ausgesprochen: *delta-klein*
[6] ausgesprochen: *delta-minimal*
[7] Für ein kleinstes Element oder Minimum gilt immer, dass alle anderen Elemente größer oder gleich groß sind. Im Fall von Matchings gibt es aber Elemente, die nicht vergleichbar sind, da sie zum Beispiel aus völlig anderen Matchingkanten bestehen (siehe Beispiel 4.6).



(a) G (b) M (c) M_4 (d) $M_4 \Delta M$ (e) M_5 (f) $M_5 \Delta M$

Abbildung 4.5: Beispiel 2 für Δ-minimale Matchings

Beispiel 4.6 Die Abbildungen 4.4 und 4.5 zeigen jeweils in (a) denselben Graphen und in (b) dasselbe Bezugsmatching M. Dazu sind insgesamt fünf verschiedene andere Matchings und deren jeweilige symmetrische Differenz zu M angegeben. Die blauen Kanten stammen immer von M und die roten von den verschiedenen Matchings. Während $M_1 \Delta M$ aus insgesamt drei (4.4d) und $M_2 \Delta M$ aus zwei (4.4f) Kreisen besteht, enthalten die anderen Differenzen jeweils nur einen Kreis (4.4h, 4.5d, 4.5f).

Aus $M_2 \Delta M \subset M_1 \Delta M$ folgt per Definition $M_2 <_{\Delta M} M_1$. Ebenso gilt $M_3 <_{\Delta M} M_2$ und $M_3 <_{\Delta M} M_1$. Obwohl $M_4 \Delta M$ (4.5d) und $M_5 \Delta M$ (4.5f) aus mehr Kanten bestehen als $M_3 \Delta M$ (4.4h), ist keines dieser Matchings Δ-kleiner als das andere. Insbesondere gilt, dass M_3, M_4 und M_5 Δ-minimale Matchings sind. Der Grund dafür liegt darin, dass die jeweilige symmetrische Differenz nur aus einem einzigen Kreis besteht und damit nach Proposition 4.3 keine echte Teilmenge enthalten kann.

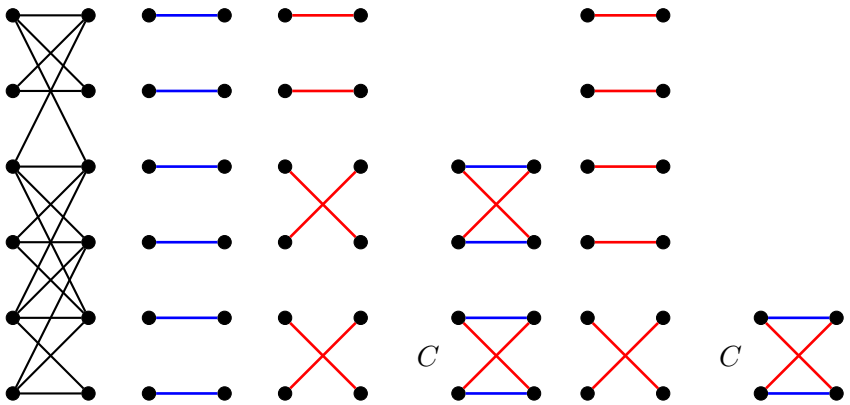
Die Charakterisierung eines Δ-minimalen Matchings in einem einfachen Graphen ist recht einfach. Es besteht aus genau einem Kreis.

Proposition 4.4 *Seien G ein Graph und M, M' zwei perfekte Matchings in G. M' ist genau dann ein Δ-minimales Matching bezüglich M, wenn $G[M' \Delta M]$ nur aus genau einem Kreis besteht.*

Beweis: „$\Rightarrow$“ (indirekt): $G[M' \Delta M]$ bestehe aus mindestens 2 Kreisen und C sei einer davon. Sei M^* eine Kantenmenge, die alle Kanten von C enthält, die auch in M' sind, und ansonsten alle Kanten aus M: $M^* = (C \cap M') \cup (M \setminus C)$. Dann ist M^* ein perfektes Matching in G und $G[M^* \Delta M]$ besteht nur aus dem Kreis C. Das widerspricht aber der Minimalität von M' nach Definition 4.3 (siehe Abbildung 4.6).

„$\Leftarrow$“: Offensichtlich, da $G[M' \Delta M]$ nach Proposition 4.3 nur aus Kreisen besteht. □

Das Ziel der weiteren Überlegungen liegt darin zu zeigen, dass für den Test auf k-assign Anonymität nur Δ-minimale Matchings betrachtet werden müssen. Somit muss für jede Labelkante $e_l = (t, s)$, die in einem beliebigen validen Matching vorkommt, ein Δ-minimales valides Matching gefunden werden, welches e_l enthält. Daraus folgt, dass nur Δ-minimale Matchings benötigt werden, um alle möglichen Zuordnungen von SA-Werten zu Tupeln zu modellieren. Folgende Lemmata bilden die Grundlage dieser Behauptung.



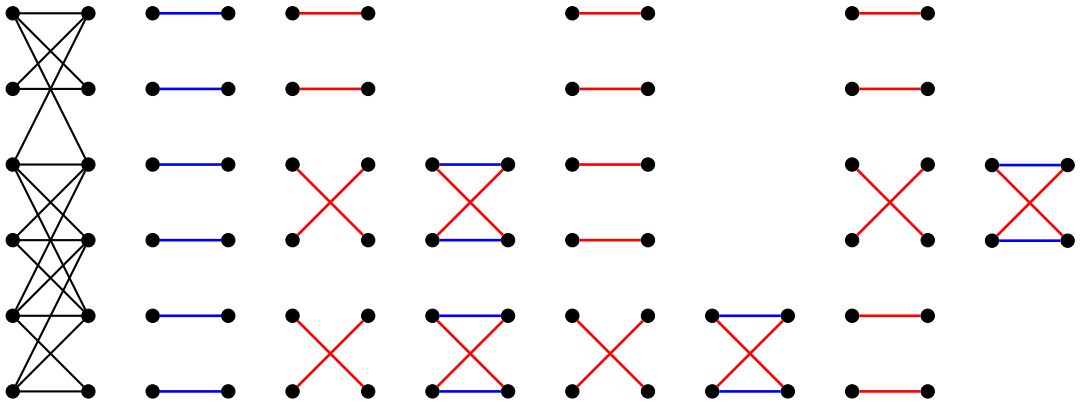
(a) G (b) M (c) M' (d) $M' \Delta M$ (e) M^* (f) $M^* \Delta M$

Abbildung 4.6: Beispiel zum Beweis von Proposition 4.4

Lemma 4.5 *Seien G ein Graph und M, M' zwei perfekte Matchings in G, wobei M' kein Δ-minimales Matching bezüglich M sei. Dann gibt es zwei zu M und M' verschiedene perfekte Matchings M_1, M_2 in G mit*

$$(M_1 \Delta M) \cup (M_2 \Delta M) = M' \Delta M, \tag{4.2}$$
$$(M_1 \Delta M) \cap (M_2 \Delta M) = \emptyset. \tag{4.3}$$

Beweis: Da M' nicht Δ-minimal ist, gibt es nach Definition 4.3 ein zu M verschiedenes perfektes Matching M_1 in G mit $M_1 \Delta M \subset M' \Delta M$. Sei $M_2 \subseteq E(G)$ eine Kantenmenge mit $M_2 \Delta M = (M' \Delta M) \setminus (M_1 \Delta M)$. Aufgrund der Wahl von M_1 (echte Teilmengenbeziehung zwischen den symmetrischen Differenzen) gilt $M_2 \Delta M \neq \emptyset$, woraus $M_2 \neq M$ folgt. Weiterhin enthält M_2 für jeden Knoten aus $M_2 \Delta M$ dieselbe Matchingkante, die auch in M' den Knoten enthält, und für jeden anderen Knoten dieselbe Matchingkante, die auch in M den Knoten enthält. Folglich ist M_2 ein perfektes Matching in G (siehe Abbildung 4.7). □



(a) G (b) M (c) M' (d) $M' \Delta M$ (e) M_1 (f) $M_1 \Delta M$ (g) M_2 (h) $M_2 \Delta M$

Abbildung 4.7: Beispiel zum Beweis von Lemma 4.5

Eine direkte Folgerung aus Lemma 4.5 ist, dass zu jedem Matching, welches nicht Δ-minimal bezüglich eines gegebenen Matchings ist, zwei Δ-kleinere Matchings existieren.

Korollar 4.6 *Seien G ein Graph und M, M' zwei perfekte Matchings in G, wobei M' kein Δ-minimales Matching bezüglich M sei. Dann gibt es zwei zu M und M' verschiedene perfekte Matchings M_1, M_2 in G mit $M_1 \neq M_2$, die beide jeweils bezüglich M Δ-kleiner sind als M'.*

Beweis: Nach Lemma 4.5 existieren in diesem Fall zwei Matchings M_1, M_2, die beide per Konstruktion (vgl. Beweis des Lemmas) bezüglich M Δ-kleiner sind als M'.[8] □

Nach den beiden vorigen Sätzen kann jedes nicht-Δ-minimale Matching in mindestens zwei Matchings aufgeteilt werden, sodass die Vereinigung der symmetrischen Differenzen beider Matchings mit dem Bezugsmatching genau der symmetrischen Differenz des betrachteten Matchings mit dem Bezugsmatching entspricht (vgl. Gleichung 4.2). Abbildung 4.7 zeigt diesen Fall, wobei M das Bezugsmatching und M' das betrachtete nicht-Δ-minimale Matching ist. Die beiden Δ-kleineren Matchings sind M_1 und M_2. Jede Kante des nicht-Δ-minimalen Matchings ist in mindestens einem der beiden entstandenen Δ-kleineren Matchings enthalten. Ebenso ist jede Kante der symmetrischen Differenz $M' \Delta M$ (siehe (d)) in genau einer der beiden symmetrischen Differenzen $M_1 \Delta M$ (siehe (f)) beziehungsweise $M_2 \Delta M$ (siehe (h)) enthalten.

Die Definition der Δ-minimalen Matchings kann auf Anfragegraphen erweitert werden. Dabei ist zu beachten, dass nur Matchings betrachtet werden, die valid (vgl. Definition 3.9) sind. Andernfalls wären nur Matchings Δ-minimal, deren symmetrische Differenz zum Bezugsmatching aus genau einem Kreis in genau einem Teilgraphen besteht.

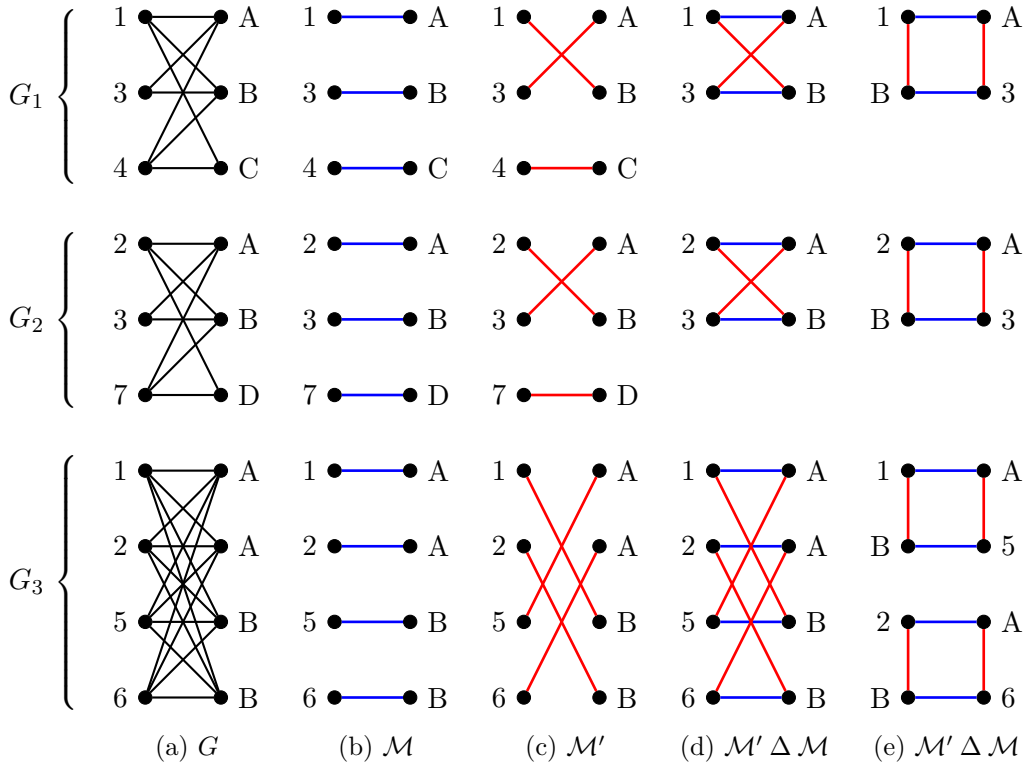
Definition 4.4 (Δ-minimales valides Matching für Anfragegraphen) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}$, $\mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Dann ist $\mathcal{M}'$ genau dann ein* Δ-minimales valides Matching *bezüglich $\mathcal{M}$, wenn für alle zu $\mathcal{M}$ verschiedenen validen Matchings $\mathcal{M}^*$ gilt: $\mathcal{M}^* \Delta \mathcal{M} \not\subset \mathcal{M}' \Delta \mathcal{M}$.*

Interessanterweise gilt Proposition 4.4 nicht für Anfragegraphen, das heißt, die symmetrische Differenz zwischen einem Δ-minimalen validen Matching und seinem Bezugsmatching kann auch aus mehreren Kreisen bestehen.

Beispiel 4.7 Abbildung 4.8 zeigt eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3)$ mit dem dazugehörigen Originalmatching $\mathcal{M}$. Weiterhin ist in (c) ein valides Matching $\mathcal{M}'$ sowie in (d) und (e) jeweils die symmetrische Differenz $\mathcal{M}' \Delta \mathcal{M}$ gegeben. Obwohl diese im Teilgraphen G_3 aus zwei Kreisen besteht, ist $\mathcal{M}'$ ein Δ-minimales Matching bezüglich $\mathcal{M}$. Das gilt, da jedes valide Matching, das die Kante $(3, \mathrm{A})$ enthält, zwangsläufig auch die Kanten $(1, \mathrm{B})$ und $(2, \mathrm{B})$ enthalten muss. Demnach kann es kein Matching geben, dessen symmetrische Differenz zu $\mathcal{M}$ aus weniger Kanten besteht als die von $\mathcal{M}'$.

Viele andere Eigenschaften von Δ-minimalen Matchings in einfachen Graphen gelten auch in Anfragegraphen, wenn valide Matchings betrachtet werden. Beispielsweise ist diese Beziehung auch hier symmetrisch, das heißt, ein Matching $\mathcal{M}$ ist genau dann Δ-minimal bezüglich eines Matchings $\mathcal{M}'$, wenn auch $\mathcal{M}'$ Δ-minimal bezüglich $\mathcal{M}$ ist (siehe Proposition B.1 im Anhang auf Seite 303). Weiterhin gelten Lemma 4.5 und Korollar 4.6 analog auch für Anfragegraphen.

[8] M_1 kann sogar so gewählt werden, dass es immer Δ-minimal bezüglich M ist.



(a) G (b) $\mathcal{M}$ (c) $\mathcal{M}'$ (d) $\mathcal{M}' \Delta \mathcal{M}$ (e) $\mathcal{M}' \Delta \mathcal{M}$

Abbildung 4.8: Symmetrische Differenz zwischen einem Δ-minimalen validen Matching und seinem Bezugsmatching

Lemma 4.7 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$, wobei $\mathcal{M}'$ kein Δ-minimales Matching bezüglich $\mathcal{M}$ sei. Dann gibt es zwei zu $\mathcal{M}$ und $\mathcal{M}'$ verschiedene valide Matchings $\mathcal{M}_1, \mathcal{M}_2$ in G mit*

$$(\mathcal{M}_1 \Delta \mathcal{M}) \cup (\mathcal{M}_2 \Delta \mathcal{M}) = \mathcal{M}' \Delta \mathcal{M}, \tag{4.4}$$

$$(\mathcal{M}_1 \Delta \mathcal{M}) \cap (\mathcal{M}_2 \Delta \mathcal{M}) = \emptyset. \tag{4.5}$$

Beweis: Die Matchings $\mathcal{M}_1$ und $\mathcal{M}_2$ existieren nach Definition 4.4 analog zum Beweis von Lemma 4.5. Zu zeigen bleibt hier nur, dass neben $\mathcal{M}_1$ auch $\mathcal{M}_2$ valid ist. Sei t ein beliebiges Tupel, für das in zwei verschiedenen Teilgraphen $G_i, G_j \in \mathcal{G}$ jeweils ein Tupelknoten $v_{t_i} \in G_i$ beziehungsweise $v_{t_j} \in G_j$ gehört. Seien $\{v_{t_i}, v_{s_i}\}$ und $\{v_{t_j}, v_{s_j}\}$ die entsprechenden Matchingkanten in $\mathcal{M}_2$, dann soll $v_{s_i} =_l v_{s_j}$ gelten. Falls $\{v_{t_i}, v_{s_i}\}$ und $\{v_{t_j}, v_{s_j}\}$ zusammen in $\mathcal{M}$ beziehungsweise $\mathcal{M}'$ enthalten sind, muss $v_{s_i} =_l v_{s_j}$ gelten, da $\mathcal{M}$ beziehungsweise $\mathcal{M}'$ valid sind. O. B. d. A. gelte andernfalls $\{v_{t_i}, v_{s_i}\} \in \mathcal{M}$ und $\{v_{t_j}, v_{s_j}\} \in \mathcal{M}'$. Dann gibt es zwei SA-Knoten $v'_{s_i} \in G_i$ und $v'_{s_j} \in G_j$ mit $\{v_{t_i}, v'_{s_i}\} \in \mathcal{M}'$ und $\{v_{t_j}, v'_{s_j}\} \in \mathcal{M}$. Aus der Validität von $\mathcal{M}$ und $\mathcal{M}'$ folgt $v_{s_j} =_l v'_{s_i}$ und $v_{s_i} =_l v'_{s_j}$. Nach der Konstruktion von $\mathcal{M}_2$ und $\mathcal{M}_1$ muss dann $\{v_{t_i}, v'_{s_i}\}, \{v_{t_j}, v'_{s_j}\} \in \mathcal{M}_1$ gelten und, da $\mathcal{M}_1$ ebenfalls valid ist, folgt $v'_{s_i} =_l v'_{s_j}$. Zusammen ergibt das $v_{s_i} =_l v'_{s_j} =_l v'_{s_i} =_l v_{s_j}$. □

Korollar 4.8 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in G, wobei $\mathcal{M}'$ kein Δ-minimales Matching bezüglich $\mathcal{M}$ sei. Dann gibt*

es zwei zu $\mathcal{M}$ und $\mathcal{M}'$ verschiedene valide Matchings $\mathcal{M}_1, \mathcal{M}_2$ in $\mathcal{G}$ mit $\mathcal{M}_1 \neq \mathcal{M}_2$, die beide jeweils bezüglich $\mathcal{M}$ Δ-kleiner sind als $\mathcal{M}'$.

Beweis: Analog zum Fall einfacher Graphen nach Korollar 4.6. □

Nach diesen Vorbetrachtungen kann bewiesen werden, dass jede Kante (t, s) eines beliebigen validen Matchings auch in mindestens einem Δ-minimalen Matching vorkommt.

Theorem 4.9 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Dann gibt es für jede Kante $e = \{v_t, v_s\} \in \mathcal{M}'$ mit $e \notin \mathcal{M}$ ein Δ-minimales Matching bezüglich $\mathcal{M}$, das e enthält.*

Beweis: Wenn $\mathcal{M}'$ bereits ein Δ-minimales Matching bezüglich $\mathcal{M}$ ist, ist nichts zu zeigen. Sei $\mathcal{M}'$ nicht Δ-minimal. Man betrachte die beiden zu $\mathcal{M}$ Δ-kleineren Matchings $\mathcal{M}_1$ und $\mathcal{M}_2$ nach Lemma 4.7 und Korollar 4.8. Für die Kante e gilt $e \in \mathcal{M}' \,\Delta\, \mathcal{M}$. Aus Gleichung 4.4 folgt $e \in \mathcal{M}_1$ oder $e \in \mathcal{M}_2$. Falls das Matching, das e enthält, nicht Δ-minimal ist, so kann Lemma 4.7 erneut angewendet werden, um die symmetrische Differenz weiter zu verkleinern. Da $\mathcal{M}'$ endlich ist, terminiert die Teilung des Matchings mit zwei Δ-kleineren Matchings, die beide Δ-minimal sein müssen und von denen eins e enthält. □

Theorem 4.9 stellt den Hauptbeitrag in diesem Abschnitt dar und charakterisiert eine weitere Beschränkung der modellierten Matchings. Um k-assign Anonymität zu berechnen, das heißt um festzustellen, ob ein Matching mit einer gegebenen Kante (t, s) existiert, reicht es aus, nur Δ-minimale Matchings zu betrachten. Als Bezugsmatching wird dabei erneut das leicht zu berechnende Originalmatching gewählt. Neben der SA-Äquivalenz ist die Δ-Minimalität die zweite Eigenschaft, die die Menge aller zu modellierenden Matchings beschränkt und dabei sicherstellt, dass weiterhin jede Zuordnung von SA-Werten zu Tupeln erhalten bleibt.

4.3.4 Δr-Matchings

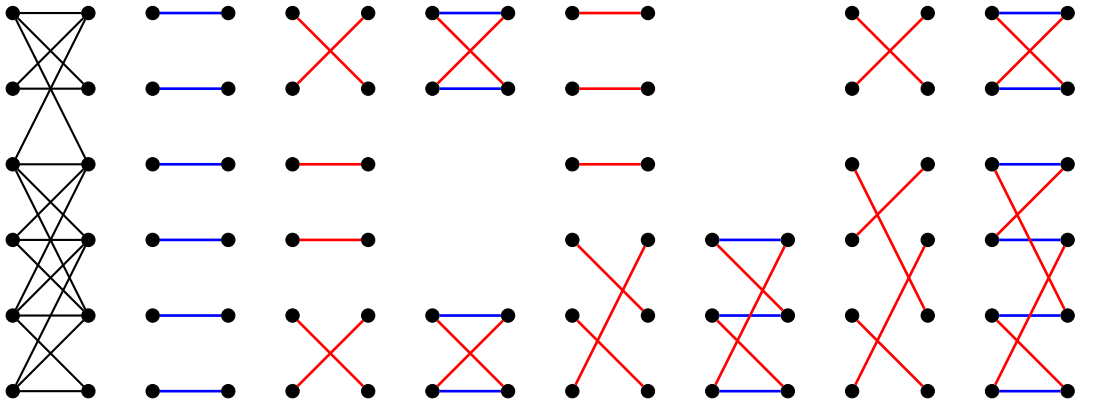
In einfachen Graphen ist die Anzahl der perfekten Matchings im Allgemeinen exponentiell in der Größe des Graphen. Auch durch die Beschränkung auf Δ-minimale Matchings können immer noch exponentiell viele Matchings existieren. Da aber eine Anforderung an den Approximationsalgorithmus war, nur polynomielle Laufzeit zu benötigen, muss die Menge aller betrachteter Matchings weiter eingeschränkt werden. Dabei dürfen zusätzlich nur Matchings betrachtet werden, die in Polynomialzeit berechnet werden können. Aus dieser Motivation ergeben sich die folgenden Einschränkungen.

Nach Proposition 4.3 gilt, dass die symmetrische Differenz zwischen zwei perfekten Matchings immer aus Kreisen gerader Länge besteht. Mithilfe dieser Länge können Matchings in Klassen eingeteilt werden. Sei ein festes Matching M gegeben, dann können alle anderen Matchings anhand der maximalen Länge eines Kreises in der symmetrischen Differenz zu M klassifiziert werden.

Definition 4.5 (Δr-Matching für Graphen) *Seien G ein Graph und M, M' zwei perfekte Matchings in G. Dann ist M' genau dann ein Δr-Matching[9] bezüglich M, wenn die maximale Länge eines Kreises in $M' \,\Delta\, M$ höchstens $2r$ ist.*

[9] ausgesprochen: *Delta-r-Matching*

Die Beschränkung auf perfekte Matchings ist für die Definition nicht zwingend erforderlich, sie ist aber sinnvoll. Andernfalls können in der symmetrischen Differenz auch Pfade vorhanden sein, die keine Kreise sind.



(a) G (b) M (c) M_1 (d) $M_1 \Delta M$ (e) M_2 (f) $M_2 \Delta M$ (g) M_3 (h) $M_3 \Delta M$

Abbildung 4.9: Bezüglich M ist M_1 ein Δ2-, M_2 ein Δ3- und M_3 ein Δ4-Matching

Beispiel 4.8 Abbildung 4.9 zeigt einen Graphen G sowie die Matchings M, M_1, M_2 und M_3. Die symmetrische Differenz $M_1 \Delta M$ (siehe (d)) enthält zwei Kreise der Länge 4 und demnach ist M_1 ein Δ2-Matching bezüglich M. Aus analogen Gründen ist M_2 ein Δ3-Matching und M_3 ein Δ4-Matching jeweils bezüglich M.

Die Definition für Δr-Matchings kann für Anfragegraphen erweitert werden, wobei die Bedingung in jedem Teilgraphen gelten soll.

Definition 4.6 (Δr-Matching für Anfragegraphen) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei perfekte Matchings in $\mathcal{G}$. Dann ist $\mathcal{M}'$ genau dann ein* Δr-Matching *bezüglich $\mathcal{M}$, wenn die maximale Länge eines Kreises in $\mathcal{M}' \Delta \mathcal{M}$ höchstens $2r$ ist.*

Die Beschränkung auf valide Matchings in Anfragegraphen ist für die Definition der Δr-Matchings nicht notwendig, obwohl in dieser Arbeit nur valide Matchings betrachtet werden.

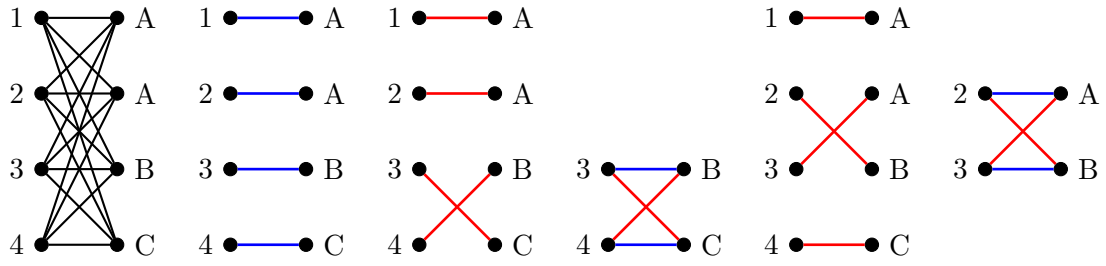
Sei das Originalmatching $\mathcal{M}_{\text{orig}}$ in einem Anfragegraphen als Bezugsmatching gewählt. Dann können alle anderen validen Matchings nach ihrem Wert für r als Δr-Matching kategorisiert werden. Bei n Tupeln gibt es damit Δ2-, Δ3-, …, Δn-Matchings.[10] Nach der Definition gilt dabei jeweils, dass jedes Δi-Matching auch ein $\Delta(i+1)$-Matching ist.

Eine Besonderheit liegt vor, wenn die Folge von Anfragegraphen aus nur einem Teilgraphen besteht. In diesem Fall kann recht einfach mithilfe von Δ2-Matchings sichergestellt werden, dass jedes Tupel mit jedem SA-Wert matcht. Dazu wird für jedes Tupel t und jeden verschiedenen SA-Wert s eine Kante (t, s) ausgewählt. Ein entsprechendes Matching $\mathcal{M}'$ entsteht aus $\mathcal{M}_{\text{orig}}$, indem die Originalkanten (t', s) und (t, s') entfernt und durch die Kanten (t, s) sowie (t', s') ersetzt werden. In diesem Fall heißt (t', s') *Gegenkante* von (t, s) und umgekehrt.

[10] Δ1-Matchings sind aus Paritätsgründen nicht möglich.

ID	Name	SA	M_1	M_2	M_3	M_6	M_7	M_9	M_{11}	M_{12}
1	Alison	A	A	A	A	A	B	B	C	C
2	Ben	A	A	A	B	C	A	C	A	B
3	Clark	B	B	C	A	B	A	A	B	A
4	Debra	C	C	B	C	A	C	A	A	A
Δr bez. M_1			–	2	2	2	2	2	2	2
Δ-min. bez. M_1			–	✓	✓	✓	✓	–	✓	–

(a) Datentabelle und SA-Werteverteilungen der Δ2-Matchings



(b) G_1 (c) M_1 (d) M_2 (e) $M_2 \Delta M_1$ (f) M_3 (g) $M_3 \Delta M_1$

Abbildung 4.10: Alle Δ2-Matchings für G_1. M_1 ist das Originalmatching, die anderen angegebenen Matchings sind jeweils Δ2-Matchings bezüglich M_1, wobei alle Matchings außer M_9 und M_{12} auch Δ-minimal bezüglich M_1 sind.

Beispiel 4.9 Abbildung 4.10 zeigt einen Anfragegraphen G_1, der aus vier Tupel- und SA-Knoten besteht. Es ist der bereits in Kapitel 3 eingeführte Graph zur Beispielanfrage Q_1 (vgl. Abbildung 3.2 und Tabelle 3.5). In (a) sind alle SA-Werteverteilungen aufgelistet, die mithilfe von Δ2-Matchings in G_1 dargestellt werden können. Die Indizes der Matchings entsprechen dabei der bereits eingeführten Nummerierung der SA-Werteverteilungen (vgl. Tabelle 3.5 auf Seite 70). Das Originalmatching ist hier mit M_1 angegeben. Matching M_2 kann beispielsweise konstruiert werden, indem in M_1 die Originalkante $(3, \mathrm{B})$ durch die Kante $(3, \mathrm{C})$ und gleichzeitig $(4, \mathrm{C})$ durch $(4, \mathrm{B})$ ersetzt wird. $(4, \mathrm{B})$ ist dabei die Gegenkante von $(3, \mathrm{C})$. Die Matchings M_2, M_3, M_6, M_7 und M_{11} sind zusätzlich Δ-minimal bezüglich M_1 und gewährleisten zusammen, dass jeder SA-Wert jedem Tupel zugewiesen werden kann.

In (d) und (f) sind exemplarisch die Matchings M_2 und M_3 sowie in (e) und (g) die Differenzen $M_2 \Delta M_1$ und $M_3 \Delta M_1$ dargestellt. Alle SA-Werteverteilungen und die dazugehörigen perfekten Matchings in G_1 sind im Anhang in Abbildung A.3 auf Seite 271 angegeben (vgl. Beispiel A.2). Neben den hier erwähnten Δ2-Matchings bezüglich M_1 sind darin auch alle Δ3-Matchings enthalten.

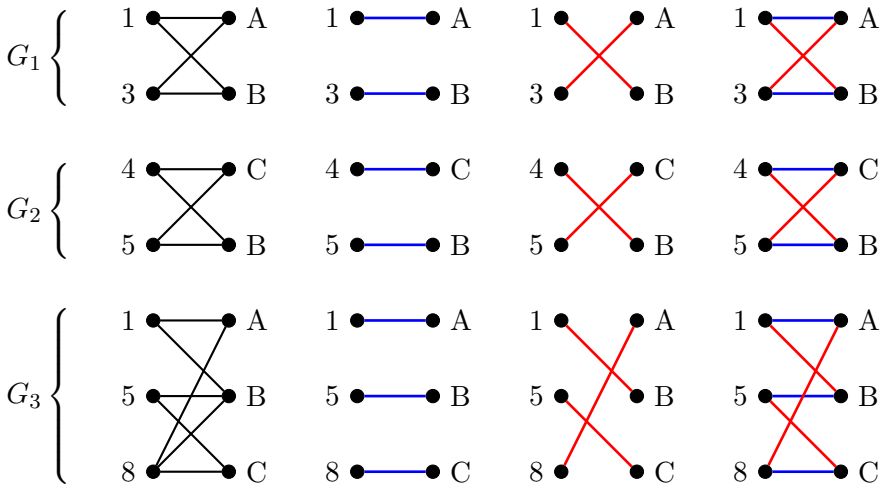
Für Anfragegraphen mit n Tupelknoten können alle Zuordnungen von sensiblen Werten zu Tupeln mithilfe von Δ3-, Δ4- bis hin zu Δn-Matchings dargestellt werden. An dieser Stelle soll aber die Menge der betrachteten Matchings verringert werden, damit die Berechnung der einzelnen Matchings vereinfacht wird. Es werden nur noch Matchings modelliert, die Δ2-Matchings bezüglich des Originalmatchings sind. Dadurch ergeben sich eine Reihe von Vorteilen, aber auch einige Nachteile. Positiv ist, dass für einen einzelnen Anfragegraphen alle Zuordnungen von SA-Werten zu Tupeln mit Δ2-Matchings abgebil-

det werden können. In diesem Fall kann k-assign Anonymität weiterhin korrekt berechnet werden. Der entscheidende Vorteil liegt aber bei der Berechnung solcher Matchings. Das zeigt sich insbesondere bei der verwendeten Datenstruktur (siehe Kapitel 4.5) als auch bei der tatsächlichen Berechnung der Matchings (siehe Kapitel 5 und 7). Die Idee ist, dass Matchings einfacher bestimmt werden können, wenn sie sich nur sehr wenig vom Originalmatching unterscheiden. Wenig heißt dabei, dass sie Δ-minimal sind und dass so wenige Tupel wie möglich nicht mit ihrem originalen SA-Wert matchen. In den nachfolgenden Kapiteln wird dieser Vorteil deutlich erkennbar.

Der Nachteil bei der Beschränkung auf $\Delta 2$-Matchings ist, dass im Allgemeinen bestimmte SA-Werteverteilungen nicht mehr abgebildet werden. Aus der Menge der verbleibenden Matchings können nicht mehr alle Zuordnungen von sensiblen Werten zu Tupeln, die ein Angreifer vermuten könnte, abgelesen werden. Eine Verletzung von k-assign Anonymität würde eventuell auch ausgegeben werden, wenn es sie gar nicht gibt (false positive). Wichtig ist aber, dass jede tatsächliche Verletzung auch gefunden wird, insgesamt also nicht mehr Möglichkeiten modelliert werden, als wirklich vorhanden sind (keine false negatives).

ID	Name	SA	$\mathcal{M}$	$\mathcal{M}_1$
1	Alison	A	A	B
3	Clark	B	B	A
4	Debra	C	C	B
5	Elaine	B	B	C
8	Helen	C	C	A

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) $\mathcal{G}$ (c) $\mathcal{M}$ (d) $\mathcal{M}_1$ (e) $\mathcal{M}_1 \,\Delta\, \mathcal{M}$

Abbildung 4.11: Anfragegraph $\mathcal{G} = (G_1, G_2, G_3)$, bei dem $\Delta 2$-Matchings nicht ausreichen, um alle Zuordnungen von SA-Werten zu Tupeln zu modellieren.

Beispiel 4.10 In Abbildung 4.11 sind in (c) und (d) zwei Matchings zu einer Folge von Anfragegraphen $\mathcal{G}$ dargestellt. $\mathcal{M}_1$ ist dabei ein $\Delta 3$-Matching bezüglich des Originalmatchings $\mathcal{M}$ (siehe (e)). Für $\mathcal{G}$ existiert kein $\Delta 2$-Matching bezüglich $\mathcal{M}$, das die Kante $(1, \mathrm{B})$

enthält. Werden Δr-Matchings mit $r > 2$ nicht modelliert, wird die Zuordnung des sensiblen Wertes B zu Tupel 1 nicht repräsentiert.

Das obige Beispiel kann so erweitert werden, dass die Diskrepanz zwischen der Anzahl tatsächlicher SA-Werteverteilungen und denen, die mithilfe von $\Delta 2$-Matchings modelliert werden, beliebig groß wird. Beispiel A.3 im Anhang auf Seite 270 zeigt eine Folge von Anfragegraphen, bei der einem Tupel durch beliebige Matchings insgesamt m verschiedene SA-Werte zugeordnet werden können. Allerdings gibt es kein $\Delta 2$-Matching, bei dem dieses Tupel auch nur mit irgendeinem anderen SA-Wert matcht als mit seinem originalen. Demzufolge würde die Beschränkung auf $\Delta 2$-Matchings zu einem minimalen Anonymitätsgrad von 1 führen, während der korrekte Wert m ist. Somit wäre eine darauf basierende Approximation von k-assign Anonymität beliebig schlecht. Eine Verallgemeinerung dieser Situation stellt Beispiel A.4 im Anhang auf Seite 273 dar. Es zeigt, dass die Beschränkung auf Δr-Matchings für jedes feste r immer dazu führen kann, dass für die resultierende Approximation keine feste Güte angegeben werden kann. Da allerdings die Vorteile der Darstellung (siehe Kapitel 4.5) und Berechnung (siehe Kapitel 5) bei $\Delta 2$-Matchings überwiegen, wurden diese für die Approximation ausgewählt. Insbesondere wird die polynomielle Laufzeit beim Bestimmen von Matchings nur für $\Delta 2$-Matchings garantiert.

Über die SA-Labels der Knoten zweier $\Delta 2$-Matchings, die auch Δ-minimal zueinander sind, kann eine Aussage getroffen werden, die für den später vorgestellten Algorithmus hilfreich ist. Es fällt auf, dass in der symmetrischen Differenz der beiden Matchings immer nur genau zwei unterschiedliche SA-Werte vorkommen. Das wird in der folgenden Proposition festgehalten.

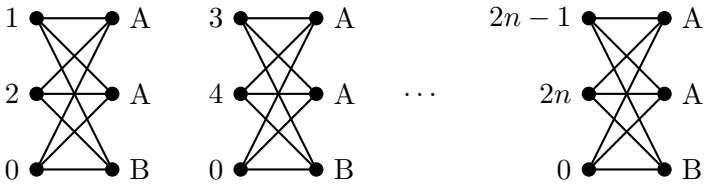
Proposition 4.10 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}$ ein valides Matching in $\mathcal{G}$. Dann gilt für alle validen Matchings $\mathcal{M}'$, die Δ-minimal und $\Delta 2$-Matchings bezüglich $\mathcal{M}$ sind, dass alle SA-Knoten in $\mathcal{G}[\mathcal{M}' \Delta \mathcal{M}]$ nur maximal zwei verschiedene Labels haben.*

Beweis: Der Beweis erfolgt indirekt. Sei $\mathcal{M}'$ ein valides perfektes Matching, das Δ-minimal und ein $\Delta 2$-Matching zu $\mathcal{M}$ ist, und $\mathcal{G}_\Delta = \mathcal{G}[\mathcal{M}' \Delta \mathcal{M}]$ habe mindestens drei verschiedene SA-Labels. Dann existieren in $\mathcal{G}_\Delta$ zwei Kreise C_1 und C_2 (der Länge 2), wobei in C_1 ein SA-Label vorkommt, das nicht in C_2 vorkommt. O. B. d. A. seien A und B die Labels in C_1. Sei $\mathcal{G}^*_\Delta$ der Graph, der aus $\mathcal{G}_\Delta$ entsteht, indem alle Kreise (der Länge 2) gelöscht werden, die nur aus SA-Knoten mit Labels A oder B und beliebigen Tupelknoten bestehen. $\mathcal{M}^*$ sei das Matching, für das gilt $\mathcal{G}^*_\Delta = \mathcal{M}^* \Delta \mathcal{M}$. $\mathcal{M}^*$ enthält man aus $\mathcal{M}'$, indem für alle Tupel t, die in $\mathcal{M}'$ mit dem SA-Wert A (beziehungsweise B) und in $\mathcal{M}$ mit SA-Wert B (beziehungsweise A) matchen, die Matchingkante aus $\mathcal{M}$ gewählt wird. Für $\mathcal{M}^*$ gilt nun $\mathcal{M}^* \neq \mathcal{M}$, da C_2 nicht aus $\mathcal{M}'$ gelöscht wurde und somit in $\mathcal{G}^*_\Delta$ vorhanden ist. Weiterhin ist $\mathcal{M}^*$ valid, denn alle Tupellabel in gelöschten Kreisen kommen nirgends anders vor (ein Tupellabel, das in einem anderen Kreis der Länge 2 vorkommen würde, in dem auch ein zu A und B verschiedenes SA-Label vorkommt, widerspricht der Validität von $\mathcal{M}$ oder $\mathcal{M}'$). Dann folgt aber mit $\mathcal{M}^* \Delta \mathcal{M} \subset \mathcal{M}' \Delta \mathcal{M}$ ein Widerspruch zur Minimalität von $\mathcal{M}'$ und demzufolge ist die Annahme falsch. □

Interessanterweise kann Proposition 4.10 nicht auf Δr-Matchings mit $r > 2$ erweitert werden. Insbesondere können in einer entsprechenden symmetrischen Differenz beliebig viele verschiedene SA-Labels vorkommen, wenn zum Beispiel $\Delta 3$-Matchings zugelassen sind. In Beispiel A.5 im Anhang auf Seite 274 wird ein ähnlicher Fall beschrieben.

ID	SA	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\cdots$	$\mathcal{M}_{2^n}$
0	B	A	A	A	A	$\cdots$	A
1	A	B	A	B	A	$\cdots$	A
2	A	A	B	A	B	$\cdots$	B
3	A	B	B	A	A	$\cdots$	A
4	A	A	A	B	B	$\cdots$	B
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$2n$	A	A	A	A	A	$\cdots$	B

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) G_1 (c) G_2 (d) G_n

Abbildung 4.12: In $\mathcal{G} = (G_1, G_2, \ldots, G_n)$ gibt es exponentiell viele Δ-minimale $\Delta 2$-Matchings.

Durch die Beschränkung der Modellierung auf $\Delta 2$-Matchings wird die Anzahl der betrachteten Matchings und damit auch der SA-Werteverteilungen stark verringert. Dennoch können im Allgemeinen weiterhin exponentiell viele $\Delta 2$-Matchings existieren, die sogar Δ-minimal sind. Dazu seien in Abbildung 4.12 n Graphen $G_1, G_2, \ldots, G_n$ gegeben, wobei jeder Graph G_i die Tupel $2i-1$, $2i$ und 0 sowie die SA-Werte A, A und B enthalte (siehe (b) bis (d)).[11] Ist nach einem Matching mit einer Kante $(0, \mathrm{A})$ gefragt, muss in jedem Graphen genau ein Tupel mit B matchen. Für G_i muss daher entweder Tupel $2i-1$ oder $2i$ mit B matchen. Somit gibt es für die Folge von Graphen $\mathcal{G}^{(n)}$ insgesamt 2^n mögliche Matchings, die jeweils auch Δ-minimal bezüglich des Originalmatchings sind (siehe (a)).

Damit die Anforderungen eingehalten werden und die Approximation eine polynomielle Laufzeit hat, muss die Menge der modellierten Matchings weiter eingeschränkt werden. Dazu wird nur ein Teil der Δ-minimalen $\Delta 2$-Matchings berechnet, der trotzdem noch möglichst viele verschiedene Zuordnungen von SA-Werten zu Tupeln repräsentiert. Im obigen Beispiel gibt es zwar mindestens 2^n verschiedene Matchings, aber jedes Tupel $1, 2, \ldots, 2n$ matcht dabei nur mit zwei verschiedenen SA-Werten. Demzufolge würden $4n$ Matchings ausreichen, um diese Zuordnungen zu gewährleisten. Wie diese Matchings ausgewählt werden, wird bei der Vorstellung des Approximationsalgorithmus in Kapitel 5 erläutert.

4.3.5 Δ-Pfadkonformität

Die Motivation für das letzte hier eingeführte Kriterium, wann ein Matching nicht betrachtet wird, liegt in der effizienten Berechnung von Matchings. Dazu wird an dieser Stelle ein kleiner Vorgriff auf den darauffolgenden Abschnitt gegeben, in dem die Datenstruktur zur

[11] In der Datentabelle wird auf das Name-Attribut verzichtet, da dieses Beispiel nicht auf der sonst verwendeten Datenbasis aus Tabelle 3.3 auf Seite 68 basiert.

Speicherung von Matchings vorgestellt wird. Die Idee ist, nicht alle Kanten eines validen Matchings zu speichern, sondern nur die, die keine Originalkanten sind. Weiterhin werden die Matchings nicht einzeln gespeichert, sondern global alle Kanten, die in irgendeinem Matching vorkommen. Damit daraus die Matchings effizient rekonstruierbar sind, muss für die Kanten eine gewisse „globale“ Eigenschaft erfüllt sein, die im Folgenden vorgestellt wird.

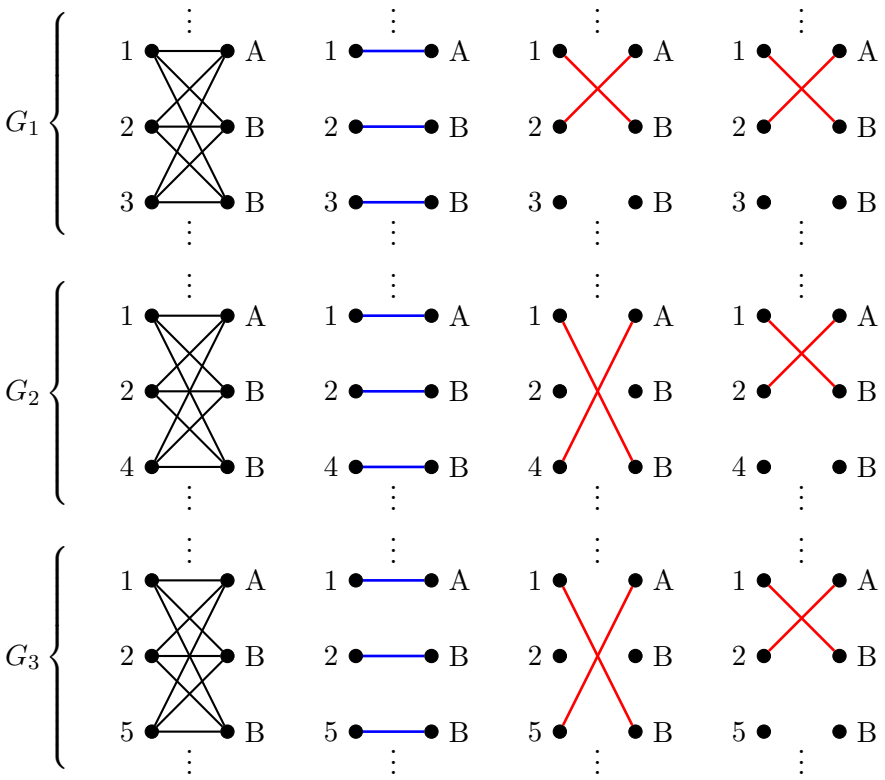


Abbildung 4.13: Anfragegraph $\mathcal{G} = (G_1, G_2, G_3)$

Sei die Folge von Anfragegraphen $\mathcal{G}$ aus Abbildung 4.13 gegeben, die aus mindestens drei Teilgraphen G_1, G_2 und G_3 bestehe. Jeder dieser Graphen enthalte die Tupel 1 und 2 sowie die SA-Werte A und B. Dazu sei in jedem Graph mindestens ein weiteres Tupel und ein zweiter SA-Wert B vorhanden (siehe (a)). (b) zeigt das Originalmatching $\mathcal{M}_{\text{orig}}$, welches aus allen horizontalen Kanten besteht. In $\mathcal{G}$ soll nun ein Matching bestimmt werden, das die Kante $(1, \text{B})$ enthält. Dazu gibt es in jedem Teilgraphen zwei Möglichkeiten. $\mathcal{M}_1$ aus (c) ist beispielsweise ein Matching, das in G_1 die Kante von 1 zum mittleren Knoten mit Label B und in G_2 sowie G_3 jeweils die Kante von 1 zum unteren Knoten mit Label B enthält. An dieser Stelle soll stets gelten, dass alle anderen Knoten in $\mathcal{G}$ ebenfalls Matchingpartner finden, sodass das Matching valid ist (dargestellt durch die Auslassungspunkte). Damit $\mathcal{M}_1$ ein $\Delta 2$-Matching ist, müssen die Tupel 2, 4 und 5 jeweils mit einem A matchen.

Sei $\mathcal{M}_2$ ein anderes Matching, in dem Tupel 1 in jedem Graphen mit dem SA-Knoten matcht, der der Matchingpartner von 2 im Originalmatching ist (siehe (c)). Das ist in allen drei Graphen der mittlere Knoten, der mit B gelabelt ist. Um die $\Delta 2$-Matching-

Eigenschaft zu erfüllen, muss nur Tupel 2 in jedem Graphen mit einem A matchen. Sollen beide Matchings abgebildet werden, muss für $\mathcal{M}_1$ für jedes Tupel in jedem Teilgraphen der entsprechende Matchingpartner gespeichert werden. Insbesondere ist das für die Tupel 1 und 2 in jedem Teilgraphen ein anderer Knoten. Um $\mathcal{M}_2$ abzubilden, muss sich nur gemerkt werden, dass die Matchingpartner von 1 und 2 bezüglich des Originalmatchings in allen Graphen vertauscht sind. Das heißt, Tupel 1 matcht in jedem Graphen, in dem es vorkommt, mit dem Knoten, der der originale Matchingpartner von Tupel 2 ist (und umgekehrt). In diesem Fall müssen die Matchingpartner von 1 nicht pro Graph, sondern nur global für alle Graphen gespeichert werden. Ein weiterer Vorteil ist, dass die symmetrische Differenz von $\mathcal{M}_2$ zum Originalmatching insgesamt weniger Kanten enthält als die von $\mathcal{M}_1$.

Basierend auf diesem Beispiel wird nun eine weitere Beschränkung der Menge der betrachteten Matchings definiert. Das Ziel dabei lautet, Matchingpartner von Tupelknoten global, das heißt unabhängig von den einzelnen Teilgraphen, zu definieren. In $\Delta 2$-Matchings sollen folglich je zwei Tupel, die in einem Teilgraphen vertauschte Matchingpartner haben, in allen anderen Teilgraphen, in denen sie zusammen vorkommen, auch vertauschte Matchingpartner haben. Damit diese Eigenschaft auch für Δr-Matchings mit $r > 2$ anwendbar ist, wird sie allgemeingültig definiert. Angenommen, in einem Teilgraphen matcht ein Tupel t mit dem originalen SA-Knoten eines anderen Tupels t'. Dann wird gefordert, dass in jedem anderen Teilgraphen, in dem t und t' vorkommen, t auch mit dem originalen SA-Knoten von t' matcht. Diese Eigenschaft kann über Pfade in den symmetrischen Differenzen der Matchings definiert werden und heißt daher Δ-*Pfadkonformität*[12].

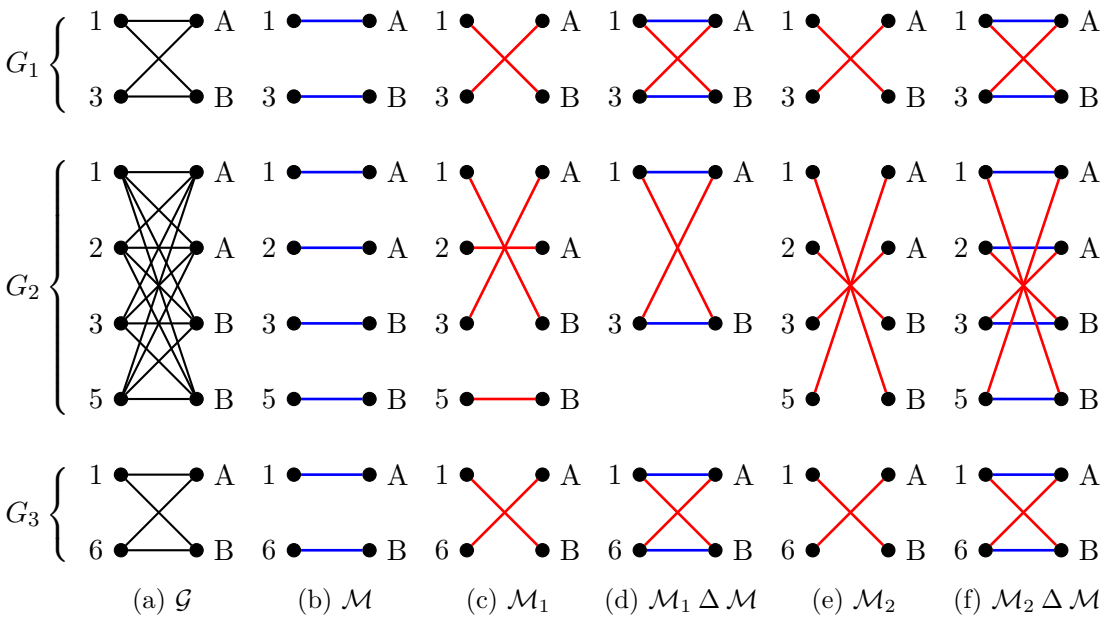
Definition 4.7 (Δ-pfadkonform) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}$, $\mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Dann ist $\mathcal{M}'$ genau dann* Δ-pfadkonform *bezüglich $\mathcal{M}$, wenn für alle Graphen $G_i, G_j \in \mathcal{G}$ gilt: Wenn es einen Pfad $P_i = v_1, v_2, v_3 \in \mathcal{M}'[G_i] \,\Delta\, \mathcal{M}[G_i]$ und Knoten $v'_1, v'_2, v'_3 \in \mathcal{M}'[G_j] \,\Delta\, \mathcal{M}[G_j]$ mit den gleichen Labels (d. h. $v_1 =_l v'_1, v_2 =_l v'_2, v_3 =_l v'_3$) gibt, dann existiert auch ein Pfad $P_j = v^*_1, v^*_2, v^*_3 \in \mathcal{M}'[G_j] \,\Delta\, \mathcal{M}[G_j]$ mit gleichen Labels (d. h. $v_1 =_l v^*_1, v_2 =_l v^*_2, v_3 =_l v^*_3$).*

Die in der Definition verwendeten Knoten v'_1, v'_2, v'_3 und v^*_1, v^*_2, v^*_3 können identisch sein, müssen das aber nicht, wenn mehrere gleiche SA-Labels in G_j vorkommen.

Beispiel 4.11 Abbildung 4.14 zeigt eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2)$ sowie die Matchings $\mathcal{M}$, $\mathcal{M}_1$ und $\mathcal{M}_2$. Während $\mathcal{M}_1$ Δ-pfadkonform bezüglich $\mathcal{M}$ ist, gilt diese Eigenschaft nicht für $\mathcal{M}_2$. In G_1 existiert in $\mathcal{M}_1 \,\Delta\, \mathcal{M}$ und $\mathcal{M}_2 \,\Delta\, \mathcal{M}$ jeweils ein Pfad von Knoten mit den Labels $1, \mathrm{B}, 3$. Da es in G_2 ebenfalls Knoten gibt, die die Labels 1, B und 3 haben, muss es für Δ-Pfadkonformität einen entsprechenden Pfad in den jeweiligen symmetrischen Differenzen geben. Das ist aber in $\mathcal{M}_2 \,\Delta\, \mathcal{M}$ nicht der Fall, denn dort gibt es nur einen Pfad von Knoten mit den Labels $1, \mathrm{B}, 5$. Der Graph G_3 spielt für die Bestimmung von Δ-Pfadkonformität übrigens keine Rolle, denn dort gibt es zwar einen Tupelknoten mit Label 1, aber keinen mit Label 3.

Als Beschränkung gilt nun, dass nur Matchings betrachtet werden, die Δ-pfadkonform bezüglich des Originalmatchings sind. Der entscheidende Vorteil davon wird im nächsten Abschnitt und in Kapitel 5 sichtbar, wenn die Berechnung der Matchings vorgestellt wird. Ein Nachteil ist, dass analog zur Beschränkung auf $\Delta 2$-Matchings ebenfalls Matchings

[12] ausgesprochen: *Delta-Pfadkonformität*



(a) $\mathcal{G}$ (b) $\mathcal{M}$ (c) $\mathcal{M}_1$ (d) $\mathcal{M}_1 \Delta \mathcal{M}$ (e) $\mathcal{M}_2$ (f) $\mathcal{M}_2 \Delta \mathcal{M}$

Abbildung 4.14: Zwei Matchings für $\mathcal{G} = (G_1, G_2, G_3)$, die jeweils die Kante $(1, \mathrm{B})$ beinhalten. $\mathcal{M}_1 \Delta \mathcal{M}$ besteht aus insgesamt weniger Kanten als $\mathcal{M}_2 \Delta \mathcal{M}$ und $\mathcal{M}_2$ ist nicht Δ-pfadkonform bezüglich $\mathcal{M}$.

und SA-Werteverteilungen nicht mehr dargestellt werden. Dadurch können Verletzungen des Schutzkriteriums registriert werden, die eigentlich nicht existieren (false positives). Das nächste Beispiel zeigt so einen Fall. Aber auch hier gilt, dass alle tatsächlichen Verletzungen auch weiterhin gefunden werden (keine false negatives).

Beispiel 4.12 Abbildung 4.15 zeigt eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3)$ und ein Matching $\mathcal{M}_1$, welches die Kante $(1, \mathrm{B})$ enthält. In der symmetrischen Differenz zwischen $\mathcal{M}_1$ und $\mathcal{M}$ gibt es in G_2 einen Pfad, der Knoten mit den Labels 1, B, 5 enthält. In G_3 kommen zwar ebenfalls Knoten mit den Labels 1, B, 5 vor, es gibt in $\mathcal{M}_1 \Delta \mathcal{M}$ aber keinen entsprechenden Pfad. $\mathcal{M}_1$ ist demnach nicht Δ-pfadkonform bezüglich des Originalmatchings $\mathcal{M}$ und würde nicht modelliert werden. Da es kein anderes valides Matching gibt, welches die Kante $(1, \mathrm{B})$ enthält und Δ-pfadkonform bezüglich $\mathcal{M}$ ist, wird im Rahmen der Approximation keine SA-Werteverteilung berechnet, die Tupel 1 den SA-Wert B zuordnet.

4.4 Algorithmische Einschränkungen

Durch die vorgestellte Klassifikation kann jetzt angegeben werden, welche Matchings in Anfragegraphen für die Approximation des Angreiferwissens benutzt werden. Für jedes verwendete Matching $\mathcal{M}$ muss dazu gelten:

- $\mathcal{M}$ ist valid.
- $\mathcal{M}$ ist nicht partiell SA-äquivalent bezüglich des Originalmatchings.
- $\mathcal{M}$ ist Δ-minimal bezüglich des Originalmatchings.

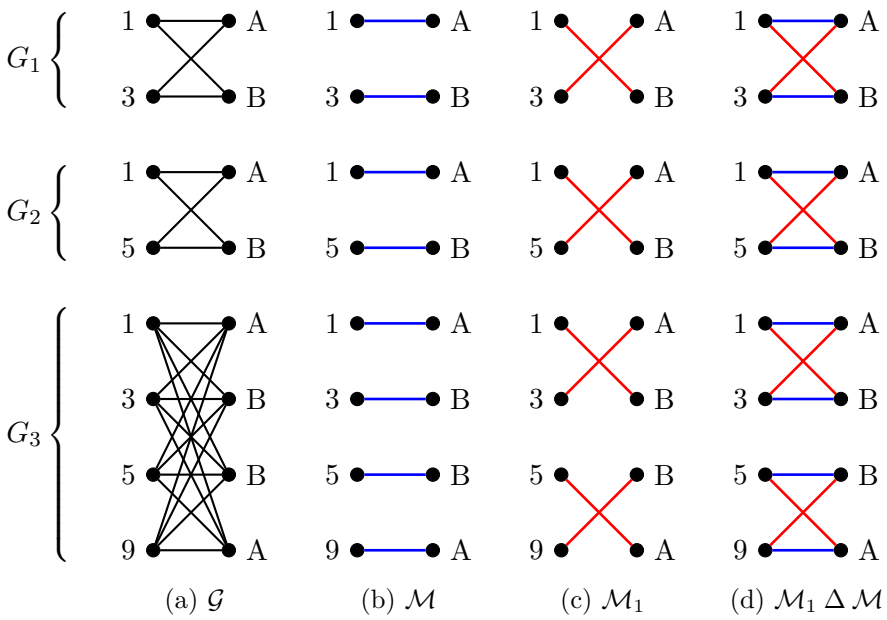


Abbildung 4.15: In $\mathcal{G} = (G_1, G_2, G_3)$ gibt es mit $\mathcal{M}_1$ nur ein Matching, das die Kante $(1, \mathrm{B})$ enthält. Es ist jedoch nicht Δ-pfadkonform.

- $\mathcal{M}$ ist ein $\Delta 2$-Matching bezüglich des Originalmatchings.
- $\mathcal{M}$ ist Δ-pfadkonform bezüglich des Originalmatchings.

Der Einfachheit halber wird nun ein Begriff eingeführt, der diese Kriterien zusammenfasst. Ein Matching $\mathcal{M}$, welches die genannten Eigenschaften erfüllt, heißt Δ-*top*[13].

Definition 4.8 (Δ-top) *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}_{orig}$ ein Originalmatching in $\mathcal{G}$. Dann heißt ein Matching $\mathcal{M}$ in $\mathcal{G}$* Δ-top*, wenn es valid und bezüglich $\mathcal{M}_{orig}$ nicht partiell SA-äquivalent, Δ-minimal, ein $\Delta 2$-Matching sowie Δ-pfadkonform ist.*

Obwohl durch das Beschränken auf Δ-top Matchings die Anzahl von modellierten Matchings reduziert wurde, ist sie im Allgemeinen noch immer exponentiell groß in der Anzahl aller Tupel und damit in der Eingabegröße. Der Beispielgraph aus Abbildung 4.12, welcher bereits zeigte, dass die Anzahl aller $\Delta 2$-Matchings exponentiell ist, erfüllt auch die Δ-top-Eigenschaft.[14] Folglich müssen weitere Kriterien angegeben werden, nach denen nur einige Matchings ausgewählt werden, sodass die Gesamtanzahl der betrachteten Matchings nur polynomielle Größe hat. Diese hängen aber von den verwendeten Algorithmen und Heuristiken ab und werden daher erst in den Kapiteln 5 und 7 vorgestellt. Die Idee dabei ist, in so einem Fall wie in Abbildung 4.12 nur eine polynomiell große Teilmenge aller Δ-top Matchings zu berechnen. Nur für den Fall, in dem genau eine Anfrage beziehungsweise ein Anfragegraph gegeben ist, kann mithilfe von Δ-top Matchings k-assign Anonymität exakt bestimmt werden (vgl. Bemerkung im Kapitel 4.3.4).

[13] ausgesprochen: *delta-top*

[14] Beispiel A.10 im Anhang auf Seite 280 beschreibt einen ähnlichen Fall, in dem ebenfalls exponentiell viele Δ-top Matchings existieren.

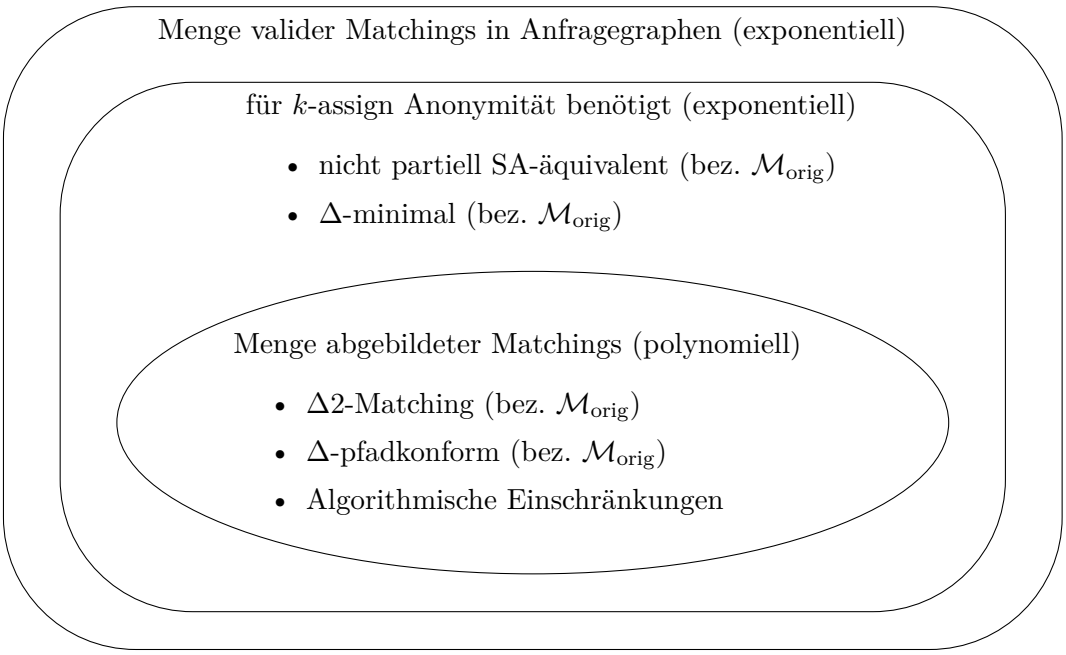


Abbildung 4.16: Modellierte Matchings

Ein Überblick über alle verwendeten Einschränkungen ist in Abbildung 4.16 zu finden. Die noch fehlenden Einschränkungen, die in den Algorithmen in Kapitel 5 und 7 vorgestellt werden, sind bereits enthalten.

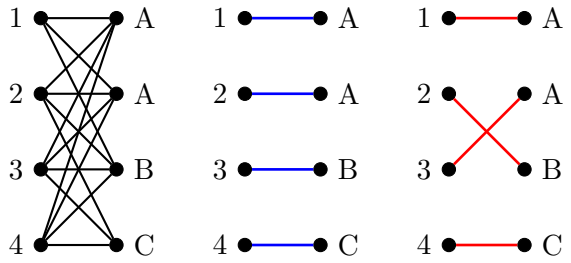
4.5 Repräsentation von Δ-top Matchings

Damit Matchings in Anfragegraphen in Polynomialzeit bestimmt werden können, werden diese rekursiv aus bereits bekannten Matchings berechnet. Das bedeutet, Matchings für einen Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ werden aus den Matchings für $\mathcal{G}^{(n-1)} = (G_1, \ldots, G_{n-1})$ konstruiert. Dafür wird eine Datenstruktur benötigt, die Matchings so repräsentiert, dass Berechnungen effizient durchgeführt werden können. Im Allgemeinen müssen alle Kanten eines Matchings gespeichert werden. Durch die hier vorgestellten Beschränkungen ist allerdings eine wesentlich kompaktere Darstellung von Matchings möglich. Die Grundlage dafür bildet die Tatsache, dass sich alle modellierten Matchings nur wenig vom Originalmatching unterscheiden. Da dieses wiederum immer aus der Datenbankinstanz abgelesen werden kann, muss es nicht explizit gespeichert werden. Für Δ-top Matchings ist es somit ausreichend, nur die Kanten zu betrachten, die verschieden vom Originalmatching sind.

Als Erstes wird dazu eine Notation für Matchingkanten eingeführt, die dafür sorgt, dass Kanten mithilfe von Labels identifiziert werden können. Darauf aufbauend wird eine Datenstruktur eingeführt, in der Matchings effizient gespeichert werden. Sei ein einzelner Anfragegraph G_i gegeben. Da jeder Tupelknoten ein eindeutiges Label besitzt, kann der zu einem Label t gehörige Knoten v_t mit t identifiziert werden. Da SA-Knoten mehrfach in G_i vorkommen können, muss zum Referenzieren dieser Knoten neben den Labels noch Tupel verwendet werden. Sei M_{orig} das Originalmatching, welches eine Kante $\{v_t, v_s\}$ enthalte,

wobei v_s ein SA-Knoten mit Label s sei. Da v_t durch t identifiziert wird und das Originalmatching fest gewählt ist, kann t zusammen mit s benutzt werden, um v_s zu identifizieren. Die Originalkante $\{v_t, v_s\}$ kann somit durch $(t, s(t))$ bezeichnet werden. Jede andere Kante $\{v_t, v_{s'}\}$, die keine Originalkante ist, kann als $(t, s'(t'))$ notiert werden, wobei t' das Tupel ist, dessen originaler SA-Wert s' ist.

Das folgende und alle weiteren Beispiele in diesem Abschnitt basieren auf den bereits in Kapitel 3 eingeführten Beispielanfragen und -graphen (siehe Abbildungen 3.2 und 3.3). Dementsprechend wird auch dieselbe Nummerierung der Matchings gewählt.



(a) G_1 (b) M_{orig} (c) M_3

Abbildung 4.17: Ein einzelner Anfragegraph mit zwei Matchings

Beispiel 4.13 In Abbildung 4.17 ist ein einzelner Anfragegraph G_1, das Originalmatching M_{orig} und ein weiteres Matching M_3 dargestellt. Mithilfe der Identifizierung von Knoten mittels Labels lassen sich beide Matchings wie folgt darstellen:

$$M_{\text{orig}} = \{(1, \text{A}(1)), (2, \text{A}(2)), (3, \text{B}(3)), (4, \text{C}(4))\},$$
$$M_3 = \{(1, \text{A}(1)), (2, \text{B}(3)), (3, \text{A}(2)), (4, \text{C}(4))\}.$$

Dabei ist zu beachten, dass es zwei SA-Knoten mit Label A gibt, die mit A(1) (oberer Knoten) beziehungsweise A(2) (unterer Knoten) identifiziert werden.

Die eingeführte Bezeichnung von Kanten kann auf Folgen von Anfragegraphen erweitert werden. Dabei ist es entscheidend, dass nur Matchings betrachtet werden, die Δ-pfadkonform bezüglich des Originalmatchings sind. Gibt es demnach eine Matchingkante $(t, s'(t'))$ in einem Teilgraphen und kommen die Knoten t und $s'(t')$ in einem anderen Teilgraphen vor, muss es auch dort eine Matchingkante $(t, s'(t'))$ geben. Die Notation $(t, s'(t'))$ kann also für alle Teilgraphen verwendet werden, in denen t und $s'(t')$ vorkommen.

Zusammenfassend wird Folgendes vereinbart:

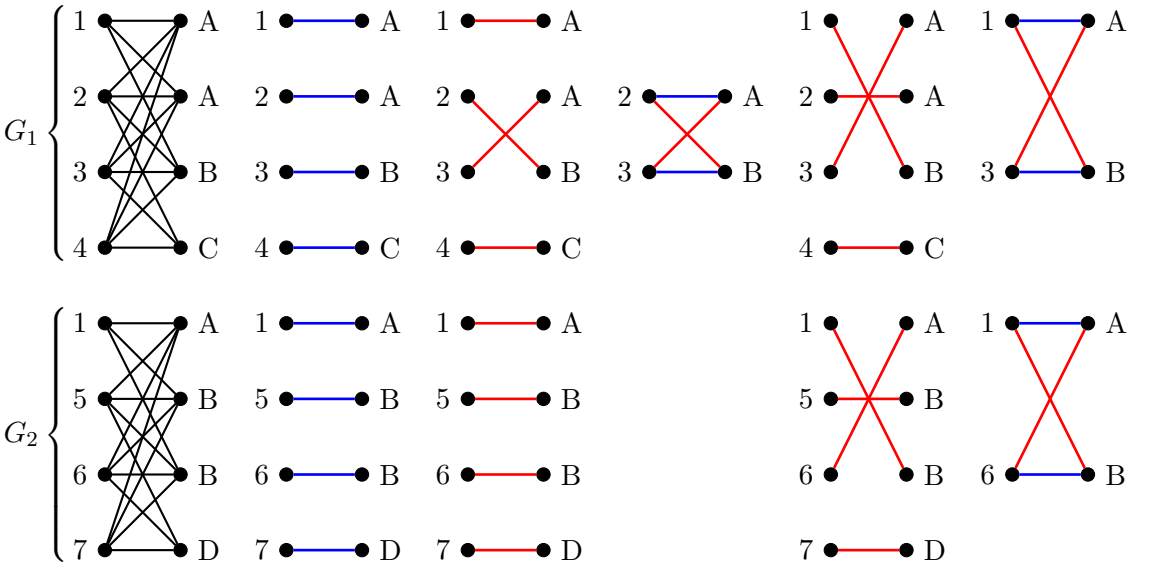
- $e = \{v_t, v_s\}$ bezeichnet wie bisher in einem (Teil-)Anfragegraphen G_i eine Kante zwischen dem Tupelknoten v_t und dem SA-Knoten v_s. Falls nichts anderes angegeben ist, ist v_t mit t und v_s mit s gelabelt.

- $e = (t, s'(t'))$ bezeichnet in allen (Teil-)Anfragegraphen $G_1, \ldots, G_n$ eine Kante zwischen dem Tupelknoten, der mit t gelabelt ist, und dem SA-Knoten, der mit s' gelabelt ist sowie im Originalmatching mit dem Tupelknoten matcht, der mit t' gelabelt ist. Damit wird in jedem G_i eindeutig eine Kante beschrieben.

- $e_l = (t, s)$ steht in allen (Teil-)Anfragegraphen $G_1, \ldots, G_n$ für eine beliebige Kante zwischen dem Tupelknoten, der mit t gelabelt ist, und einem SA-Knoten, der mit s gelabelt ist. Wenn der SA-Wert s mehrfach in einem G_i vorkommt, gibt es mehrere Kanten mit dieser Bezeichnung.

Als Vereinfachung können in Matchings alle Kanten mit gleichem Tupelknoten zusammengefasst werden, indem die Tupelreferenzen der SA-Knoten vereinigt werden. Die Kanten $(t, s(t_1))$ und $(t, s(t_2))$ werden mit $(t, s(t_1, t_2))$ bezeichnet und analog können bereits zusammengefasste Knotenmengen weiter vereinigt werden.[15] Durch diese Notation können Matchingkanten mithilfe der Labels unabhängig von den einzelnen Teilgraphen als sogenannte *Δ-top Matchingkanten* angegeben werden.

Definition 4.9 (Δ-top Matchingkante) *Eine* Δ-top Matchingkante $e_\Delta = (t, s, S_T)$ *ist ein Tripel, bestehend aus einem Tupel t, einem SA-Wert s und einer Menge von Tupeln S_T. s ist hierbei der originale SA-Wert aller Tupel in S_T. Für e_Δ wird die Schreibweise $e_\Delta = (t, s(S_T))$ verwendet.*[16]



(a) $\mathcal{G}$ (b) $\mathcal{M}_{\text{orig}}$ (c) $\mathcal{M}_{3.1}$ (d) $\mathcal{M}_{3.1} \Delta \mathcal{M}_{\text{orig}}$ (e) $\mathcal{M}_{7.3}$ (f) $\mathcal{M}_{7.3} \Delta \mathcal{M}_{\text{orig}}$

Abbildung 4.18: Anfragegraph $\mathcal{G} = (G_1, G_2)$ und die Matchings $\mathcal{M}_{3.1}$ sowie $\mathcal{M}_{7.3}$

Beispiel 4.14 In Abbildung 4.18 sind eine Folge von Anfragegraphen $\mathcal{G}$, das Originalmatching $\mathcal{M}_{\text{orig}}$ und zwei weitere Matchings $\mathcal{M}_{3.1}$ sowie $\mathcal{M}_{7.3}$ dargestellt. Mithilfe der

[15] Mathematisch gesehen ist (t_1, t_2) eine Menge von Tupeln $S_T = \{t_1, t_2\}$. Zur übersichtlicheren Darstellung wird hier auf die Mengenklammern verzichtet. Ebenso kann $s(t_1, t_2) = s(S_T)$ als Element (s, S_T) einer zweistelligen Relation angesehen werden, die für jedes Tupel t unterschiedlich definiert ist.

[16] An dieser Stelle sei vermerkt, dass bei der Verwendung der in Kapitel 3.10.1 vorgestellten Hypergraphen anstelle der Folgen von Anfragegraphen eine Δ-top Matchingkante genau einer Hyperkante im Graphen entspricht.

Identifizierung von Knoten mittels Labels lassen sich die Matchings als Menge von Δ-top Matchingkanten wie folgt darstellen:

$$\mathcal{M}_{\text{orig}} = \{(1, \text{A}(1)), (2, \text{A}(2)), (3, \text{B}(3)), (4, \text{C}(4)), (5, \text{B}(5)), (6, \text{B}(6)), (7, \text{D}(7))\},$$
$$\mathcal{M}_{3.1} = \{(1, \text{A}(1)), (2, \text{B}(3)), (3, \text{A}(2)), (4, \text{C}(4)), (5, \text{B}(5)), (6, \text{B}(6)), (7, \text{D}(7))\},$$
$$\mathcal{M}_{7.3} = \{(1, \text{B}(3, 6)), (2, \text{A}(2)), (3, \text{A}(1)), (4, \text{C}(4)), (5, \text{B}(5)), (6, \text{A}(1)), (7, \text{D}(7))\}.$$

Da das Originalmatching jeweils aus den tatsächlichen sensiblen Werten abgelesen werden kann, reicht es aus, für jedes Matching nur die Kanten zu speichern, in denen sich das Matching vom Originalmatching unterscheidet. Alle Tupel, die keine explizit erwähnte Matchingkante haben, matchen immer mit ihrem originalen SA-Wert. Für ein Matching $\mathcal{M}^{(n)}$ wird demzufolge nur $\mathcal{M}^{(n)} \setminus \mathcal{M}_{\text{orig}}$ gespeichert.

Beispiel 4.15 Um die beiden Matchings $\mathcal{M}_{3.1}$ und $\mathcal{M}_{7.3}$ aus Abbildung 4.18 zu speichern, werden nur die jeweiligen Differenzen zum Originalmatching berechnet.

$$\mathcal{M}_{3.1} \setminus \mathcal{M}_{\text{orig}} = \{(2, \text{B}(3)), (3, \text{A}(2))\}$$
$$\mathcal{M}_{7.3} \setminus \mathcal{M}_{\text{orig}} = \{(1, \text{B}(3, 6)), (3, \text{A}(1)), (6, \text{A}(1))\}$$

Das entspricht jeweils nur den rot markierten Kanten in der symmetrischen Differenz mit dem Originalmatching (siehe (d) und (f)). Ein komplettes Matching erhält man, indem für jeden nicht erwähnten Tupelknoten (z. B. $1, 4, 5, 6, 7$ bei $\mathcal{M}_{3.1}$) standardmäßig immer die Originalkante (blaue Kante) gewählt wird.

Jede Kante kann in mehreren Matchings vorkommen. Beispielsweise gibt es zum Anfragegraphen aus Abbildung 4.18 ein Matching $\mathcal{M}_{7.1}$, das aus den Kanten $(1, \text{B}(3, 5))$, $(3, \text{A}(1))$ und $(5, \text{A}(1))$ besteht.[17] Die Kanten $(1, \text{B}(3))$ und $(3, \text{A}(1))$ kommen dabei auch im Matching $\mathcal{M}_{7.3}$ vor, denn $\mathcal{M}_{7.1}$ und $\mathcal{M}_{7.3}$ sind in G_1 identisch. Wenn es mehrere Matchings gibt, die in vielen Teilgraphen identisch sind und sich nur in einem einzigen unterscheiden, werden viele Kanten mehrfach gespeichert (jeweils einmal pro Matching). Für eine Aussage über die Verletzung von k-assign Anonymität ist aber nicht entscheidend, wie viele Matchings mit einer Kante $e_l = (t, s)$ existieren, sondern nur, ob mindestens eins existiert. Daher wird an dieser Stelle erneut eine Vereinfachung eingeführt, die in Kapitel 5 die Grundlage für die polynomielle Laufzeit der Approximation ist. Es werden nicht alle Matchings einzeln gespeichert, sondern nur alle Kanten, die in mindestens einem Matching vorkommen. Dazu werden die Kanten aller modellierter Δ-top Matchings in einer Datenstruktur vereinigt, die *Δ-top Matchingtabelle* $\mathcal{T}_\Delta^{(n)}$ genannt wird. Sie enthält für eine Folge von Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ die Kanten von Δ-top Matchings $\mathcal{M}^{(n)}$, wobei nur die Kanten aufgelistet werden, in denen sich die jeweiligen Matchings vom Originalmatching unterscheiden. Für $\mathcal{M}^{(n)}$ wird demzufolge nur $\mathcal{M}^{(n)} \setminus \mathcal{M}_{\text{orig}}$ gespeichert. Für jedes Tupel t enthält $\mathcal{T}_\Delta^{(n)}$ genau eine Zeile, in der der originale SA-Wert von t und alle Δ-top Matchingkanten stehen, die t enthalten. Jede Kante $(t, s'(t'))$ wird in der Zeile für t als Eintrag $s'(t')$ angegeben. Zusammengesetzte Kanten $(t, s'(S_T))$, wobei S_T eine Menge von Tupeln ist, werden als $s'(S_T)$ dargestellt.

[17] Alle Δ-top Matchings in $\mathcal{G}^{(2)}$ sind im Anhang in Abbildung A.7 auf Seite 276 dargestellt (vgl. Beispiel A.6).

ID	Name	SA	$\mathcal{M}_{3.1}$	$\mathcal{M}_{7.1}$	$\mathcal{M}_{7.3}$
1	Alison	A		B(3, 5)	B(3, 6)
2	Ben	A	B(3)		
3	Clark	B	A(2)	A(1)	A(1)
4	Debra	C			
5	Elaine	B		A(1)	
6	Fiona	B			A(1)
7	Gary	D			

(a) Datentabelle und drei Matchings

ID	Name	SA	$\mathcal{G}^{(2)}$
1	Alison	A	B(3, 5), B(3, 6)
2	Ben	A	B(3)
3	Clark	B	A(1), A(2)
4	Debra	C	
5	Elaine	B	A(1)
6	Fiona	B	A(1)
7	Gary	D	

(b) Δ-top Matchingtabelle $\mathcal{T}_{\Delta}^{(2)}$

Tabelle 4.3: Δ-top Matchingtabelle und enthaltene Matchings

Beispiel 4.16 Tabelle 4.3 zeigt in (a) die Repräsentation der drei Δ-top Matchings $\mathcal{M}_{3.1}$, $\mathcal{M}_{7.1}$ und $\mathcal{M}_{7.3}$ (siehe Beispiel 4.15). Die entsprechende Δ-top Matchingtabelle $\mathcal{T}_{\Delta}^{(2)}$ ist in (b) dargestellt und enthält als Tupelidentifikator sowohl die ID als auch das Name-Attribut. Die Matchingkanten der drei Matchings sind in Spalte $\mathcal{G}^{(2)}$ aufgelistet. Dabei ist zu beachten, dass in $\mathcal{G}^{(2)}$ weitere Δ-top Matchings existieren, die in diesem Beispiel nicht angegeben sind.

Aus Δ-top Matchingtabellen sind sehr leicht die SA-Werte ablesbar, die einem Tupel durch Matchings beziehungsweise SA-Werteverteilungen zugeordnet werden können. Dabei ist zu beachten, dass nicht jedes Matching und demzufolge auch nicht jede Zuordnung modelliert wird. Demzufolge ist die Menge aller aufgelisteter SA-Werte nur eine Teilmenge aller möglicher Werte. Um verschiedene Approximationen durch Δ-top Matchingtabellen zu bewerten, werden die bereits eingeführten Begriffe Signatur, Anonymitätsgrad (vgl. Definition 3.6) und k-assign Anonymität (vgl. Definition 3.7) auch für Δ-top Matchingtabellen verwendet.

Definition 4.10 (Signatur und Anonymitätsgrad bez. Δ-top Matchingtabelle) *Sei $\mathcal{T}_{\Delta}^{(n)}$ eine Δ-top Matchingtabelle, dann ist die* Signatur *$Sig(t)$ eines Tupels t bezüglich $\mathcal{T}_{\Delta}^{(n)}$ die Menge von SA-Werten aus Δ-top Matchingkanten von t in $\mathcal{T}_{\Delta}^{(n)}$. Der* Anonymitätsgrad *$a(t)$ von t bezüglich $\mathcal{T}_{\Delta}^{(n)}$ ist die Größe der Signatur: $a(t) = |Sig(t)|$. Der* minimale Anonymitätsgrad *$a_{\min}(\mathcal{T}_{\Delta}^{(n)})$ von Tupeln bezüglich $\mathcal{T}_{\Delta}^{(n)}$ ist der kleinste vorkommende Anonymitätsgrad: $a_{\min}(\mathcal{T}_{\Delta}^{(n)}) = \min_t a(t)$.*

Definition 4.11 (k-assign Anonymität bez. Δ-top Matchingtabelle) *Sei $\mathcal{T}_{\Delta}^{(n)}$ eine Δ-top Matchingtabelle. Dann erfüllt $\mathcal{T}_{\Delta}^{(n)}$* k-assign Anonymität*, wenn der Anonymitätsgrad jedes Tupels $t \in \mathcal{T}_{\Delta}^{(n)}$ bezüglich $\mathcal{T}_{\Delta}^{(n)}$ mindestens k ist.*

Bei der Berechnung der Signatur und des Anonymitätsgrades müssen die Kanten des Originalmatchings, die nur implizit in einer Δ-top Matchingtabelle gespeichert sind, ebenfalls beachtet werden.

Beispiel 4.17 Die Signatur der Tupel 1, 2, 3, 5 und 6 aus Tabelle 4.3b ist jeweils $\{\mathrm{A}, \mathrm{B}\}$. Diese Tupel haben dadurch den Anonymitätsgrad 2. Tupel 4 hat die Signatur $\{\mathrm{C}\}$ und Tupel 7 $\{\mathrm{D}\}$. Der Anonymitätsgrad beider Tupel ist somit 1. Offensichtlich sollte für beide

Tupel je ein Matching gespeichert werden, in dem sie mit einem anderen SA-Wert matchen als ihrem originalen. Dann wäre ihr Anonymitätsgrad ebenfalls 2. In der abgebildeten Form erfüllt Tabelle 4.3b nur 1-assign Anonymität.

Eine Δ-top Matchingtabelle kann sehr einfach erzeugt werden, indem Kanten aller zu speichernder Matchings zunächst vereinigt und dann in die entsprechenden Zeilen (d. h. nach den enthaltenen Tupeln getrennt) einsortiert werden. Dahingegen ist das Extrahieren eines Matchings aus der Tabelle nicht immer offensichtlich. Allerdings können die Eigenschaften von Δ-top Matchings genutzt werden, um Kanten aus der Tabelle den Matchings zuzuordnen. Ist beispielsweise aus Tabelle 4.3b ein Matching $\mathcal{M}$ gesucht, das die Labelkante $(3, \mathrm{A})$ enthält, muss in der Zeile für Tupel 3 nach einer Matchingkante gesucht werden, die das Label A enthält. Mit $(3, \mathrm{A}(1))$ und $(3, \mathrm{A}(2))$ gibt es dafür zwei verschiedene Möglichkeiten. Wird $(3, \mathrm{A}(2))$ gewählt, muss $\mathcal{M}$ auch die Gegenkante $(2, \mathrm{B}(3))$ enthalten, denn $\mathcal{M}$ ist ein $\Delta 2$-Matching bezüglich des Originalmatchings. Es entsteht das Matching $\mathcal{M} = \mathcal{M}_{3.1}$ aus Tabelle 4.3a. Wird hingegen $(3, \mathrm{A}(1))$ gewählt, gibt es für die Gegenkante $(1, \mathrm{B}(3))$ zwei mögliche Δ-top Matchingkanten, nämlich $(1, \mathrm{B}(3, 5))$ oder $(1, \mathrm{B}(3, 6))$. In beiden Fällen muss rekursiv weiter nach Gegenkanten gesucht werden, das heißt, $(5, \mathrm{A}(1))$ oder $(6, \mathrm{A}(1))$ gehört ebenfalls zum gesuchten Matching $\mathcal{M}$. In diesen Fällen werden die Matchings $\mathcal{M}_{7.1}$ beziehungsweise $\mathcal{M}_{7.3}$ extrahiert.

Das beschriebene Verfahren kann aber auch zu Sackgassen führen, wenn bestimmte Kanten nicht zum Matching hinzugefügt werden können, ohne die Matchingeigenschaft zu verletzen. Das kann zum Beispiel passieren, wenn nach einer Gegenkante $(t, s'(t'))$ gesucht wird, im bisher konstruierten Matching aber bereits eine andere Kante vorhanden ist, die den SA-Knoten $s'(t')$ enthält. Folglich müsste ein Backtracking durchgeführt und andere Kanten extrahiert werden. In den folgenden Kapiteln wird dieser Fall noch genauer untersucht und gezeigt, dass unter bestimmten Voraussetzungen die Extraktion von Matchings aus einer Δ-top Matchingtabelle in Polynomialzeit ohne Backtracking möglich ist. Dazu ist es aber zwingend erforderlich, dass nur Δr-Matchings mit $r = 2$ gespeichert werden. Für $r > 2$ sind die vorgestellten Algorithmen nicht anwendbar. Das ist auch der Hauptgrund, warum das Angreiferwissen nur mithilfe von $\Delta 2$-Matchings approximiert wird.

Zusammenfassung

Ein Ziel der vorliegenden Arbeit lautet, einen Approximationsalgorithmus zu entwickeln, der den Schutz der Privatsphäre bei der Anfragebearbeitung sicherstellt, nur polynomielle Laufzeit hat und damit praktischen Anforderungen genügt. Eine wichtige Anforderung dabei ist, dass jede Verletzung des Schutzkriteriums weiterhin gefunden wird. Die Approximation darf sich somit nur dann irren, wenn sie eine Verletzung ausgibt, tatsächlich aber keine vorhanden ist.

In Kapitel 3 wurde gezeigt, wie valide Matchings in Anfragegraphen verwendet werden können, um k-assign Anonymität zu erfüllen und damit die Privatsphäre zu schützen. Da allerdings das Bestimmen eines Matchings exponentielle Laufzeit benötigt, wurden in diesem Kapitel Verfeinerungen und Einschränkungen der untersuchten Matchings erstellt. Es wurde festgelegt, dass für die Approximation nur Δ-top Matchings betrachtet werden. Das Besondere an ihnen ist, dass sie sich nur wenig vom Originalmatching unterscheiden, welches wiederum der realen Zuordnung von sensiblen Werten zu Individuen entspricht. Als

weitere Grundlage für einen Approximationsalgorithmus wurde mit der Δ-top Matchingtabelle eine Datenstruktur vorgestellt, mit der Kanten von bereits berechneten Matchings effizient gespeichert werden können. Damit ist es beispielsweise möglich, Matchings für n Anfragegraphen sukzessive aus denen für $n-1$ Anfragegraphen zu konstruieren. Hierdurch sind die Voraussetzungen geschaffen worden, um in den Kapiteln 5 und 7 Algorithmen anzugeben, die Δ-top Matchings in Polynomialzeit berechnen.

5 Approximation des Angreiferwissens für beliebige Anfragen

Basierend auf den Kapiteln 3 und 4 wird in diesem Kapitel ein Approximationsalgorithmus vorgestellt, der valide Matchings in Anfragegraphen in Polynomialzeit findet. Der Grundgedanke dabei lautet, Matchings sukzessive aus anderen Matchings zu konstruieren. Anhand eines Beispiels wird zunächst erläutert, welche Matchings für einen einzelnen Graphen berechnet werden (Kapitel 5.1) und wie diese zu anderen Matchings erweitert werden können, wenn mehrere Graphen gegeben sind (Kapitel 5.2). Der Approximationsalgorithmus für den allgemeinen Fall wird danach detailliert eingeführt (Kapitel 5.3), wobei besonders auf zusätzliche Einschränkungen bei der Berechnung eingegangen wird (Kapitel 5.4). Mit seiner Hilfe kann ein Verfahren angegeben werden, das den Schutz der Privatsphäre bei der Anfragebearbeitung sicherstellt (Kapitel 5.5). Am Ende dieses Kapitels wird eine Möglichkeit vorgestellt, mit der Antworten auf Anfragen auch bei einer Verletzung des Schutzkriteriums ausgegeben werden können (Kapitel 5.6).

5.1 Der erste Graph

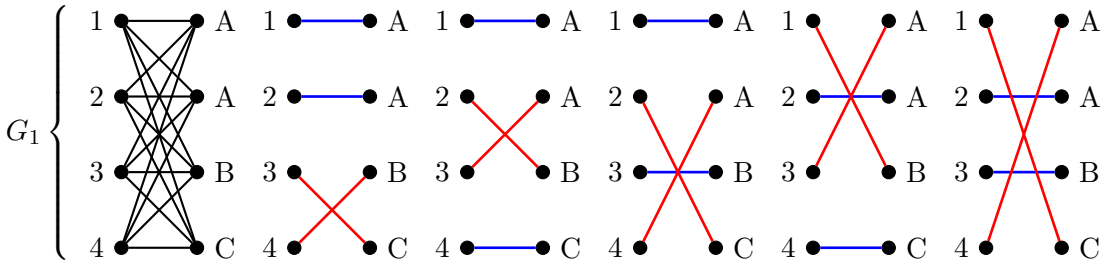
Als Erstes wird ein Spezialfall untersucht, bei dem die Folge von Anfragegraphen aus genau einem Teilgraphen besteht. Danach wird gezeigt, wie Matchings in Anfragegraphen berechnet werden, die aus mehreren Teilgraphen bestehen. Die hier verwendete Referenzierung von Knoten anhand der Labels wurde in Kapitel 4.5 vorgestellt. t steht demzufolge für einen Tupelknoten in einem beliebigen Graphen, der mit t gelabelt ist, während $s(t)$ einen SA-Knoten darstellt, der mit s gelabelt ist und im Originalmatching mit t matcht.

Sei das bereits in Kapitel 3 vorgestellte Beispiel mit den Anfragegraphen G_1, G_2 und G_3 gegeben. Abbildung 5.1 zeigt in (a) den Anfragegraphen G_1, der für das Ergebnis der ersten Anfrage steht.[1] Darin kommen Tupel für Alison (ID 1), Ben (2), Clark (3) und Debra (4) zusammen mit den SA-Werten A, A, B und C vor. In (b) bis (f) sind alle Δ-top Matchings in $\mathcal{G}^{(1)} = (G_1)$ dargestellt.[2] Die Bezeichnung orientiert sich dabei an der Nummerierung der SA-Werteverteilungen, wie sie zum Beispiel bereits in Tabelle 3.5 auf Seite 70 verwendet wurde. Von den ursprünglich zwölf verschiedenen Werteverteilungen beziehungsweise Matchings in $\mathcal{G}^{(1)}$ erfüllen nur fünf die Δ-top Eigenschaft. Diese werden gespeichert und sind in der Δ-top Matchingtabelle $\mathcal{T}_{\Delta}^{(1)}$ in (g) aufgelistet. Es fällt auf, dass jeder sensible Wert jedem Tupel zugeordnet werden kann. Das bedeutet, dass trotz der Beschränkung auf Δ-top Matchings alle möglichen Zuordnungen von sensiblen Werten zu Tupeln auch modelliert werden.[3]

[1] Vgl. Tabelle 3.4 auf Seite 69.

[2] Alle SA-Werteverteilungen und dazugehörige Matchings in $\mathcal{G}^{(1)}$ sind im Anhang in Abbildung A.3 auf Seite 271 dargestellt. Dabei wurden insbesondere auch die berücksichtigt, die nicht Δ-top sind.

[3] Vgl. Beispiel 4.1 auf Seite 100.



(a) $\mathcal{G}^{(1)}$ (b) $\mathcal{M}_2^{(1)}$ (c) $\mathcal{M}_3^{(1)}$ (d) $\mathcal{M}_6^{(1)}$ (e) $\mathcal{M}_7^{(1)}$ (f) $\mathcal{M}_{11}^{(1)}$

ID	Name	SA	$\mathcal{G}^{(1)}$
1	Alison	A	B(3), C(4)
2	Ben	A	B(3), C(4)
3	Clark	B	A(1), A(2), C(4)
4	Debra	C	A(1), A(2), B(3)

(g) Δ-top Matchingtabelle $\mathcal{T}_\Delta^{(1)}$ für $\mathcal{G}^{(1)}$

Abbildung 5.1: Für den ersten Graphen $\mathcal{G}^{(1)}$ werden alle Δ-top Matchings gespeichert, welche in (b) bis (f) angegeben sind.

Im Allgemeinen verläuft die Behandlung des ersten Graphen analog zum oben vorgestellten Beispiel. Für einen gegebenen Anfragegraphen werden alle Δ-top Matchings erzeugt, indem aus allen vorkommenden Tupeln jeweils zwei ausgewählt werden, die unterschiedliche originale SA-Werte haben und deren Matchingpartner im Vergleich zum Originalmatching vertauscht werden. Das bedeutet im obigen Fall, dass kein Matching mit den Kanten $(1, \mathrm{A}(2))$ oder $(2, \mathrm{A}(1))$ gespeichert wird, denn A ist jeweils der originale SA-Wert von Alison (1) und Ben (2). Besteht der Graph aus n Tupeln, entstehen höchstens $\binom{n}{2} = \mathcal{O}(n^2)$ Matchings, deren Erstellung offensichtlich in Polynomialzeit möglich ist. Interessanterweise gehört jede Δ-top Matchingkante nach der Bearbeitung des ersten Graphen eindeutig zu genau einem Δ-top Matching. Diese Eigenschaft muss ab dem zweiten Graphen aber nicht mehr gelten.

5.2 Mehrere Graphen

Sei eine Folge von Anfragegraphen $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$ gegeben und sei nach validen Matchings in $\mathcal{G}^{(n+1)}$ gefragt. Der Ansatz dazu lautet, Matchings rekursiv aus bereits bekannten Matchings zu berechnen. Das bedeutet konkret, dass zunächst Matchings in G_1 berechnet werden, aus denen dann Matchings in (G_1, G_2) erzeugt werden. Daraus werden wiederum Matchings in (G_1, G_2, G_3) und so weiter berechnet. Allgemein werden somit Matchings für $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$ aus denen für $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ konstruiert.

Im Folgenden sind daher immer Matchings $\mathcal{M}^{(n)}$ für eine Folge von Anfragegraphen $\mathcal{G}^{(n)}$ und ein neuer Anfragegraph G_{n+1} gegeben und es sind Matchings in $\mathcal{G}^{(n+1)}$ gesucht. Im Wesentlichen wird dabei für jedes Matching $\mathcal{M}^{(n)}$ getestet, ob es ein Matching $\mathcal{M}^{(n+1)}$ in $\mathcal{G}^{(n+1)}$ gibt, das in den bisher behandelten Teilgraphen $G_1, \ldots, G_n$ mit $\mathcal{M}^{(n)}$ übereinstimmt. Genauer gesagt werden dabei nur die Tupel beachtet, die in $\mathcal{M}^{(n)}$ in einer anderen

Matchingkante enthalten sind als im Originalmatching. Dieser Schritt heißt *Erweiterung* eines Matchings und bildet das Kernstück des hier vorgestellten Approximationsalgorithmus. Falls in $\mathcal{G}^{(n+1)}$ Tupel vorkommen, die bisher nicht in $\mathcal{G}^{(n)}$ vorkamen, können auch sogenannte *neue Matchings* entstehen. Das sind Matchings, die in $G_1, \ldots, G_n$ identisch mit dem Originalmatching sind und dadurch auch als Erweiterung des Originalmatchings gelten. Wichtig dabei ist, dass alle erweiterten Matchings die Δ-top Eigenschaften weiterhin erfüllen.

Ein Matching $\mathcal{M}^{(n+1)}$ ist eine Erweiterung eines Matchings $\mathcal{M}^{(n)}$, wenn beide folgenden Eigenschaften erfüllt sind. Erstens müssen alle Tupel, die in $\mathcal{M}^{(n)}$ in einer anderen Kante als im Originalmatching enthalten sind, genau in dieser Kante auch in $\mathcal{M}^{(n+1)}$ enthalten sein. Dazu wird die symmetrische Differenz zwischen den Matchings und dem Originalmatching betrachtet und gefordert, dass $\mathcal{M}^{(n)} \,\Delta\, \mathcal{M}^{(n)}_{\text{orig}} \subset \mathcal{M}^{(n+1)} \,\Delta\, \mathcal{M}^{(n+1)}_{\text{orig}}$ gilt. Tupel, die in $\mathcal{M}^{(n)}$ denselben Matchingpartner haben wie im Originalmatching, können in einem erweiterten Matching einen anderen Matchingpartner haben. Zweitens soll sich $\mathcal{M}^{(n+1)}$ in gewisser Weise von $\mathcal{M}^{(n)}$ unterscheiden. Das bedeutet, dass $\mathcal{M}^{(n+1)}$ eine Kante $(t, s'(t'))$ enthalten soll, die nicht in $\mathcal{M}^{(n)}$ vorkommt. Somit gibt es mindestens ein Tupel, das im erweiterten Matching mit einem anderen SA-Wert matcht als in $\mathcal{M}^{(n)}$.

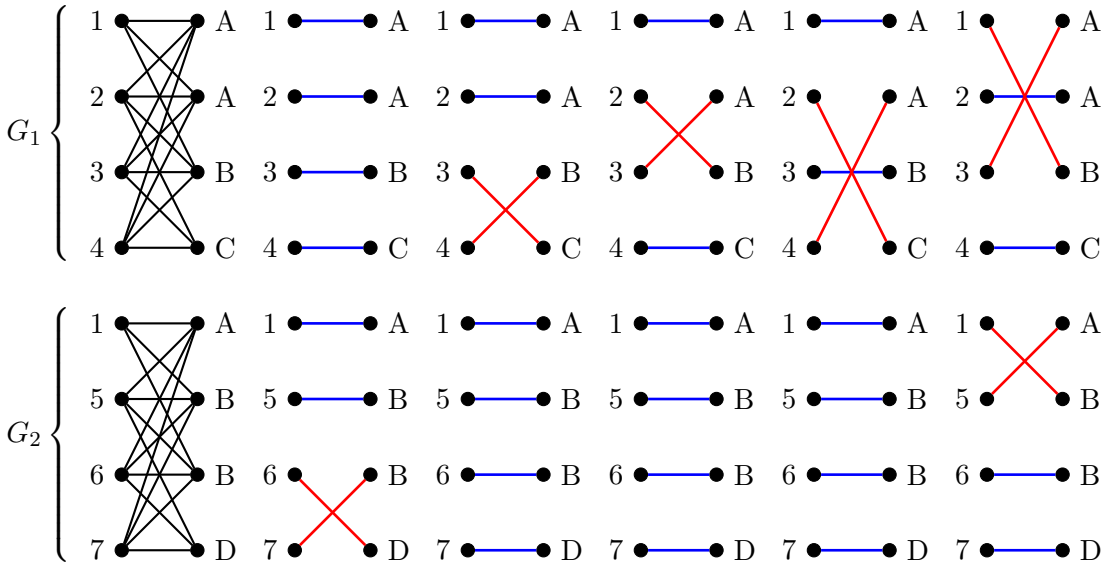
Definition 5.1 (Erweiterung eines Matchings) *Seien $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen, $\mathcal{M}^{(n)}_{orig}$ das Originalmatching und $\mathcal{M}^{(n)}$ ein Δ-top Matching in $\mathcal{G}^{(n)}$. Weiterhin sei G_{n+1} ein Anfragegraph und $\mathcal{M}^{(n+1)}$ ein Δ-top Matching in $\mathcal{G}^{(n+1)}$. Dann heißt $\mathcal{M}^{(n+1)}$ genau dann* Erweiterung von *$\mathcal{M}^{(n)}$, wenn $\mathcal{M}^{(n)} \Delta \mathcal{M}^{(n)}_{orig} \subset \mathcal{M}^{(n+1)} \Delta \mathcal{M}^{(n+1)}_{orig}$ gilt und $\mathcal{M}^{(n+1)}$ in G_{n+1} mindestens eine Kante $(t, s'(t'))$ enthält, die nicht in $\mathcal{M}^{(n)}$ vorkommt. Man sagt in diesem Fall auch, dass $\mathcal{M}^{(n)}$ zu $\mathcal{M}^{(n+1)}$* erweitert *werden kann.*

Entsprechend Definition 5.1 heißt der Berechnungsschritt, um $\mathcal{M}^{(n+1)}$ aus $\mathcal{M}^{(n)}$ zu konstruieren, *Erweiterungsschritt*. Ein Tupel t, für das in $\mathcal{M}^{(n+1)}[\mathcal{G}^{(n+1)}]$ eine Matchingkante $(t, s'(t'))$ existiert, die nicht in $\mathcal{M}^{(n)}$ vorkommt, heißt *mit t'* beziehungsweise *mit $s'(t')$*[4] *erweitert*. Analog heißt eine Δ-top Matchingkante $e_\Delta = (t, s'(S_T))$ in $\mathcal{T}^{(n)}_\Delta$ *mit t'* beziehungsweise *$s'(t')$ erweitert*, wenn es eine Kante $(t, s'(S_T \cup \{t'\}))$ in $\mathcal{T}^{(n+1)}_\Delta$ gibt. Für Δ2-Matchings gilt in diesem Fall immer, dass auch t' mit t erweitert ist. Wenn zu einem gegebenen Matching $\mathcal{M}^{(n)}$ und Graphen G_{n+1} kein Matching $\mathcal{M}^{(n+1)}$ mit $\mathcal{M}^{(n)} \,\Delta\, \mathcal{M}^{(n)}_{\text{orig}} \subset \mathcal{M}^{(n+1)} \,\Delta\, \mathcal{M}^{(n+1)}_{\text{orig}}$ existiert, wird auch davon gesprochen, dass $\mathcal{M}^{(n)}$ in $\mathcal{G}^{(n+1)}$ *gelöscht* wurde. Matchings und Δ-top Matchingkanten, die weder erweitert noch gelöscht werden, bleiben in $\mathcal{G}^{(n+1)}$ beziehungsweise $\mathcal{T}^{(n+1)}_\Delta$ *erhalten*.

Bevor der Algorithmus für die Erweiterung von Matchings vorgestellt wird, werden wichtige Schritte und Begriffe anhand des bereits eingeführten Beispiels erläutert. Dazu sollen zunächst die fünf Δ-top Matchings im Anfragegraphen aus Abbildung 5.1 erweitert werden. Sei G_2 ein weiterer Anfragegraph, der Knoten für Alison (ID 1), Elaine (5), Fiona (6) und Gary (7) sowie für die SA-Werte A, B, B und D enthält. Abbildung 5.2 zeigt in (a) die Folge von Anfragegraphen $\mathcal{G}^{(2)} = (G_1, G_2)$ und in (g) die Δ-top Matchingtabelle $\mathcal{T}^{(2)}_\Delta$ (siehe Spalte $\mathcal{G}^{(2)}$). Darin sind insgesamt sieben Δ-top Matchings gespeichert, von denen in (b) bis (f) aus Platzgründen nur fünf abgebildet sind.[5] Zur besseren Übersicht

[4] Als Kurzform für ein beliebiges t' wird auch der Ausdruck „*mit s' erweitert*“ verwendet.

[5] Im Anhang sind in Abbildung A.7 auf Seite 276 alle sieben Matchings dargestellt.



(a) $\mathcal{G}^{(2)}$ (b) $\mathcal{M}^{(2)}_{1.2}$ (c) $\mathcal{M}^{(2)}_{2.1}$ (d) $\mathcal{M}^{(2)}_{3.1}$ (e) $\mathcal{M}^{(2)}_{6.1}$ (f) $\mathcal{M}^{(2)}_{7.1}$

ID	Name	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$
1	Alison	A	B(3), C(4)	B(3, 5), B(3, 6)*
2	Ben	A	B(3), C(4)	B(3), C(4)
3	Clark	B	A(1), A(2), C(4)	A(1), A(2), C(4)
4	Debra	C	A(1), A(2), B(3)	A(2), B(3)
5	Elaine	B		A(1), D(7)*
6	Fiona	B		A(1)*, D(7)
7	Gary	D		B(5)*, B(6)

(g) Δ-top Matchingtabelle $\mathcal{T}^{(2)}_{\Delta}$ für $\mathcal{G}^{(2)}$

Abbildung 5.2: Für $\mathcal{G}^{(2)}$ werden sieben Δ-top Matchings gespeichert, von denen in (b) bis (f) fünf dargestellt sind. Die Δ-top Matchingtabelle $\mathcal{T}^{(2)}_{\Delta}$ in (g) enthält alle sieben (Kanten fehlender Matchings sind mit „*“ gekennzeichnet), die auch im Anhang in Abbildung A.7 auf Seite 276 angegeben sind.

beinhaltet $\mathcal{T}^{(2)}_{\Delta}$ auch die Spalte $\mathcal{G}^{(1)}$ aus Abbildung 5.1, an der sich die Veränderungen durch den zweiten Graphen ablesen lassen.

Mit Alison (1) gibt es ein Tupel, welches sowohl in G_1 als auch in G_2 vorkommt. Daher muss in allen validen Matchings in $\mathcal{G}^{(2)}$ der entsprechende Tupelknoten für 1 in beiden Graphen mit jeweils demselben SA-Label matchen. Beispielsweise enthalten $\mathcal{M}^{(2)}_{2.1}$, $\mathcal{M}^{(2)}_{3.1}$ und $\mathcal{M}^{(2)}_{6.1}$ jeweils die Kante $(1, \mathrm{A})$. Diese drei Matchings sind aus den in $\mathcal{T}^{(1)}_{\Delta}$ gespeicherten Matchings $\mathcal{M}^{(1)}_{2}$, $\mathcal{M}^{(1)}_{3}$ beziehungsweise $\mathcal{M}^{(1)}_{6}$ konstruiert worden (vgl. Abbildung 5.1) und stimmen in G_1 mit ihrem Ursprungsmatching überein. Da sie aber jeweils in G_2 identisch zum Originalmatching sind, gilt $\mathcal{M}^{(1)} \,\Delta\, \mathcal{M}^{(1)}_{\text{orig}} = \mathcal{M}^{(2)} \,\Delta\, \mathcal{M}^{(2)}_{\text{orig}}$. Nach Definition 5.1 gelten demzufolge $\mathcal{M}^{(2)}_{2.1}$, $\mathcal{M}^{(2)}_{3.1}$ und $\mathcal{M}^{(2)}_{6.1}$ nicht als Erweiterungen und man sagt, dass $\mathcal{M}^{(1)}_{2}$, $\mathcal{M}^{(1)}_{3}$ und $\mathcal{M}^{(1)}_{6}$ in $\mathcal{G}^{(2)}$ „erhalten geblieben“ sind. Andererseits würden Matchings, die in

G_2 verschieden vom Originalmatching wären, zwar durchaus Erweiterungen für die drei genannten Matchings darstellen, würden aber gegen die Minimalitätseigenschaft von Δ-top Matchings verstoßen. Folglich gibt es keine Erweiterungen von $\mathcal{M}_2^{(1)}$, $\mathcal{M}_3^{(1)}$ und $\mathcal{M}_6^{(1)}$, die im Rahmen dieser Approximation berechnet werden.

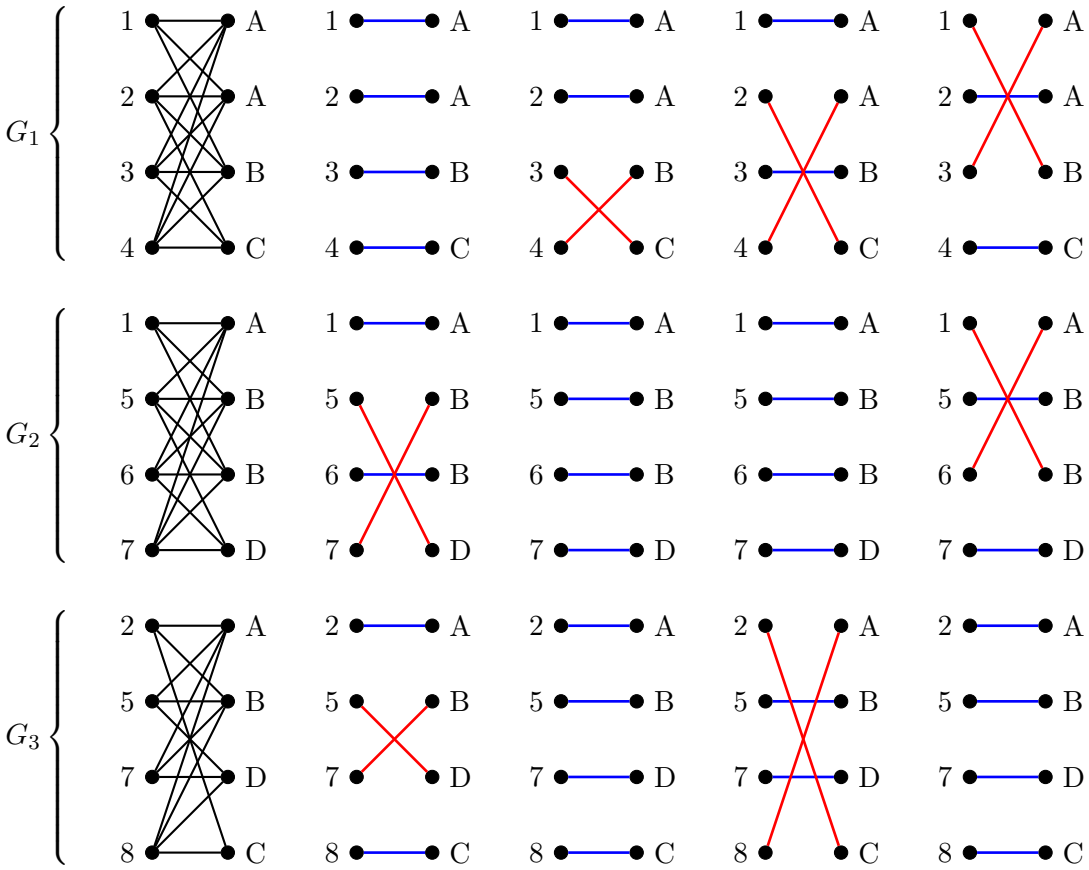
Für $\mathcal{M}_7^{(1)}$, das die Kante $(1, \mathrm{B}(3))$ enthält, gibt es in $\mathcal{G}^{(2)}$ zwei Erweiterungen, denn mit B(5) und B(6) kommen zwei SA-Knoten vor, die mit B gelabelt sind. $\mathcal{M}_{7.1}^{(2)}$ ist eine dieser Erweiterungen und in (f) dargestellt, das andere erweiterte Matching $\mathcal{M}_{7.3}^{(2)}$ befindet sich im Anhang in Abbildung A.7 auf Seite 276. Die hier verwendete Bezeichnung orientiert sich an der Nummerierung der entsprechenden SA-Werteverteilungen, wie sie in Tabelle 3.8 auf Seite 72 verwendet wurde. In $\mathcal{M}_{7.1}^{(2)}$ ist Tupel 1 beziehungsweise die Δ-top Matchingkante $(1, \mathrm{B}(3))$ mit B(5) erweitert worden, in $\mathcal{M}_{7.3}^{(2)}$ analog mit B(6). In $\mathcal{T}_\Delta^{(2)}$ entstehen auf diese Weise die Kanten $(1, \mathrm{B}(3,5))$ und $(1, \mathrm{B}(3,6))$.

$\mathcal{M}_{11}^{(1)}$ enthält die Kante $(1, \mathrm{C}(4))$. Da es aber in G_2 kein SA-Label C gibt, kann es auch kein valides Matching in $\mathcal{G}^{(2)}$ geben, in dem 1 mit C matcht. Somit muss $\mathcal{M}_{11}^{(1)}$ gelöscht werden.

Das letzte hier dargestellte Matching $\mathcal{M}_{1.2}^{(2)}$ entspricht in G_1 dem Originalmatching und wird daher auch als Erweiterung des Originalmatchings beziehungsweise als neues Matching bezeichnet. Auch hier gibt es noch eine andere Erweiterung, die die Nummer 1.3 trägt, die Kanten $(5, \mathrm{D}(7))$ sowie $(7, \mathrm{B}(5))$ enthält und nur im Anhang dargestellt ist.

Sei als letzter Anfragegraph der bereits eingeführte Graph G_3 gegeben, der die Tupel 2, 5, 7 und 8 sowie die SA-Werte A, B, C und D enthalte. Abbildung 5.3 zeigt in (b) bis (e) alle vier Δ-top Matchings in $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$. Matching $\mathcal{M}_{6.1}^{(3)}$ ist dabei eine Erweiterung des Matchings $\mathcal{M}_{6.1}^{(2)}$ aus Abbildung 5.2. Es hat dieselbe Nummer, da es keine andere Erweiterung von $\mathcal{M}_{6.1}^{(2)}$ gibt. Die Matchings $\mathcal{M}_{2.1}^{(3)}$ und $\mathcal{M}_{7.3}^{(3)}$ entstehen direkt aus den Matchings $\mathcal{M}_{2.1}^{(2)}$ und $\mathcal{M}_{7.3}^{(2)}$. Da sie jeweils in G_3 identisch zum Originalmatching sind, gelten sie nicht als Erweiterungen. Ebenfalls entsteht $\mathcal{M}_{1.3}^{(3)}$ zwar aus $\mathcal{M}_{1.3}^{(2)}$, gilt aber auch nicht als Erweiterung. Die Bedingung $\mathcal{M}^{(n)} \,\Delta\, \mathcal{M}_{\text{orig}}^{(n)} \subset \mathcal{M}^{(n+1)} \,\Delta\, \mathcal{M}_{\text{orig}}^{(n+1)}$ ist erfüllt, die Matchingkanten $(5, \mathrm{D}(7))$ und $(7, \mathrm{B}(5))$ in G_3 kommen jedoch schon in G_2 beziehungsweise $\mathcal{M}_{1.3}^{(2)}$ vor. Auch hier ist dieselbe Nummerierung gewählt worden wie bei den bereits eingeführten äquivalenten SA-Werteverteilungen (vgl. Tabelle 3.10 auf Seite 73). Alle anderen Matchings in $\mathcal{G}^{(2)}$ werden gelöscht, da sie nicht erweitert werden können.

Interessanterweise gibt es kein Δ-top Matching, das eine Labelkante $(2, \mathrm{B})$ enthält, obwohl in jedem Graphen, in dem 2 vorkommt, auch ein B vorkommt. Dieses Ergebnis kann auch logisch hergeleitet werden. Würde die Kante $(2, \mathrm{B})$ in einem Matching vorkommen, müsste dieses auch die Kanten $(3, \mathrm{A}(2))$ (in G_1) und $(5, \mathrm{A}(2))$ (in G_3) enthalten. Folglich muss es auch die Kanten $(5, \mathrm{A}(1))$ und $(1, \mathrm{B}(5))$ in G_2 besitzen, was dazu führt, dass Tupel 1 in G_1 mit B(3) matchen würde. Die beiden Kanten $(3, \mathrm{A}(2))$ und $(1, \mathrm{B}(3))$ in G_1 verletzten jedoch die Δ-top Eigenschaft. Aus den gegebenen Matchings kann dieser Zusammenhang ebenfalls abgeleitet werden und ist graphisch in Abbildung 5.4 dargestellt. $\mathcal{M}_{3.1}^{(2)}$ (siehe (b)) ist das einzige Δ-top Matching, das die Labelkante $(2, \mathrm{B})$ enthält. Dieses kann in G_3 nur mit der Kante $(2, \mathrm{B}(5))$ erweitert werden, woraus folgt, dass 5 auch in G_2 mit einem A matchen muss. $\mathcal{M}_{7.1}^{(2)}$ (siehe (c)) ist wiederum das einzige Δ-top Matching, das die Labelkante $(5, \mathrm{A})$ enthält. Eine Erweiterung von $\mathcal{M}_{3.1}^{(2)}$ ist also nichts anderes als ein Zusammenlegen der Matchings $\mathcal{M}_{3.1}^{(2)}$ und $\mathcal{M}_{7.1}^{(2)}$ (siehe (d)), wobei nur die Kanten betrach-

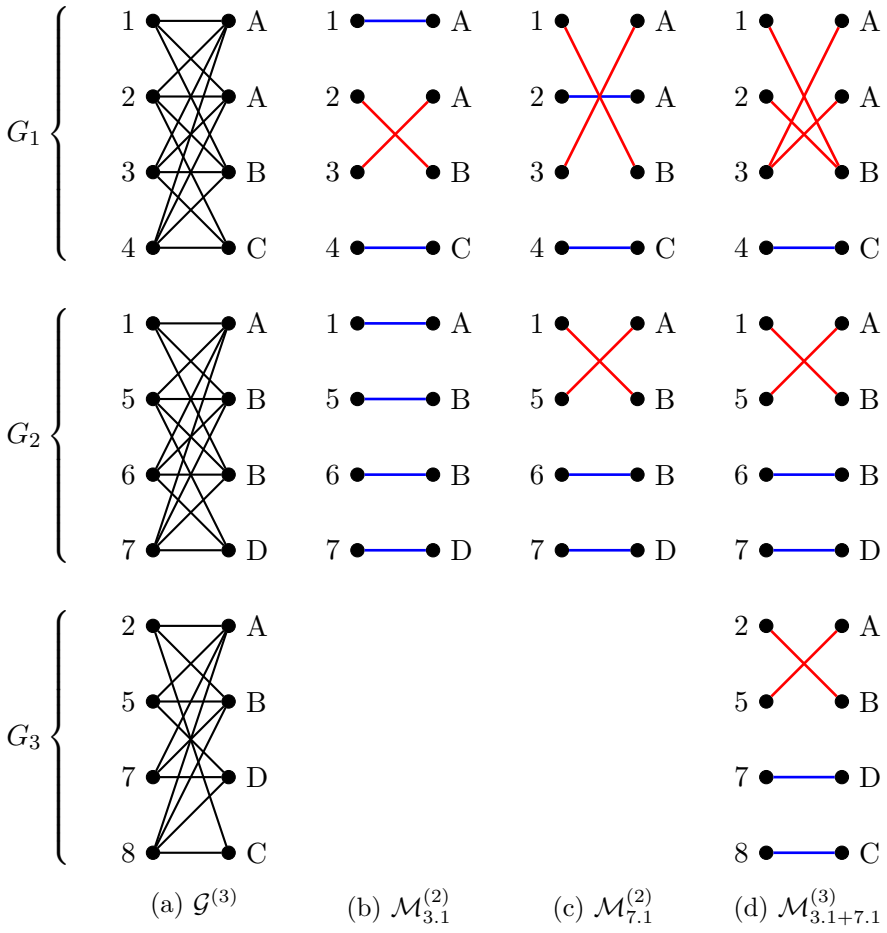


(a) $\mathcal{G}^{(3)}$ (b) $\mathcal{M}^{(3)}_{1.3}$ (c) $\mathcal{M}^{(3)}_{2.1}$ (d) $\mathcal{M}^{(3)}_{6.1}$ (e) $\mathcal{M}^{(3)}_{7.3}$

ID	Name	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$
1	Alison	A	B(3), C(4)	B(3,5), B(3,6)	B(3,6)
2	Ben	A	B(3), C(4)	B(3), C(4)	C(4,8)
3	Clark	B	A(1), A(2), C(4)	A(1), A(2), C(4)	A(1), C(4)
4	Debra	C	A(1), A(2), B(3)	A(2), B(3)	A(2), B(3)
5	Elaine	B		A(1), D(7)	D(7)
6	Fiona	B		A(1), D(7)	A(1)
7	Gary	D		B(5), B(6)	B(5)
8	Helen	C			A(2)

(f) Δ-top Matchingtabelle $\mathcal{T}^{(3)}_{\Delta}$ für $\mathcal{G}^{(3)}$

Abbildung 5.3: Alle modellierten Matchings für $\mathcal{G}^{(3)}$



(a) $\mathcal{G}^{(3)}$ (b) $\mathcal{M}^{(2)}_{3.1}$ (c) $\mathcal{M}^{(2)}_{7.1}$ (d) $\mathcal{M}^{(3)}_{3.1+7.1}$

Abbildung 5.4: Das Zusammenlegen der Δ-top Matchings $\mathcal{M}^{(2)}_{3.1}$ und $\mathcal{M}^{(2)}_{7.1}$ funktioniert in diesem Fall nicht. Dabei werden alle Kanten übernommen, die verschieden vom Originalmatching und hier rot dargestellt sind. Die entstehende Kantenmenge $\mathcal{M}^{(2)}_{3.1+7.1}$ ist kein Matching mehr.

tet werden, die verschieden vom Originalmatching sind (dargestellt als rote Kanten). In G_1 ist zu erkennen, dass 3 sowohl mit A(1) als auch mit A(2) eine Matchingkante bilden soll und dass 1 und 2 mit B(3) matchen. Demzufolge verstößt die entstehende Kantenmenge gegen die Matchingdefinition. Aus dem gleichen Grund gibt es auch kein Δ-top Matching, welches die Kante $(5, \mathrm{A})$ enthält.

Während es in $\mathcal{G}^{(3)}$ auch insgesamt kein valides Matching gibt, in dem die Kante $(2, \mathrm{B})$ vorkommt, entsteht durch die Beschränkung auf Δ-top Matchings ein Verlust anderer Zuordnungen. Tabelle 3.11 auf Seite 73 gibt eine Übersicht über die SA-Werte, die den einzelnen Tupeln bei vollständiger Betrachtung aller SA-Werteverteilungen zugeordnet werden können. Ein Vergleich mit der Matchingtabelle $\mathcal{T}^{(3)}_{\Delta}$ in Abbildung 5.3f (siehe Spalte $\mathcal{G}^{(3)}$) zeigt, dass zum Beispiel Elaine (ID 5) kein A zugeordnet wird. Das liegt daran, dass beide validen Matchings $\mathcal{M}^{(3)}_{9.1}$ und $\mathcal{M}^{(3)}_{9.2}$, die zu einer möglichen SA-Werteverteilung mit Elaine

und A führen, Δ3- beziehungsweise Δ4-Matchings bezüglich des Originalmatchings sind.[6] Gleiches trifft für die Zuordnungen von Fiona zu D, Gary zu A, Helen zu B und Helen zu D zu. Somit wird deutlich, dass die vorgeschlagene Modellierung nur eine Approximation für das Wissen des Angreifers ist.

5.3 Konstruktion von Matchings

Um Erweiterungen von Matchings für einen Anfragegraphen G_{n+1} zu berechnen, spielt es eine große Rolle, ob Tupel aus G_{n+1} in bereits bearbeiteten Graphen $G_1, \ldots, G_n$ vorkamen. Daher werden Tupel in *alte* und *neue Tupel* unterschieden. Erstere sind solche, die bereits in mindestens einem vorigen Graphen G_i mit $1 \leq i < n+1$ vorkamen, während letztere bisher nur in G_{n+1} vorkommen. Für alte Tupel existieren bereits Mengen von sensiblen Werten, welche den Tupeln zugeordnet werden können. Diese Mengen können nicht vergrößert, sondern nur noch verkleinert werden, wenn bestimmte Möglichkeiten wegfallen. Für neue Tupel werden Angaben über mögliche sensible Werte erst mit den nun berechneten beziehungsweise erweiterten Matchings gemacht. Theoretisch können ihnen alle SA-Werte zugeordnet werden, die in G_{n+1} vorkommen, vorausgesetzt, es existiert ein entsprechendes Matching in $\mathcal{G}^{(n+1)}$. Formal werden alte und neue Tupel wie folgt definiert.

Definition 5.2 (alte und neue Tupel) *Sei $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und bezeichne $S_T(\mathcal{G}^{(n)})$ die Menge aller Tupel, die als Label in $\mathcal{G}^{(n)}$ vorkommen sowie $S_T(G_n)$ die Menge aller Tupel, die als Label in G_n vorkommen. Sei G_{n+1} ein weiterer Anfragegraph, dann heißt $S_{alt}(G_{n+1}) \coloneqq S_T(\mathcal{G}^{(n)}) \cap S_T(G_{n+1})$ die Menge der* alten Tupel *und $S_{neu}(G_{n+1}) \coloneqq S_T(G_{n+1}) \setminus S_T(\mathcal{G}^{(n)})$ die Menge der* neuen Tupel *in G_{n+1}.*

Beispiel 5.1 In G_2 aus Abbildung 5.2 auf Seite 134 ist 1 ein altes Tupel, da es bereits in G_1 vorkam. 5, 6 und 7 sind neue Tupel. In G_3 (siehe Abbildung 5.3 auf Seite 136) ist nur 8 ein neues Tupel und alle anderen sind alt.

$S_T(\mathcal{G}^{(1)}) = \{1,2,3,4\}$	$S_T(\mathcal{G}^{(2)}) = \{1,2,3,4,5,6,7\}$	$S_T(\mathcal{G}^{(3)}) = \{1,2,3,4,5,6,7,8\}$
$S_T(G_1) = \{1,2,3,4\}$	$S_T(G_2) = \{1,5,6,7\}$	$S_T(G_3) = \{2,5,7,8\}$
$S_{\text{alt}}(G_1) = \emptyset$	$S_{\text{alt}}(G_2) = \{1\}$	$S_{\text{alt}}(G_3) = \{2,5,7\}$
$S_{\text{neu}}(G_1) = \{1,2,3,4\}$	$S_{\text{neu}}(G_2) = \{5,6,7\}$	$S_{\text{neu}}(G_3) = \{8\}$

Seien $\mathcal{T}_\Delta^{(n)}$ eine Δ-top Matchingtabelle für die Folge von Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ und G_{n+1} ein weiterer Anfragegraph. Der hier vorgestellte Algorithmus, der Matchings für $\mathcal{G}^{(n+1)}$ aus $\mathcal{T}_\Delta^{(n)}$ berechnet, arbeitet nicht matching-, sondern kantenorientiert. Das bedeutet, es werden Entscheidungen, ob ein Matching erweitert, erhalten oder gelöscht wird, nur anhand einzelner Δ-top Matchingkanten getroffen. Dieser Ansatz ist eine starke Vereinfachung der Berechnung, der wesentlich effizienter umzusetzen ist, als komplette Matchings aus der Tabelle zu extrahieren. Kanten, die in mehreren verschiedenen Matchings vorkommen, müssen nämlich nur einmal betrachtet werden.

Sei t ein altes Tupel in G_{n+1} mit originalem SA-Wert s und $e_\Delta = (t, s'(S_T))$ eine Δ-top Matchingkante von t. Wie die Beispiele bereits gezeigt haben, müssen Matchings mit e_Δ

[6] Diese Behauptung ist aus den SA-Werteverteilungen der Tabelle 3.10 auf Seite 73 abzulesen. Die entsprechenden Matchings sind in dieser Arbeit graphisch nicht dargestellt.

gelöscht werden, wenn in G_{n+1} kein SA-Knoten mit Label s' vorkommt. In diesem Fall kann es in $\mathcal{G}^{(n+1)}$ kein valides Matching geben, das die Labelkante (t, s') enthält. e_Δ muss ebenfalls gelöscht werden, wenn mindestens zwei Tupel aus S_T in G_{n+1} vorkommen. In diesem Fall würde ein valides Matching in $\mathcal{G}^{(n+1)}$, das die Kante e_Δ beinhaltet, nicht Δ-pfadkonform bezüglich des Originalmatchings sein.[7] Damit wäre das Matching nicht Δ-top und wird somit hier nicht modelliert. Wenn genau ein Tupel aus S_T in $\mathcal{G}^{(n+1)}$ vorkommt, bleibt e_Δ in der aktuellen Form erhalten. e_Δ kann erweitert werden, wenn kein Tupel aus S_T in $\mathcal{G}^{(n+1)}$ vorkommt. Dazu muss es aber ein Tupel t' in $\mathcal{G}^{(n+1)}$ geben, das den originalen SA-Wert s' hat und dem s zugeordnet werden kann. Das ist der Fall, wenn eine Δ-top Matchingkante $e'_\Delta = (t', s(S'_T))$ existiert oder t' ein neues Tupel ist. Für neue Tupel existieren noch keine Matchings und demzufolge können ihnen prinzipiell alle SA-Werte zugeordnet werden, die in der aktuellen Anfrage vorkommen. Das heißt insbesondere, dass jedes neue Tupel auch mit jedem anderen neuen Tupel erweitert werden kann. Dabei wird beiden Tupeln der SA-Wert des jeweils anderen Tupels zugeordnet und es entstehen sogenannte neue Matchings, die als Erweiterung des Originalmatchings gelten.

Algorithmus 5.1 Berechne Δ-top Matchings (Kurzform)

Eingabe: $\mathcal{T}_\Delta^{(n)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$, G_{n+1}: Graph
Ausgabe: $\mathcal{T}_\Delta^{(n+1)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$

```
1: T_Δ^(n+1) ← T_Δ^(n)
2: for all alte Tupel t in G_{n+1} do
3:    for all Δ-top Matchingkanten e_Δ = (t, s'(S_T)) in T_Δ^(n+1) do
4:       if s' kommt nicht in G_{n+1} vor then
5:          Lösche Matchings mit e_Δ
6:       end if
7:       Vergleiche S_T mit S_T(G_{n+1})
8:          Fall 1: genau ein Tupel aus S_T in G_{n+1} ⇒ e_Δ muss nicht erweitert werden
9:          Fall 2: mind. zwei Tupel aus S_T in G_{n+1} ⇒ lösche Matchings mit e_Δ
10:         Fall 3: andernfalls ⇒ erweitere e_Δ mittels Erweiterungsalgorithmus
11:   end for
12: end for
13: for all neue Tupel t, t' mit verschiedenen SA-Werten do
14:    Erzeuge neues Matching mit Kanten (t, s'(t')) und (t', s(t)), wobei s SA-Wert von t
       und s' SA-Wert von t'
15: end for
16: return T_Δ^(n+1)
```

Algorithmus 5.1 gibt zunächst einen groben Überblick über die Konstruktion von Matchings für $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$. Als Erstes wird die Δ-top Matchingtabelle als $\mathcal{T}_\Delta^{(n+1)}$ übernommen und es werden nur darin Änderungen durchgeführt. Dazu werden alle gespeicherten Δ-top Matchingkanten $e_\Delta = (t, s'(S_T))$ der alten Tupel untersucht und deren SA-Werte und Tupel mit denen aus G_{n+1} verglichen. e_Δ wird gelöscht, wenn s' nicht in G_{n+1} vorkommt. Beim Löschen von Kanten muss weiterhin sichergestellt werden, dass alle Matchings, die diese Kanten enthalten, auch gelöscht werden (vgl. Algorithmus 5.2).

[7] Vgl. Definition 4.7 in Kapitel 4.3.5.

Wenn s' in G_{n+1} vorkommt, wird die Tupelmenge S_T der Kante mit den Tupeln aus G_{n+1} verglichen. Dafür werden drei verschiedene Fälle unterschieden. Als Erstes kann genau ein Tupel aus S_T in G_{n+1} vorkommen (Fall 1), wobei das Matching beziehungsweise die Kante erhalten bleibt. Da bereits alle Kanten nach $\mathcal{T}_\Delta^{(n+1)}$ übertragen wurden, ist in diesem Fall nichts weiter zu tun. Die zweite Möglichkeit ist, dass zwei oder mehr Tupel aus S_T in G_{n+1} vorkommen (Fall 2). Das würde in $\mathcal{G}^{(n+1)}$ zu einem Matching führen, das nicht Δ-pfadkonform bezüglich des Originalmatchings ist. Somit werden erneut e_Δ und alle dazugehörigen Matchings gelöscht. Im letzten Fall ist kein Tupel aus S_T in G_{n+1} vorhanden und e_Δ kann erweitert werden. Dieser Schritt ist das Herzstück des gesamten Algorithmus und nicht trivial. Es muss gewährleistet werden, dass einerseits entstehende Matchings Δ-top sind und andererseits nicht zu viele solcher Matchings entstehen. In Kapitel 4 wurde bereits festgelegt, dass die Anzahl der berechneten Matchings nur polynomiell in der Anzahl der im Anfragegraphen enthaltenen Tupel sein soll, damit der Approximationsalgorithmus in Polynomialzeit läuft. Demzufolge soll beispielsweise eine exponentielle Anzahl von Matchings vermieden werden. Für die Umsetzung dieser Bedingungen wird ein sogenannter *Erweiterungsalgorithmus* verwendet, auf den in den nächsten Abschnitten und im Kapitel 6 näher eingegangen wird. Als Letztes werden mithilfe von Algorithmus 5.1 neue Matchings berechnet, welche Erweiterungen des Originalmatchings darstellen. Für jedes Paar neuer Tupel, welche unterschiedliche originale SA-Werte haben, wird ein Matching erstellt. Dieses besteht aus genau zwei Δ-top Matchingkanten und unterscheidet sich vom Originalmatching nur in G_{n+1}. Im ersten Anfragegraphen G_1, in dem nur neue Tupel vorkommen, werden Matchings nur mithilfe dieser Zeilen konstruiert.

Algorithmus 5.2 Lösche

Eingabe: $e_\Delta = (t, s'(S_T))$: Δ-top Matchingkante
Ausgabe: – (löscht alle Matchings mit e_Δ aus der Δ-top Matchingtabelle $\mathcal{T}_\Delta$)

```
1: Lösche Kante e_Δ aus T_Δ
2: for all Tupel t' ∈ S_T do
3:   if t' kommt in keiner anderen Kante e*_Δ = (t, s'(S*_T)) von t vor then
4:     for all Kanten e'_Δ = (t', s(S'_T)) von t', in der t vorkommt (d. h. t ∈ S'_T) do
5:       Lösche(e'_Δ)
6:     end for
7:   end if
8: end for
9: return
```

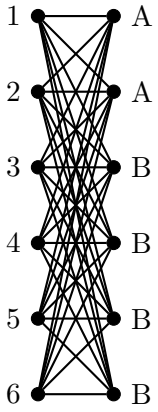
Algorithmus 5.2 (*Lösche*) zeigt, wie Matchings gelöscht werden, die eine bestimmte Δ-top Matchingkante $e_\Delta = (t, s'(S_T))$ beinhalten. Nachdem e_Δ aus der Matchingtabelle entfernt wurde, werden alle Tupel $t' \in S_T$ dahingehend überprüft, ob sie in anderen Kanten von t vorkommen. Ist das nicht der Fall, existiert kein Matching mehr, das die Kante $(t, s'(t'))$ enthält. Entsprechend der Definition von Δ2-Matchings müssen nun auch alle Gegenkanten $(t', s(t))$ aus der Tabelle gelöscht werden, was durch ein rekursives Aufrufen der Methode umgesetzt wird.[8] Der Algorithmus wird für alle Stellen in Algorithmus 5.1 verwendet, an denen eine Kante gelöscht werden soll.

[8] Vgl. Kapitel 4.3.4.

Im nächsten Abschnitt wird der Erweiterungsschritt von Matchings genauer untersucht. Es werden zwei Kriterien eingeführt, die die Anzahl möglicher Erweiterungen einschränken und dadurch für eine polynomielle Laufzeit sorgen. Darauf aufbauend wird eine detaillierte Form von Algorithmus 5.1 vorgestellt.

5.4 Erweiterung von Matchings

Die zentrale Idee beim Erweitern von Matchings lautet, Entscheidungen anhand von einzelnen Δ-top Matchingkanten und nicht mithilfe kompletter Matchings zu treffen. Seien $\mathcal{T}_\Delta^{(n)}$ eine Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und G_{n+1} ein weiterer Anfragegraph. Weiterhin seien t ein altes Tupel mit originalem SA-Wert s und $e_\Delta = (t, s'(S_T))$ eine Δ-top Matchingkante von t. Damit e_Δ beziehungsweise alle Matchings, die e_Δ enthalten, erweitert werden können, müssen zwei Bedingungen gelten. Zum einen darf kein Tupel aus S_T in G_{n+1} vorkommen und zum anderen muss es ein Tupel t' in G_{n+1} geben, das den SA-Wert s' hat und dem s zugeordnet werden kann. Wenn t' ein altes Tupel ist, muss dafür eine Δ-top Matchingkante $e'_\Delta = (t', s(S'_T))$ existieren, die den SA-Wert s enthält. Ist t' ein neues Tupel, kann s in jedem Fall t' zugeordnet werden und die Erweiterung ist prinzipiell möglich.



(a) G_{n+1}

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B(3), B(7)	B(3), B(7, 4), B(7, 5), B(7, 6)
2	A	B(8), B(9)	B(8, 3), B(9, 3), B(8, 4), B(9, 4), B(8, 5), B(9, 5), B(8, 6), B(9, 6)
3	B	A(1)	A(1), A(2)
4	B	A(10)	A(10, 1), A(10, 2)
5	B		A(1)
6	B		A(2)

(b) Δ-top Matchingtabelle

Abbildung 5.5: Erweiterungen von Matchingkanten (ohne Beschränkungen). Die Tupel 7, 8, 9 und 10 aus der Tabelle kommen in Graphen G_i mit $1 \leq i \leq n$ vor, die hier nicht dargestellt sind.

Abbildung 5.5 zeigt in (a) einen Anfragegraphen G_{n+1}, zu dem Matchings berechnet werden sollen. In (b) sind in Spalte $\mathcal{G}^{(n)}$ bereits alle Δ-top Matchingkanten gespeichert, die für Matchings in $\mathcal{G}^{(n)}$ stehen, welche nun erweitert werden sollen. Die Graphen $G_1, \ldots, G_n$ sind hier nicht dargestellt, ebenso fehlen Angaben zu weiteren Tupeln in der Tabelle. Tupel 1 bis 4 kamen bereits in $\mathcal{G}^{(n)}$ vor und sind damit alte Tupel in G_{n+1}. Für Tupel 5 und 6 gibt es noch keine Matchings, da sie neue Tupel sind.

Um alle gespeicherten Matchings zu erweitern, werden nur die Kanten in $\mathcal{G}^{(n)}$ betrachtet. $(1, \mathrm{B}(3))$ und $(3, \mathrm{A}(1))$ müssen nicht erweitert werden, da die Tupel 3 beziehungsweise 1 in G_{n+1} vorkommen. Die dazugehörigen Matchings bleiben in $\mathcal{G}^{(n+1)}$ erhalten. Alle anderen Kanten sind in Spalte $\mathcal{G}^{(n+1)}$ mit allen in Frage kommenden Tupeln erweitert worden.

Beispielsweise wurde die Kante $(1, \mathrm{B}(7))$ mit dem alten Tupel 4 sowie den neuen Tupeln 5 und 6 erweitert. Gleiches gilt für die Kanten von Tupel 2, wobei hier zusätzlich auch mit Tupel 3 erweitert wurde. Eine Erweiterung von $(1, \mathrm{B}(7))$ mit 3 würde zu Matchings führen, die nicht Δ-pfadkonform sind, denn Tupel 1 hat bereits eine andere Matchingkante mit Tupel 3.[9]

Im Folgenden werden die Matchings untersucht, die durch die erweiterten Δ-top Matchingkanten entstehen. Dabei wird eine Unterscheidung durchgeführt, ob das Tupel, mit dem erweitert wurde, ein altes oder ein neues Tupel ist. In beiden Fällen ist unterschiedlich zu verfahren, denn nicht alle im Beispiel durchgeführten Erweiterungen führen auch zu Δ-top Matchings in $\mathcal{G}^{(n+1)}$. Dafür werden zwei Bedingungen eingeführt, nach denen die Erweiterung von Matchingkanten in jedem Fall zu Δ-top Matchings führt. Die tatsächlich durchführbaren Erweiterungen für den obigen Beispielgraphen werden am Ende dieses Unterkapitels aufgelistet.

5.4.1 Erweiterung mit alten Tupeln

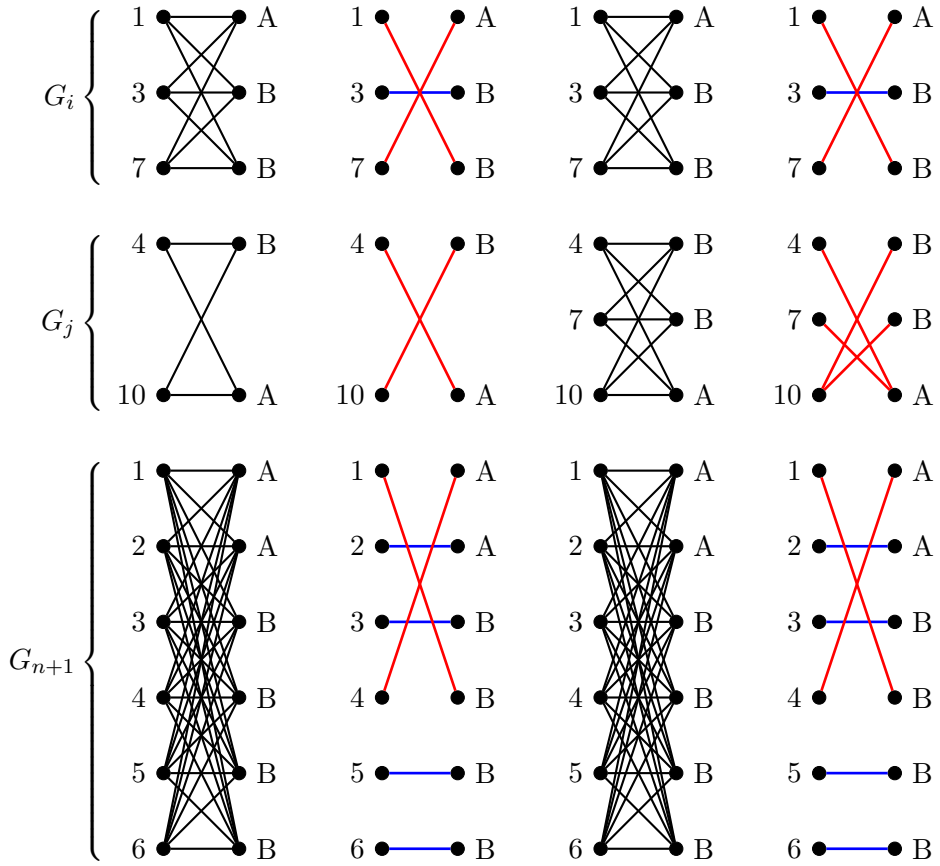
Als Erstes wird der Fall von Erweiterungen mit alten Tupeln untersucht. Für den Graphen in Abbildung 5.5 wurde zum Beispiel die Kante $(1, \mathrm{B}(7))$ mit dem alten Tupel 4 zu $(1, \mathrm{B}(7, 4))$ erweitert. Dazu werden zwei mögliche Folgen von Anfragegraphen betrachtet, die in Abbildung 5.6 dargestellt sind. In (a) ist ein Anfragegraph G_i gegeben, der aus den Tupeln 1, 3 und 7 besteht, sowie ein Graph G_j, in dem die Tupel 4 und 10 vorkommen. Weiterhin seien zwei Δ-top Matchings gegeben, wobei eins in G_i die Kanten $(1, \mathrm{B}(7))$ und $(7, \mathrm{A}(1))$ sowie das andere in G_j die Kanten $(4, \mathrm{A}(10))$ und $(10, \mathrm{B}(4))$ enthalte. Dadurch entstehen genau die Einträge der Matchingtabelle, die in Abbildung 5.5b in Spalte $\mathcal{G}^{(n)}$ dargestellt sind. Ein Erweitern der Kante $(1, \mathrm{B}(7))$ mit Tupel 4 führt dazu, dass beide beschriebenen Matchings zusammenlegt werden, wobei nur Kanten betrachtet werden, die verschieden vom Originalmatching sind. (b) zeigt das entstehende Matching $\mathcal{M}_1^{(n+1)}$, welches die Δ-top-Eigenschaft hat. Die Erweiterung von $(1, \mathrm{B}(7))$ mit 4 ist in diesem Fall problemlos möglich.

In (c) ist eine andere mögliche Folge von Anfragegraphen gegeben, die sich zur ersten Variante nur im Graphen G_j unterscheidet. Dieser besteht diesmal aus den Tupeln 4, 7 und 10. Dadurch müssen in dem Matching in $\mathcal{G}^{(n)}$, das erneut die Kante $(1, \mathrm{B}(7))$ enthält, in G_j auch die Kanten $(7, \mathrm{A}(10))$ und $(10, \mathrm{B}(7))$ enthalten sein. Durch die Erweiterung von $(1, \mathrm{B}(7))$ mit 4 wird dieses Matching mit demjenigen vereinigt, in dem in G_j wiederum die Kanten $(4, \mathrm{A}(10))$ und $(10, \mathrm{B}(4))$ vorkommen. Die auf diese Weise entstehende Kantenmenge $\mathcal{M}_2^{(n+1)}$ ist in (d) angegeben. Da es Kanten mit gemeinsamen Knoten gibt, ist $\mathcal{M}_2^{(n+1)}$ kein Matching. Es ist leicht zu sehen, dass kein Δ-top Matching existiert, welches die Kante $(1, \mathrm{B}(7, 4))$ enthält. Somit ist in diesem Fall die Erweiterung von $(1, \mathrm{B}(7))$ mit 4 nicht möglich.

Beispiele A.8 und A.9 im Anhang stellen weitere Situationen dar, in denen die Erweiterung mit alten Tupeln einmal zu einem Δ-top Matching führt und einmal nicht. Prinzipiell funktioniert es, wenn die Knotenmengen[10] der Matchings zueinander disjunkt sind und auch alle Δ-top Kriterien im aktuellen Graphen G_{n+1} eingehalten werden. An dieser Stelle

[9] Vgl. Definition 4.7 in Kapitel 4.3.5.

[10] Damit sind nur Tupel- und SA-Knoten gemeint, die im Matching einen anderen Matchingpartner haben als im Originalmatching.



(a) $\mathcal{G}_1^{(n+1)}$ (b) $\mathcal{M}_1^{(n+1)}$ (c) $\mathcal{G}_2^{(n+1)}$ (d) $\mathcal{M}_2^{(n+1)}$

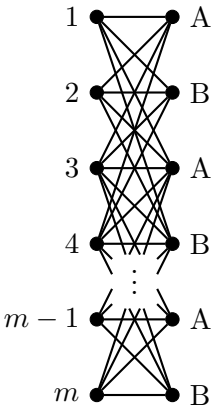
Abbildung 5.6: Erweiterungen mit alten Tupeln

liegt jedoch ein großes Augenmerk auf der Komplexität eines Tests, ob eine Kantenmenge ein Δ-top Matching darstellt. Insbesondere wird gezeigt, dass durch eine Erweiterung mit alten Tupeln exponentiell viele Matchings entstehen können, was gegen die zuvor festgelegten Kriterien des Approximationsalgorithmus verstößt.

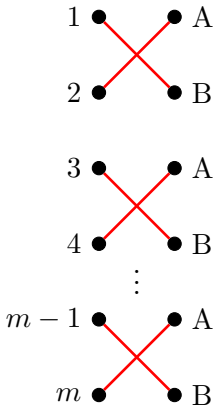
In Abbildung 5.7 ist ein weiteres etwas komplizierteres Beispiel skizziert. Gegeben sei eine Folge von Anfragegraphen $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$. Der Graph G_{n+1} bestehe aus m Tupel- und SA-Knoten, wobei die SA-Knoten abwechselnd mit A und B gelabelt seien. Für $\mathcal{G}^{(n)}$ sei ein Δ-top Matching $\mathcal{M}_1^{(n)}$ gegeben, das jedem Tupel 1 bis m ein B zuordnet. Es existieren weitere Tupel, die mit A matchen, aber hier nicht dargestellt sind. Alle Tupel mit ungerader Nummer $u \leq m$ haben damit in $\mathcal{M}_1^{(n)}$ als Matchingpartner den jeweils anderen SA-Wert, während alle Tupel mit gerader Nummer $g \leq m$ mit ihren Originalwert matchen. In jeder Erweiterung müssen somit alle Tupel u mit B matchen, während alle Tupel g einen beliebigen Matchingpartner haben können. Weiterhin sei für jede gerade Zahl $g \leq m$ mindestens ein Matching $\mathcal{M}_g^{(n)}$ gegeben, das dem Tupel g ein A zuordnet, allen anderen Tupeln $1, \ldots, m$ aber den jeweiligen tatsächlichen Wert (vgl. Matchings $\mathcal{M}_2^{(n)}, \mathcal{M}_4^{(n)}$ in (a)).

ID	SA	$\mathcal{M}_1^{(n)}$	$\mathcal{M}_2^{(n)}$	$\mathcal{M}_4^{(n)}$	...
1	A	B	A	A	...
2	B	B	A	B	...
3	A	B	A	A	...
4	B	B	B	A	...
⋮	⋮	⋮	⋮	⋮	⋱
$m-1$	A	B	A	A	...
m	B	B	B	B	...
⋮	⋮	⋮	⋮	⋮	⋱

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) G_{n+1}



(c) $\mathcal{M}_1^{(n+1)}[G_{n+1}]$

Abbildung 5.7: Erweiterung eines Matchings

Für $\mathcal{M}_1^{(n)}$ ist nun eine Erweiterung für G_{n+1} gesucht. Da es insgesamt $\frac{m}{2}$ SA-Knoten mit Label B gibt, gibt es $\frac{m}{2}!$ Möglichkeiten, Matchingpartner für die Tupel mit ungerader Nummer zu finden. Sollen also alle möglichen Erweiterungen gespeichert werden, müssen exponentiell viele Matchings abgebildet werden. Da die SA-Werteverteilung aller dieser Matchings für die Tupel $1, \ldots, m$ identisch ist,[11] könnte man anstatt aller Erweiterungen nur eine (oder wenige) modellieren. Beispielsweise würden die Kanten $(1, \mathrm{B}(2)), (2, \mathrm{A}(1)), (3, \mathrm{B}(4)), (4, \mathrm{A}(3)), \ldots, (n-1, \mathrm{B}(n)), (n, \mathrm{A}(n-1))$ ein Matching in G_{n+1} darstellen, das dort Δ-top ist (siehe (c)). In einem Matching $\mathcal{M}_1^{(n+1)}$, das in der kompletten Anfragenfolge $\mathcal{G}^{(n+1)}$ Δ-top ist, muss jedes ungerade Tupel ein B (wie in $\mathcal{M}_1^{(n)}$) und jedes gerade ein A matchen (wie in den einzelnen $\mathcal{M}_g^{(n)}$). Das bedeutet, $\mathcal{M}_1^{(n+1)}$ entsteht durch die Zusammenlegung von $\mathcal{M}_1^{(n)}$ mit allen Matchings $\mathcal{M}_g^{(n)}$. Seien für jedes gerade Tupel g mindestens zwei verschiedene Matchings $\mathcal{M}_{g.1}^{(n)}$ und $\mathcal{M}_{g.2}^{(n)}$ gegeben, in denen g mit A matcht (analog zu $\mathcal{M}_2^{(n)}$ und $\mathcal{M}_4^{(n)}$), aber mindestens einem anderen hier nicht dargestellten Tupel ein anderer SA-Wert zugeordnet wird. Dann gibt es insgesamt $2^{m/2}$ mögliche Matchings für $\mathcal{G}^{(n+1)}$, die zwar alle in G_{n+1} übereinstimmen, bei denen aber jeweils mindestens ein Tupel mit einem anderen SA-Wert matcht. Um also nur eins der infrage kommenden Matchings zu modellieren, muss eins dieser potentiellen Matchings ausgewählt und auf die Δ-top Eigenschaften getestet werden. Ein einfacher Algorithmus, der über alle Matchings iteriert und das erste auswählt, welches Δ-top ist, hat im schlechtesten Fall exponentielle Laufzeit, wenn zum Beispiel kein Δ-top Matching existiert. Der Algorithmus würde in diesem Fall alle Matchings testen. Ein formaler Beweis, dass es keinen anderen Algorithmus gibt, der die Erweiterung in polynomieller Zeit findet, wird in dieser Arbeit nicht geführt. Beispiel A.10 im Anhang auf Seite 280 zeigt allerdings für genau diesen Fall eine Folge von Anfragegraphen, die vermuten lässt, dass es wohl keinen schnelleren Algorithmus gibt.

Da die gesamte Anfragebearbeitung in Polynomialzeit ausgeführt werden soll, wird als Konsequenz dieser Betrachtungen die Erweiterung von Matchings eingeschränkt. Sei t ein

[11] Knoten mit ungeradem Label matchen ein B und Knoten mit geradem Label ein A.

altes Tupel, das in $\mathcal{M}^{(n)}$ mit einem SA-Wert s' matcht, der nicht sein originaler Wert ist. Dann werden als mögliche Matchingpartner in G_{n+1} nur SA-Knoten $s'(t')$ betrachtet, bei denen entweder die Kante $(t, s'(t'))$ bereits in $\mathcal{M}^{(n)}$ enthalten oder t' ein neues Tupel ist. Da ersteres keine Erweiterung nach Definition 5.1 darstellt, bedeutet letzteres nichts anderes, als dass Δ-top Matchingkanten beziehungsweise alte Tupel nur noch mit neuen Tupeln erweitert werden.

Der große Vorteil dieser Beschränkung ist, dass sie leicht überprüft werden kann und immer zu Δ-top Matchings führt. Da neue Tupel in keinem Matching von $\mathcal{G}^{(n)}$ vorkommen, können sie in $\mathcal{G}^{(n+1)}$ mit jedem beliebigen SA-Knoten matchen. Wird also ein Matching $\mathcal{M}^{(n)}$ zu einem Matching $\mathcal{M}^{(n+1)}$ erweitert, dann stimmt $\mathcal{M}^{(n+1)}$ auf $\mathcal{G}^{(n)}$ immer mit $\mathcal{M}^{(n)}$ überein. Matchings werden somit nicht mehr vereinigt, sondern bestehende Matchings werden nur um Kanten im aktuellen Graphen $\mathcal{G}^{(n+1)}$ ergänzt.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B(3), B(7)	B(3), B(7, 5), B(7, 6)
2	A	B(8), B(9)	B(8, 5), B(9, 5), B(8, 6), B(9, 6)
3	B	A(1)	A(1)
4	B	A(10)	–
5	B		A(1)
6	B		A(2)

Tabelle 5.1: Erweiterungen von Matchingkanten (inkl. Beachtung der ersten Einschränkung)

Für das anfangs eingeführte Beispiel in Abbildung 5.5 zeigt Tabelle 5.1 alle Δ-top Matchingkanten, die die erste hier aufgeführte Einschränkung erfüllen. Alle Kanten der Tupel 1 und 2 dürfen nicht mit den alten Tupeln 3 oder 4 erweitert werden. Das gleiche gilt auch andersherum, wodurch Tupel 4 jetzt keine Δ-top Matchingkanten mehr besitzt.

Beispiel 5.2 Auch alle Erweiterungen aus den Abbildungen 5.2 und 5.3 sind konsistent zu dieser Einschränkung. Beispielsweise ist $\mathcal{M}_{6.1}^{(3)}$ aus Abbildung 5.3d ein Matching, das die Kante $2, \mathrm{C}$ enthält. Da 8 ein neues Tupel in G_3 ist, kann hier 2 mit 8 erweitert werden, wodurch die Kanten $(2, \mathrm{C}(8))$ und $(8, \mathrm{B}(2))$ entstehen. Wenn 8 bereits in einem vorigen Graphen vorgekommen wäre, würde diese Erweiterung gegen die Einschränkung verstoßen, da dann sowohl 2 als auch 8 alte Tupel wären, die nicht miteinander erweitert werden dürfen.

Für Tupel 1 in Abbildung 5.7, das in $\mathcal{G}^{(n+1)}$ mit einem B matchen soll, kommt B(2) als möglicher Matchingpartner nur genau dann infrage, wenn die Kante $(1, \mathrm{B}(2))$ bereits in einem Matching in $\mathcal{G}^{(n)}$ vorhanden war oder Tupel 2 nicht in $\mathcal{G}^{(n)}$ vorkommt. Im zweiten Fall gibt es dann aber kein wie in der vorigen Betrachtung gefordertes Matching $\mathcal{M}_2^{(n)}$. Der Vorteil ist jetzt, dass Matching $\mathcal{M}_1^{(n)}$ nun recht einfach erweitert werden kann. Für alle Knoten mit ungeradem Tupellabel wird entweder der SA-Knoten als Matchingpartner gewählt, den der Knoten bereits in einem vorigen Graphen hatte, oder ein neues Tupel zum Erweitern ausgesucht. Wenn alle geraden Tupel $2, 4, \ldots, m$ in G_{n+1} neue Tupel sind, gibt es zwar wieder $\frac{m}{2}!$ mögliche Erweiterungen von $\mathcal{M}_1^{(n)}$. Diesmal kann aber einfach eine beliebige ausgewählt werden, da jede Kombination ein Δ-top Matching darstellt (z. B. $\mathcal{M}_1^{(n+1)}$ aus (c)).

Ein Nachteil der hier eingeführten Einschränkung ist, dass damit nicht mehr alle Δ-top Matchings berechnet werden. Es kann vorkommen, dass Zuordnungen von SA-Werten zu Tupeln nicht abgebildet werden, obwohl es Matchings gibt, die genau das ausdrücken. Beispiel A.11 im Anhang auf Seite 282 zeigt genau so einen Fall, bei dem durch die Einhaltung der Beschränkung bestimmte SA-Werteverteilungen nicht mehr modelliert werden.

5.4.2 Tupel mit identischem SA-Wert

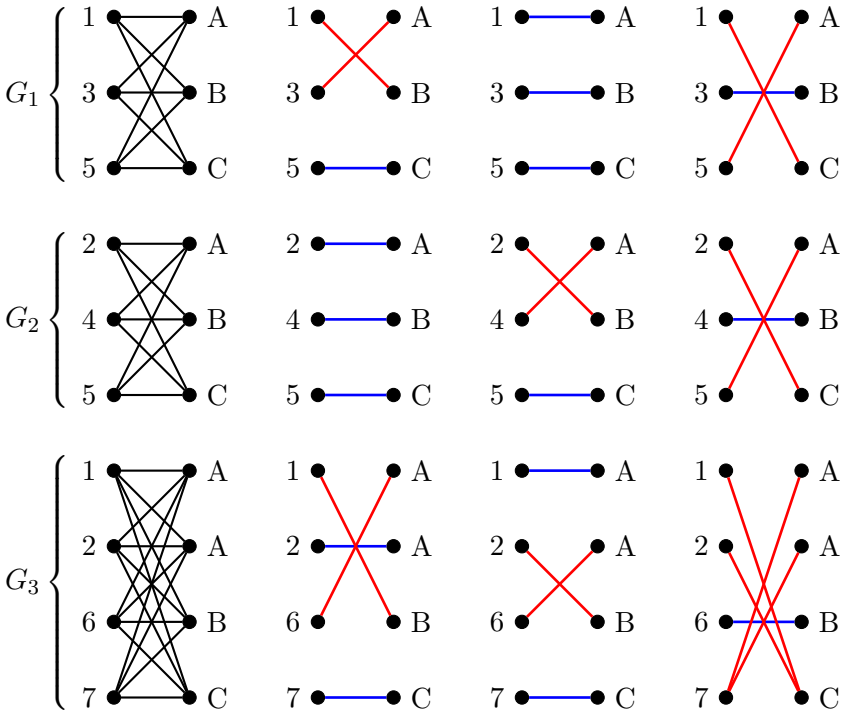
Eine zweite Einschränkung folgt für die Erweiterung mit neuen Tupeln. Sei dazu in Abbildung 5.8 ein einfaches Beispiel in Form von drei Anfragegraphen gegeben. G_3 bestehe aus Knoten für die alten Tupel 1 und 2, die neuen Tupel 6 und 7 sowie die angegebenen SA-Werte. Von besonderem Interesse sind die Erweiterungen der Kanten $(1, \mathrm{B}(3))$, $(1, \mathrm{C}(5))$, $(2, \mathrm{B}(3))$ und $(2, \mathrm{C}(5))$, zu denen die dazugehörigen Matchings in (b) bis (d) dargestellt sind. Man beachte dabei zunächst nur die beiden ersten Anfragegraphen G_1 und G_2.

Werden Entscheidungen über Erweiterungen von Matchings nur mithilfe einzelner Δ-top Matchingkanten getroffen, können prinzipiell beide alte Tupel 1 und 2 mit jeweils beiden neuen Tupeln 6 und 7 erweitert werden. Das führt zu den Einträgen in Spalte $\mathcal{G}^{(3)}$. Während es zu den Δ-top Matchingkanten $(1, \mathrm{B}(3, 6))$ und $(2, \mathrm{B}(4, 6))$ in $\mathcal{G}^{(3)}$ jeweils ein Δ-top Matching gibt (siehe (b) beziehungsweise (c)), gibt es kein Matching, in dem Tupel 1 oder 2 der SA-Wert C zugeordnet wird. Das Erweitern der Kanten $(1, \mathrm{C}(5))$ und $(2, \mathrm{C}(5))$ mit C(7) führt zu einer Kantenmenge in (d), die kein Matching darstellt. Ebenso wenig kann auch nur genau eine der beiden Kanten mit C(7) erweitert werden.

Um korrekte Erweiterungen durchzuführen, ist es im obigen Beispiel notwendig, nicht nur einzelne Matchingkanten, sondern gesamte Matchings zu betrachten. Dennoch ist das Ziel des Approximationsalgorithmus, nicht gesamte Δ-top Matchings aus einer Matchingtabelle zu extrahieren, sondern möglichst schnell erweiterte Matchings zu berechnen. Da jede Matchingkante in mehreren Matchings vorkommen kann, ist es am effizientesten, jede Kante nur einmal zu untersuchen. Wünschenswert ist daher ein kantenbasierter Ansatz, bei dem Entscheidungen über Erweiterungen nur mithilfe einzelner Δ-top Matchingkanten getroffen werden.

Die Folge dieser Überlegungen ist die Festlegung eines Kriteriums, das garantiert, wann Erweiterungen mit neuen Tupeln in jedem Fall zu Δ-top Matchings führen. Als Konsequenz wird dabei in Kauf genommen, dass dadurch eine Ungenauigkeit des Modells entstehen kann. Das bedeutet, es werden nicht alle möglichen Erweiterungen von Δ-top Matchings berechnet. Seien t, t' zwei alte Tupel, die denselben SA-Wert haben. Dann sollen die Menge aller neuer Tupel, mit denen t erweitert wird, und die Menge aller neuer Tupel, mit denen t' erweitert wird, disjunkt sein. Mit anderen Worten dürfen alte Tupel, die denselben originalen SA-Wert haben, nicht mit demselben neuen Tupel erweitert werden.

Im obigen Beispiel in Abbildung 5.8 bedeutet dies, dass Tupel 1 und 2 nicht beide mit 7 erweitert werden dürfen. Der Fall der entstehenden Kantenmenge $\mathcal{M}_3^{(n+1)}$ (siehe (d)), die kein Matching ist, ist damit ausgeschlossen. Ein kantenbasierter Algorithmus darf allerdings eines der beiden alten Tupel mit 7 erweitern. Wird beispielsweise $(1, \mathrm{C}(5))$ mit 7 zu $(1, \mathrm{C}(5, 7))$ erweitert, muss $(2, \mathrm{C}(5))$ gelöscht werden, denn es gibt kein weiteres neues Tupel mit SA-Wert C. Dadurch muss aus der Matchingtabelle das komplette Matching gelöscht werden, das die Kante $(2, \mathrm{C}(5))$ enthält. Die weiteren Kanten sind $(5, \mathrm{A}(1))$ (G_1, G_2) und auch $(1, \mathrm{C}(5))$ (G_1) beziehungsweise die gerade erweiterte Kante $(1, \mathrm{C}(5, 7))$ (G_1, G_3),



(a) $\mathcal{G}^{(3)}$ (b) $\mathcal{M}_1^{(n+1)}$ (c) $\mathcal{M}_2^{(n+1)}$ (d) $\mathcal{M}_3^{(n+1)}$

ID	SA	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$
1	A	B(3), C(5)	B(3, 6), C(5, 7)
2	A	B(4), C(5)	B(4, 6), C(5, 7)
6	B		A(1), A(2)
7	C		A(1), A(2)

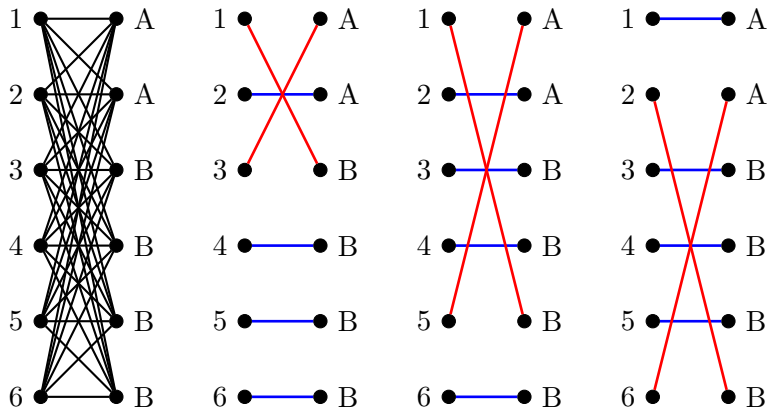
(e) (Unvollständige) Δ-top Matchingtabelle

Abbildung 5.8: Erweiterungen mit neuen Tupeln (ohne Beschränkungen). In der Tabelle in (e) sind nur die Matchingkanten der Tupel aus G_3 angegeben, die zu den Matchings in (b) bis (d) gehören.

welche alle ebenfalls gelöscht werden. Somit kann auch nicht nur genau ein Tupel mit 7 erweitert werden.

Der Vorteil dieser zweiten Einschränkung ist, dass sie sehr einfach und vor allem kantenbasiert umgesetzt werden kann. Dazu müssen nur die Δ-top Matchingkanten aller alten Tupel mit identischem SA-Wert zusammen betrachtet werden. Weiterhin wird gewährleistet, dass die Anzahl der erweiterten Matchings nicht zu groß wird. Im Beispiel in Abbildung 5.7 gibt es $\frac{m}{2}$ alte Tupel mit identischem SA-Wert, denen ebenso viele neue Tupel zum Erweitern zur Verfügung stehen, falls alle Tupel mit gerader Nummer neu sind. Insgesamt führt dieser Ansatz zu $\frac{m}{2}!$ und damit zu exponentiell vielen Erweiterungen eines Matchings. Nach der zweiten Einschränkung wird aber nur eines dieser Matchings berechnet, da jede Kante nur mit einem neuen Tupel erweitert wird.

Ein Nachteil der Einschränkung ist, dass erneut mögliche Zuordnungen von SA-Werten zu Tupeln eventuell nicht abgebildet werden, obwohl es entsprechende Matchings gibt. Beispielsweise dürfen in Abbildung 5.8 die Tupel 1 und 2 ebenfalls nicht beide mit 6 erweitert werden. Dadurch muss ein Matching gelöscht werden, das die Kante $(1, \mathrm{B})$ oder $(2, \mathrm{B})$ enthält. Wie aber in (b) und (c) zu sehen ist, gibt es zu beiden Zuordnungen jeweils ein Δ-top Matching. In diesem Fall führt die Einschränkung zum Verlust einer möglichen SA-Werteverteilung.



(a) G_{n+1} (b) M_1 (c) M_2 (d) M_3

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B(3), B(7)	B(3), B(7, 5)
2	A	B(8), B(9)	B(8, 6), B(9, 6)
3	B	A(1)	A(1)
4	B	A(10)	–
5	B		A(1)
6	B		A(2)

(e) Δ-top Matchingtabelle

Abbildung 5.9: Erweiterung eines Matchings

Für das bereits in Abbildung 5.5 eingeführte Beispiel zeigt Abbildung 5.9 in (e) die finale Δ-top Matchingtabelle. Dabei sind beide Einschränkungen umgesetzt worden. Da Tupel 5 und 6 neu sind, dürfen sie nur entweder Tupel 1 oder Tupel 2 zugeordnet werden. Dementsprechend wurde Tupel 1 nur mit 5 und 2 nur mit 6 erweitert. In (b) bis (d) sind die drei Matchings in G_{n+1} dargestellt, die durch die Kanten in der Tabelle repräsentiert werden.

5.4.3 Detaillierter Algorithmus

In den beiden vorigen Abschnitten wurden zwei Einschränkungen eingeführt, nach denen Erweiterungen von Matchings berechnet werden:

1. Alte Tupel werden nur mit neuen Tupeln erweitert.

2. Alte Tupel, die denselben originalen SA-Wert besitzen, werden nicht mit demselben neuen Tupel erweitert.

Algorithmus 5.3 zeigt den detaillierten Ablauf der Konstruktion von Matchings für eine Folge von Anfragegraphen $\mathcal{G}^{(n+1)}$, wobei beide Einschränkungen umgesetzt sind. Er ist eine Verfeinerung des bereits vorgestellten Algorithmus 5.1. Dabei ist zu beachten, dass alle Δ-top Matchingkanten, die erweitert werden können, zunächst in einer Menge E_{alt} gesammelt werden. Entscheidend hierbei ist, dass nur die Kanten gespeichert werden, die mit einem neuen Tupel erweitert werden können. An dieser Stelle kann es daher auch zum Löschen von Kanten kommen (vgl. Zeile 18). Gleichzeitig werden alte Tupel, die in erweiterbaren Kanten vorkommen, in einer Menge S_{alt} gespeichert. Anschließend werden alle neuen Tupel, mit denen erweitert werden kann, zu einer Menge S_{neu} hinzugefügt.

Als Letztes werden die gesammelten Kanten in Zeile 32 mithilfe eines zusätzlichen Algorithmus erweitert. Dabei erfolgt für jeden SA-Wert s eines alten Tupels in S_{alt} ein Aufruf. $S_{\text{alt}}(s)$ sei hierbei die Teilmenge von S_{alt}, die nur Tupel mit SA-Wert s enthält. Die Bezeichnung $E_{\text{alt}}(s)$ wird analog verwendet. Zusätzlich werden folgende Bezeichnungen eingeführt.

Definition 5.3 *Sei E_Δ eine Menge von Δ-top Matchingkanten. Dann ist*

$$
\begin{aligned}
E_\Delta(t) &\coloneqq \{(t', s(S_T) \in E_\Delta) \mid t' = t\},\\
E_\Delta(s) &\coloneqq \{(t, s'(S_T) \in E_\Delta) \mid s' = s\},\\
E_\Delta(t, s) &\coloneqq \{(t', s'(S_T) \in E_\Delta) \mid t' = t, s' = s\}.
\end{aligned}
$$

Der Grund für die mehrfache Ausführung des Erweiterungsalgorithmus sind die beschriebenen Beschränkungen. Alle Tupel mit demselben SA-Wert werden in einer Erweiterung behandelt, wodurch sichergestellt werden kann, dass sie nicht mit demselben neuen Tupel erweitert werden. Matchingkanten, die dabei nicht erweitert wurden, werden zum Schluss gelöscht (siehe Zeile 34).

Sei t ein altes Tupel mit originalem SA-Wert s, welches in G_{n+1} vorkommt. Dann gilt zusammenfassend für jede Δ-top Matchingkante $e_\Delta = (t, s'(S_T))$ nach Algorithmus 5.3:

1. e_Δ wird gelöscht, wenn
   - in G_{n+1} kein SA-Knoten mit Label s' vorkommt oder
   - in G_{n+1} mindestens zwei Tupel aus S_T vorkommen oder
   - e_Δ nicht erweitert wird.
2. e_Δ bleibt erhalten, wenn
   - genau ein Tupel aus S_T in G_{n+1} vorkommt.
3. e_Δ kann erweitert werden, wenn
   - kein Tupel aus S_T in G_{n+1} vorkommt und
   - ein neues Tupel in G_{n+1} vorkommt, das den SA-Wert s' hat.

Im letzten Fall muss beim Erweitern sichergestellt werden, dass Tupel mit demselben SA-Wert nicht mit demselben neuen Tupel erweitert werden. Dieses Prinzip wird durch

Algorithmus 5.3 Berechne Δ-top Matchings

Eingabe: $\mathcal{T}_\Delta^{(n)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$, G_{n+1}: Graph
Ausgabe: $\mathcal{T}_\Delta^{(n+1)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$

$\mathcal{T}_\Delta^{(n+1)} \leftarrow \mathcal{T}_\Delta^{(n)}$
$S_{\text{alt}} \leftarrow \emptyset$ # Menge erweiterbarer alter Tupel
$E_{\text{alt}} \leftarrow \emptyset$ # Menge erweiterbarer Matchingkanten
$S_{\text{neu}} \leftarrow \emptyset$ # Menge neuer Tupel für Erweiterung
for all $t \in S_{\text{alt}}(G_{n+1})$ **do**
  for all Δ-top Matchingkanten $e_\Delta = (t, s'(S_T))$ in $\mathcal{T}_\Delta^{(n+1)}$ **do**
    if s' kommt nicht in G_{n+1} vor **then**
      `Lösche(`e_Δ`)`
    else if genau ein Tupel aus S_T kommt in G_{n+1} vor **then**
      continue # übertrage Kanten
    else if mindestens zwei Tupel aus S_T kommen in G_{n+1} vor **then**
      `Lösche(`e_Δ`)`
    else # kein Tupel aus S_T in G_{n+1}
      if $\exists t' \in S_{\text{neu}}(G_{n+1})$ mit SA-Wert s' **then**
        $S_{\text{alt}} \leftarrow S_{\text{alt}} \cup \{t\}$ # speichere für Erweiterung
        $E_{\text{alt}} \leftarrow E_{\text{alt}} \cup \{e_\Delta\}$ # speichere für Erweiterung
      else
        `Lösche(`e_Δ`)`
      end if
    end if
  end for
end for
for all $t \in S_{\text{neu}}(G_{n+1})$ **do**
  if $\exists e_\Delta \in E_{\text{alt}}$ mit SA-Wert von t **then**
    $S_{\text{neu}} \leftarrow S_{\text{neu}} \cup \{t\}$ # speichere für Erweiterung
  end if
end for
for all $t, t' \in S_{\text{neu}}(G_{n+1})$ mit verschiedenen SA-Werten **do**
  Erzeuge neues Matching mit Kanten $(t, s'(t'))$ und $(t', s(t))$, wobei s SA-Wert von t und s' SA-Wert von t'
end for
for all SA-Werte s von Tupeln aus S_{alt} **do**
  `Erweiterungsalgorithmus(`$S_{\texttt{alt}}(s), E_{\texttt{alt}}(s), S_{\texttt{neu}}$`)`
  for all Matchingkanten $e_\Delta \in E_{\text{alt}}(s)$, die nicht erweitert wurden, **do**
    `Lösche(`e_Δ`)`
  end for
end for
return $\mathcal{T}_\Delta^{(n+1)}$

sogenannte Erweiterungsalgorithmen umgesetzt, die zusätzlich bestimmte Vorgaben garantieren. Es entsteht ein weiteres Optimierungsproblem, welches ausführlich im Kapitel 6 besprochen wird. An dieser Stelle wird davon ausgegangen, dass Erweiterungen entsprechend den Einschränkungen in Polynomialzeit berechnet werden können.

Wenn in den Anfragegraphen n Tupel vorkommen, sind die Anzahlen der alten und neuen Tupel sowie der SA-Werte durch n beschränkt. Ebenfalls können in einer Δ-top Matchingkante nicht mehr als n Tupel vorkommen. Das bedeutet, der einzige Schritt in Algorithmus 5.3, der eventuell nicht in Polynomialzeit geht, ist die Iteration über alle Δ-top Matchingkanten (Zeile 6), wodurch auch der Algorithmus *Lösche* keine polynomielle Laufzeit hätte. Theoretisch können in Δ-top Matchingkanten sämtliche Kombinationen von Tupeln vorkommen, wodurch die Anzahl der Kanten exponentiell in n ist. Das Beispiel in Abbildung 4.12 auf Seite 118 zeigt genau so einen Fall, bei dem insgesamt 2^n verschiedene Δ-top Matchings entstehen. Daher wird an dieser Stelle eine Beschränkung der Anzahl der Δ-top Matchingkanten eingeführt, die ein bestimmtes Tupel t und einen SA-Wert s enthalten. Sei E_Δ die Menge aller Δ-top Matchingkanten in einer Δ-top Matchingtabelle, dann gilt für alle t und s

$$|E_\Delta(t, s)| \leq c_\Delta, \tag{5.1}$$

wobei c_Δ eine Konstante ist. Daraus folgt, dass für jedes Tupel höchstens linear viele Matchingkanten berechnet werden.

$$|E_\Delta(t)| = \mathcal{O}(n) \tag{5.2}$$

Es sei vereinbart, dass diese Beschränkung durch die Erweiterungsalgorithmen umgesetzt wird, auch wenn sie dort nicht explizit erwähnt wird. Das liegt daran, dass sie für das jeweilige Konzept des entsprechenden Algorithmus nicht ausschlaggebend ist. Zusammenfassend folgt aus diesen Betrachtungen, dass Algorithmus 5.3 polynomielle Laufzeit hat.

5.5 Test auf k-assign Anonymität

Mithilfe von Algorithmus 5.3 kann approximativ überprüft werden, ob eine gegebene Folge von Anfragen $\mathcal{Q}^{(n)}$ mit Ergebnissen $\mathcal{R}^{(n)}$ eine Verletzung der Privatsphäre darstellt oder nicht. Sei k-assign Anonymität das Privatsphärenkriterium, dann muss getestet werden, ob für jedes Tupel t mindestens k SA-Werteverteilungen existieren, die t jeweils einen anderen SA-Wert zuordnen. Für die dazugehörige Folge von Anfragegraphen $\mathcal{G}^{(n)}$ bedeutet das, dass für jedes t insgesamt k Matchings existieren, bei denen der Matchingpartner von t jeweils ein anderes SA-Label hat.

Ausgehend von einer leeren Δ-top Matchingtabelle wird für $\mathcal{Q}^{(1)}, \mathcal{Q}^{(2)}, \ldots, \mathcal{Q}^{(n)}$ jeweils eine Menge von Matchings berechnet und gespeichert. Interessanterweise müssen dafür die Anfragegraphen $\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \ldots, \mathcal{G}^{(n)}$ nicht materialisiert werden, wenn Algorithmus 5.3 so modifiziert wird, dass er anstelle der Graphen die Ergebnisse der Anfragen als Eingabe erhält. Nach jeder Bearbeitung einer weiteren Anfrage wird getestet, ob für alle Tupel mindestens k Δ-top Matchingkanten existieren, die jeweils einen anderen SA-Wert enthalten. Ist das nicht der Fall, wird eine Verletzung der Privatsphäre ausgegeben. Dabei kann es passieren, dass durch die Beschränkungen des Approximationsalgorithmus weniger Zuordnungen von SA-Werten zu Tupeln modelliert werden als real vorkommen. Demzufolge

kann eine Verletzung ausgegeben werden, obwohl keine existiert. In jedem Fall wird aber jede tatsächliche Verletzung auch gefunden.

5.6 k-assign Anonymität durch Hinzufügen künstlicher sensibler Werte

Neben dem bisher beschriebenen Ansatz, Verletzungen des Privatsphärenkriteriums nur zu erkennen und bei Bedarf auszugeben, gibt es eine weitere Möglichkeit, den Schutz der Privatsphäre zu gewährleisten. Dabei sind die Fälle von Bedeutung, bei denen eine Verletzung gefunden wird. Der bisherige Ansatz würde eine solche Anfrage verweigern und kein Ergebnis liefern. Eine andere Variante ist, ein Ergebnis zu der Anfrage auszugeben, welches das Schutzkriterium nicht verletzt. Offensichtlich muss dazu das korrekte Ergebnis R_n so zu einem R'_n verändert werden, dass die Folge von Ergebnissen $\mathcal{R}'^{(n)} = (R'_1, \ldots, R'_n)$ das Privatsphärenkriterium erfüllt.

k-assign Anonymität ist verletzt, wenn es mindestens ein Tupel t gibt, dessen Anonymitätsgrad kleiner als k ist. Dieser Fall kann auf zwei verschiedene Weisen eintreten. Erstens kann t ein altes Tupel sein, bei dem mindestens ein Matching gelöscht wird. Die zweite Möglichkeit ist, dass t ein neues Tupel ist, dem nicht genügend SA-Werte zugeordnet werden können. Der in der vorliegenden Arbeit verfolgte Ansatz ist, in beiden Fällen künstliche sensible Werte[12] in das Ergebnis hinzuzufügen. Die entsprechenden Anfragegraphen bekommen dadurch zusätzliche SA-Knoten, wodurch insgesamt mehr valide Matchings möglich sind.

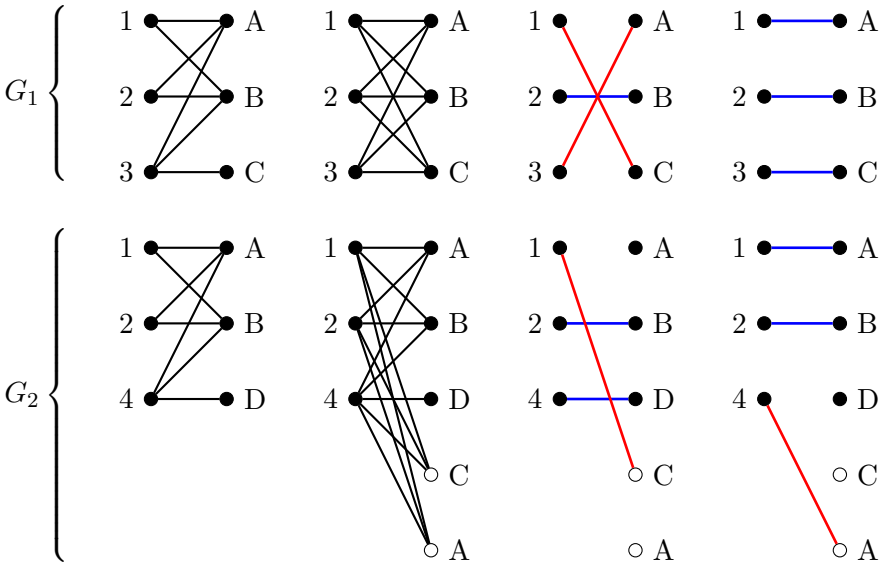
Sei k-assign Anonymität mit $k = 3$ gefordert. Abbildung 5.10 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2)$. Die berechneten Matchings in G_1 sind in der Matchingtabelle in (e) aufgelistet (Spalte $\mathcal{G}^{(1)}$). Da jedes Tupel mit drei verschiedenen SA-Werten matchen kann, erfüllt $\mathcal{T}_\Delta^{(1)}$ 3-assign Anonymität. Durch das Hinzufügen von G_2 müssen Matchings gelöscht werden, denn der SA-Wert C kommt nicht in G_2 vor. Daher werden die Δ-top Matchingkanten $(1, \mathrm{C}(3))$, $(2, \mathrm{C}(3))$, $(3, \mathrm{A}(1))$ und $(3, \mathrm{B}(2))$ mithilfe von Algorithmus 5.3 aus der Tabelle entfernt. $\mathcal{T}_\Delta^{(2)}$ (Spalte $\mathcal{G}^{(2)}$) verletzt nun 3-assign Anonymität sowohl für die alten Tupel 1, 2 und 3 als auch für das neue Tupel 4.

In (b) beziehungsweise Spalte $\mathcal{G}'^{(2)}$ der Tabelle ist bereits die Lösung dargestellt. Es werden insgesamt zwei künstliche SA-Knoten in G_2 eingefügt, sodass Matchings erhalten bleiben und neue Matchings entstehen. Wie und warum genau diese Werte erzeugt werden, wird im Folgenden beschrieben.

5.6.1 Verletzung bei alten Tupeln

Eine Verletzung von k-assign Anonymität kann bei einem alten Tupel nur dann vorliegen, wenn Matchings beziehungsweise Δ-top Matchingkanten aus der Matchingtabelle entfernt werden. Algorithmus 5.3 verwendet dafür die Subroutine *Lösche*, die an insgesamt vier Stellen[13] angerufen wird. Jeweils wird eine übergebene Kante inklusive aller Gegenkanten entfernt (vgl. Algorithmus 5.2). Dabei können Zuordnungen von SA-Werten zu mehreren Tupeln entfernt werden, welche wiederum zu einer Verletzung des Schutzkriteriums führen können. Das kann verhindert werden, indem in den ursprünglichen Graphen SA-Knoten

[12] Bei Xiao und Tao [162] werden diese Werte *Fälschungen* (engl. *counterfeits*) genannt.
[13] Zeilen 8, 12, 18 und 34



(a) $\mathcal{G}$ (b) $\mathcal{G}'$ (c) $\mathcal{M}_1$ (d) $\mathcal{M}_2$

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}'^{(2)}$
1	A	B(2), C(3)	B(2)	B(2), C(3)
2	B	A(1), C(3)	A(1)	A(1), C(3)
3	C	A(1), B(2)	–	A(1), B(2)
4	D		–	C(), A()

(e) Δ-top Matchingtabelle $\mathcal{T}_\Delta$

Abbildung 5.10: Anfragegraphen mit künstlichen SA-Knoten

eingefügt werden, die mit dem SA-Werten gelabelt sind, deren Fehlen für die Löschungen verantwortlich war.

Im Beispiel in Abbildung 5.10 fehlt in G_2 der SA-Wert C. Dadurch werden die Kanten $(1, \mathrm{C}(3))$ und $(2, \mathrm{C}(3))$ sowie die Gegenkanten $(3, \mathrm{A}(1))$ und $(3, \mathrm{B}(2))$ aus der Tabelle entfernt.[14] Bei den alten Tupeln 1, 2 und 3 entstehen Verletzungen von 3-assign Anonymität. Wird im Graphen G_2 ein SA-Knoten mit Label C eingefügt, gibt es ein Δ-top Matching $\mathcal{M}_1$, das die Kanten $(1, \mathrm{C}(3))$ und $(3, \mathrm{A}(1))$ enthält (siehe (c)). Dieses Matching ist zwar nicht perfekt, da in G_2 die beiden Knoten mit dem SA-Wert A keinen Matchingpartner haben. Es ist aber nach Definition 3.9 valid, denn alle Tupelknoten sind überdeckt. Da auf analoge Weise ein Matching konstruiert werden kann, das die Kanten $(2, \mathrm{C}(3))$ und $(3, \mathrm{B}(2))$ enthält, bleiben alle vier zuvor gelöschten Kanten in der Matchingtabelle erhalten.

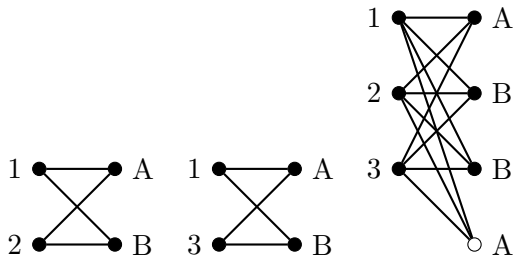
Die beschriebene Idee wird im Algorithmus 5.3 umgesetzt, indem eine modifizierte Version der Subroutine *Lösche* verwendet wird. Diese testet, ob durch das Löschen der übergebenen Matchingkante eine Verletzung von k-assign Anonymität eintritt. Wenn das der Fall ist, werden alle eventuell im Rekursionsschritt bereits durchgeführten Löschungen rückgängig gemacht. Anstatt die von Algorithmus 5.3 übergebene Matchingkante $e_\Delta = (t, s'(S_T))$

[14] Vgl. Zeile 8 in Algorithmus 5.3.

zu entfernen, wird ein zusätzlicher SA-Knoten mit dem Label s' in den aktuellen Graphen eingefügt. Matchings mit der Kante e_Δ können so erhalten bleiben.

Weiterhin stellt die modifizierte Version von *Lösche* sicher, dass die eingefügten SA-Knoten „wiederverwendet“ und nicht jedes Mal neu eingefügt werden. Beispielsweise erfolgt jeweils ein Löschaufruf in Abbildung 5.10 für die beiden Matchingkanten $(1, \mathrm{C}(3))$ und $(2, \mathrm{C}(3))$, da sowohl Tupel 1 als auch 2 ein Matching mit dem SA-Wert C benötigen. Insgesamt reicht es in diesem Fall aus, nur einen weiteren SA-Knoten einzufügen. Damit Algorithmus 5.3 aber weiterhin korrekte Δ-top Matchings berechnet, müssen auch für künstliche SA-Knoten die zuvor eingeführten Einschränkungen gelten. Folglich dürfen alte Tupel, die denselben SA-Wert haben, nicht mit demselben künstlichen SA-Knoten erweitert werden. Hätte also Tupel 2 denselben originalen SA-Wert wie Tupel 1, müsste für beide Tupel jeweils ein zusätzlicher Knoten mit SA-Wert C erstellt werden.

Eine letzte Besonderheit liegt vor, wenn eine Matchingkante aufgrund einer Verletzung von Δ-Pfadkonformität gelöscht werden soll und dadurch k-assign Anonymität nicht mehr erfüllt ist.[15] In diesem Fall wird der für die Kante $e_\Delta = (t, s'(S_T))$ erstellte SA-Knoten nicht wie oben beschrieben mit s', sondern mit dem originalen SA-Wert von t gelabelt. Er stellt nämlich keinen möglichen Matchingpartner von t, sondern einen für einen Knoten aus S_T dar. Insbesondere müssen in diesem Schritt mehrere SA-Knoten eingefügt werden, wenn in S_T mehr als zwei Knoten vorkommen, die auch im aktuellen Graphen vorhanden sind.



(a) G_1 (b) G_2 (c) G'_3

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}'^{(3)}$
1	A	B(2)	B(2, 3)	–	B(2, 3)
2	B	A(1)	A(1)	–	A(1)
3	B		A(1)	–	A(1)

(d) Δ-top Matchingtabelle

Abbildung 5.11: Verletzung der Δ-Pfadkonformität durch künstliche SA-Knoten

In Abbildung 5.11 ist eine Folge von drei Anfragegraphen gegeben. Ohne die Verwendung von künstlichen SA-Knoten müssten nach der Bearbeitung von G_3 alle Δ-top Matchingkanten entfernt werden (vgl. Spalte $\mathcal{G}^{(3)}$ in (d)). Die Kante, die zu diesen Löschungen führt, ist $(1, \mathrm{B}(2, 3))$, denn die beiden Tupel 2 und 3 kommen in G_3 vor. Demzufolge würde ein eventuelles Matching mit dieser Kante Δ-Pfadkonformität verletzen.[16] Durch das Hinzufügen eines künstlichen SA-Knotens mit Label A existiert allerdings ein Matching in $\mathcal{G} = (G_1, G_2, G_3)$, in dem Tupel 1 mit B sowie die Tupel 2 und 3 jeweils mit A matchen. In G_3 ist dazu der künstliche SA-Knoten ein Matchingpartner von 2 oder 3. Dieses Matching bildet eine Ausnahme bei der Approximation, denn es wird zwar modelliert, verstößt aber weiterhin gegen Δ-Pfadkonformität. Da die Verletzung jedoch nur den Knoten betrifft, der mit dem künstlichen SA-Knoten matcht, kann es in Algorithmus 5.3 wie ein normales Δ-top Matching verwendet werden.

[15] Vgl. Zeile 12 in Algorithmus 5.3.
[16] Vgl. Definition 4.7 auf Seite 120.

5.6.2 Verletzung bei neuen Tupeln

Eine Verletzung des Privatsphärenkriteriums liegt bei einem neuen Tupel t vor, wenn nicht genügend Matchingpartner gefunden werden. Das passiert, wenn insgesamt zu wenige Tupel im Anfragegraphen vorkommen oder die potentiellen Partner nicht mit dem originalen SA-Wert von t matchen können. Letzteres ist in Abbildung 5.10 für das neue Tupel 4 der Fall. In G_2 können die beiden alten Tupel 1 und 2 nicht mit dem SA-Wert D matchen, wodurch es keine Δ-top Matchings gibt, die eine Kante $(4, \mathrm{A})$ oder $(4, \mathrm{B})$ enthalten. Analog zur Verletzung bei alten Tupeln werden auch in diesem Fall künstliche SA-Knoten in den Graphen eingefügt. Dabei ist zu beachten, dass bereits ein zusätzlicher Knoten mit Label C in G_2 enthalten ist und somit auch ein Matching mit der Kante $(4, \mathrm{C})$ existiert. Da C ein künstlicher Wert ist, wird die entsprechende Δ-top Matchingtabelle mit $(4, \mathrm{C}())$ bezeichnet. Damit der Anonymitätsgrad von Tupel 4 mindestens 3 beträgt, wird noch ein weiterer SA-Knoten mit Label A eingefügt (vgl. Matching $\mathcal{M}_2$ in (d)). Der SA-Wert kann prinzipiell beliebig gewählt werden, da für Tupel 4 noch keine Matchings existieren. Es ist aber sinnvoll, Werte zu nehmen, die bereits in der Anfrage vorkommen und dadurch semantisch mit dem neuen Tupel assoziiert werden können. Anstelle von A hätte demzufolge auch B verwendet werden können.

5.6.3 Ergebnisse mit künstlichen sensiblen Werten

Auf eine detaillierte Angabe der veränderten *Lösche*-Routine wird an dieser Stelle verzichtet. Beispiel A.12 im Anhang auf Seite 282 erläutert die Erweiterung von Algorithmus 5.3 und das Hinzufügen künstlicher SA-Knoten anhand eines komplexen Anfragegraphen.

Die entstehenden Graphen entsprechen nicht dem realen Ergebnis einer Anfrage, sondern enthalten zusätzliche SA-Knoten. Sie werden in der vorliegenden Arbeit der Einfachheit halber *erweiterter Anfragegraph* genannt und mit einem Strich gekennzeichnet. Ein Ergebnis, das entsprechend einem so konstruierten Graphen ausgegeben wird, enthält dieselben künstlichen SA-Werte, die als Labels im Graphen vorkommen. Das gewährleistet, dass die Zuordnungen von sensiblen Werten zu Individuen, die durch die berechneten Matchings abgebildet werden, auch durch das Ergebnis mithilfe von SA-Werteverteilungen konstruiert werden können.

ID	SA
1	
2	A, B, D
4	

(a) Reales Ergebnis R_2

ID	SA
1	
2	A, A, B, C, D
4	

(b) Ergebnis R_2' (inkl. künstlicher SA-Werte)

Abbildung 5.12: Reales und zurückgegebenes Ergebnis für die Anfrage Q_2 (vgl. G_2 und G_2' in Abbildung 5.10)

Sei Q_2 die Anfrage, welche die Tupel 1, 2 und 4 mit ihren SA-Werten A, B und D selektiert und für die der Anfragegraph G_2 aus Abbildung 5.10 erstellt wurde. Abbildung 5.12a zeigt das dazugehörige reale Ergebnis R_2 und Abbildung 5.12b das zurückgegebene Ergebnis R_2'. Dabei wurde die vereinfachte Darstellung von Ergebnissen aus Kapitel 3.2 verwendet, welche leicht in eine Generalisierung oder Bucketisierung (vgl. 2.2.3) überführt werden kann.

Zusammenfassung

In diesem Kapitel wurde ein Algorithmus vorgestellt, der valide Matchings in Anfragegraphen berechnet. Mit seiner Hilfe können die Probleme QGM und SA-Werteverteilung approximiert werden. Die Grundidee lautet, erstens nur Δ-top Matchings zu betrachten und zweitens diese rekursiv aus bereits zuvor bestimmten Matchings zu konstruieren. Für eine gegebene Folge von n Anfragegraphen werden somit Matchings aus denen für $n-1$ Graphen abgeleitet. Dieser Schritt wird Erweiterung von Matchings genannt und bringt zwei zusätzliche Einschränkungen mit sich. Zum einen dürfen sogenannte alte Tupel nur mit neuen Tupeln erweitert werden und zum anderen dürfen jeweils zwei alte Tupel, die denselben originalen SA-Wert haben, nur mit verschiedenen neuen Tupeln erweitert werden. Dadurch ist gewährleistet, dass die Berechnung in Polynomialzeit durchgeführt werden kann, wobei detaillierte Erweiterungsalgorithmen erst im Kapitel 6 vorgestellt werden.

In Kapitel 3 wurde das Modell der k-assign Anonymität definiert, welche zur Bestimmung von Verletzungen der Privatsphäre dient. Mithilfe des hier eingeführten Approximationsalgorithmus wurde ein Verfahren angegeben, das die Einhaltung dieses Schutzkriteriums bei einer gegebenen Folge von Anfragen inklusive Ergebnissen überprüft und insbesondere alle Fälle findet, in denen es nicht erfüllt ist. Darüber hinaus wurde eine Methode eingeführt, bei der immer eine Antwort auf eine Anfrage ausgegeben werden kann. Wenn das korrekte Ergebnis zu einer Verletzung von k-assign Anonymität führen würde, werden zusätzlich künstliche SA-Werte hinzugefügt. Damit wird garantiert, dass kein Bruch des Schutzkriteriums vorliegt.

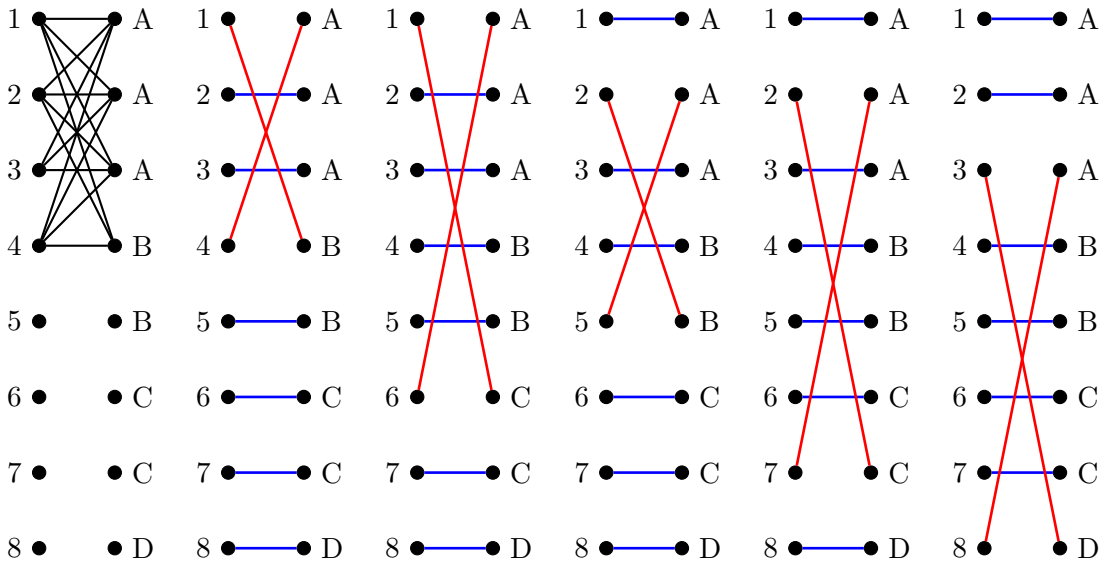
6 Erweiterungsalgorithmen

Dieses Kapitel schließt inhaltlich direkt an Kapitel 5 an und stellt den dort eingeführten Erweiterungsschritt von Matchings detailliert vor. Es werden Kriterien eingeführt, die Erweiterungen erfüllen sollen, damit die entstehende Approximation möglichst genau ist (Kapitel 6.1). Darauf aufbauend wird die Erweiterung von Matchings als Optimierungsproblem definiert (Kapitel 6.2) und es werden Algorithmen vorgestellt, die dieses Problem bestmöglich lösen (Kapitel 6.3 bis 6.6).

6.1 Gute und schlechte Erweiterungen

In Kapitel 5.4 wurden für das Erweitern von Δ-top Matchings zwei Einschränkungen eingeführt, von denen eine bereits durch Algorithmus 5.3 umgesetzt wird: Matchingkanten beziehungsweise alte Tupel werden nur mit neuen Tupeln erweitert. Die zweite Restriktion besagt, dass alte Tupel, die denselben originalen SA-Wert haben, nicht mit demselben neuen Tupel erweitert werden dürfen. Sie wird durch spezielle Erweiterungsalgorithmen realisiert, die gleichzeitig versuchen, möglichst „gute“ Erweiterungen durchzuführen. Beispielsweise können Erweiterungen sehr einfach erzeugt werden, indem Δ-top Matchingkanten von alten Tupeln sequenziell neue Tupel zugewiesen werden, wenn sie mit diesen erweitert werden können. Das ist der Fall, wenn der SA-Wert des neuen Tupels in der Matchingkante vorkommt. Man spricht in diesem Zusammenhang auch von einer *Tupelverteilung*, womit die Zuordnung der neuen Tupel zu den alten gemeint ist. Ein neues Tupel kann dabei auch mehreren alten zugewiesen werden, wenn diese verschiedene originale SA-Werte haben. Dieses Vorgehen kann allerdings schnell zu „besseren“ oder „schlechteren“ Erweiterungen führen, wenn die Zuordnungen beliebig durchgeführt werden. Das folgende Beispiel veranschaulicht, was intuitiv unter einer „schlechten“ Erweiterung zu verstehen ist.

Beispiel 6.1 Abbildung 6.1 zeigt ein Beispiel für mögliche Erweiterungen. In (a) ist ein Anfragegraph G_{n+1} gegeben, der aus den alten Tupeln 1, 2, 3 und 4 sowie den neuen Tupeln 5, 6, 7 und 8 besteht. Damit der Graph nicht unübersichtlich wird, sind Kanten nur zwischen alten Knoten gezeichnet. Fehlende Kanten können leicht aus den gespeicherten Matchings abgelesen werden, die in Spalte $\mathcal{G}^{(n)}$ der Δ-top Matchingtabelle aufgelistet sind (siehe (g)). Hierbei steht jedes x_i für eine Menge (bzw. Liste) beliebiger Tupel, die jeweils nicht in G_{n+1} vorkommen. Beispielsweise hat Tupel 2 zwei unterschiedliche Matchingkanten mit einem C, da alle x_i zueinander verschieden sind. Alle neun Kanten mit einem x_i sollen erweitert werden. Insgesamt gibt es aber nur vier neue Tupel und da 1, 2 und 3 jeweils denselben SA-Wert haben, dürfen sie nach den Einschränkungen nicht mit denselben Tupeln erweitert werden. Beispielsweise können aber die Kanten $(2, \mathrm{C}(x_{2.3}))$ und $(2, \mathrm{C}(x_{2.4}))$ mit demselben neuen Tupel erweitert werden, da sie zum selben alten Tupel 2 gehören.



(a) G_{n+1} (b) M_1 (c) M_2 (d) M_3 (e) M_4 (f) M_5

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$\mathcal{G}_{v2}^{(n+1)}$
1	A	B(4), B($x_{1.1}$), C($x_{1.2}$), D($x_{1.3}$)	B(4), B($x_{1.1}$, 5), C($x_{1.2}$, 6), D($x_{1.3}$, 8)	B(4), C($x_{1.2}$, 6)
2	A	B($x_{2.1}$), B($x_{2.2}$), C($x_{2.3}$), C($x_{2.4}$), D($x_{2.5}$)	C($x_{2.3}$, 7), C($x_{2.4}$, 7)	B($x_{2.1}$, 5), B($x_{2.2}$, 5), C($x_{2.3}$, 7), C($x_{2.4}$, 7)
3	A	D(x_3)	–	D(x_3, 8)
4	B	A(1)	A(1)	A(1)
5	B		A(1)	A(2)
6	C		A(1)	A(1)
7	C		A(2)	A(2)
8	D		A(1)	A(3)

(g) Δ-top Matchingtabelle

Abbildung 6.1: Verschiedene Erweiterungen für Matchings für $\mathcal{G}^{(n)}$. Im Graphen G_{n+1} in (a) sind der Übersicht halber nur die Kanten zwischen alten Knoten gezeichnet. In (g) sind die Matchingtabellen für zwei verschiedene Erweiterungen ($\mathcal{G}_{v1}^{(n+1)}$ und $\mathcal{G}_{v2}^{(n+1)}$) angegeben. Die Matchings der zweiten Variante sind graphisch in (b) bis (f) dargestellt.

Eine mögliche Erweiterung ist in Spalte $\mathcal{G}_{v1}^{(n+1)}$ angegeben. Tupel 1 ist mit 5, 6 und 8 erweitert, Tupel 2 nur mit 7 und für Tupel 3 bleibt kein neues Tupel zum Erweitern übrig. Während Tupel 1 jetzt vier verschiedene SA-Werte zugeordnet werden können (A, B, C, D), hat Tupel 2 nur zwei (A, C) und 3 sogar nur noch seinen originalen SA-Wert (A). Der minimale Anonymitätsgrad bezüglich dieser Erweiterung ist demnach $a_{\min} = a(3) = 1$, wodurch in jedem Fall eine Verletzung des Schutzkriteriums vorliegt.

In Spalte $\mathcal{G}_{v2}^{(n+1)}$ ist eine andere Variante der Erweiterungen gegeben, die auch graphisch in (b) bis (f) dargestellt ist. Tupel 3 wurde hier mit 7, 1 mit 6 und 2 mit 5 sowie 7 erweitert. Ein Vergleich der Anonymitätsgrade beider Fälle zeigt, dass der maximale Wert hier kleiner ist (jetzt 3, vorher 4) und der minimale größer (jetzt 2, vorher 1). Insgesamt führen die Erweiterungen bei dieser Variante zu einer gleichmäßigeren Verteilung der Anonymitätsgrade und eventuell zu keiner Verletzung der Privatsphäre.[1] Insgesamt sind sechs Kanten erweitert worden, während bei der ersten Variante nur fünf Kanten erweitert wurden. Intuitiv ist dadurch der zweite Fall besser als der erste, denn es werden insgesamt mehr Kanten erweitert und der minimale Anonymitätsgrad ist größer.

Im Folgenden werden Kriterien definiert, nach denen entschieden werden kann, wann eine gegebene Menge von Erweiterungen besser ist als eine andere. Dabei wird die Menge der neuen Tupel auf die Menge der alten abgebildet, wobei ein Element (t, t') besagt, dass alle Matchingkanten $(t, s'(S_T))$, in denen der originale SA-Wert s' des neuen Tupels t' vorkommt, mit t' erweitert werden. Die Menge dieser Paare heißt *Tupelverteilung*. Das Problem der Erweiterung wird damit im Folgenden als Optimierungsproblem dargestellt und es werden insgesamt vier sogenannte *Erweiterungsalgorithmen* vorgestellt, die eine optimale Tupelverteilung berechnen.

6.2 Erweiterung als Optimierungsproblem

Bei der Erweiterung von Δ-top Matchingkanten beziehungsweise Tupelverteilung sind eine Menge von alten und eine Menge von neuen Tupeln sowie eine Menge von Δ-top Matchingkanten der alten Tupel gegeben. Gesucht ist eine Zuordnung der neuen zu den alten Tupeln,[2] wobei ein neues Tupel einem alten nur dann zugeordnet werden kann, wenn es auch eine Matchingkante mit dem alten Tupel gibt, in der der SA-Wert des neuen Tupels vorkommt. Weiterhin darf jedes neue Tupel nicht zwei alten Tupeln zugeordnet werden, die denselben originalen SA-Wert haben. Wird daher eine Menge alter Tupel mit gleichem SA-Wert betrachtet, ist mathematisch gesehen eine Abbildung der Menge der neuen auf die Menge der alten Tupel gesucht.

Im Allgemeinen existieren mehrere mögliche Erweiterungen. Die bestmögliche Variante ist dabei eine Zuordnung, bei der alle Matchingkanten erweitert werden. In diesem Fall muss kein Matching gelöscht werden und der Anonymitätsgrad der Tupel verringert sich nicht. Somit kann auch keine (weitere) Verletzung des Schutzkriteriums entstehen. Das funktioniert aber nur, wenn genügend neue Tupel zum Erweitern vorhanden sind und darüber hinaus diese auch die benötigten SA-Werte haben. Eine Matchingkante $e_\Delta = (t, s'(S_T))$ muss andererseits auf jeden Fall entfernt werden, wenn kein neues Tupel den sensiblen Wert s' hat.

[1] Man beachte, dass der minimale Anonymitätsgrad aus Spalte $\mathcal{G}^{(n)}$ bereits 2 war.

[2] Mathematisch gesehen ist eine Relation zwischen der Menge der alten und der Menge der neuen Tupel gesucht.

Ist es nicht möglich, alle Matchingkanten zu erweitern, sollte eine Tupelverteilung gewählt werden, durch die möglichst wenige falsche Verletzungen der Privatsphäre (false positives) entstehen, damit die Güte der Approximation nicht zu gering wird. Das kann näherungsweise erreicht werden, indem zum Beispiel möglichst viele Kanten erweitert werden. Das Problem der Tupelerweiterung kann dadurch als einfaches Optimierungsproblem aufgefasst werden.

Problem 6.1 (Erweiterung/Tupelverteilung) *Gegeben seien eine Menge von alten Tupeln S_{alt}, die jeweils denselben SA-Wert haben, eine Menge von neuen Tupeln S_{neu} und eine Menge von Matchingkanten E_{alt}. Eine zulässige Lösung ist eine Abbildung $f\colon S_{neu} \to S_{alt}$ der neuen auf die alten Tupel, wobei ein neues Tupel einem alten nur dann zugeordnet werden kann, wenn es eine Matchingkante mit dem alten Tupel gibt, in der der SA-Wert des neuen Tupels vorkommt.*

Gesucht ist eine Abbildung f, bei der die Anzahl aller erweiterter Kanten aus E_{alt} maximal ist.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$\mathcal{G}_{v2}^{(n+1)}$
1	A	B($x_{1.1}$), B($x_{1.2}$), C($x_{1.3}$), C($x_{1.4}$)	B($x_{1.1}, 3$), B($x_{1.2}, 3$), C($x_{1.3}, 4$), C($x_{1.4}, 4$)	B($x_{1.1}, 3$), B($x_{1.2}, 3$)
2	A	B($x_{2.1}$), C($x_{2.2}$)	–	C($x_{2.2}, 4$)
3	B		A(1)	A(1)
4	C		A(1)	A(2)

Tabelle 6.1: Optimale Erweiterungen

Allerdings garantiert eine optimale Lösung von Problem 6.1 nicht zwangsläufig auch eine minimale Anzahl an (evtl. falschen) Verletzungen des Schutzkriteriums. In Tabelle 6.1 sind beispielsweise zwei alte Tupel 1 und 2 mit jeweiligen Matchingkanten zu den SA-Werten B und C sowie zwei neue Tupel 3 und 4 gegeben. Eine Erweiterung, die die meisten Kanten beinhaltet, ist in Spalte $\mathcal{G}_{v1}^{(n+1)}$ dargestellt. Dabei werden insgesamt vier Kanten erweitert und zwei gelöscht. Da Tupel 1 jeweils mehr Kanten mit den SA-Werten B und C hat als Tupel 2, werden beide Kanten von Tupel 2 nicht erweitert und gelöscht. Folglich entsteht hier eine Verletzung des Schutzkriteriums. Andererseits führt eine Erweiterung von nur zwei Kanten von 1 (z. B. mit B) und einer Kante von 2 (z. B. mit C) nicht zwangsläufig zu einer Verletzung (siehe Spalte $\mathcal{G}_{v2}^{(n+1)}$).[3]

Ist k-assign Anonymität gefordert, müssen alle alten Tupel insgesamt mindestens k Matchingkanten mit paarweise verschiedenen SA-Werten haben.[4] Um Fälle wie $\mathcal{G}_{v1}^{(n+1)}$ in Tabelle 6.1 auszuschließen, kann eine andere Zielfunktion für das Problem der Tupelverteilung gewählt werden. Gesucht sind dann Erweiterungen, bei denen alle alten Tupel mit möglichst vielen verschiedenen SA-Werten erweitert werden. Genauer gesagt soll der minimale Anonymitätsgrad der alten Tupel maximal sein.

[3] Hierbei sei vorausgesetzt, dass das Löschen von Matchings der Tupel in den Mengen x_i zu keiner Verletzung führt.

[4] Das ist ein notwendiges Kriterium, aber kein hinreichendes. Wenn auch nur eine Kante beziehungsweise ein Matching gelöscht wird, kann das Schutzkriterium bei einem Tupel verletzt sein, das in dem Matching vorkommt. Dieses Tupel muss nicht zwangsläufig auch in der aktuellen Anfrage vorkommen.

Problem 6.2 (Erweiterung/Tupelverteilung) *Gegeben seien S_{alt}, S_{neu}, E_{alt} und f wie bei Problem 6.1. Ferner bezeichne $a_{\min}(S_{alt})$ den minimalen Anonymitätsgrad eines alten Tupels $t \in S_{alt}$ nach der Erweiterung (d. h. in $\mathcal{T}_{\Delta}^{(n+1)}$).*

Gesucht ist eine Abbildung f, bei der der minimale Anonymitätsgrad der alten Tupel maximal ist:

$$\textit{maximiere } a_{\min}(S_{alt}). \tag{6.1}$$

Algorithmen, die Problem 6.2 optimieren, sollten zusätzlich versuchen, möglichst wenige Kanten zu streichen, denn durch jedes Löschen eines Matchings kann eine Verletzung des Schutzkriteriums entstehen. Weiterhin sollte der Anonymitätsgrad möglichst vieler alter Tupel mindestens k sein, wenn $a_{\min}(S_{\text{alt}}) < k$ gilt.

Beispiel 6.2 In Abbildung 6.1g sind in den Spalten $\mathcal{G}_{v1}^{(n+1)}$ und $\mathcal{G}_{v2}^{(n+1)}$ zwei mögliche Erweiterungen der Kanten dargestellt. Der minimale Anonymitätsgrad der alten Tupel beträgt in der ersten Variante 1 und in der zweiten 2. Die erste Variante ist demnach keine optimale Lösung von Problem 6.2. Da in der zweiten Variante mehr Kanten als in der ersten Variante erweitert wurden (sechs zu fünf), ist die erste Variante auch keine optimale Lösung von Problem 6.1.

Wichtig ist hierbei festzuhalten, dass optimale Lösungen der hier vorgestellten Probleme nicht gewährleisten, dass auch die Anzahl von Verletzungen des Schutzkriteriums minimiert wird. Es kann eine Tupelverteilung geben, die gegen die hier definierten Zielfunktionen verstößt, aber insgesamt zu weniger Verletzungen führt als die optimale Verteilung. Das liegt daran, dass das Löschen bestimmter Kanten immer zu Verletzungen des Schutzkriteriums führen kann, während andere gefahrlos gelöscht werden dürfen. Beispiel A.13 im Anhang auf Seite 285 zeigt einen solchen Fall. Daneben kann auch die Verwendung einer Tupelverteilung f, durch die für den n-ten Anfragegraphen mehr Verletzungen entstehen als bei einer anderen Verteilung f', dazu führen, dass für den $(n+1)$-ten Graphen eine Tupelverteilung existiert, die weniger Verletzungen verursacht als bei der Verwendung von f' (vgl. Beispiel A.14 im Anhang auf Seite 285). Aus ähnlichen wie im vorigen Kapitel genannten Gründen kann ein ausführlicher Test aller Kanten beziehungsweise Matchings oder das Betrachten aller möglicher Erweiterungen schnell zu exponentiellen Algorithmen führen. Andererseits gibt es polynomielle Algorithmen, die eine optimale Lösung der Probleme 6.1 beziehungsweise 6.2 berechnen. Auf einen Beweis, der zeigt, dass das Finden einer Erweiterung mit einer minimalen Anzahl von Verletzungen des Schutzkriteriums in Polynomialzeit möglich oder NP-hart ist, wird an dieser Stelle verzichtet.

Im Folgenden werden Algorithmen vorgestellt, die eine Verteilung der neuen auf die alten Tupel berechnen. Alle Verfahren haben die Gemeinsamkeit, dass die ideale Verteilung (d. h., alle Kanten können erweitert werden) auch gefunden wird, sofern sie existiert. Müssen aber Kanten gelöscht werden, findet Algorithmus *Maximale Kanten* ein Optimum für das Problem 6.1, während Algorithmus *Klonmatching* ein Optimum für das Problem 6.2 berechnet. Daneben werden noch die beiden einfachen Algorithmen *Maximaler Anonymitätsgrad* und *Einfaches Matching* vorgestellt, die das Optimum des Problems 6.2 approximieren. Es handelt sich dabei um einfache und dafür schnelle Heuristiken.

6.3 Erweiterungsalgorithmus: Maximale Kanten

Ein einfacher Erweiterungsalgorithmus ordnet ein neues Tupel demjenigen alten Tupel zu, das die meisten Kanten mit dem SA-Wert des neues Tupels besitzt. Auf diese Weise wird eine Erweiterung erzeugt, die die Anzahl aller erweiterter Kanten maximiert (vgl. Problem 6.1).

Algorithmus 6.1 Erweiterungsalgorithmus: Maximale Kanten

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel
Ausgabe: Erweiterung von alten Tupeln (Tupelverteilung)

1: **for all** SA-Werte s aus E_{alt} **do**
2: Erstelle Liste aller alten Tupel $t_{\text{alt}} \in S_{\text{alt}}$ mit $|E_{\text{alt}}(t_{\text{alt}}, s)| > 0$, die absteigend nach $|E_{\text{alt}}(t_{\text{alt}}, s)|$ sortiert ist
3: **end for**
4: **for all** $t_{\text{neu}} \in S_{\text{neu}}$ **do**
5: $t_{\text{alt}} \leftarrow$ erstes Tupel in Liste für SA-Wert von t_{neu}
6: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$ # entspricht $f(t_{\text{neu}}) = t_{\text{alt}}$
7: Verschiebe t_{alt} vom Anfang der Liste ans Ende
8: **end for**

Algorithmus 6.2 Erweitere

Eingabe: t_{alt}: altes Tupel, t_{neu}: neues Tupel, E_{alt}: Menge erweiterbarer Matchingkanten
Ausgabe: Erweiterung aller möglicher Kanten von t_{alt} mit t_{neu}

1: **for all** $e_\Delta = (t_{\text{alt}}, s_{\text{neu}}(S_T)) \in E_{\text{alt}}$, wobei s_{neu} SA-Wert von t_{neu} ist, **do**
2: Erzeuge erweiterte Kante zu e_Δ durch das Hinzufügen von t_{neu} zur Menge S_T
3: Erstelle neue Kante $(t_{\text{neu}}, s_{\text{alt}}(\{t_{\text{alt}}\}))$, wobei s_{alt} SA-Wert von t_{alt} ist
4: Füge beide Kanten in aktuelle Matchingtabelle ein
5: **end for**

Das entsprechende Vorgehen ist in Algorithmus 6.1 gegeben und heißt *Maximale Kanten*. Zunächst werden für jeden SA-Wert s Listen der alten Tupel erstellt, die absteigend nach Häufigkeit von s in Matchingkanten der alten Tupel sortiert sind. Danach wird für jedes Tupel $t_{\text{neu}} \in S_{\text{neu}}$ (mit SA-Wert s_{neu}) das oberste Tupel t_{alt} derjenigen Liste ausgewählt, die für den SA-Wert s_{neu} erstellt wurde. t_{alt} entspricht dem alten Tupel, das die meisten erweiterbaren Kanten mit s_{neu} hat, und folglich werden diese Kanten auch mit t_{neu} erweitert (vgl. Algorithmus 6.2 (*Erweitere*)). Dadurch entsteht die gesuchte Abbildung f mit $f(t_{\text{neu}}) = t_{\text{alt}}$. Zum Schluss wird t_{alt} ans Ende der Liste verschoben, damit für das nächste neue Tupel mit SA-Wert s_{neu} das alte Tupel mit den zweitmeisten Kanten mit s_{neu} ausgewählt wird. Somit wird jedes neue Tupel mit demselben SA-Wert immer einem anderen alten Tupel zugeordnet. Wurden die Kanten aller alten Tupel einmal erweitert, beginnt der Auswahlprozess von vorn mit dem Tupel, das nun die meisten Kanten mit dem gewünschten SA-Wert hat.

Satz 6.1 *Algorithmus Maximale Kanten findet eine Lösung von Problem 6.1 in* $\mathcal{O}(n^2 \cdot \log n)$*, wobei* n *die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Bezeichne $|S_{\text{neu}}[\text{SA}]|$ die Anzahl verschiedener SA-Werte in S_{neu}. Dann gibt es $|S_{\text{alt}}|$ alte Tupel, die in $|S_{\text{alt}}| \log |S_{\text{alt}}|$ für alle SA-Werte sortiert werden können. Für jedes neue Tupel werden in einem Erweiterungsschritt höchstens $\max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|$ Kanten erweitert, sodass eine Laufzeit von $\mathcal{O}(|S_{\text{neu}}[\text{SA}]| \cdot |S_{\text{alt}}| \log |S_{\text{alt}}| + |S_{\text{neu}}| \cdot \max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|)$ entsteht. Wird die Anzahl der alten und neuen Tupel mit der Anzahl aller Tupel in der Datentabelle n und $|E_{\text{alt}}(t_{\text{alt}})|$ nach Gleichung 5.2 abgeschätzt, folgt eine Laufzeit von $\mathcal{O}(n \cdot n \log n + n \cdot n) = \mathcal{O}(n^2 \log n)$. □

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B($x_{1.1}$), B($x_{1.2}$), C($x_{1.3}$), C($x_{1.4}$), D($x_{1.5}$)	B($x_{1.1}, 4$), B($x_{1.2}, 4$), B($x_{1.1}, 6$), B($x_{1.2}, 6$), C($x_{1.3}, 8$), C($x_{1.4}, 8$), –
2	A	B($x_{2.1}$), C($x_{2.2}$), C($x_{2.3}$), C($x_{2.4}$), D($x_{2.5}$), D($x_{2.6}$)	B($x_{2.1}, 5$), C($x_{2.2}, 7$), C($x_{2.3}, 7$), C($x_{2.4}, 7$), –
3	A	C($x_{3.1}$), D($x_{3.2}$), D($x_{3.3}$), D($x_{3.4}$)	C($x_{3.1}, 9$), D($x_{3.2}, 10$), D($x_{3.3}, 10$), D($x_{3.4}, 10$)
4–6	B		A(1)/A(2)/A(1)
7–9	C		A(2)/A(1)/A(3)
10	D		A(3)

Tabelle 6.2: Erweiterungen nach Algorithmus 6.1 (*Maximale Kanten*)

Beispiel 6.3 Seien 1, 2 und 3 drei alte Tupel, die zusammen mit sieben neuen Tupeln in G_{n+1} vorkommen. Tabelle 6.2 zeigt die entsprechende Δ-top Matchingtabelle mit den erweiterbaren Matchingkanten. Der Einfachheit halber sind die neuen Tupel 4 bis 10 nach SA-Werten zusammengefasst. Nach Algorithmus 6.1 werden für die SA-Werte B, C und D drei sortierte Listen der alten Tupel erstellt. Die Liste für B enthält nur die Tupel 1 und 2 (in dieser Reihenfolge), denn 1 hat zwei Matchingkanten mit B, 2 nur eine und 3 gar keine. Dementsprechend werden die neuen Tupel mit SA-Wert B (Tupel 4, 5 und 6) zuerst Tupel 1, dann Tupel 2 und zum Schluss erneut Tupel 1 zugeordnet. Die Liste für den SA-Wert C enthält die Tupel 2, 1 und 3, wodurch die Zuordnung der Tupel 7, 8 und 9 bestimmt wird. Da es nur ein neues Tupel mit einem D gibt und Tupel 3 die meisten Kanten mit einem D hat, werden nur die Kanten von 3 mit einem D (Tupel 10) erweitert. Die resultierenden Matchingkanten für $\mathcal{G}^{(n+1)}$ sind in der letzten Spalte dargestellt.

6.4 Erweiterungsalgorithmus: Maximaler Anonymitätsgrad

Da Lösungen des Problems 6.1 zu mehr Verletzungen des Schutzkriteriums führen können als nötig, wird in der vorliegenden Arbeit auch ein Algorithmus angegeben, der eine Lösung des Problems 6.2 berechnet. Die Herausforderung dabei ist, eine Erweiterung zu berechnen, die den minimalen Anonymitätsgrad der alten Tupel maximiert. In diesem Abschnitt wird dazu zunächst eine einfache Heuristik eingeführt. Der Algorithmus, der das Problem exakt löst, wird in Kapitel 6.6 vorgestellt.

Algorithmus 6.3 Erweiterungsalgorithmus: Maximaler Anonymitätsgrad

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel

Ausgabe: Erweiterung von alten Tupeln (Tupelverteilung)

1: $E_{\text{Erw}} \coloneqq \{(t_{\text{alt}}, s(S_T)) \in E_{\text{alt}} \mid \exists t_{\text{neu}} \in S_{\text{neu}} \colon t_{\text{neu}}[\text{SA}] = s\}$
2: $E^*_{\text{Erw}} \coloneqq \{e_\Delta \in E_{\text{Erw}} \mid e_\Delta \text{ noch nicht erweitert}\}$
3: Für $t_{\text{alt}} \in S_{\text{alt}}$ sei $E_{\text{Erw}}(t_{\text{alt}}) \coloneqq E_{\text{Erw}} \cap E_{\text{alt}}(t_{\text{alt}})$ und $E^*_{\text{Erw}}(t_{\text{alt}}) \coloneqq E^*_{\text{Erw}} \cap E_{\text{alt}}(t_{\text{alt}})$
4: Sortiere Tupel $t_{\text{alt}} \in S_{\text{alt}}$ aufsteigend nach ihrem Erweiterungsgrad $d(t_{\text{alt}})$ bezüglich E^*_{Erw}
5: Sortiere Tupel $t_{\text{neu}} \in S_{\text{neu}}$ absteigend nach Häufigkeit ihres SA-Wertes in S_{neu}
6: **while** $S_{\text{neu}} \neq \emptyset$ und $E^*_{\text{Erw}} \neq \emptyset$ **do** # Phase 1
7: $t_{\text{alt}} \leftarrow$ erstes Tupel aus S_{alt} mit minimalem aktuellen Anonymitätsgrad
8: Betrachte dabei nur alte Tupel t mit $E^*_{\text{Erw}}(t) \neq \emptyset$
9: $t_{\text{neu}} \leftarrow$ erstes Tupel aus S_{neu}, dessen SA-Wert in einer Kante aus $E^*_{\text{Erw}}(t_{\text{alt}})$ vorkommt
10: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$
11: Entferne t_{neu} aus S_{neu}, aktualisiere E_{Erw}, E^*_{Erw} und die Sortierungen von S_{alt} sowie S_{neu}
12: **end while**
13: Sortiere Tupel $t_{\text{alt}} \in S_{\text{alt}}$ aufsteigend nach Ihrem Erweiterungsgrad $d(t_{\text{alt}})$ bezüglich E_{Erw}
14: **while** $S_{\text{neu}} \neq \emptyset$ **do** # Phase 2
15: $t_{\text{alt}} \leftarrow$ erstes Tupel aus S_{alt}, das in dieser Phase am seltensten erweitert wurde
16: Betrachte dabei nur alte Tupel t mit $E_{\text{Erw}}(t) \neq \emptyset$
17: $t_{\text{neu}} \leftarrow$ erstes Tupel aus S_{neu}, dessen SA-Wert in einer Kante aus $E_{\text{Erw}}(t_{\text{alt}})$ vorkommt
18: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$
19: Entferne t_{neu} aus S_{neu}, aktualisiere E_{Erw} sowie die Sortierungen von S_{alt} und S_{neu}
20: **end while**

Algorithmus 6.3 heißt *Maximaler Anonymitätsgrad* und erweitert alte Tupel in zwei Phasen. Das Ziel der ersten Phase lautet, wenn möglich jede Kante genau einmal zu erweitern. Dazu werden zwei Mengen E_{Erw} und E^*_{Erw} berechnet. E_{Erw} ist die Menge aller Δ-top Matchingkanten aus E_{alt}, die noch erweitert werden können. Das heißt, für diese Kanten gibt es ein neues Tupel in S_{neu}, das den SA-Wert der Kante hat. Für Phase 1 ist E^*_{Erw} die Teilmenge der Kanten aus E_{Erw}, die dabei noch nicht erweitert wurden.

Im Gegensatz zum Algorithmus *Maximale Kanten* wird ausgehend von jeweils einem alten Tupel t_{alt} ein neues Tupel t_{neu} gesucht, mit dem t_{alt} erweitert werden kann. Entscheidend ist, in welcher Reihenfolge die alten und die neuen Tupel bearbeitet werden. Dazu wird der Begriff des *Erweiterungsgrades* eingeführt, der für die Anzahl verschiedener SA-Werte steht, die in erweiterbaren Matchingkanten vorkommen.

Definition 6.1 (Erweiterungsgrad) *Sei E_Δ eine Menge von Δ-top Matchingkanten. Der* Erweiterungsgrad $d(t)$ *eines Tupels t bezüglich E_Δ ist die Anzahl verschiedener SA-Werte in Δ-top Matchingkanten in $E_\Delta(t)$.*

Alte Tupel mit einem kleinen Erweiterungsgrad sind potentiell schwieriger zu erweitern, da es für sie weniger passende neue Tupel gibt. Daher werden die alten Tupel nach ihrem Erweiterungsgrad sortiert und es wird mit dem Tupel begonnen, das den kleinsten Grad hat. Um auch immer viele verschiedene SA-Werte zum Erweitern zur Verfügung zu haben, wird mit dem neuen Tupel zuerst erweitert, dessen SA-Wert am häufigsten vorkommt. Die Menge S_{neu} wird somit absteigend nach der Häufigkeit der SA-Werte der neuen Tupel sortiert.

Da eine Lösung von Problem 6.2 eine Tupelverteilung ist, bei der der minimale Anonymitätsgrad in der Matchingtabelle $\mathcal{T}_\Delta^{(n+1)}$ maximal ist, werden in der ersten Phase nur alte Tupel betrachtet, die einen kleinen Anonymitätsgrad in $\mathcal{T}_\Delta^{(n+1)}$ haben. Da diese Tabelle aber während der Ausführung eines Erweiterungsalgorithmus noch nicht vollständig ist, wird die Bezeichnung *aktueller Anonymitätsgrad* des Tupels t eingeführt. Dieser bezieht sich auf die Kanten, die bereits in $\mathcal{T}_\Delta^{(n+1)}$ eingefügt beziehungsweise bearbeitet wurden.

Definition 6.2 (aktueller Anonymitätsgrad) *Sei $\mathcal{T}_\Delta^{(n)}$ eine Δ-top Matchingtabelle, aus der nach Algorithmus 5.3 eine Tabelle $\mathcal{T}_\Delta^{(n+1)}$ berechnet wird. Sei $\mathcal{T}_\Delta^{*(n+1)}$ der Teil von $\mathcal{T}_\Delta^{(n+1)}$, der zu einem gegebenen Zeitpunkt (Schritt im Algorithmus 5.3) aus bereits abgearbeiteten Kanten besteht. Dann ist der* aktuelle Anonymitätsgrad *$a^*(t)$ eines Tupels t bezüglich $\mathcal{T}_\Delta^{*(n+1)}$ die Anzahl verschiedener SA-Werte in Matchingkanten von t in $\mathcal{T}_\Delta^{*(n+1)}$.*

Hierbei ist zu beachten, dass die Erweiterungsalgorithmen von Algorithmus 5.3 verwendet werden und für den aktuellen Anonymitätsgrad demzufolge auch alle Kanten betrachtet werden, die durch die Erweiterungsalgorithmen in die Matchingtabelle eingefügt werden. Ebenso werden auch die Kanten des Originalmatchings beachtet. Daraus folgt, dass der aktuelle Anonymitätsgrad eines Tupels immer mindestens 1 ist. Um eine gewisse Konsistenz in den Bezeichnungen zu erhalten, wird auch für einen unvollständigen Teil einer Δ-top Matchingtabelle die Bezeichnung $\mathcal{T}_\Delta^{(n+1)}$ anstelle von $\mathcal{T}_\Delta^{*(n+1)}$ gewählt.

Sind im Algorithmus 6.3 keine Kanten mehr vorhanden, die noch erweitert werden können ($E_{\text{Erw}}^* \neq \emptyset$), beginnt Phase 2, in der Kanten ein weiteres Mal erweitert werden können. Da der aktuelle Anonymitätsgrad hierbei nicht steigt, werden alte Tupel nur noch entsprechend der Sortierung von S_{alt} abgearbeitet, die sich jetzt nach E_{Erw} (anstelle von E_{Erw}^*) richtet. Zusätzlich wird aber gewährleistet, dass alle alten Tupel gleichmäßig erweitert werden. Das heißt, es wird immer das alte Tupel ausgewählt, das in dieser Phase am seltensten erweitert worden ist. Die weiteren Schritte sind analog zur ersten Phase.

Der Algorithmus endet, wenn keine neuen Tupel zum Erweitern mehr übrig sind ($S_{\text{neu}} = \emptyset$). Diese Situation kann auch bereits in Phase 1 eintreten.

Satz 6.2 *Algorithmus Maximaler Anonymitätsgrad findet eine Erweiterung von Matchings in $\mathcal{O}(n^4)$, wobei n die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Da in jedem Durchlauf einer der beiden While-Schleifen ein neues Tupel zugeordnet wird, gibt es insgesamt $|S_{\text{neu}}|$ Durchläufe. Das Bestimmen eines gewünschten alten und eines dazu passenden neuen Tupels geht in $\mathcal{O}(|S_{\text{neu}}| \cdot |S_{\text{alt}}|)$, wobei höchstens $\max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|$ Kanten erweitert werden. Die Laufzeit ist demnach $\mathcal{O}(|S_{\text{neu}}|^2 \cdot |S_{\text{alt}}| \cdot \max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|)$. Wird die Anzahl der alten und neuen Tupel mit der Anzahl aller Tupel in der Datentabelle n und $|E_{\text{alt}}(t_{\text{alt}})|$ nach Gleichung 5.2 abgeschätzt, folgt eine Laufzeit von $\mathcal{O}(n^2 \cdot n \cdot n) = \mathcal{O}(n^4)$. □

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B(3)	B(3)
		C($x_{1.1}$)	–
		D($x_{1.2}$)	D($x_{1.2}, 7$)
		E($x_{1.3}$)	E($x_{1.3}, 8$)
2	A	B($x_{2.1}$)	B($x_{2.1}, 4$)
			B($x_{2.1}, 5$)
		C($x_{2.2}$)	C($x_{2.2}, 6$)
		D($x_{2.3}$)	–
		F($x_{2.4}$)	–
3	B	A(1)	A(1)
4	B		A(2)
5	B		A(2)
6	C		A(2)
7	D		A(1)
8	E		A(1)

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)

ID	SA	E^*_{Erw}	Phase 1.1	Phase 1.2	Phase 1.3	E_{Erw}	Phase 2
1	A	C($x_{1.1}$)	C($x_{1.1}$)	–	–		–
		D($x_{1.2}$)	D($x_{1.2}$)	D($x_{1.2}, 7$)	D($x_{1.2}, 7$)		D($x_{1.2}, 7$)
		E($x_{1.3}$)	E($x_{1.3}$)	E($x_{1.3}$)	E($x_{1.3}, 8$)		E($x_{1.3}, 8$)
2	A	B($x_{2.1}$)	B($x_{2.1}, 4$)	B($x_{2.1}, 4$)	B($x_{2.1}, 4$)	B($x_{2.1}$)	B($x_{2.1}, 4$)
							B($x_{2.1}, 5$)
		C($x_{2.2}$)	C($x_{2.2}$)	C($x_{2.2}, 6$)	C($x_{2.2}, 6$)		C($x_{2.2}, 6$)
		D($x_{2.3}$)	D($x_{2.3}$)	–	–		–
4	B		A(2)	A(2)	A(2)		A(2)
5	B						A(2)
6	C			A(2)	A(2)		A(2)
7	D			A(1)	A(1)		A(1)
8	E				A(1)		A(1)

(b) Ablauf des Algorithmus

Abbildung 6.2: Erweiterung von Matchings mithilfe des Algorithmus 6.3 (*Maximaler Anonymitätsgrad*)

Beispiel 6.4 Abbildung 6.2 zeigt in (a) die Matchingtabelle für eine Folge von n Anfragegraphen (siehe Spalte $\mathcal{G}^{(n)}$). In einem weiteren Graphen G_{n+1} kommen die alten Tupel 1, 2 und 3 sowie die neuen Tupel 4, 5, 6, 7 und 8 vor. Da der SA-Wert von 1 und 2 identisch ist, wird Algorithmus 6.3 für diese beiden alten Tupel aufgerufen. Der Ablauf ist dabei in (b) dargestellt. Für jedes Tupel (Spalte ID) ist der entsprechende SA-Wert (Spalte SA) angegeben. Alle Kanten, die in Phase 1 erweitert werden können, sind in Spalte E^*_{Erw} aufgelistet. Kante $(2, \text{F}(x_{2.4}))$ kann nicht erweitert werden und fehlt in E^*_{Erw}, da es kein neues Tupel mit SA-Wert F gibt. Die vorletzte Spalte E_{Erw} zeigt die einzige Kante, die in Phase 2 ein weiteres Mal erweitert werden kann. In den übrigen Spalten sind sowohl die bereits erweiterten Kanten als auch die Kanten dargestellt, welche in der jeweiligen Phase noch erweitert werden können.

Zu Beginn sind die aktuellen Anonymitätsgrade der alten Tupel $a^*(1) = 2$ und $a^*(2) = 1$, denn es gibt bereits die Matchingkante $(1, \text{B}(3))$ in $\mathcal{T}_\Delta^{(n+1)}$ (siehe Spalte $\mathcal{G}^{(n+1)}$ in (a)). Diese Kante muss nicht erweitert werden (vgl. Zeile 10 von Algorithmus 5.3) und konnte daher von $\mathcal{T}_\Delta^{(n)}$ übertragen werden. Als Erstes (Phase 1.1) wird daher nur Tupel 2 erweitert. Der häufigste SA-Wert der neuen Tupel ist B, welcher zweimal vorkommt. Die Sortierung der neuen Tupel sei daher der Einfachheit halber 4, 5, 6, 7 und 8. Somit wird 2 mit B(4) erweitert und es entsteht die Kante $(2, \text{B}(x_{2.1}, 4))$ (Spalte Phase 1.1 enthält neben der erweiterten Kante auch alle noch nicht erweiterten Kanten aus E^*_{Erw}).

Jetzt haben beide alten Tupel den aktuellen Anonymitätsgrad 2 und die Wahl des nächsten alten Tupels wird durch deren Sortierung entschieden. Da Tupel 2 mit C und D zwei sowie Tupel 1 mit C, D und E drei verschiedene SA-Werte in Kanten aus E^*_{Erw} hat, liegt in der Reihenfolge der alten Tupel 2 vor 1.[5] Somit wird 2 mit C(6) und danach 1 mit D(7) erweitert, was in Spalte Phase 1.2 dargestellt ist. Die Kanten $(1, \text{C}(x_{1.1}))$ und $(2, \text{D}(x_{2.3}))$ sind gelöscht worden, da es kein weiteres Tupel mit einem SA-Wert C beziehungsweise D gibt. Damit hat nur noch Tupel 1 mit $(1, \text{E}(x_{1.3}))$ eine Kante in E^*_{Erw}, welche im letzten Durchlauf der Phase 1 mit E(8) erweitert wird.

Für Phase 2 werden alle Kanten berücksichtigt, die ein weiteres Mal erweitert werden können. Da es aber nur noch das neue Tupel 5 mit SA-Wert B gibt, kann nur noch die Kante $(2, \text{B}(x_{2.1}))$ erweitert werden. Die von Algorithmus 6.3 berechnete Lösung ist in der letzten Spalte dargestellt. In Spalte $\mathcal{G}^{(n+1)}$ aus (a) sind diese Kanten zusammen mit den beiden Matchingkanten angegeben, die nicht erweitert werden mussten. Zu einer vollständigen Matchingtabelle $\mathcal{T}_\Delta^{(n+1)}$ fehlen allerdings die neuen Matchings, die nur aus neuen Tupeln entstehen (vgl. Zeile 29 in Algorithmus 5.3), die der Übersicht halber hier weggelassen wurden.

Im Anhang befinden sich noch zwei weitere Beispiele, die auch die Feinheiten von Algorithmus 6.3 erläutern: Beispiel A.15 auf Seite 286 und Beispiel A.16 auf Seite 289. Es ist zu beachten, dass der Algorithmus *Maximaler Anonymitätsgrad* nur eine einfache Heuristik ist, die versucht, eine optimale Erweiterung entsprechend dem Problem 6.2 zu erzeugen. Es ist nicht garantiert, dass das Optimum auch immer gefunden wird. Das liegt daran, dass Entscheidungen, welches alte Tupel mit welchem neuen Tupel erweitert wird, immer nur nach lokalen Gesichtspunkten getroffen werden. Es ist nicht sichergestellt, dass es in allen Fällen besser ist, zum Beispiel immer das neue Tupel mit dem häufigsten SA-Wert auszuwählen. Ein Beispiel dazu wird im nächsten Kapitel gegeben.

[5] Die Erweiterungsgrade bezüglich E^*_{Erw} sind $d(1) = 3$ und $d(2) = 2$.

6.5 Erweiterungsalgorithmus: Einfaches Matching

Anhand eines Beispiels kann gezeigt werden, dass Algorithmus 6.3 (*Maximaler Anonymitätsgrad*) nur eine Näherung für Problem 6.2 darstellt, die das Optimum nicht immer findet. Sei dazu eine Matchingtabelle für $\mathcal{G}^{(n)}$ wie in Teil (a) von Abbildung 6.3 gegeben. Ein weiterer Graph G_{n+1} bestehe aus den alten Tupeln 1, 2 und 3 (mit SA-Wert A) sowie den neuen Tupeln 4, 5 und 6. Alle alten Tupel haben jeweils denselben aktuellen Anonymitätsgrad von 1 und Erweiterungsgrad von 2. Ebenso kommt jeder SA-Wert der neuen Tupel nur einmal vor. Die Reihenfolge, nach der Algorithmus 6.3 eine Erweiterung berechnet, ist daher beliebig. Wird beispielsweise Tupel 1 zuerst mit B(4) erweitert, können Tupel 2 und 3 nur noch mit D(6) erweitert werden. Das bedeutet, die Kanten von einem Tupel können nicht erweitert werden. In einem weiteren Schritt kann nur noch 1 mit C(5) erweitert werden. Dieser Fall ist in (a) und (b) mit $v1$ gekennzeichnet.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$\mathcal{G}_{v2}^{(n+1)}$
1	A	B($x_{1.1}$) C($x_{1.2}$)	B($x_{1.1}, 4$) C($x_{1.2}, 5$)	– C($x_{1.2}, 5$)
2	A	B($x_{2.1}$) D($x_{2.2}$)	– D($x_{2.2}, 6$)	B($x_{2.1}, 4$) –
3	A	B($x_{3.1}$) D($x_{3.2}$)	– –	– D($x_{3.2}, 6$)
4	B		A(1)	A(2)
5	C		A(1)	A(1)
6	D		A(2)	A(3)

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)

ID	SA	E^*_{Erw}	Phase 1.1_{v1}	Phase 1.2_{v1}	Phase 1_{v2}
1	A	B($x_{1.1}$) C($x_{1.2}$)	B($x_{1.1}, 4$) C($x_{1.2}$)	B($x_{1.1}, 4$) C($x_{1.2}, 5$)	– C($x_{1.2}, 5$)
2	A	B($x_{2.1}$) D($x_{2.2}$)	– D($x_{2.2}, 6$)	– D($x_{2.2}, 6$)	B($x_{2.1}, 4$) –
3	A	B($x_{3.1}$) D($x_{3.2}$)	– –	– –	– D($x_{3.2}, 6$)
4	B		A(1)	A(1)	A(2)
5	C			A(1)	A(1)
6	D		A(2)	A(2)	A(3)

(b) Ablauf des Algorithmus

Abbildung 6.3: Erweiterung von Matchings mithilfe des Algorithmus 6.3 (*Maximaler Anonymitätsgrad*)

Wird hingegen Tupel 1 zuerst mit C(5) erweitert, kann Tupel 2 noch mit B(5) und Tupel 3 danach mit D(6) erweitert werden (vgl. Version $v2$ in beiden Tabellen). In diesem Fall hat jedes Tupel den Anonymitätsgrad 2, was nach Problem 6.2 optimal ist.[6]

[6]Es gibt Graphen, in denen Algorithmus 6.3 in jedem Fall keine optimale Lösung findet. Wenn in G_{n+1} zum Beispiel noch ein altes Tupel mit SA-Wert A und Signatur ABD sowie ein neues Tupel mit SA-

Der Nachteil von Algorithmus 6.3 ist, dass Zuordnungen von neuen zu alten Tupeln nur nach lokalen Entscheidungskriterien für jedes alte Tupel einzeln durchgeführt werden. Eine durchgeführte Zuordnung kann für andere Tupel nachteilig sein. Eine erste Verbesserung ist daher, alle alten Tupel in einem Schritt zu betrachten, sodass möglichst viele erweitert werden können. Dazu wird an dieser Stelle ein anderer Erweiterungsalgorithmus eingeführt, der Zuordnungen mithilfe von Matchings zwischen alten und neuen Tupeln bestimmt. Darauf aufbauend wird im Kapitel 6.6 ein Algorithmus vorgestellt, der das Optimum in jedem Fall findet.

Die Idee des Erweiterungsalgorithmus ist, einen sogenannten *Erweiterungsgraphen* zu konstruieren, der für jedes alte und jedes neue Tupel einen Knoten enthält. Kanten werden zwischen alten und neuen Tupelknoten eingefügt, wenn das alte Tupel mit den neuen erweitert werden kann. In diesem Graphen wird ein größtes Matching bestimmt und danach werden genau die Erweiterungen beziehungsweise Zuordnungen durchgeführt, die für die im Matching enthaltenen Kanten stehen. Das heißt, für jede Matchingkante zwischen einem Knoten für ein altes und einem für ein neues Tupel wird das entsprechende alte Tupel mit dem neuen erweitert. Das Bestimmen von größten Matchings erfolgt iterativ in mehreren Schritten, solange noch neue Tupel existieren, die noch nicht zugeordnet sind.

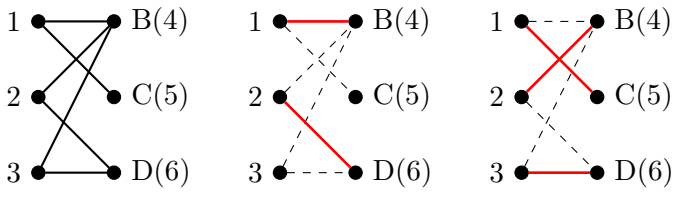
Definition 6.3 (Erweiterungsgraph) *Seien S_{alt} eine Menge von alten und S_{neu} eine Menge von neuen Tupeln sowie E_{alt} eine Menge von Δ-top Matchingkanten zwischen Tupeln aus S_{alt} und sensiblen Werten von Tupeln aus S_{neu}. Dann ist ein Erweiterungsgraph $G_{Erw} = (V_{Erw}, E_{Erw})$ wie folgt definiert:*

- *Für jedes Tupel $t \in S_{alt}$ enthalte V_{Erw} einen Knoten, der mit t gelabelt ist und* alter Knoten *genannt wird.*
- *Für jedes Tupel $t \in S_{neu}$ enthalte V_{Erw} einen Knoten, der mit $s(t)$ gelabelt ist, wobei s der sensible Wert von t ist. Dieser Knoten wird* neuer Knoten *genannt.*
- *Für jede Kante $(t, s'(S_T)) \in E_{alt}$ enthalte E_{Erw} Kanten vom Knoten mit Label t zu allen Knoten mit Label $s'(t')$, wobei t' ein beliebiges Tupel ist.*

Der Unterschied zwischen alten und neuen Tupeln ist, dass die Knoten für letztere sowohl mit dem entsprechenden Tupel als auch dem SA-Wert des Tupels gelabelt werden. Man spricht auch vom SA-Wert eines neuen Knotens, wobei der SA-Wert des neuen Tupels gemeint ist, für das der Knoten erzeugt wurde. Offensichtlich ist jeder Erweiterungsgraph bipartit, das heißt, größte Matchings können effizient mithilfe bekannter Algorithmen bestimmt werden (vgl. Kapitel 1.6.3).

In Abbildung 6.4 ist in (a) der Erweiterungsgraph für das zuvor eingeführte Beispiel gegeben. Die Knoten auf der linken Seite stehen für die alten Tupel 1, 2 und 3, die Knoten auf der rechten Seite für die neuen Tupel 4, 5, und 6. Die letztgenannten sind im Graphen auch mit ihrem jeweiligen sensiblen Wert B, C und D gekennzeichnet. Kanten stehen für mögliche Erweiterungen, die in Abbildung 6.3a in Spalte $\mathcal{G}^{(n)}$ gegeben sind. Beispielsweise kann 1 mit B und C erweitert werden, da es in $\mathcal{G}^{(n)}$ Matchingkanten $(1, \mathrm{B}(x_{1.1}))$ und $(1, \mathrm{C}(x_{1.2}))$ gibt.

Wert B vorkommt, würde auf Tupel 1 zwangsläufig mit einem B erweitert werden. Dazu gibt es dann drei Tupel mit Signatur ABD, aber nur je ein neues Tupel mit SA-Wert B und D. Demnach können nicht alle alten Tupel erweitert werden.



(a) G (b) M_{v1} (c) M_{v2}

Abbildung 6.4: Erweiterung von Matchings mithilfe des Algorithmus 6.4 (*Einfaches Matching*)

In (b) und (c) sind zwei mögliche maximale[7] Matchings im Graphen dargestellt. Eine Erweiterung nach (b) würde 1 mit B(4) und 2 mit D(6) erweitern, wodurch C(5) in diesem Schritt übrig bliebe. In einem weiteren Erweiterungsschritt, in dem nur noch nicht zugeordnete neue Knoten beachtet werden, wird ein Matching mit der Kante $(1, \text{C}(5))$ erstellt und damit 1 mit C(5) erweitert. In diesem Fall entsteht dieselbe Erweiterung, die zuvor als Version $v1$ in Abbildung 6.3a angegeben war. Die Erweiterung nach (c) ordnet jedem alten ein neues Tupel zu und ist identisch zur Version $v2$ aus Abbildung 6.3a. Offensichtlich ist das zweite Matching größer als das erste und gewährleistet, dass mehr alte Tupel in einem Schritt erweitert werden.

Algorithmus 6.4 Erweiterungsalgorithmus: Einfaches Matching

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel
Ausgabe: Erweiterung von Kanten (Tupelverteilung)

1: Konstruiere Erweiterungsgraphen G_{Erw} nach Definition 6.3
2: **while** $\exists$ neue Knoten mit mind. einer ausgehenden Kante **do** # Phase 1
3: Berechne größtes Matching M in G_{Erw} mit folgenden Bedingungen:
4: Ignoriere alle alten Tupel/Knoten ohne ausgehende Kante
5: Betrachte nur Knoten von alten Tupeln mit minimalem aktuellen Anonymitätsgrad
6: **for all** Kanten $e \in M$ **do**
7: Sei t_{alt} das alte Tupel und t_{neu} das neue Tupel in e
8: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$
9: Entferne aus G_{Erw} den Knoten für t_{neu}
10: Entferne aus G_{Erw} alle Kanten von t_{alt} zu Knoten, die für neue Tupel stehen, die denselben SA-Wert haben wie t_{neu}
11: **end for**
12: **end while**
13: **if** $\exists$ neue Knoten in G_{Erw} **then** # Phase 2
14: Füge in G_{Erw} alle möglichen zuvor entfernten Kanten ein
15: Führe Zeilen 2 bis 12 erneut aus ohne Zeile 10 und ohne Beschränkung auf minimalen aktuellen Anonymitätsgrad
16: **end if**

[7] Ein Matching heißt maximal, wenn keine weitere Kante hinzugefügt werden kann, ohne die Matchingbedingung zu verletzen (vgl. Kapitel 1.6.3).

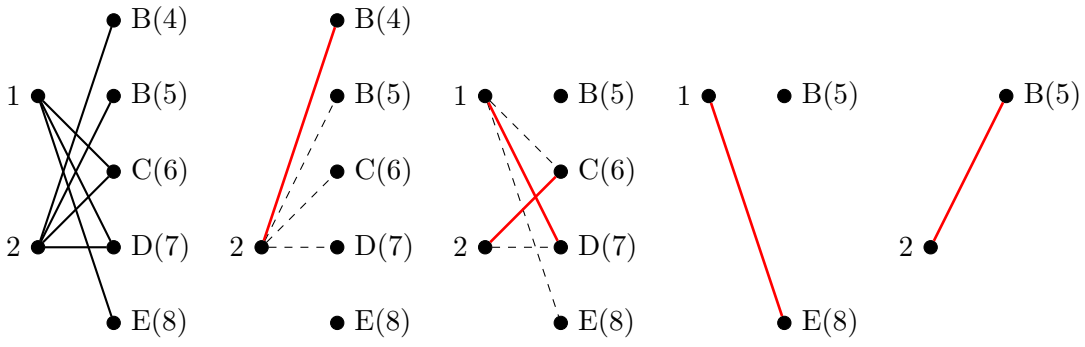
Algorithmus 6.4 erweitert Tupel mithilfe von Erweiterungsgraphen und heißt *Einfaches Matching*. Analog zum Algorithmus *Maximaler Anonymitätsgrad* arbeitet er auch in zwei Phasen, wobei in der ersten Phase jede Kante maximal einmal erweitert wird. Dazu wird der Erweiterungsgraph nach Definition 6.3 erstellt und ein größtes Matching bestimmt. Hierbei werden nur die alten Knoten betrachtet, deren dazugehöriges Tupel den kleinsten aktuellen Anonymitätsgrad hat, wobei auch nur Tupel beziehungsweise Knoten beachtet werden, die mindestens eine ausgehende Kante haben. Alle anderen Tupel werden in dieser Phase nicht mehr erweitert.

Jede im Matching enthaltene Kante repräsentiert eine Erweiterung eines alten Tupels t_{alt} mit einem neuen Tupel t_{neu}, die entsprechend durchgeführt wird. Damit Δ-top Matchingkanten von t_{alt} in dieser Phase nicht mehrfach erweitert werden, werden danach alle Kanten zwischen t_{alt} und neuen Knoten entfernt, die denselben SA-Wert haben wie t_{neu}.

In Phase 2 werden Matchingkanten erneut erweitert. Dazu werden die zusätzlich gelöschten Kanten wieder in den Graphen eingefügt und die While-Schleife erneut durchlaufen. Da sich der aktuelle Anonymitätsgrad in dieser Phase nicht ändert, wird er bei der Betrachtung von Tupeln beziehungsweise Knoten ignoriert. Wenn alle neuen Tupel zugeordnet sind, endet der Algorithmus.

Satz 6.3 *Algorithmus Einfaches Matching findet eine Erweiterung von Matchings in* $\mathcal{O}(n^{\frac{7}{2}})$*, wobei* n *die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Der konstruierte Erweiterungsgraph hat höchstens $|S_{\text{neu}}| \cdot |S_{\text{alt}}|$ Kanten. Ein Matching kann darin mithilfe des Algorithmus von Hopcroft und Karp (vgl. Satz 1.5 auf Seite 19) in $\mathcal{O}(\sqrt{|S_{\text{neu}}| + |S_{\text{alt}}|} \cdot |S_{\text{neu}}| \cdot |S_{\text{alt}}|) = \mathcal{O}((|S_{\text{neu}}| + |S_{\text{alt}}|)^{\frac{5}{2}})$ gefunden werden und wird höchstens $|S_{\text{neu}}|$-mal berechnet. Für jedes neue Tupel werden höchstens $\max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|$ Kanten erweitert, wodurch eine Laufzeit von $\mathcal{O}(|S_{\text{neu}}| \cdot ((|S_{\text{neu}}| + |S_{\text{alt}}|)^{\frac{5}{2}} + \max_{t_{\text{alt}}} |E_{\text{alt}}(t_{\text{alt}})|))$ entsteht. Wird die Anzahl der alten und neuen Tupel mit der Anzahl aller Tupel in der Datentabelle n und $|E_{\text{alt}}(t_{\text{alt}})|$ nach Gleichung 5.2 abgeschätzt, folgt eine Laufzeit von $\mathcal{O}(n \cdot (n^{\frac{5}{2}} + n)) = \mathcal{O}(n^{\frac{7}{2}})$. □



(a) G_{Erw} (b) M_1 (c) M_2 (d) M_3 (e) M_4

Abbildung 6.5: Ablauf des Algorithmus 6.4 (*Einfaches Matching*). Die farbigen Kanten symbolisieren jeweils das berechnete Matching, die gestrichelten Kanten stehen für alternative Matchingkanten. Alte Knoten, die im jeweiligen Schritt nicht betrachtet werden, sind auch nicht dargestellt.

Beispiel 6.5 Sei dieselbe Δ-top Matchingtabelle wie in Beispiel 6.4 (s. Abbildung 6.2a) gegeben. 1 und 2 seinen zwei alte Tupel mit gleichem SA-Wert und 4 bis 8 fünf neue Tupel. Abbildung 6.5 zeigt den Ablauf des Algorithmus 6.4, wobei der Erweiterungsgraph G_{Erw} in (a) dargestellt ist.

Anfangs sind die aktuellen Anonymitätsgrade $a^*(1) = 2$ und $a^*(2) = 1$, das heißt, es wird zunächst nur Tupel 2 betrachtet. Ein mögliches Matching ist in (b) berechnet und repräsentiert die Erweiterung von 2 mit B(4). Das ist die gleiche Zuordnung wie in Beispiel 6.4. Durch Zeile 10 des Algorithmus wird die Kante $\{2, \text{B}(5)\}$ in G_{Erw} entfernt, da das Tupel 5 mit B denselben SA-Wert hat wie Tupel 4.

Nach dieser Erweiterung haben alle Tupel den aktuellen Anonymitätsgrad 2 und (c) zeigt ein mögliches Matching. Es steht für die Erweiterungen von 1 mit D(7) und 2 mit C(6). Der Knoten für Tupel 2 hat aber keine ausgehende Kante mehr und wird in Phase 1 nicht weiter beachtet. Als Letztes in dieser Phase kann noch Tupel 1 mit E(8) erweitert werden (siehe (c)).

Für Phase 2 wird die zuvor entfernte Kante $\{2, \text{B}(5)\}$ wieder in G_{Erw} eingefügt und ein letztes Matching berechnet. Daraus ergibt sich die Erweiterung von Tupel 2 mit B(5). Die resultierende Δ-top Matchingtabelle entspricht der gleichen wie in Beispiel 6.4 (siehe Abbildung 6.2a), wobei durch Algorithmus 6.4 auch andere Matchings beziehungsweise Erweiterungen hätten entstehen können.

Im Anhang befinden sich zwei weitere Beispiele, die die Feinheiten von Algorithmus 6.4 erläutern: Beispiel A.17 auf Seite 291 und Beispiel A.18 auf Seite 291. Der Vorteil von Algorithmus *Einfaches Matching* gegenüber dem Algorithmus *Maximaler Anonymitätsgrad* ist, dass mehr Erweiterungen in einem Schritt durchgeführt werden. Allerdings ist auch er nur eine Heuristik für Problem 6.2, die nicht in jedem Fall eine optimale Lösung findet. Darauf aufbauend wird daher im nächsten Abschnitt ein Algorithmus vorgestellt, der Problem 6.2 löst, da er eine Erweiterung berechnet, bei der der minimale Anonymitätsgrad in jedem Fall maximal ist.

6.6 Erweiterungsalgorithmus: Klonmatching

6.6.1 Motivation und Ansatz

Zur Motivation für einen exakten Erweiterungsalgorithmus für Problem 6.2 sei in Abbildung 6.6 ein weiteres Beispiel gegeben. Dabei bestehe ein Graph G_{n+1} aus sieben Tupeln, wobei 1 und 2 alte Tupel seien, die erweitert werden sollen.[8] In (a) ist die Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ vor der Erweiterung dargestellt. Möglich wäre nun, Tupel 1 mit B(3) und C(5) sowie Tupel 2 mit B(4), D(6) und E(7) zu erweitern. Dann wäre der minimale Anonymitätsgrad der alten Tupel 3 und damit maximal, denn es gibt keine andere Erweiterung, die zu einem größeren minimalen Anonymitätsgrad führt. Die resultierende Matchingtabelle ist in Spalte $\mathcal{G}_{v1}^{(n+1)}$ zu finden.[9]

In (b) beziehungsweise (c) bis (g) sind die Abläufe der Algorithmen 6.3 (*Maximaler Anonymitätsgrad*) beziehungsweise 6.4 (*Einfaches Matching*) dargestellt. Dabei treffen beide Algorithmen analoge Entscheidungen, sodass jeweils dasselbe Ergebnis berechnet wird. Im ersten Schritt des Algorithmus 6.3 wird Tupel 1 mit B(3) erweitert, denn der SA-Wert B

[8] Dieses Beispiel ist sehr ähnlich aber nicht identisch zu Beispiel 6.4 auf Seite 167

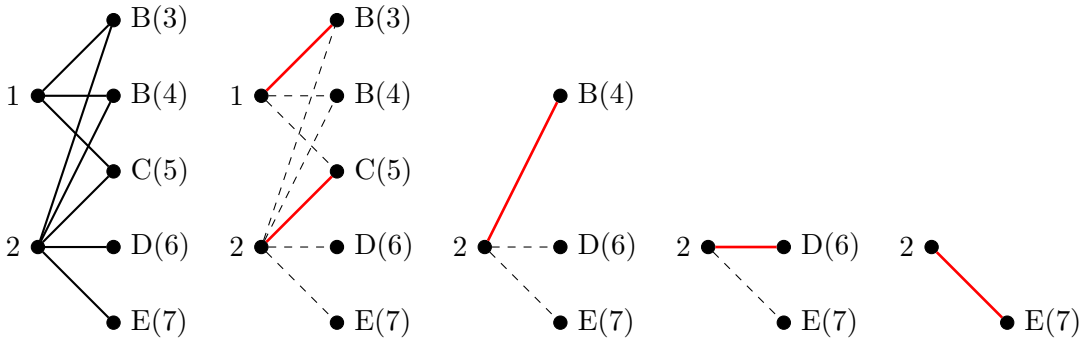
[9] Der Übersicht halber sind die Kanten weggelassen worden, die zu neuen Matchings gehören.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$\mathcal{G}_{v2}^{(n+1)}$
1	A	B($x_{1.1}$) C($x_{1.2}$)	B($x_{1.1}, 3$) C($x_{1.2}, 5$)	B($x_{1.1}, 3$) –
2	A	B($x_{2.1}$) C($x_{2.2}$) D($x_{2.3}$) E($x_{2.4}$)	B($x_{2.1}, 4$) – D($x_{2.3}, 6$) E($x_{2.4}, 7$)	B($x_{2.1}, 4$) C($x_{2.2}, 5$) D($x_{2.3}, 6$) E($x_{2.3}, 7$)
3	B		A(1)	A(1)
4	B		A(2)	A(2)
5	C		A(1)	A(2)
6	D		A(2)	A(2)
7	E		A(2)	A(2)

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)

ID	SA	E^*_{alt}	Phase 1.1$_{v2}$	Phase 1.2$_{v2}$	Phase 1.3$_{v2}$	Phase 1.4$_{v2}$
1	A	B($x_{1.1}$) C($x_{1.2}$)	B($x_{1.1}, 3$) –	B($x_{1.1}, 3$) –	B($x_{1.1}, 3$) –	B($x_{1.1}, 3$) –
2	A	B($x_{2.1}$) C($x_{2.2}$) D($x_{2.3}$) E($x_{2.4}$)	B($x_{2.1}$) C($x_{2.2}, 5$) D($x_{2.3}$) E($x_{2.4}$)	B($x_{2.1}, 4$) C($x_{2.2}, 5$) D($x_{2.3}$) E($x_{2.4}$)	B($x_{2.1}, 4$) C($x_{2.2}, 5$) D($x_{2.3}, 6$) E($x_{2.4}$)	B($x_{2.1}, 4$) C($x_{2.2}, 5$) D($x_{2.3}, 6$) E($x_{2.4}, 7$)
3	B		A(1)	A(1)	A(1)	A(1)
4	B			A(1)	A(1)	A(1)
5	C		A(2)	A(2)	A(2)	A(2)
6	D				A(2)	A(2)
7	E					A(2)

(b) Ablauf des Algorithmus 6.3 (*Maximaler Anonymitätsgrad*)



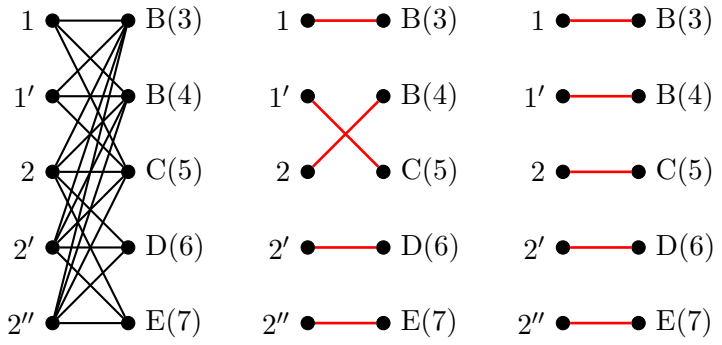
(c) G_{Erw} (d) M_1 (e) M_2 (f) M_3 (g) M_4

Abbildung 6.6: Erweiterung von Matchings mithilfe der Algorithmen 6.3 (*Maximaler Anonymitätsgrad*) und 6.4 (*Einfaches Matching*)

kommt zweimal vor. Tupel 2 kann danach mit einem beliebigen neuen Tupel erweitert werden, da alle SA-Werte genau einmal vorkommen. Angenommen, 2 wird mit C(5) erweitert, dann kann Tupel 1 in Phase 1 nicht erneut erweitert werden, da kein anderes neues Tupel mit dem SA-Wert C vorhanden ist (siehe Spalte Phase 1.1_{v2} in (b)). (d) zeigt ein Matching für Algorithmus 6.4, aus dem dieselben Zuordnungen hervorgehen.

In den weiteren Schritten der Phase 1 wird Tupel 2 mit B(4), D(6) und E(7) erweitert. Dadurch entsteht eine andere Variante der Δ-top Matchingtabelle, die mit $v2$ bezeichnet wird und in (a) in Spalte $\mathcal{G}_{v2}^{(n+1)}$ angegeben ist. Es ist leicht zu sehen, dass diese Variante nicht optimal ist. Tupel 1 hat nur noch eine Matchingkante mit SA-Wert B und demzufolge einen Anonymitätsgrad von 2, während Tupel 2 einen Grad von 5 aufweist. Hätten beide Algorithmen jeweils für Tupel 2 als Erstes eine andere Erweiterung (B(4), D(6) oder E(7)) gewählt, hätten beide Algorithmen auch die optimale Variante $v1$ berechnet.

Der Nachteil beider bisher vorgestellter Verfahren für Problem 6.2 ist, dass sie iterativ mehrere Durchläufe absolvieren und sich in einem Schritt getroffene Entscheidungen in späteren Durchläufen negativ auf weitere Möglichkeiten auswirken können. Daher ist die Idee eines exakten Erweiterungsalgorithmus, alle Erweiterungen in einem Schritt zu berechnen. Dadurch kann gewährleistet werden, dass die optimale Erweiterung in jedem Fall erzeugt wird.



(a) G_{Erw} (b) M_{v1} (c) M_{v3}

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$\mathcal{G}_{v3}^{(n+1)}$
1	A	$B(x_{1.1})$	$B(x_{1.1}, 3)$	$B(x_{1.1}, 3)$
		$C(x_{1.2})$	$C(x_{1.1}, 5)$	$B(x_{1.1}, 4)$
2	A	$B(x_{2.1})$	$B(x_{2.2}, 4)$	–
		$C(x_{2.2})$	–	$C(x_{2.2}, 5)$
		$D(x_{2.3})$	$D(x_{2.3}, 6)$	$D(x_{2.3}, 6)$
		$E(x_{2.4})$	$E(x_{2.4}, 7)$	$E(x_{2.4}, 7)$
3	B		A(1)	A(1)
4	B		A(2)	A(1)
5	C		A(1)	A(2)
6	D		A(2)	A(2)
7	E		A(2)	A(2)

(d) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)

Abbildung 6.7: Mögliche Erweiterungen von Matchings

Der hier vorgestellte Erweiterungsalgorithmus basiert auf dem Algorithmus *Einfaches Matching* und den Erweiterungsgraphen. Zusätzlich werden aber für alte Tupel mehrere Knoten erstellt, die jeweils die gleichen ausgehenden Kanten besitzen. Abbildung 6.7 zeigt in (a) einen Erweiterungsgraphen G_{Erw}, bei dem es zwei Knoten für Tupel 1 und drei Knoten für Tupel 2 gibt. Zur besseren Unterscheidung sind diese neben dem Tupellabel auch mit einem Strich gezeichnet. Ein größtes Matching M_{v1} in G_{Erw} ist in (b) dargestellt und steht für die bereits vorgestellte optimale Erweiterung (siehe Spalte $\mathcal{G}_{v1}^{(n+1)}$ in Tabelle (d)). Tupel 1 wird mit B(3) und C(5) sowie 2 mit B(4), D(6) und E(7) erweitert. In dem Graphen ist erkennbar, dass diese fünf Erweiterungen in einem Schritt durchgeführt werden können, indem ein größtes Matching in G_{Erw} berechnet wird. Werden Standardalgorithmen zum Bestimmen von Matchings genutzt, kann es aber auch passieren, dass ein anderes größtes Matching berechnet wird, wie zum Beispiel M_{v3} in (c). Tupel 1 wird in diesem Fall mit beiden neuen Tupeln mit SA-Wert B (3 und 4) sowie Tupel 2 mit C, D und E erweitert. Dadurch hat Tupel 1 nur noch Matchingkanten mit dem SA-Wert B, während ihm durch M_{v1} noch zwei verschiedene SA-Werte (B und C) zugeordnet werden konnten. M_{v3} ist daher erneut keine optimale Lösung für das gegebene Beispiel.

Die Herausforderung an einen Erweiterungsalgorithmus, der die optimale Lösung von Problem 6.2 berechnen soll, besteht demnach darin, zunächst einen Graphen G_{Erw} zu konstruieren, der für alte Tupel eine bestimmte Anzahl Knoten enthält. Dieser Graph wird *Klongraph* genannt und kann für verschiedene alte Tupel auch eine verschiedene Anzahl von Knoten enthalten. Danach soll in G_{Erw} ein Matching bestimmt werden, in dem alle Knoten für dasselbe alte Tupel mit Knoten für neue Tupel matchen, die jeweils einen anderen SA-Wert haben. Dadurch wird erreicht, dass alte Tupel nicht mehrfach mit demselben SA-Wert erweitert werden. Das Konstruieren von Klongraphen und das Berechnen von Matchings wird im Folgenden vorgestellt.

6.6.2 Klongraphen und eindeutige Matchings

Ein *Klonerweiterungsgraph* (oder kurz *Klongraph*) entsteht aus einem Erweiterungsgraphen, indem Knoten für alte Tupel vervielfältigt („geklont“) werden. Dabei werden auch die Labels und alle mit den Knoten inzidenten Kanten dupliziert. Verschiedene Knoten, die für dasselbe alte Tupel stehen, werden *Klonknoten* genannt. Analog heißen paarweise verschiedene Kanten *Klonkanten*, wenn die enthaltenen alten Knoten Klonknoten und die SA-Werte der enthaltenen neuen Knoten identisch sind. Kanten, die denselben alten Knoten, aber verschiedene neue Knoten mit jeweils demselben SA-Wert enthalten, werden *SA-äquivalente Kanten*[10] genannt. Um Klonknoten besser voneinander zu unterscheiden, werden die Labels zum Beispiel zusätzlich mit Strichen markiert.

Wie beim Erweiterungsalgorithmus *Einfaches Matching* wird auch in Klongraphen ein Matching gesucht, allerdings mit einer Nebenbedingung. Für alle alten Knoten mit identischem Label soll gelten, dass die sensiblen Werte der Labels ihrer Matchingpartner paarweise verschieden sind. Verkürzt ausgedrückt sind Matchings gesucht, die keine Klonkanten enthalten. Ein Matching M, das diese Bedingung erfüllt, heißt *eindeutiges Matching*. Wird zusätzlich jeder alte Knoten durch M überdeckt, heißt M *vollständiges Matching*. Da jeder Klongraph offensichtlich bipartit ist, ist jedes vollständige Matching auch ein größtes.

[10] Die Bezeichnung wird analog zu Kanten in Anfragegraphen (vgl. Definition 4.1 auf Seite 103) verwendet.

Beispiel 6.6 Matching M_{v1} aus Abbildung 6.7b ist ein eindeutiges und vollständiges Matching in G_{Erw}, da es keine Klonkanten enthält und alle alten Tupel überdeckt. Dagegen sind die beiden Kanten $\{1, \text{B}(3)\}$ und $\{1', \text{B}(4)\}$ im Matching M_{v3} aus Abbildung 6.7c Klonkanten, denn die enthaltenen alten Knoten 1 und $1'$ sind Klonknoten und die beiden SA-Werte sind identisch. M_{v3} ist zwar ein größtes Matching in G_{Erw}, aber kein eindeutiges.

SA-äquivalente Kanten sind zum Beispiel $\{1, \text{B}(3)\}$ und $\{1, \text{B}(4)\}$. Da sie immer inzident sind, können sie nicht im gleichen Matching vorkommen. Sie spielen aber für die Konstruktion von eindeutigen Matchings eine wichtige Rolle.

Das Ziel des hier vorgestellten Algorithmus lautet, eine optimale Zuordnung der neuen zu den alten Tupeln zu berechnen. Nach Problem 6.2 ist eine Erweiterung gesucht, bei der der minimale Anonymitätsgrad der alten Tupel maximal ist. Die Konstruktion von Klongraphen ist allerdings intuitiver, wenn diese Forderung zunächst dahingehend verändert wird, dass Erweiterungen zunächst unabhängig von bereits vorhandenen Matchings, die nicht erweitert werden müssen, durchgeführt werden. Das bedeutet, es wird nicht der aktuelle Anonymitätsgrad, sondern die Anzahl von Erweiterungen für jedes alte Tupel berücksichtigt. Im Folgenden wird daher eine Lösung zu einem vereinfachten Problem der Tupelverteilung berechnet. Der Algorithmus, der ein Optimum nach dem ursprünglichen Problem 6.2 berechnet, wird am Ende dieses Kapitels vorgestellt.

Problem 6.3 (Erweiterung/Tupelverteilung (vereinfacht)) *Gegeben seien S_{alt}, S_{neu}, E_{alt} und f wie bei Problem 6.1. Bezeichne $f^{-1}(t) \subseteq S_{neu}$ die Menge von neuen Tupeln, die dem alten Tupel $t \in S_{alt}$ zugeordnet werden, und $f^{-1}(t)[SA]$ die Menge von SA-Werten dieser neuen Tupel.*

Gesucht ist eine Abbildung f, bei der die minimale Anzahl von Erweiterungen mit verschiedenen SA-Werten pro altem Tupel maximal ist:

$$\textit{maximiere} \min_{t \in S_{alt}} |f^{-1}(t)[SA]|. \tag{6.2}$$

Verschieden bezieht sich in diesem Kontext auf den SA-Wert der neuen Tupel, mit denen das alte Tupel erweitert wird. Es reicht demzufolge aus, jedes alte Tupel höchstens einmal mit jedem möglichen SA-Wert zu erweitern. Genau das wird durch ein eindeutiges Matching in einem Klongraphen beschrieben, bei dem die minimale Anzahl von Matchingpartner pro altem Tupel maximal ist. Anders ausgedrückt wird ein Matching gesucht, bei dem keinem alten Tupel mehrere neue Tupel mit gleichem SA-Wert zugeordnet werden und bei dem die Anzahl der wenigsten Zuordnungen zu einem alten Tupel maximal ist. Die Idee dabei ist, einen Klongraphen so zu konstruieren, dass durch ein vollständiges Matching genau dieses Optimum erreicht wird.

Für jedes alte Tupel t bezeichne $d(t)$ den Erweiterungsgrad von t bezüglich E_{alt}, also die Anzahl von verschiedenen SA-Werten in Δ-top Matchingkanten aus $E_{\text{alt}}(t)$.[11] In einem beliebigen Klongraphen G_{Erw} sei $n(t)$ die Anzahl der Klonknoten für t, das heißt die Anzahl der Knoten, die für t stehen und mit t gelabelt sind. $n(t)$ heißt auch der *Klongrad* von t. Damit in G_{Erw} ein vollständiges Matching existieren kann, muss offensichtlich $n(t) \leq d(t)$ für alle alten Tupel t gelten.[12] Bezeichnen wie gehabt S_{alt} die Menge der alten und S_{neu} die Menge der neuen Tupel. Die minimale Anzahl von Erweiterungen mit verschiedenem SA-Wert kann damit pro altem Tupel höchstens $\left\lfloor \frac{|S_{\text{neu}}|}{|S_{\text{alt}}|} \right\rfloor$ sein. Da diese Anzahl identisch mit der

[11] Vgl. Definition 6.1 auf Seite 164.

[12] Das ist ein notwendiges, aber kein hinreichendes Kriterium.

minimalen Anzahl von Knoten für ein Tupel in G_{Erw} sein soll, wird sie mit $n_{\min}$ bezeichnet. Eine erste Abschätzung für $n_{\min}$ ist demnach $n_{\min} = \min\{\left\lfloor \frac{|S_{\text{neu}}|}{|S_{\text{alt}}|} \right\rfloor, \min_{t \in S_{\text{alt}}} d(t)\}$.

Beispiel 6.7 Sei erneut die Δ-top Matchingtabelle aus Abbildung 6.7d gegeben. Die beiden alten Tupel 1 und 2 sollen mit den neuen Tupeln 3, 4, 5, 6 und 7 erweitert werden. Tupel 1 hat nur Δ-top Matchingkanten mit einem B und C, Tupel 2 mit B, C, D und E. Die Erweiterungsgrade sind dementsprechend $d(1) = 2$ und $d(2) = 4$. Da es insgesamt fünf neue Tupel gibt, können bei gleichmäßiger Verteilung jedem alten Tupel höchstens $\left\lfloor \frac{5}{2} \right\rfloor = 2$ neue Tupel zugeordnet werden.

In Abbildung 6.7a ist ein Klongraph für diese Erweiterung dargestellt. Er enthält zwei Knoten für Tupel 1 ($n(1) = 2$) und drei für Tupel 2 ($n(2) = 3$). Wenn es noch ein sechstes neues Tupel mit SA-Wert B, C, D oder E gäbe, könnten auch vier Knoten für Tupel 2 erstellt werden, da der Erweiterungsgrad für dieses Tupel 4 beträgt. Aus analogem Grund ($d(1) = 2$) kann es auch nicht mehr als zwei Knoten für Tupel 1 geben. Die Konstruktion dieses Graphen erfolgte durch Algorithmus 6.5 und wird in Beispiel 6.8 vorgestellt.

Ein einfacher Erweiterungsalgorithmus erzeugt einen Klongraphen G_{Erw}, bei dem für jedes alte Tupel $n_{\min}$ Knoten vorhanden sind. Enthält G_{Erw} ein vollständiges Matching, ist eine optimale Tupelverteilung nach Problem 6.3 gefunden. Andernfalls wird für jedes alte Tupel ein Knoten aus G_{Erw} entfernt und erneut nach einem vollständigen Matching gesucht. Wird erneut kein vollständiges Matching gefunden, endet das Entfernen von Knoten und Bestimmen von Matchings spätestens, wenn im Graphen kein alter Knoten mehr vorhanden ist. Das bedeutet, dass es keine Erweiterung gibt, bei der alle alten Tupel mindestens einmal erweitert werden können. Danach kann zum Beispiel Algorithmus 6.4 (*Einfaches Matching*) verwendet werden, um weitere Matchingkanten zu erweitern.

An dieser Stelle wird ein Erweiterungsalgorithmus vorgestellt, der im Gegensatz zum eben vorgestellten Verfahren eine kleine Verbesserung aufweist. Die zusätzliche Idee ist, auch nachdem die minimale Anzahl von Erweiterungen bestimmt ist, noch nicht zugeordnete neue Tupel weiterhin gleichmäßig auf die alten Tupel zu verteilen. Dadurch wird gewährleistet, dass alle alten Tupel möglichst viele Erweiterungen mit verschiedenen SA-Werten erhalten. Das wird erreicht, indem ein Klongraph G_{Erw} mit möglichst vielen Knoten für alte Tupel konstruiert wird. Insbesondere können auch mehr als $\left\lfloor \frac{|S_{\text{neu}}|}{|S_{\text{alt}}|} \right\rfloor$ Knoten für ein altes Tupel existieren. Da in G_{Erw} aber ein vollständiges Matching gesucht wird, darf die Gesamtanzahl N_T der alten Knoten nicht größer sein als die Anzahl der neuen Tupel. Daneben bleibt die bereits vorgestellte obere Schranke für alte Tupel bestehen, sodass für kein altes Tupel mehr als $d(t)$ Knoten erzeugt werden.

Algorithmus 6.5 konstruiert einen Klongraphen nach den oben formulierten Überlegungen. Dazu wird eine Variable $d_{\max}$ berechnet, die angibt, wie oft für jedes alte Tupel t ein Knoten in den Graphen eingefügt werden kann, wenn jeder Knoten gleich häufig vorkommt. Dabei wird geachtet, dass kein Knoten mehr als $d(t)$-mal vorkommt. Das folgende Beispiel veranschaulicht die genaue Vorgehensweise.

Beispiel 6.8 Sei erneut die Δ-top Matchingtabelle aus Abbildung 6.6 gegeben. Algorithmus 6.5 wird mit den alten Tupeln 1 und 2, den Matchingkanten aus der Matchingtabelle sowie den fünf neuen Tupeln 3 bis 7 aufgerufen. Tabelle 6.3 gibt einen Überblick über die sukzessive Konstruktion des Klongraphen. Zu den alten Tupeln sind die Erweiterungsgrade $d(1) = 2$ und $d(2) = 4$ in der 2. Spalte angegeben. Entscheidend ist die Berechnung von $d_{\max} = \lfloor (|S_{\text{neu}}| - N_T)/|S_T| \rfloor$ in der letzten Zeile.

Algorithmus 6.5 Konstruiere Klongraph (vereinfacht)

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel
Ausgabe: Klongraph G_{Erw}

Entferne aus S_{alt} lokal alle Tupel t mit $E_{\text{alt}}(t) = \emptyset$
Entferne aus S_{neu} lokal alle Tupel, deren SA-Wert in keiner Kante aus E_{alt} vorkommt
for all Tupel $t \in S_{\text{alt}}$ **do**
  $d(t) \leftarrow$ Anzahl verschiedener SA-Werte in $E_{\text{alt}}(t)$ # Erweiterungsgrad von t
  $n(t) \leftarrow 1$ # Klongrad: Anz. Knoten für t
end for
$N_T \leftarrow |S_{\text{alt}}|$ # Anzahl alter Knoten
$S_T \leftarrow \{t \in S_{\text{alt}} \mid d(t) > n(t)\}$ # Tupel zum Klonen
while $S_T \neq \emptyset$ **and** $d_{\max} \leftarrow \lfloor (|S_{\text{neu}}| - N_T)/|S_T| \rfloor \geq 1$ **do**
  for all $t \in S_T$ **do**
    $n(t) \leftarrow n(t) + \min\{d(t) - n(t), d_{\max}\}$
    $N_T \leftarrow N_T + \min\{d(t) - n(t), d_{\max}\}$
    if $n(t) = d(t)$ **then**
      $S_T \leftarrow S_T \setminus \{t\}$ # Tupel maximal vorhanden
    end if
  end for
end while
Konstruiere Klongraphen G_{Erw} nach folgenden Regeln:
  Erzeuge für jedes alte Tupel $t \in S_{\text{alt}}$ $n(t)$ Knoten
  Erzeuge für jedes neue Tupel $t \in S_{\text{neu}}$ einen Knoten
  Für jede Matchingkante $e_\Delta = (t_{\text{alt}}, s'(S_T)) \in E_{\text{alt}}$ erzeuge Kanten von allen Knoten für t_{alt} zu allen Knoten für neue Tupel, die den SA-Wert s' haben

Im ersten Durchlauf der While-Schleife (siehe Spalte Runde 1) ist $N_T = 2$, $S_T = \{1, 2\}$ und demnach $d_{\max} = \lfloor (5-2)/2 \rfloor = 1 \geq 1$. Da für die alten Tupel $d(1) - n(1) = 2 - 1 = 1 = d_{\max}$ beziehungsweise $d(2) - n(2) = 4 - 1 = 3 > d_{\max}$ gilt, werden für beide Tupel die Klongrade jeweils um $d_{\max} = 1$ erhöht ($n(1) = n(2) = 2$). Da Tupel 1 jetzt maximal oft vorhanden ist ($n(1) = d(1)$), wird es aus S_T entfernt.

Danach gilt $d_{\max} = \lfloor (5-4)/1 \rfloor = 1 \geq 1$ und für Tupel 2 $d(2) - n(2) = 4 - 2 = 2 > d_{\max}$. $n(2)$ kann somit erneut um $d_{\max}$ erhöht werden. Nun entspricht aber die Anzahl aller Knoten N_T der Anzahl neuer Tupel $|S_{\text{neu}}|$ und es folgt $d_{\max} = \lfloor (5-5)/1 \rfloor = 0 < 1$. Das ist ein Abbruchkriterium der Konstruktion. Folglich entsteht der Klongraph G_{Erw} aus Abbildung 6.7a, der zwei Knoten für Tupel 1 und drei Knoten für Tupel 2 besitzt.

Das Verfahren, welches Erweiterungen mithilfe vollständiger Matchings in Klongraphen berechnet, heißt *Klonmatching* und ist in Algorithmus 6.6 angegeben. Dabei definiere $G_{\text{Erw}}^{[i]} \subseteq G_{\text{Erw}}$ den Teilgraphen von G_{Erw}, in dem für jedes Tupel t genau $\min\{n(t), i\}$ Knoten enthalten sind. $G_{\text{Erw}}^{[i]}$ ist damit ein Klongraph mit beschränkter Anzahl Knoten für jedes alte Tupel.

Zuerst erstellt Algorithmus 6.6 mithilfe von Algorithmus 6.5 einen Klongraphen G_{Erw} ohne weitere Einschränkungen. Existiert in G_{Erw} kein vollständiges Matching, werden die Anzahlen der alten Knoten verringert, bis ein solches Matching gefunden wird oder für

ID	$d(t)$	Runde 1		Runde 2		Runde 3	
		$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$
1	2	1	1	2	–	2	–
2	4	1	3	2	2	3	1
N_T		2		4		5	
S_T		$\{1,2\}$		$\{1\}$		$\{1\}$	
$d_{\max}$		1		1		0	

Tabelle 6.3: Konstruktion des Klongraphen nach Algorithmus 6.5

Algorithmus 6.6 Erweiterungsalgorithmus: Vereinfachtes Klonmatching

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel

Ausgabe: Erweiterung von Kanten (Tupelverteilung)

1: $E_{\text{Erw}} \coloneqq \{(t_{\text{alt}}, s(S_T)) \in E_{\text{alt}} \mid \exists t_{\text{neu}} \in S_{\text{neu}} \colon t_{\text{neu}}[\text{SA}] = s\}$
2: $E^*_{\text{Erw}} \coloneqq \{e_\Delta \in E_{\text{Erw}} \mid e_\Delta$ noch nicht erweitert$\}$
3: **while** $E^*_{\text{Erw}} \neq \emptyset$ **do** # Phase 1
4: Konstruiere Klongraph G_{Erw} nach Algorithmus 6.5 mit Eingabe $S_{\text{alt}}, E^*_{\text{Erw}}, S_{\text{neu}}$
5: $n_{\max} \leftarrow \max_{t \in S_{\text{alt}}} n(t)$
6: Bestimme maximales i mit $1 \leq i \leq n_{\max}$, für das im Klongraphen $G^{[i]}_{\text{Erw}}$ ein vollständiges Matching M existiert
7: Falls i nicht existiert, bestimme ein größtes Matching M in $G^{[1]}_{\text{Erw}}$
8: **for all** Kanten $e \in M$ **do**
9: Sei t_{alt} das alte Tupel und t_{neu} das neue Tupel in e
10: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$
11: Entferne aus S_{neu} Tupel t_{neu}
12: **end for**
13: **end while**
14: **if** $S_{\text{neu}} \neq \emptyset$ **then** # Phase 2
15: Führe Zeilen 1 bis 12 aus Algorithmus 6.4 aus ohne Zeile 10, ohne Beschränkung auf minimalen aktuellen Anonymitätsgrad und mit Eingabe $S_{\text{alt}}, E_{\text{Erw}}, S_{\text{neu}}$
16: **end if**

jedes alte Tupel nur noch ein Knoten in G_{Erw} vorhanden ist. Im letztgenannten Fall wird nur ein größtes Matching bestimmt. Die restlichen Schritte sind analog zum Algorithmus *Einfaches Matching*. Es werden Erweiterungen entsprechend den Kanten im gefundenen Matching durchgeführt und diese Schritte so lange wiederholt, bis keine noch nicht erweiterten Kanten existieren, die erweitert werden können (d. h. $E^*_{\text{Erw}} = \emptyset$). Danach können nur noch Kanten erweitert werden, die bereits einmal erweitert worden sind, wofür Algorithmus 6.4 (*Einfaches Matching*) verwendet wird.

Satz 6.4 *Algorithmus 6.6 (Vereinfachtes Klonmatching) findet eine optimale Erweiterung nach Problem 6.3.*

Beweis: Wenn die Ausgabe von Algorithmus 6.6 kein Optimum von Problem 6.3 darstellt, muss es eine Erweiterung geben, bei der die minimale Anzahl von Erweiterungen mit

verschiedenem SA-Wert für die alten Tupel größer ist. Das steht aber im Widerspruch zur Maximalität von i in Zeile 6. □

Im Anhang befinden sich zwei komplexere Beispiele, die auch die Feinheiten der Konstruktion von Klongraphen nach Algorithmus 6.5 und der Berechnung von Erweiterungen nach Algorithmus 6.6 erläutern: Beispiel A.19 auf Seite 292 und Beispiel A.21 auf Seite 296.

Um die Laufzeit von Algorithmus 6.6 abzuschätzen, muss ein Verfahren angegeben werden, mit dem vollständige Matchings in Klongraphen effizient berechnet werden können. Die nächsten beiden Abschnitte beschäftigen sich mit dieser Fragestellung.

6.6.3 Berechnung von eindeutigen Matchings in Klongraphen

Der entscheidende Schritt in Algorithmus 6.6 ist das Berechnen eines vollständigen Matchings M im Klongraphen $G_{\text{Erw}}^{[i]}$ (vgl. Zeile 6). Da M immer auch ein größtes Matching in einem bipartiten Graphen ist, ist es naheliegend, dafür bekannte Verfahren zu verwenden. Allerdings soll für M gelten, dass es keine Klonkanten enthält (d. h., M ist ein eindeutiges Matching). Die in Kapitel 1.6.3 vorgestellten Matchingalgorithmen *Bipartites Matching* und *Hopcroft-Karp* können zwar größte Matchings in bipartiten Graphen berechnen, doch ist zunächst unklar, wie diese zusätzliche Nebenbedingung integriert werden kann.

Die beiden Algorithmen 1.1 (*Bipartites Matching*, siehe Seite 18) und 1.2 (*Hopcroft-Karp*, siehe Seite 20) basieren auf der Idee, Matchings M mithilfe augmentierender Pfade so lange zu vergrößern, bis das Maximum erreicht ist. Das sind Pfade, die abwechselnd aus Matching- und Nicht-Matchingkanten bestehen, wobei Anfangs- und Endknoten nicht von M überdeckt sind. Durch ein Vertauschen dieser Kanten bezüglich des Matchings erhält man ein um eins größeres Matching. Satz 1.1 liefert die theoretische Grundlage für dieses Vorgehen und besagt, dass ein Matching M genau dann ein größtes Matching ist, wenn kein M-augmentierender Pfad mehr existiert. Der Unterschied zwischen beiden vorgestellten Verfahren ist, dass der Algorithmus von Hopcroft und Karp in jedem Schritt simultan mithilfe mehrerer kürzester Pfade augmentiert. In diesem Fall gilt, dass sich die Länge eines kürzesten Pfades nach dem Augmentieren stets vergrößert (vgl. Lemma 1.4 auf Seite 18), wodurch der Algorithmus eine bessere Laufzeit als die Standardvariante erreicht.

Offensichtlich führt nicht jedes beliebige Augmentieren mithilfe eines Pfades in einem Klongraphen zu einem eindeutigen Matching, da die Nebenbedingung sehr leicht verletzt werden kann. Um die Matchingalgorithmen dennoch verwenden zu können, muss für eindeutige Matchings in Klongraphen gezeigt werden, dass sie durch spezielle augmentierende Pfade vergrößert werden können. Dabei müssen erneut eindeutige Matchings entstehen. Das entspricht einer Aussage analog zum Satz von Berge (siehe Satz 1.1) für größte Matchings. Daneben müssen die Algorithmen angepasst werden, damit diese speziellen sogenannten *eindeutig augmentierenden Pfade* effizient gefunden werden. Für den schnelleren Algorithmus von Hopcroft und Karp muss zusätzlich eine Aussage analog zum Lemma 1.4 getroffen werden, damit das simultane Augmentieren von Pfaden einen Laufzeitvorteil bringt.

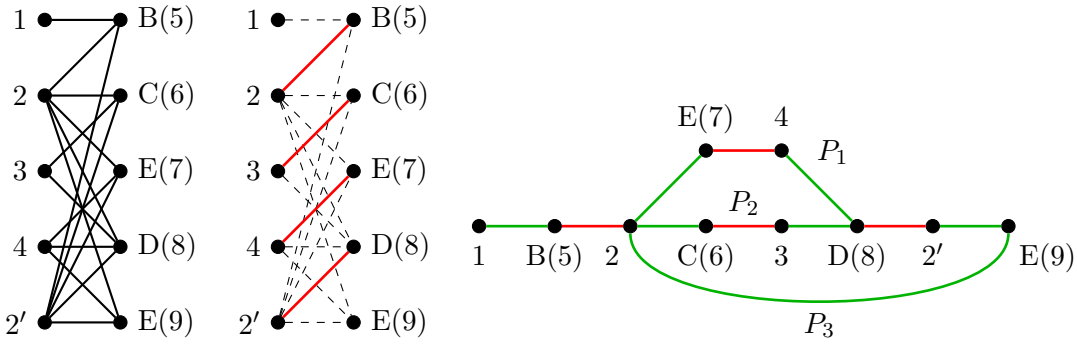
Definition 6.4 (eindeutig augmentierender Pfad) *Sei M ein eindeutiges Matching in einem Klongraphen G_{Erw}. Für einen Pfad P in G_{Erw} definiere $E_M(P) \coloneqq E(P) \cap M$ die Menge der Matchingkanten und $E_{\bar{M}}(P) \coloneqq E(P) \setminus M$ die Menge der Nicht-Matchingkanten*

von P. Dann heißt P eindeutig M-augmentierend*, wenn er M-augmentierend ist und $E_{\bar{M}}(P) \cup M \setminus E_M(P)$ keine Klonkanten enthält.*

Mithilfe von eindeutig augmentierenden Pfaden können eindeutige Matchings so erweitert werden, dass die entstehenden Matchings erneut eindeutig sind.

Korollar 6.5 *Seien M ein eindeutiges Matching in einem Klongraphen und P ein eindeutig M-augmentierender Pfad. Dann ist das Matching $M' \coloneqq M \,\Delta\, E(P)$, welches aus M durch Augmentieren entlang von P entsteht, ebenfalls eindeutig.*

Beweis: Wenn M' nicht eindeutig ist, gibt es mindestens zwei Klonkanten e_1 und e_2 in M'. Aus der Eindeutigkeit von M folgt, dass mindestens eine dieser beiden Kanten in $E_{\bar{M}}(P)$ sein muss. Beide Kanten können nicht in $E_{\bar{M}}(P)$ sein, denn dann wäre P nicht eindeutig. O. B. d. A. sei $e_1 \in E_{\bar{M}}(P)$ und $e_2 \notin E_{\bar{M}}(P)$. Dann muss aber $e_2 \in M$ gelten, was erneut ein Widerspruch zur Eindeutigkeit von P ist. □



(a) G_{Erw} (b) M (c) Drei M-augmentierende Pfade

Abbildung 6.8: Vergrößern von Matchings mithilfe augmentierender Pfade

Beispiel 6.9 In Abbildung 6.8 ist zu einem Klongraphen G_{Erw} ein Matching M gegeben (siehe farbige Kanten in (b)). In G_{Erw} gibt es zwei Knoten für das alte Tupel 2, die zur besseren Unterscheidung mit 2 und $2'$ gelabelt sind. In Teil (c) sind exemplarisch drei M-augmentierende Pfade dargestellt: $P_1 = 1, \text{B}(5), 2, \text{E}(7), 4, \text{D}(8), 2', \text{E}(9)$, $P_2 = 1, \text{B}(5), 2, \text{C}(6), 3, \text{D}(8), 2', \text{E}(9)$ und $P_3 = 1, \text{B}(5), 2, \text{E}(9)$, wobei die Knoten hier mithilfe ihrer Labels beschrieben werden. Die Matchingkanten $E_M(P)$ von P sind farblich wie die Matchingkante von M gekennzeichnet, die Menge der grünen Kanten bildet $E_{\bar{M}}(P)$. P_1 ist offensichtlich kein eindeutig M-augmentierender Pfad, da hier die beiden Klonkanten $\{2, \text{E}(7)\}$ und $\{2', \text{E}(9)\}$ vorkommen. P_2 und P_3 hingegen erfüllen die Eigenschaften eindeutig augmentierender Pfade.

Um zu testen, ob ein M-augmentierender Pfad P auch eindeutig ist, muss nach Definition 6.4 überprüft werden, ob in $E_{\bar{M}}(P) \cup M \setminus E_M(P)$ zwei Klonkanten vorkommen. Ist M eindeutig, können nicht beide Kanten in M vorkommen, sondern entweder beide in $E_{\bar{M}}(P)$ oder eine Kante in $E_{\bar{M}}(P)$ und eine in $M \setminus E_M(P)$. Das folgende Lemma zeigt, dass es ausreicht, nur Klonkanten zu untersuchen, von denen eine in $E_{\bar{M}}(P)$ und eine in M ist. Es liefert dadurch ein effizientes Verfahren, um augmentierende Pfade zu berechnen, die eindeutig sind.

Lemma 6.6 *Seien M ein eindeutiges Matching in einem Klongraphen und P ein M-augmentierender Pfad. Dann existiert ein M-augmentierender Pfad P', der keine Klonknoten und keine SA-äquivalente Kanten enthält.*

Beweis: Der Beweis erfolgt indirekt durch eine Fallunterscheidung. Im ersten Fall sei P ein kürzester M-augmentierender Pfad, welcher zwei Klonknoten t und t' enthält. Da P ungerade Länge hat, können nicht beide Klonknoten Endknoten des Pfades sein. Sei t Endknoten, dann hat P die Form $P = t, P_1, s_1, t', s_2, P_2$ (oder genau umgekehrt), wobei s_1 und s_2 zwei neue Knoten sowie P_1 und P_2 Teile des Pfades sind. Somit ist t nicht überdeckt und $\{t', s_2\}$ keine Matchingkante aus M. Aus der Konstruktion des Klongraphen folgt, dass es auch eine Kante $\{t, s_2\}$ geben muss, die ebenfalls keine Matchingkante ist. Folglich ist $P' \coloneqq t, s_2, P_2$ ein augmentierender Pfad, der kürzer ist als P. Ein ganz analoges Argument liefert einen Widerspruch, wenn t und t' keine Endknoten sind. P hat dann die Form $P = P_1, s_1, t, P_2, t', s_2, P_3$, wobei entweder $\{s_1, t\}$ oder $\{t', s_2\}$ eine Matchingkante ist. Wenn $\{s_1, t\} \in M$ gilt, dann ist $P' \coloneqq P_1, s_1, t, s_2, P_3$, und wenn $\{t', s_2\} \in M$ gilt, dann ist $P'' \coloneqq P_1, s_1, t', s_2, P_3$ ein jeweils kürzerer augmentierender Pfad.

Im zweiten Fall sei P erneut kürzester M-augmentierender Pfad, der diesmal zwei SA-äquivalente Kanten $\{t_1, s_1\}$ und $\{t_1, s_2\}$ enthält. O. B. d. A. sei s_1 kein Endknoten, dann hat P die Form $P = P_1, t_0, s_1, t_1, s_2, P_2$, wobei t_0 ein alter Knoten sowie P_1 und P_2 Teile des Pfades sind. Wenn $\{t_0, s_1\}$ und $\{t_1, s_2\}$ Matchingkanten sind, kann P_2 in $P_2 = t_2, P_2'$ aufgeteilt werden (t_2 ist erneut ein alter Knoten und P_2' ein Teil des Pfades). Da s_1 und s_2 zu SA-äquivalenten Kanten gehören, haben sie denselben SA-Wert und es gibt nach Konstruktion des Graphen auch eine Kante $\{s_1, t_2\}$, die ungematcht ist. Somit ist $P' \coloneqq P_1, t_0, s_1, t_2, P_2'$ ein augmentierender Pfad, der kürzer ist als P. Wenn $\{s_1, t_1\}$ eine Matchingkante ist, muss es im Graphen auch eine Nicht-Matchingkante $\{t_0, s_2\}$ geben und erneut ist mit $P' \coloneqq P_1, t_0, s_2, P_2$ ein kürzerer augmentierender Pfad gefunden. □

Korollar 6.7 *Seien M ein eindeutiges Matching in einem Klongraphen und P ein kürzester M-augmentierender Pfad. Dann enthält P keine Klonknoten und keine SA-äquivalenten Kanten.*

Beweis: Falls P Klonknoten oder SA-äquivalente Kanten enthält, kann analog zum Beweis von Lemma 6.6 ein kürzerer M-augmentierender Pfad gefunden werden. □

Beispiel 6.10 Der Klongraph G_{Erw} in Abbildung 6.8 hat die beiden Klonknoten 2 und $2'$ sowie mit E(7) und E(9) zwei neue Knoten mit demselben SA-Wert. Wenn es in G_{Erw} zu einem beliebigen eindeutigen Matching M einen beliebigen M-augmentierenden Pfad gibt, dann existiert nach Lemma 6.6 auch immer ein M-augmentierender Pfad P', in dem nicht beide Knoten für Tupel 2 vorkommen. Weiterhin enthält P' auch nicht zwei Kanten der Form $\{t, \text{E}(7)\}$ und $\{t, \text{E}(9)\}$, wobei t ein beliebiger alter Knoten ist.[13] Zu den in Teil (c) angegebenen Pfaden P_1 und P_2 existiert also auch immer noch ein anderer (kürzerer) Pfad, wie zum Beispiel P_3.

Man beachte, dass Lemma 6.6 eine Aussage über (allgemeine) augmentierende Pfade ist und die speziellen eindeutigen Pfade zunächst nicht weiter beachtet werden. Es besagt aber, dass es einen augmentierenden Pfad P' gibt, der keine Klonknoten und damit auch

[13] In diesem Fall kann t nur ein Knoten für 2 oder 4 sein, da das die einzigen Knoten sind, die Kanten zu E(7) und E(9) haben.

keine Klonkanten enthält. Ein Augmentieren mithilfe von P' würde auch keinen alten Knoten t „SA-äquivalent ummatchen“, das heißt, aus einer Matchingkante $\{t, s\}$ eine andere Matchingkante $\{t, s'\}$ konstruieren, die denselben SA-Wert hat.[14] P' ist damit eindeutig augmentierend bezüglich eines eindeutigen Matchings M, wenn es nicht zwei Klonkanten gibt, von denen eine in $E_{\bar{M}}(P')$ und eine in M ist. Die anderen oben genannten Möglichkeiten sind durch Lemma 6.6 für P' ausgeschlossen.

Mit diesen Überlegungen kann ein Algorithmus angegeben werden, der eindeutig M-augmentierende Pfade findet. Die einzigen beiden Unterschiede zum Standardverfahren für allgemeine augmentierende Pfade ist, dass erstens alle Klonkanten von Kanten aus M nicht benutzt werden dürfen und zweitens pro Pfad auch keine Klonknoten vorkommen dürfen. Wie im Beweis von Lemma 6.6 gesehen, erfüllen alle kürzesten augmentierenden Pfade immer die zweite Einschränkung, sodass diese sehr leicht ohne zusätzlichen Aufwand überprüft werden kann. Insgesamt folgt, dass die Suche nach einem eindeutig augmentierenden Pfad den gleichen Aufwand erfordert wie die Suche nach einem beliebigen kürzesten augmentierenden Pfad.[15]

Durch das Augmentieren entlang eines eindeutigen Pfades vergrößert sich ein Matching um 1. Damit auf diese Weise analog zu einfachen Graphen auch ein größtes Matching gefunden wird, muss gezeigt werden, dass in einem größten eindeutigen Matching kein eindeutig augmentierender Pfad existiert. Das ist ein ähnlicher Satz wie der Satz von Berge für allgemeine Matchings (vgl. Satz 1.1).

Satz 6.8 *Ein Matching M ist genau dann ein größtes eindeutiges Matching in einem Klongraphen, wenn kein eindeutig M-augmentierender Pfad existiert.*

Beweis: Falls es einen eindeutig M-augmentierenden Pfad P gibt, so ist M offensichtlich kein größtes Matching, denn durch Vertauschen von Matchingkanten und Nicht-Matchingkanten entlang P erhält man nach Korollar 6.5 ein größeres eindeutiges Matching. Somit gibt es bezüglich eines größten eindeutigen Matchings M keine eindeutig M-augmentierenden Pfade.

Man nehme nun an, dass M kein größtes eindeutiges Matching ist. Sei M' Matching mit $|M'| > |M|$. Dann kann M' nach einer einfachen Regel in ein anderes eindeutiges Matching derselben Größe transformiert werden. Betrachte dazu jede Kante $\{t, s\} \in M'$, zu der eine Klonkante $\{t', s'\}$ in M vorkommt, für die $s' \neq s$ gilt. Ersetze $\{t, s\}$ durch die Kante $\{t', s'\}$. Falls t' in M' durch eine Kante $\{t', s^*\}$ überdeckt ist, ersetze diese Kante durch $\{t, s^*\}$. Analog ersetze die Kante $\{t^*, s'\}$ durch $\{t^*, s\}$, falls s' durch die Kante $\{t^*, s'\}$ überdeckt war.

Durch die Transformation ändert sich offensichtlich weder die Größe noch die Eindeutigkeit von M'. Da die Anzahl Kanten, die zwar in M', aber nicht in M sind, in jedem Schritt um mindestens 1 verringert wird, terminiert die wiederholte Anwendung des Ersetzungsschrittes mit einem eindeutigen Matching, welches mit M^* bezeichnet wird. Betrachte nun die symmetrische Differenz $M \,\Delta\, M^*$. Diese induziert einen Graphen mit Maximalgrad 2 und besteht somit aus der Vereinigung von geraden Kreisen und Pfaden. Da $|M^*| > |M|$ gilt, muss es insbesondere auch mindestens einen Pfad P geben, der mit einer Kante aus

[14] Da in P' keine Klonknoten vorkommen, kann ein solches „SA-äquivalente Ummatchen“ nur durch einen Teilpfad s, t, s' gelingen, wobei in P' zwei SA-äquivalente Kanten vorkommen würden.

[15] Das zusätzliche Verbieten der Benutzung von Kanten geht offensichtlich in Laufzeit $\mathcal{O}(m)$, was höchstens so groß ist wie die Konstruktion eines Pfades.

M^* beginnt und mit einer endet. Dieser ist also M-augmentierend. Es bleibt zu zeigen, dass P auch eindeutig ist.

Wenn P nicht eindeutig ist, muss es zwei Klonkanten $\{t, s\}$ und $\{t', s'\}$ in $E_{\bar{M}}(P) \cup M \setminus E_M(P)$ geben. Da M und M^* eindeutig sind sowie $E_{\bar{M}}(P) \subseteq M^*$ gilt, muss die eine Klonkante in $E_{\bar{M}}(P)$ und die andere in $M \setminus E_M(P)$ vorkommen. Seien $\{t, s\} \in E_{\bar{M}}(P)$ und $\{t', s'\} \in M \setminus E_M(P)$, dann gilt $\{t, s\} \in M^*$ und $\{t', s'\} \in M$. Ferner ist $\{t', s'\} \notin E(P)$, wodurch $s \neq s'$ folgt. Genau dieser Fall ist aber durch die vorige Transformation des Matchings ausgeschlossen worden, was beweist, dass P auch eindeutig ist. □

Der letzte Satz besagt, dass das Vergrößern eines Matchings mithilfe eindeutig augmentierender Pfade zu einem größten eindeutigen Matching in einem Klongraphen führt. Der folgende Algorithmus 6.7 berechnet auf diese Weise ein größtes eindeutiges Matching mithilfe kürzester augmentierender Pfade.

Algorithmus 6.7 Größtes eindeutiges Klonmatching

Eingabe: G_{Erw}: Klongraph
Ausgabe: M: größtes eindeutiges Matching in G_{Erw}

```
1:  M ← ∅
2:  repeat
3:    Entferne aus G_Erw alle Klonkanten von Kanten aus M
4:    S ← Menge nicht überdeckter alter Knoten
5:    T ← Menge nicht überdeckter neuer Knoten
6:    Bestimme mittels Breitensuche einen kürzesten augmentierenden Pfad P zwischen
      einem Knoten aus S und einem aus T
7:    if P existiert then
8:      Augmentiere M entlang P
9:      Füge in G_Erw die zuvor entfernten Kanten wieder ein
10:   end if
11: until P existiert nicht
12: return M
```

Satz 6.9 *Algorithmus 6.7 bestimmt ein größtes eindeutiges Matching in einem Klongraphen und hat Laufzeit $\mathcal{O}(nm)$.*

Beweis: Da berechnete augmentierende Pfade keine Kanten benutzen, die Klonkanten zu Kanten aus M sind, folgt der Beweis der Korrektheit direkt aus Lemma 6.6 und Satz 6.8.

Ohne Einschränkung kann $n = \mathcal{O}(m)$ angenommen werden. Die Repeat-Until-Schleife wird maximal $n/2$-mal durchlaufen, da das Matching in jedem Schritt um mindestens 1 vergrößert wird. Das Entfernen und Hinzufügen von Kanten benötigt Laufzeit $\mathcal{O}(m)$ und die Breitensuche $\mathcal{O}(n + m)$. Hierbei werden in jedem Schritt ausgehend von einem alten Knoten aus S alle dessen Nachbarn besucht, diese aber nicht in die Warteschlange aufgenommen, sondern alle alten Knoten, die mit diesen Nachbarn über eine Matchingkante verbunden sind. Das ergibt insgesamt eine Laufzeit von $\mathcal{O}(nm)$. □

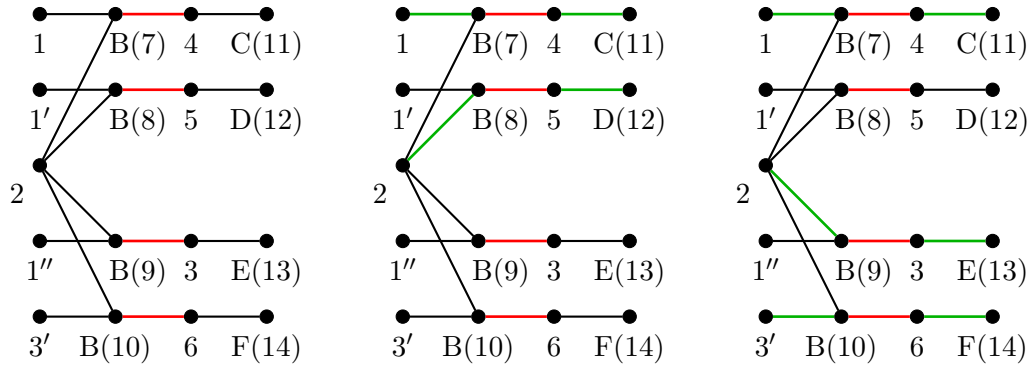
6.6.4 Algorithmus von Hopcroft und Karp für Klongraphen

Der schnellste bekannte Algorithmus für das Matchingproblem in einfachen bipartiten Graphen stammt von Hopcroft und Karp (siehe Algorithmus 1.2) und hat eine Laufzeit

von $\mathcal{O}(\sqrt{n}m)$. Er basiert darauf, dass sich die Länge eines kürzesten augmentierenden Pfades bezüglich eines Matchings M vergrößert, wenn M gleichzeitig mithilfe einer inklusionsmaximalen Menge knotendisjunkter Pfade augmentiert wird (vgl. Lemma 1.4). Um diese Idee auch auf eindeutige Matchings anzuwenden, muss zusätzlich beachtet werden, dass das gleichzeitige Vergrößern von Matchings mithilfe beliebiger jeweils eindeutiger Pfade nicht notwendigerweise zu eindeutigen Matchings führen muss. Enthält beispielsweise ein eindeutig augmentierender Pfad P_i eine Nicht-Matchingkante e, darf nicht gleichzeitig mit einem anderen Pfad P_j augmentiert werden, der eine Klonkante zu e ebenfalls als Nicht-Matchingkante enthält. Das gilt sogar, wenn P_i und P_j knotendisjunkt und jeweils eindeutig augmentierend sind. Weiterhin kann aber gleichzeitig mit einem Pfad P_k augmentiert werden, der selbst nicht eindeutig ist, da er eine Klonkante zu einer Matchingkante von P_i als Nicht-Matchingkante enthält.

Definition 6.5 ((streng) eindeutig augmentierende Menge) *Sei M ein eindeutiges Matching in einem Klongraphen G_{Erw}. Für eine Menge S_P von Pfaden $P_1, \dots, P_t$ definiere $E_M(S_P) \coloneqq E_M(P_1) \cup \dots \cup E_M(P_t)$ die Menge der Matchingkanten und $E_{\bar{M}}(S_P) \coloneqq E_{\bar{M}}(P_1) \cup \dots \cup E_{\bar{M}}(P_t)$ die Menge der Nicht-Matchingkanten von S_P. Dann heißt S_P* eindeutig M-augmentierend*, wenn alle P_i knotendisjunkt und M-augmentierend sind sowie $E_{\bar{M}}(S_P) \cup M \setminus E_M(S_P)$ keine Klonkanten enthält. S_P heißt* streng eindeutig M-augmentierend*, wenn zusätzlich alle P_i eindeutig M-augmentierend sind.*

Zur Unterscheidung von eindeutig und streng eindeutig augmentierenden Mengen von Pfaden werden erstere auch *schwach eindeutig augmentierende Mengen* genannt. Den Unterschied zwischen beiden Begriffen veranschaulicht das folgende Beispiel.



(a) M (b) Augmentierende Pfade (c) Augmentierende Pfade

Abbildung 6.9: Eindeutig M-augmentierende Menge von Pfaden im Klongraphen G_{Erw}. Die farbigen Kanten in (a) zeigen ein eindeutiges Matching M in einem Klongraphen, wobei der Übersicht halber hier nicht alle Kanten von G_{Erw} dargestellt sind. In (b) und (c) sind zwei eindeutig M-augmentierende Mengen von Pfaden dargestellt (beginnend und endend mit grünen Nicht-Matchingkanten), wobei der Pfad $3', \text{B}(10), 6, \text{F}(14)$ selbst nicht eindeutig M-augmentierend ist.

Beispiel 6.11 In Abbildung 6.9 ist in (a) ein Klongraph dargestellt, bei dem der Übersicht halber nicht alle Kanten angegeben sind. Die farbig markierten Kanten bilden ein eindeutiges Matching M. Teil (b) zeigt zwei augmentierende Pfade $P_1 = 1, \text{B}(7), 4, \text{C}(11)$ und

$P_2 = 2, \mathrm{B}(8), 5, \mathrm{D}(12)$. Die Menge $S_{P_{12}} = \{P_1, P_2\}$ ist nach Definition 6.5 eine streng eindeutig M-augmentierende Menge. Das Hinzufügen von $P_3 = 1'', \mathrm{B}(9), 3, \mathrm{E}(13)$ zu $S_{P_{12}}$ würde die Eindeutigkeit verletzen, denn dann wären die Klonkanten $\{1, B(7)\}$ und $\{1'', B(9)\}$ in $E_{\bar{M}}(S_{P_{12}})$. Ebenfalls kann $P_4 = 3', \mathrm{B}(10), 6, \mathrm{F}(14)$ nicht hinzugefügt werden, da dann die Kanten $\{3, B(9)\}$ und $\{3', B(10)\}$ in $E_{\bar{M}}(S_{P_{12}}) \cup M \setminus E_M(S_{P_{12}})$ enthalten wären. Sofern es keine anderen augmentierenden Pfade gibt, ist $S_{P_{12}}$ inklusionsmaximal.

In (c) sind insgesamt drei augmentierende Pfade abgebildet. Auch die Menge $S_{P_{145}}$ dieser drei Pfade ist eindeutig M-augmentierend. Hierbei ist der Pfad P_4 selbst nicht eindeutig M-augmentierend, da er die Nicht-Matchingkante $\{3', \mathrm{B}(10)\}$ enthält, zu der die Matchingkante $\{3, \mathrm{B}(9)\}$ eine Klonkante darstellt. Da allerdings $P_5 = 2, \mathrm{B}(9), 3, \mathrm{E}(13)$ ebenfalls Element von $S_{P_{145}}$ ist, gilt $\{3, \mathrm{B}(9)\} \in E_M(S_{P_{145}})$. Die Menge $S_{P_{145}}$ ist damit schwach eindeutig. Sie ist aber ebenfalls inklusionsmaximal, denn das Hinzufügen des einzigen zu $S_{P_{145}}$ knotendisjunkten Pfades $1', \mathrm{B}(8), 5, \mathrm{D}(12)$ $S_{P_{145}}$ würde die Eindeutigkeit verletzen (Klonkanten $\{1, \mathrm{B}(7)\}$ und $\{1', \mathrm{B}(8)\}$).

Direkt aus Definition 6.5 folgt, dass beim Augmentieren mithilfe eindeutig augmentierender Mengen die Eigenschaft der Eindeutigkeit von Matchings erhalten bleibt.

Korollar 6.10 *Seien M ein eindeutiges Matching in einem Klongraphen und $S_P = \{P_1, \ldots, P_t\}$ eine eindeutig M-augmentierende Menge knotendisjunkter Pfade. Dann ist das Matching $M' \coloneqq M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)$, welches aus M durch Augmentieren entlang der Pfade $P_1, \ldots, P_t$ entsteht, ebenfalls eindeutig.*

Beweis: Wenn M' nicht eindeutig ist, gibt es mindestens zwei Klonkanten e_1 und e_2 in M'. Aus der Eindeutigkeit von M folgt, dass mindestens eine dieser beiden Kanten in $E_{\bar{M}}(S_P)$ enthalten sein muss. Beide Kanten können nicht in $E_{\bar{M}}(S_P)$ sein, denn dann wäre die Menge S_P nicht eindeutig. O. B. d. A. sei $e_1 \in E_{\bar{M}}(S_P)$ und $e_2 \notin E_{\bar{M}}(S_P)$. Dann muss aber $e_2 \in M$ gelten, was erneut ein Widerspruch zur Eindeutigkeit von S_P ist. □

Analog zum Matchingproblem in einfachen Graphen muss für den Algorithmus von Hopcroft und Karp eine inklusionsmaximale eindeutig augmentierende Menge S_P von Pfaden bestimmt werden. Das ist zum einen schwieriger als das Bestimmen einer beliebigen Menge augmentierender Pfade, denn die einzelnen Pfade können nicht mehr unabhängig voneinander gewählt werden. Zum anderen muss auch bewiesen werden, dass dadurch ein Laufzeitgewinn gegenüber dem Standardverfahren (siehe Algorithmus 6.7) erreicht wird. Das bedeutet insbesondere, dass sich die Länge kürzester eindeutig augmentierender Pfade immer vergrößern soll, wenn ein Matching mithilfe von S_P augmentiert wird.

Die folgenden beiden Sätze zeigen, dass dieses Vorgehen korrekt ist und die gewünschte Laufzeit garantiert. Dabei werden analog zum Matching auf einfachen Graphen Aussagen über Anzahl und Länge kürzester eindeutig augmentierender Pfade getroffen (vgl. Lemmata 1.2 und 1.4). Der für Klongraphen modifizierte Algorithmus von Hopcroft und Karp wird danach in Algorithmus 6.8 vorgestellt.

Korollar 6.11 *Seien M_1 und M_2 zwei eindeutige Matchings in einem Klongraphen mit $|M_1| > |M_2|$. Dann existieren ein eindeutiges Matching M_1^* mit $|M_1^*| = |M_1|$ und mindestens $|M_1| - |M_2|$ viele (knoten-)disjunkte eindeutig M_2-augmentierende Pfade, die nur Kanten aus $M_1^* \Delta M_2$ enthalten. Zusätzlich ist die Menge S_P, die genau diese Pfade enthält, streng eindeutig M_2-augmentierend.*

Beweis: Der Beweis der Existenz der Pfade folgt direkt aus dem Beweis von Satz 6.8, wobei M_2^* das durch die angegebene Transformation konstruierte Matching ist. Da M_1 und M_2^* eindeutig sind, ist S_P offensichtlich streng eindeutig M_2-augmentierend. □

Für ein eindeutiges Matching M sei $l_e(M)$ die Länge eines kürzesten eindeutig M-augmentierenden Pfades. Falls kein eindeutig augmentierender Pfad existiert, so setze $l_e(M) = \infty$. Nach Korollar 6.7 gilt für kürzeste M-augmentierende Pfade P $E_{\bar{M}}(P) \cap E_M(P) = \emptyset$. Demnach ist P auch eindeutig M-augmentierend, wenn es keine Klonkanten in $E_{\bar{M}}(P) \cup M$ gibt.

Lemma 6.12 *Seien M ein eindeutiges Matching in einem Klongraphen und $S_P = \{P_1, \ldots, P_t\}$ eine (inklusions-)maximale streng eindeutig M-augmentierende Menge knotendisjunkter Pfade der Länge $l_e(M)$. Dann gilt*

$$l_e(M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)) > l_e(M).$$

Beweis: Sei Q ein eindeutig augmentierender Pfad in $M' \coloneqq M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)$. Dann gilt $|M' \,\Delta\, E(Q)| = |M| + t + 1$, denn $|M'| = |M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)| = |M| + t$ und Q ist M'-augmentierend. Nach Korollar 6.11 gibt es somit ein Matching M^* mit $|M^*| = |M' \,\Delta\, E(Q)|$ und $t+1$ knotendisjunkte M-augmentierende Pfade $Q_1, \ldots, Q_{t+1}$ in $M \,\Delta\, M^*$, die alle Länge mindestens $l_e(M)$ haben. Man erhält:

$$(t+1) \cdot l_e(M) \leq |E(Q_1)| + \cdots + |E(Q_{t+1})| \leq |M \,\Delta\, M^*|.$$

Die letzte Ungleichung gilt wegen $E(Q_i) \subseteq M \,\Delta\, M^*$. M^* kann nach Korollar 6.11 direkt aus $M' \,\Delta\, E(Q)$ konstruiert werden, wobei Kanten aus $M' \,\Delta\, E(Q)$ durch Klonkanten aus M ersetzt werden (vgl. Beweis von Satz 6.8). Folglich gilt $|M \,\Delta\, M^*| \leq |M \,\Delta\, (M' \,\Delta\, E(Q))|$ und damit

$$(t+1) \cdot l_e(M) \leq |M \,\Delta\, (M' \,\Delta\, E(Q))|.$$

Aus der Definition von M' und der Eigenschaft, dass die P_i knotendisjunkt sind, folgt[16]

$$\begin{aligned} |M \,\Delta\, (M' \,\Delta\, E(Q))| &= |E(P_1) \,\Delta \cdots \Delta\, E(P_t) \,\Delta\, E(Q)| \\ &= |(E(P_1) \cup \cdots \cup E(P_t)) \,\Delta\, E(Q)| \\ &= |E(P_1) \cup \cdots \cup E(P_t)| + |E(Q)| - 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| \\ &= t \cdot l_e(M) + |E(Q)| - 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)|. \end{aligned}$$

Somit gilt also

$$|E(Q)| \geq l_e(M) + 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)|.$$

Falls Q nicht knotendisjunkt zu $P_1 \cup \cdots \cup P_t$ ist, so ist Q auch nicht kantendisjunkt zu $P_1 \cup \cdots \cup P_t$ (da in M' alle Knoten der P_i überdeckt sind) und es gilt daher $|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| > 0$. Somit erhält man in diesem Fall

$$|E(Q)| \geq l_e(M) + 2|(E(P_1) \cup \cdots \cup E(P_t)) \cap E(Q)| > l_e(M).$$

[16] Man beachte hierbei, dass für zwei Mengen A und B gilt: $|A \,\Delta\, B| = |A \setminus B| + |B \setminus A| = |A| - |A \cap B| + |B| - |A \cap B|$.

Falls Q knotendisjunkt zu $P_1 \cup \cdots \cup P_t$ und eindeutig M-augmentierend ist, so muss auch $S_P \cup \{Q\}$ streng eindeutig M-augmentierend sein, denn andernfalls wäre Q nicht eindeutig M'-augmentierend.[17] Damit muss $|E(Q)| > l_e(M)$ gelten, denn andernfalls wären die P_i nicht inklusionsmaximal.

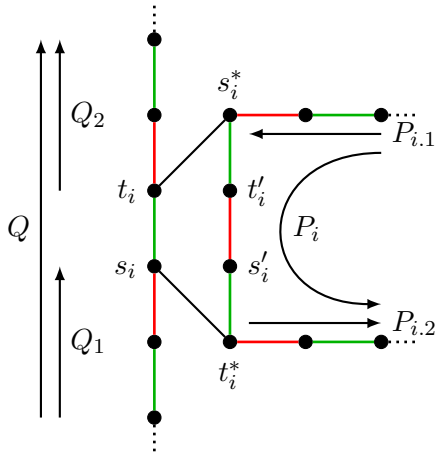


Abbildung 6.10: Augmentierende Pfade Q und P_i

Es bleibt zu zeigen, dass die Länge von Q ebenfalls größer als $l_e(M)$ ist, wenn Q in diesem Fall nicht eindeutig M-augmentierend ist. Hierbei muss es mindestens eine Kante $\{t_i, s_i\} \in E_{\bar{M}}(Q)$ geben, die eine Klonkante $\{t_i', s_i'\} \in M$ hat (siehe Abbildung 6.10). Da Q eindeutig M'-augmentierend ist, muss es einen Pfad $P_i \in S_P$ geben mit $\{t_i', s_i'\} \in E_M(P_i)$. Dann hat P_i mindestens Länge 3 und die Form $P_i = P_{i.1}, s_i^*, t_i', s_i', t_i^*, P_{i.2}$. Sei $Q = Q_1, s_i, t_i, Q_2$, dann muss es nach Konstruktion des Klongraphen auch die Kanten $\{t_i, s_i^*\}$ und $\{t_i^*, s_i\}$ geben, die offensichtlich nicht in M vorkommen. Man nehme an, dass $l(Q) = |E(Q)| \leq l_e(M)$ gilt, dann sind $P_1 = Q_1, s_i, t_i^*, P_{i.2}$ und $P_2 = P_{i.1}, s_i^*, t_i, Q_2$ zwei Pfade und es folgt

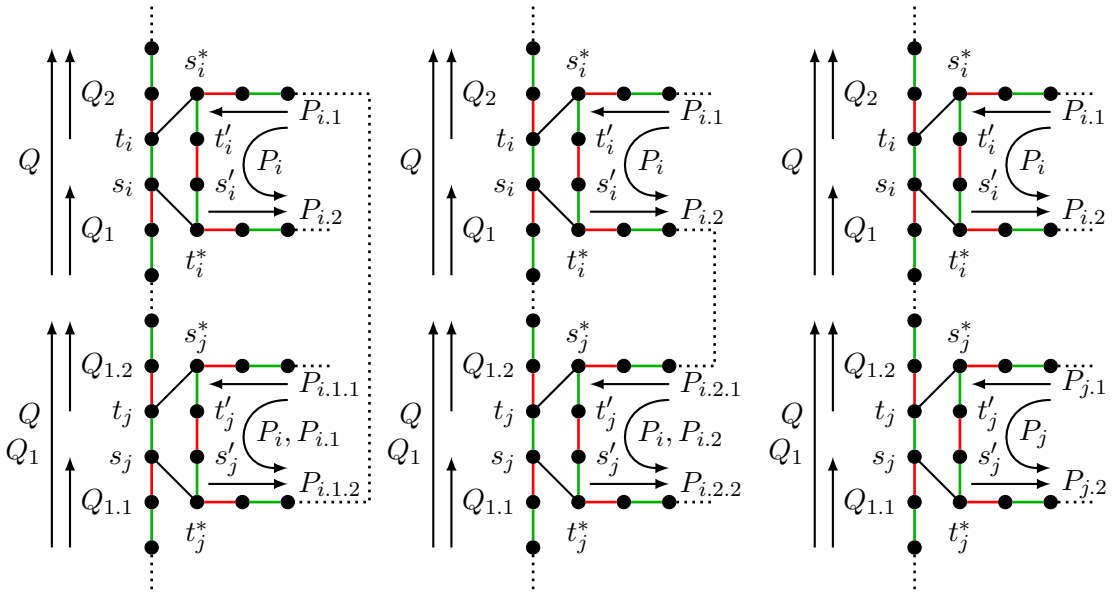
$$\begin{aligned} l(P_1) + l(P_2) &= l(Q_1) + 1 + l(P_{i.2}) + l(P_{i.1}) + 1 + l(Q_2) \\ &= l(Q) + l(P_i) - 2 \\ &\leq 2l_e(M) - 2 \\ &< 2l_e(M). \end{aligned}$$

Somit muss P_1 oder P_2 kürzer sein als $l_e(M)$. O. B. d. A. sei $l(P_1) < l_e(M)$ und die Kante $\{t_i, s_i\} \in Q$ so gewählt (falls mehrere solcher Kanten existieren), dass der Teilpfad Q_1 am kürzesten ist. Zu zeigen ist nun, dass P_1 auch eindeutig M-augmentierend ist. Da Q knotendisjunkt zu allen Pfaden aus S_P ist, gilt $E_{\bar{M}}(Q) = E_{\bar{M}'}(Q)$ und $E_M(Q) = E_{M'}(Q)$ und folglich ist Q und damit P_1 M-augmentierend. Wenn P_1 nicht eindeutig ist, muss es eine zweite Kante $\{t_j, s_j\} \in E_{\bar{M}}(Q_1)$ geben, die eine Klonkante $\{t_j', s_j'\}$ in M beziehungsweise S_P hat.[18] $\{t_j, s_j\}$ sei die letzte solche Kante auf dem Teilpfad Q_1, wodurch Q_1 die Form $Q_{1.1}, s_j, t_j, Q_{1.2}$ hat (d. h., $Q_{1.2}$ enthält keine andere Klonkante zu einer

[17] Ein formaler Beweis zu dieser offensichtlichen Behauptung befindet sich im Anhang in Lemma B.2 auf Seite 303.

[18] Aufgrund der minimalen Länge von P und Q_1 und der Eigenschaft, dass Q eindeutig M'-augmentierend ist, können zwei Klonkanten nicht in $E_{\bar{M}}(P_1)$ vorkommen (vgl. Korollar 6.7).

Matchingkante). Der Pfad, der die Klonkante $\{t'_j, s'_j\}$ enthält, muss erneut zwei Knoten t^*_j und s^*_j mit den Nicht-Matchingkanten $\{t'_j, s^*_j\}$ und $\{t^*_j, s'_j\}$ enthalten. Im Klongraphen existieren ebenso die Nicht-Matchingkanten $\{t_j, s^*_j\}$ und $\{t^*_j, s_j\}$ (siehe Abbildung 6.11).



(a) Fall 1: $\{t'_j, s'_j\} \in P_{i.1}$ (b) Fall 2: $\{t'_j, s'_j\} \in P_{i.2}$ (c) Fall 3: $\{t'_j, s'_j\} \in P_j$

Abbildung 6.11: Mögliche Fälle, welcher Pfad die 2. Klonkante $\{t'_j, s'_j\}$ enthält

Nun gibt es drei Möglichkeiten, welcher Pfad die Klonkante $\{t'_j, s'_j\}$ enthalten kann. Sei im ersten Fall $P_{i.1}$ dieser Pfad und $P_{i.1} = P_{i.1.1}, s^*_j, t'_j, s'_j, t^*_j, P_{i.1.2}$[19] (siehe (a)). Dann betrachte die Pfade $P_{1.1} = Q_{1.1}, s_j, t^*_j, P_{i.1.2}, s^*_i, t'_i, s'_i, t^*_i, P_{i.2}$ und $P_{1.2} = P_{i.1.1}, s^*_j, t_j, Q_{1.2}, s_i, t^*_i, P_{i.2}$. Es folgt

$$\begin{aligned} l(P_{1.1}) + l(P_{1.2}) &= l(Q_{1.1}) + 1 + l(P_{i.1.2}) + 3 + l(P_{i.2}) \\ &\quad + l(P_{i.1.1}) + 1 + l(Q_{1.2}) + 1 + l(P_{i.2}) \\ &= l(Q_1) + l(P_{i.2}) + l(P_{i.1}) + l(P_{i.2}) + 2 \\ &= l(P_1) + l(P_i) - 2 \\ &< 2l_e(M). \end{aligned}$$

Somit muss $P_{1.1}$ oder $P_{1.2}$ kürzer sein als $l_e(M)$. $l(P_{1.1}) < l_e(M)$ verstößt gegen die Wahl von $\{t_i, s_i\} \in Q$, denn $Q_{1.1}$ ist kürzer als Q_1. Da $P_{1.2}$ offensichtlich eindeutig M-augmentierend ist,[20] verletzt $l(P_{1.2}) < l_e(M)$ die Wahl der P_t.

Im zweiten Fall sei $\{t'_j, s'_j\} \in P_{i.2}$ und $P_{i.2} = P_{i.2.1}, s^*_j, t'_j, s'_j, t^*_j, P_{i.2.2}$[21] (siehe (b)). Dann betrachte den Pfad $P_{1.1} = Q_{1.1}, s_j, t^*_j, P_{i.2.2}$. Offensichtlich gilt $l(P_{1.1}) < l(P_1) < l_e(M)$, was allerdings gegen die Wahl von $\{t_i, s_i\} \in Q$ verstößt, denn $Q_{1.1}$ ist kürzer als Q_1.

[19] Da der Klongraph bipartit und $P_{i.1}$ M-alternierend ist, ist die andere Möglichkeit $P_{i.1} = P_{i.1.1}, t^*_j, s'_j, t'_j, s^*_j, P_{i.1.2}$ ausgeschlossen. Im Spezialfall $t^*_j = t'_i, s'_j = s^*_i$ gilt $P_{i.1} = P_{i.1.1}, s^*_j, t'_j, s'_j$ beziehungsweise $l(P_{i.1.2}) = -1$.

[20] Man beachte, dass $Q_{1.2}$ so gewählt ist, dass keine andere Klonkante zu einer Matchingkante vorkommt.

[21] Da der Klongraph bipartit und $P_{i.2}$ M-alternierend ist, ist die andere Möglichkeit $P_{i.2} = P_{i.2.1}, t^*_j, s'_j, t'_j, s^*_j, P_{i.2.2}$ ausgeschlossen. Im Spezialfall $t'_j = t^*_i, s^*_j = s'_i$ gilt $P_{i.2} = t'_j, s'_j, t^*_j, P_{i.2.2}$.

Sei im dritten Fall $P_j \in S_P$ der Pfad, der die Klonkante $\{t'_j, s'_j\}$ enthält (siehe (c)). Dann gilt $P_j = P_{j.1}, s^*_j, t'_j, s'_j, t^*_j, P_{j.2}$ und es existieren zwei Pfade $P_{1.1} = Q_{1.1}, s_j, t^*_j, P_{j.2}$ und $P_{1.2} = P_{j.1}, s^*_j, t_j, Q_{1.2}, s_i, t^*_i, P_{i.2}$. Es folgt

$$\begin{aligned} l(P_{1.1}) + l(P_{1.2}) &= l(Q_{1.1}) + 1 + l(P_{j.2}) + l(P_{j.1}) + 1 + l(Q_{1.2}) + 1 + l(P_{i.2}) \\ &= l(Q_1) + l(P_{i.2}) + l(P_j) - 1 \\ &= l(P_1) + l(P_j) - 2 \\ &< 2l_e(M). \end{aligned}$$

Somit muss auch hier $P_{1.1}$ oder $P_{1.2}$ kürzer sein als $l_e(M)$. $l(P_{1.1}) < l_e(M)$ verstößt erneut gegen die Wahl von $\{t_i, s_i\} \in Q$, denn $Q_{1.1}$ ist kürzer als Q_1. Da $P_{1.2}$ offensichtlich eindeutig M-augmentierend ist,[22] verletzt $l(P_{1.2}) < l_e(M)$ die Wahl der P_t.

Da in jedem der Fälle ein Widerspruch entsteht, muss die Annahme falsch sein und es folgt $l(Q) = |E(Q)| > l_e(M)$. □

Algorithmus 6.8 Hopcroft-Karp für Klongraphen

Eingabe: G_{Erw}: Klongraph
Ausgabe: M: größtes eindeutiges Matching in G_{Erw}

1: $M \leftarrow \emptyset$
2: **while** $l_e(M) < \infty$ **do**
3: Entferne aus G_{Erw} alle Klonkanten von Kanten aus M
4: $S \leftarrow$ Menge nicht überdeckter alter Knoten
5: $T \leftarrow$ Menge über alternierende Pfade der Länge $l_e(M)$ von S aus erreichbare (neue) Knoten
6: Bestimme mittels Tiefensuche von den Knoten in T eine maximale streng eindeutig augmentierende Menge von Pfaden der Länge $l_e(M)$ und augmentiere diese
7: Füge in G_{Erw} die zuvor entfernten Kanten wieder ein
8: **end while**
9: **return** M

Satz 6.13 *Der Algorithmus von Hopcroft und Karp findet ein größtes eindeutiges Matching in Klongraphen in* $\mathcal{O}(\sqrt{n}m)$.

Beweis: Die Korrektheit des Algorithmus folgt direkt aus Lemma 6.12. Dabei ist zu beachten, dass Klonkanten von Nicht-Matchingkanten eines gefundenen Pfades jeweils aus G_{Erw} entfernt werden, damit die berechnete Menge von Pfaden streng eindeutig ist (Zeile 6).

Für die Laufzeit gelten dieselben Betrachtungen wie für den *Hopcroft-Karp*-Algorithmus auf einfachen Graphen (vgl. Beweis von Satz 1.5 auf Seite 19). Daraus folgt, dass die While-Schleife $\mathcal{O}(\sqrt{n})$-mal durchlaufen wird und jeweils Laufzeit $\mathcal{O}(m)$ hat. Das zusätzliche Entfernen und Hinzufügen von Kanten geht ebenfalls in $\mathcal{O}(m)$, denn jede Kante wird innerhalb der While-Schleife maximal einmal entfernt und wieder hinzugefügt. □

[22] Man beachte, dass $Q_{1.2}$ so gewählt ist, dass keine andere Klonkante zu einer Matchingkante vorkommt.

Die Algorithmen 6.7 und 6.8 vervollständigen den vorgestellten Erweiterungsalgorithmus *Vereinfachtes Klonmatching* (siehe Algorithmus 6.6), indem im dort erstellten Klongraphen ein größtes Matching berechnet wird.

Satz 6.14 *Algorithmus 6.6 (Vereinfachtes Klonmatching) hat eine Laufzeit von* $\mathcal{O}(\log n \cdot n^{\frac{7}{2}})$*, wobei* n *die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Ein Klongraph enthält höchstens $2 \cdot \max\{|S_{\text{neu}}|, |S_{\text{alt}}|\} = \mathcal{O}(n)$ Knoten und demnach $\mathcal{O}(n^2)$ Kanten. Wird die Anzahl der erweiterbaren Kanten $|E_{\text{alt}}(t_{\text{alt}})|$ nach Gleichung 5.2 mit $\mathcal{O}(n)$ abgeschätzt, kann der Graph mittels Algorithmus 6.5 in $\mathcal{O}(n^2)$ erstellt werden. Somit besteht der einzige Laufzeitunterschied zum Algorithmus *Einfaches Matching* im Bestimmen von i, das durch eine binäre Suche in $\mathcal{O}(\log n)$ gefunden werden kann. Da vollständige Matchings nach Satz 6.13 in $\mathcal{O}(\sqrt{n}m)$ berechnet werden können, ergibt sich eine Gesamtlaufzeit von $\mathcal{O}(\log n \cdot n^{\frac{7}{2}})$ (analog zum Beweis von Satz 6.3). □

6.6.5 Ausführliches Klonmatching

Bisher wurde für den Klonmatchingansatz ein vereinfachtes Optimierungsproblem betrachtet, bei dem die Anzahl von Erweiterungen eines Tupels mit verschiedenem SA-Wert maximiert wird.[23] Der Unterschied zum originalen Problem ist, dass dabei der minimale Anonymitätsgrad maximiert wird.[24] Das bedeutet, es müssen auch Matchings beachtet werden, die nicht erweitert werden müssen. Das sind insbesondere Kanten, die von $\mathcal{T}_\Delta^{(n)}$ nach $\mathcal{T}_\Delta^{(n+1)}$ übertragen werden können.[25]

Abbildung 6.12 erläutert den Unterschied zwischen den beiden Problemen 6.3 und 6.2 anhand eines einfachen Beispiels. Dazu ist in (a) die Δ-top Matchingtabelle zu einer Folge von Anfragegraphen $\mathcal{G}^{(n)}$ gegeben. Ein weiterer Graph G_{n+1} besteht aus den alten Tupeln 1, 2, 3 und 4 sowie den neuen Tupeln 5 und 6. Da Tupel 1 und 2 denselben SA-Wert haben, werden sie zusammen mithilfe des Klonmatchingansatzes erweitert.

Die Matchingkanten $(1, \text{B}(3))$ und $(1, \text{C}(4))$ gehören zu Matchings, die nicht erweitert werden müssen und in die Matchingtabelle $\mathcal{T}_\Delta^{(n+1)}$ nur übernommen werden. Im bisher vorgestellten vereinfachten Algorithmus 6.6 werden beide Kanten nicht beachtet und ein Klongraph $G_{\text{Erw}v1}$, wie in (b) gegeben, erstellt. (c) zeigt ein mögliches Matching, nach dem erweitert werden kann. Die resultierende Δ-top Matchingtabelle für $\mathcal{G}^{(n+1)}$ ist in (a) in Spalte $\mathcal{G}_{v1}^{(n+1)}$ dargestellt. Es sind beide alte Tupel mit genau einem neuen Tupel erweitert worden, was optimal nach Problem 6.3 ist. Die Anonymitätsgrade dabei sind $a(1) = 3$ und $a(2) = 2$.

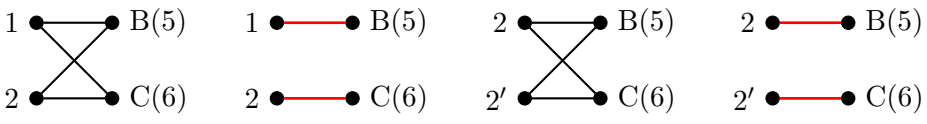
(d) zeigt einen anderen Klongraphen $G_{\text{Erw}v2}$, der nur aus zwei Knoten für Tupel 2 und den neuen Tupeln besteht. Ein entsprechendes Matching ist in (e) dargestellt und führt dazu, dass Tupel 2 sowohl mit B(5) als auch mit C(6) erweitert wird. Spalte $\mathcal{G}_{v2}^{(n+1)}$ aus (a) zeigt die Matchingtabelle nach diesen Erweiterungen bei der die Anonymitätsgrade $a(1) = a(2) = 3$ sind. Offensichtlich ist diese Erweiterung (Tupelverteilung) optimal nach Problem 6.2, während dies die erste Variante nicht ist.

Der Unterschied zwischen beiden Lösungen liegt in der Beachtung der bereits vorhandenen Δ-top Matchingkanten $(1, \text{B}(3))$ und $(1, \text{C}(4))$. Der Anonymitätsgrad von Tupel 1

[23] Vgl. Problem 6.3.
[24] Vgl. Problem 6.2.
[25] Vgl. Zeile 10 im Algorithmus 5.3.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$a(t)_{v1}$	$\mathcal{G}_{v2}^{(n+1)}$	$a(t)_{v2}$
1	A	B(3) B($x_{1.1}$) C(4) C($x_{1.2}$)	B(3) B($x_{1.1}, 5$) C(4) –	3	B(3) – C(4) –	3
2	A	B($x_{2.1}$) C($x_{2.2}$)	– C($x_{2.2}, 6$)	2	B($x_{2.1}, 5$) C($x_{2.2}, 6$)	3
3	B	A(1) C(4)	A(1) C(4)	3	A(1) C(4)	3
4	C	A(1) B(3)	A(1) B(3)	3	A(1) B(3)	3
5	B		A(1)		A(2)	
6	C		A(2)		A(2)	

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)



(b) $G_{\text{Erw}v1}$ (c) M_{v1} (d) $G_{\text{Erw}v2}$ (e) M_{v2}

Abbildung 6.12: Optimale Erweiterungen für die Probleme 6.3 ($v1$) und 6.2 ($v2$)

beträgt nach der Erweiterung immer 3, auch wenn die beiden Kanten $(1, \text{B}(x_{1.1}))$ und $(1, \text{C}(x_{1.2}))$ nicht erweitert werden. Tupel 2 hingegen hat noch keine Matchingkanten in $\mathcal{T}_{\Delta}^{(n+1)}$, wodurch $a(2)$ allein durch die Anzahl von Erweiterungen mit verschiedenem SA-Wert bestimmt wird.

Um aus dem Algorithmus für das vereinfachte Optimierungsproblem ein Verfahren zu konstruieren, das auch das komplette Problem löst, müssen an allen Stellen bereits vorhandene Matchingkanten und der aktuelle Anonymitätsgrad beachtet werden. Erweiterungen von Matchingkanten, zu denen bereits eine Kante mit demselben SA-Wert existiert, werden zunächst nicht durchgeführt. Folglich muss sowohl die Konstruktion des Klongraphen als auch die Berechnung von Matchings im Klongraphen modifiziert werden.

Algorithmus 6.9 (*Konstruiere Klongraph*) berechnet einen Klongraphen, bei dem die Anzahl von Knoten für alte Tupel auch vom aktuellen Anonymitätsgrad abhängt. Dabei soll zunächst gelten, dass die Summe aus Knotenanzahl und aktuellem Anonymitätsgrad für alle Tupel t identisch ist. Dazu wird der maximale aktuelle Anonymitätsgrad $a_{\max}^{*}$ berechnet und für alle Tupel $a_{\max}^{*} - a^{*}(t)$ Knoten erstellt. Die bereits vorgestellte Schranke von maximal $d(t)$ Knoten pro altem Tupel darf auch hier nicht überschritten werden (siehe Zeile 6). Bei der vereinfachten Konstruktion des Graphen, bei der der Anonymitätsgrad nicht beachtet wird, wurde für jedes Tupel zunächst ein Knoten erstellt. Um konsistent zu diesem Ansatz zu sein, werden im Fall, wenn alle Anonymitätsgrade identisch sind und demzufolge $n(t) = 0$ für alle t gilt, alle Klongrade ebenfalls auf 1 gesetzt. Die weiteren Schritte sind analog zur Konstruktion fürs vereinfachte Klonmatching in Algorithmus 6.5.

Algorithmus 6.10 (*Klonmatching*) berechnet Erweiterungen, die optimal entsprechend von Problem 6.2 sind. Im Unterschied zum vereinfachten Klonmatching besteht er aus drei

Algorithmus 6.9 Konstruiere Klongraph

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel
Ausgabe: Klongraph G_{Erw}

1: Entferne aus S_{alt} lokal alle Tupel t mit $E_{\text{alt}}(t) = \emptyset$
2: Entferne aus S_{neu} lokal alle Tupel, deren SA-Wert in keiner Kante aus E_{alt} vorkommt
3: $a^*_{\max} \leftarrow \max_{t \in S_{\text{alt}}} a^*(t)$ # max. akt. Anonymitätsgrad
4: **for all** Tupel $t \in S_{\text{alt}}$ **do**
5: $d(t) \leftarrow$ Anzahl verschiedener SA-Werte in $E_{\text{alt}}(t)$ # Erweiterungsgrad von t
6: $n(t) \leftarrow \min\{a^*_{\max} - a^*(t), d(t)\}$ # Klongrad: Anz. Knoten für t
7: **end for**
8: $N_T \leftarrow \sum_{t \in S_{\text{alt}}} n(t)$
9: **if** $N_T = 0$ **then**
10: Setze für alle Tupel t: $n(t) \leftarrow 1$
11: $N_T \leftarrow S_{\text{alt}}$
12: **end if**
13: $S_T \leftarrow \{t \in S_{\text{alt}} \mid d(t) > n(t)\}$ # Tupel zum Klonen
14: Führe Zeilen 9 bis 21 von Algorithmus 6.5 aus

Teilen. In Phase 1 werden zunächst alle Kanten aus E_{alt} entfernt, zu denen es bereits ein Matching mit einer Kante mit demselben SA-Wert in $\mathcal{T}_{\Delta}^{(n+1)}$ gibt. Erweiterungen dieser Kanten würden den Anonymitätsgrad des betroffenen alten Tupels nicht erhöhen. Mithilfe von Algorithmus 6.9 wird dann ein Klongraph G_{Erw} erstellt und darin ein vollständiges Matching gesucht. Dabei sei $G_{\text{Erw}}^{\langle i \rangle} \subseteq G_{\text{Erw}}$ der Teilgraph von G_{Erw}, in dem für jedes Tupel t genau $\max\{\min\{n(t), i - a^*(t)\}, 0\}$ Knoten existieren. Falls in G_{Erw} kein vollständiges Matching existiert, wird der Klongrad der alten Tupel verringert. Dabei muss beachtet werden, dass die Summe des aktuellen Anonymitätsgrades und des Klongrades den Anonymitätsgrad nach der Erweiterung bestimmt. Da der letztgenannte maximiert werden soll, wird genau diese Summe beschränkt, aus der die Anzahl von Knoten für jedes alte Tupel leicht berechnet werden kann. Der kleinste Klongraph ist der, in dem für alle Tupel mit minimalem $a^*(t)$ nur ein Knoten vorkommt, was $G_{\text{Erw}}^{\langle a^*_{\min}+1 \rangle}$ entspricht. Existiert auch in diesem Graphen kein vollständiges Matching, wird nur ein größtes Matching berechnet. Der minimale Anonymitätsgrad nach der Erweiterung entspricht in diesem Fall dem minimalem aktuellen Anonymitätsgrad vor der Erweiterung.

In Phase 2 von Algorithmus 6.10 werden Erweiterungen der anfangs entfernten Kanten berechnet. Dabei wird Algorithmus 6.4 (*Einfaches Matching*) verwendet. Die dritte und letzte Phase entspricht der Phase 2 des vereinfachten Klonmatchingalgorithmus. Hierbei werden bereits einmal erweiterte Kanten erneut erweitert, wobei erneut Algorithmus 6.4 verwendet wird.

Satz 6.15 *Algorithmus 6.10 (Klonmatching) findet eine optimale Erweiterung nach Problem 6.2 in $\mathcal{O}(\log n \cdot n^{\frac{7}{2}})$, wobei n die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Der Beweis der Korrektheit ist analog zum Beweis von Satz 6.4 und der Beweis der Laufzeit ist analog zum Beweis von Satz 6.14. □

Algorithmus 6.10 Erweiterungsalgorithmus: Klonmatching

Eingabe: S_{alt}: Menge erweiterbarer alter Tupel, E_{alt}: Menge erweiterbarer Δ-top Matchingkanten, S_{neu}: Menge neuer Tupel

Ausgabe: Erweiterung von Kanten (Tupelverteilung)

1: $E_{\text{Erw}} \coloneqq \{(t_{\text{alt}}, s(S_T)) \in E_{\text{alt}} \mid \exists t_{\text{neu}} \in S_{\text{neu}} \colon t_{\text{neu}}[\text{SA}] = s\}$
2: $E^*_{\text{Erw}} \coloneqq \{e_\Delta \in E_{\text{Erw}} \mid e_\Delta \text{ noch nicht erweitert}\}$
3: Entferne aus E_{alt} alle Kanten $(t, s'(S_T))$, für die es bereits ein Matching mit einer Kante $(t, s'(S'_T))$ in $\mathcal{T}_\Delta^{(n+1)}$ gibt # Phase 1
4: **while** $E^*_{\text{Erw}} \neq \emptyset$ **do**
5: Konstruiere Klongraph G_{Erw} nach Algorithmus 6.9 mit Eingabe $S_{\text{alt}}, E^*_{\text{Erw}}, S_{\text{neu}}$
6: $n^*_{\max} \leftarrow \max_{t \in S_{\text{alt}}}(n(t) + a^*(t))$
7: $a^*_{\min} \leftarrow \min_{t \in S_{\text{alt}}} a^*(t)$
8: Bestimme maximales i mit $a^*_{\min} + 1 \leq i \leq n^*_{\max}$, für das im Klongraphen $G^{\langle i \rangle}_{\text{Erw}}$ ein vollständiges Matching M existiert
9: Falls i nicht existiert, bestimme ein größtes Matching M in $G^{\langle a^*_{\min}+1 \rangle}_{\text{Erw}}$
10: **for all** Kanten $e \in M$ **do**
11: Sei t_{alt} das alte Tupel und t_{neu} das neue Tupel in e
12: `Erweitere`$(t_{\texttt{alt}}, t_{\texttt{neu}}, E_{\texttt{alt}})$
13: Entferne aus S_{neu} Tupel t_{neu}
14: **end for**
15: **end while**
16: Füge alle durch Zeile 3 gelöschten Kanten wieder in E_{alt} ein # Phase 2
17: **if** $S_{\text{neu}} \neq \emptyset$ **then**
18: Führe Zeilen 1 bis 12 aus Algorithmus 6.4 aus ohne Beschränkung auf minimalen aktuellen Anonymitätsgrad und mit Eingabe $S_{\text{alt}}, E^*_{\text{Erw}}, S_{\text{neu}}$
19: **end if**
20: Entferne aus S_{neu} alle Tupel, deren dazugehöriger Knoten durch Zeile 9 des Algorithmus 6.4 entfernt wurde # Phase 3
21: **if** $S_{\text{neu}} \neq \emptyset$ **then**
22: Führe Zeilen 1 bis 12 aus Algorithmus 6.4 aus ohne Zeile 10, ohne Beschränkung auf minimalen aktuellen Anonymitätsgrad und mit Eingabe $S_{\text{alt}}, E_{\text{Erw}}, S_{\text{neu}}$
23: **end if**

Beispiele zu den vorgestellten Algorithmen 6.9 und 6.10 sind im Anhang aufgeführt. In Beispiel A.20 auf Seite 292 und in Beispiel A.21 auf Seite 296 werden zwei Erweiterungen berechnet, wobei im letztgenannten Fall die Erweiterung nach Algorithmus 6.10 identisch zu der nach dem vereinfachten Klonmatching ist.

Zusammenfassung

In diesem Kapitel wurde der Erweiterungsschritt des im Kapitel 5 vorgestellten Algorithmus zur Berechnung von validen Matchings in Anfragegraphen untersucht. Es wurde gezeigt, was unter einer optimalen Erweiterung zu verstehen ist. Dazu wurden die Probleme 6.1, 6.2 und 6.3 definiert und die Erweiterungsalgorithmen *Maximale Kanten*, *Maximaler Anonymitätsgrad*, *Einfaches Matching* und *Klonmatching* vorgestellt. Der letztgenannte

garantiert eine optimale Erweiterung nach Problem 6.2, was als bestes Ergebnis angesehen wird. Er konstruiert einen speziellen Klongraphen, in dem ein modifizierter *Hopcroft-Karp-Algorithmus* ein sogenanntes vollständiges Matching bestimmt. Da alle eingeführten Verfahren polynomielle Laufzeit haben, ist auch die Approximation aus Kapitel 5 (siehe Algorithmus 5.3) in Polynomialzeit möglich. In Kapitel 8 werden alle Erweiterungsalgorithmen experimentell miteinander verglichen.

7 Approximation des Angreiferwissens für SA-eindeutige Anfragen

In diesem Kapitel wird ein alternativer Ansatz vorgestellt, mit dem analog zum Algorithmus in Kapitel 5 Matchings in Anfragegraphen berechnet werden können. Die Modellierung in Kapitel 4 bildet erneut die Grundlage, denn auch hier werden nur Δ-top Matchings gesucht. Allerdings werden die betrachteten Anfragegraphen dahingehend eingeschränkt, dass jeder SA-Wert nur noch höchstens einmal pro Graph vorkommen darf (Kapitel 7.1 und 7.2). Es wird gezeigt, dass in diesem Fall Erweiterungen durchgeführt werden können, die keinen weiteren Einschränkungen wie denen aus Kapitel 5.4 unterliegen (Kapitel 7.3 und 7.4). Um diese Methoden auch mit beliebigen Anfragegraphen verwenden zu können, wird ein Clusteransatz entwickelt, der einen gegebenen Graphen in mehrere Teilgraphen teilt, sodass in keinem der entstehenden Graphen ein SA-Wert mehrfach vorkommt (Kapitel 7.5). Danach wird ein Verfahren eingeführt, mit dem Antworten auf Anfragen auch bei einer Verletzung des Schutzkriteriums ausgegeben werden können (Kapitel 7.6). Am Ende dieses Kapitels wird der hier vorgestellte Ansatz mit dem aus Kapitel 5 kombiniert (Kapitel 7.7).

7.1 Verwandte Ansätze

Im Kapitel 5 wurde ein Algorithmus vorgestellt, der Matchings in Anfragegraphen berechnet und dazu zwei Einschränkungen voraussetzt. Zum einen dürfen alte Tupel nur mit neuen erweitert werden und zum anderen dürfen alte Tupel mit identischem SA-Wert nur mit verschiedenen neuen Tupeln erweitert werden. Dadurch wird die Menge der modellierten Matchings beschränkt und eventuell werden Tupeln bestimmte SA-Werte nicht mehr zugeordnet, obwohl dies möglich wäre.

Eine andere Herangehensweise an den Schutz der Privatsphäre bei der Anfragebearbeitung liefern die Ansätze aus dem Bereich der dynamischen Veröffentlichung von Daten aus Kapitel 2.5. Das dort vorgestellte Modell der m-Invarianz gibt Kriterien vor, die einen Schutz bei der Publikation mehrerer sich überschneidender Datentabellen garantieren. Beispielsweise benutzten Xiao und Tao [163] dieses Prinzip, um aus einer gegebenen Datentabelle mehrere anonymisierte Versionen zu erstellen. Die konstruierten Tabellen wurden benutzt, um mehrere Zählanfragen mit höherer Genauigkeit beantworten zu können, als es mit nur einer anonymisierten Tabelle möglich gewesen wäre. Diese Idee kann weiter verfeinert werden, indem einerseits beliebige Anfragen zugelassen sind und andererseits nur die Ergebnisse der Anfragen anonymisiert werden und nicht die gesamte Datentabelle. Das Veröffentlichen von Ergebnissen entspricht demnach dem dynamischen Veröffentlichen von Tabellen, wie es ebenfalls von Xiao und Tao [162] beschrieben wurde. Danach gilt, dass eine Folge von Ergebnissen $\mathcal{R}^{(n)} = (R_1, \ldots, R_n)$ k-assign Anonymität erfüllt, wenn $(R_1, \ldots, R_n)$ m-invariant mit $m \geq k$ ist.

Allerdings sind die Kriterien, die bei m-Invarianz gelten müssen, für die Anfragebearbeitung zu restriktiv. So wird gefordert, dass jedes Ergebnis einer Anfrage aus mindestens m Tupeln besteht, wobei kein sensibler Wert mehrfach vorkommen darf. Eine andere Eigenschaft betrifft die Tupel, die in mehreren Ergebnissen vorkommen. Für sie muss gelten, dass die Menge aller sensibler Werte in den jeweiligen Ergebnissen identisch ist. Insbesondere das letzte Kriterium stellt eine starke Beschränkung dar, denn es lässt nur wenige Anfragen zu. Andererseits gestattet der Ansatz von m-Invarianz, künstliche sensible Werte einer Veröffentlichung hinzuzufügen, um die definierten Kriterien zu erfüllen. Bei der Anfragebearbeitung werden auf diese Weise sehr viele zusätzliche Werte benötigt, wodurch die Aussagekraft der Ergebnisse schwindet.

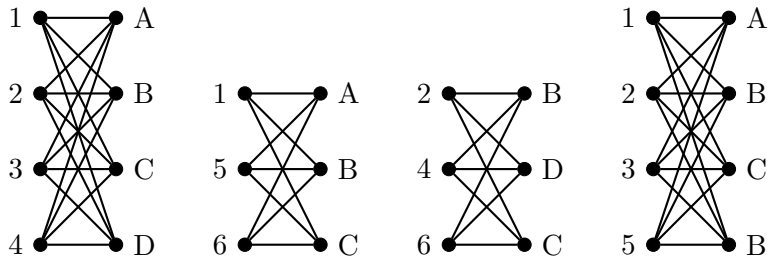
Diesem Problem steht positiv gegenüber, dass durch das Vermeiden von mehrfach vorkommenden sensiblen Werten die Gültigkeit des Schutzkriteriums sehr einfach überprüft werden kann. Der Algorithmus, der m-invariante Tabellen erstellt, ist effizienter als die im Kapitel 6 eingeführten Erweiterungsalgorithmen. Das ist die Motivation für den Ansatz, der in diesem Kapitel vorgestellt wird. Wenn in den Ergebnissen zu Anfragen sensible Werte nicht mehrfach vorkommen, kann ein vereinfachter Algorithmus angegeben werden, der den Schutz der Privatsphäre garantiert. Insbesondere ist in diesem Fall die Berechnung und Erweiterung von Matchings in Anfragegraphen wesentlich einfacher und effizienter als zuvor.

7.2 SA-eindeutige Anfragegraphen

Für die in diesem Kapitel betrachteten Anfragegraphen gilt die zusätzliche Bedingung, dass SA-Werte nicht mehrfach vorkommen dürfen. Das bedeutet, in den jeweiligen Ergebnismengen, zu denen die Graphen erstellt wurden, kommt jeder sensible Wert höchstens einmal vor. Die entstehenden Anfragegraphen werden *SA-eindeutig* genannt.

Definition 7.1 (SA-eindeutig) *Ein Anfragegraph G heißt* SA-eindeutig, *wenn kein SA-Label mehrfach vorkommt.*

Entsprechend Definition 7.1 heißt eine Folge von Anfragegraphen $\mathcal{G} = (G_1, \ldots, G_n)$ *SA-eindeutig*, wenn alle Graphen G_i SA-eindeutig sind. Das bedeutet aber nicht, dass die SA-Label verschiedener Graphen G_i und G_j ebenfalls unterschiedlich sein müssen. Die Eigenschaft der Eindeutigkeit bezieht sich immer nur auf einen einzelnen Anfragegraphen.



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Abbildung 7.1: G_1, G_2 und G_3 sind SA-eindeutige Anfragegraphen. Da in G_4 zweimal der SA-Wert B vorkommt, ist G_4 nicht SA-eindeutig.

Beispiel 7.1 Abbildung 7.1 zeigt die drei SA-eindeutigen Anfragegraphen G_1, G_2 und G_3. Eine Folge von Anfragegraphen $\mathcal{G}^{(3)}$, die aus diesen drei Graphen besteht, ist ebenfalls SA-eindeutig. In $\mathcal{G}^{(3)}$ würden allerdings einige Kanten fehlen (z. B. $(1, \mathrm{D})$ und $(4, \mathrm{A})$ in G_1). G_4 ist nicht SA-eindeutig, da der SA-Wert B zweimal vorkommt.

Für Matchings in SA-eindeutigen Folgen von Anfragegraphen gelten ein paar zusätzliche Eigenschaften. Zum einen gibt es in der symmetrischen Differenz zwischen zwei Matchings höchstens einen Kreis pro Anfragegraph. Zum anderen gibt es keine validen Matchings, die partiell SA-äquivalent oder nicht Δ-pfadkonform bezüglich des Originalmatchings sind. Das vereinfacht die Bedingungen, die für Δ-top Matchings gelten müssen.

Korollar 7.1 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine SA-eindeutige Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Wenn $\mathcal{M}$ Δ-minimal und ein Δ2-Matching bezüglich $\mathcal{M}'$ ist, dann besteht die symmetrische Differenz $\mathcal{M} \,\Delta\, \mathcal{M}'$ in jedem Teilgraphen G_i aus höchstens einem Kreis.*

Beweis: Nach Proposition 4.10 kommen in $\mathcal{M}' \,\Delta\, \mathcal{M}$ nur maximal zwei verschiedene Labels vor. Da jeder Anfragegraph SA-eindeutig ist, kann es in jedem Anfragegraphen nur höchstens einen Kreis geben. □

Korollar 7.2 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine SA-eindeutige Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Dann ist $\mathcal{M}$ Δ-pfadkonform und nicht partiell SA-äquivalent bezüglich $\mathcal{M}'$.*

Beweis: Da $\mathcal{G}$ SA-eindeutig ist, folgt der Beweis direkt aus Definition 4.1 (SA-äquivalent) und Definition 4.7 (Δ-pfadkonform). □

Da in SA-eindeutigen Anfragegraphen jeder SA-Wert höchstens einmal vorkommt, existiert nur ein Originalmatching. Andere Matchings sind Δ-top, wenn sie valid und bezüglich des Originalmatchings Δ-minimal sowie Δ2-Matchings sind. Insbesondere kann kein valides Matching gegen die Eigenschaft der Δ-Pfadkonformität verstoßen. Das vereinfacht die Algorithmen zur Erweiterung von Matchings. Zusätzlich besagt der folgende Satz, dass jede Kante eines Anfragegraphen in höchstens einem Δ-top Matching enthalten ist.

Proposition 7.3 *Sei $\mathcal{G} = (G_1, \ldots, G_n)$ eine SA-eindeutige Folge von Anfragegraphen. Dann existiert für jede Labelkante $(t, s') \in G_i$ mit $1 \leq i \leq n$, bei der s' nicht der originale SA-Wert von t ist, höchstens ein Δ-top Matching $\mathcal{M}$, welches (t, s') enthält.*

Beweis: Der Beweis erfolgt indirekt. Seien $\mathcal{M}_{\text{orig}}$ das Originalmatching in $\mathcal{G}$ sowie $\mathcal{M}_1$ und $\mathcal{M}_2$ zwei verschiedene Δ-top Matchings, die beide die Kante (t, s') enthalten. Da s' nicht der SA-Wert von t ist, gilt $(t, s') \notin \mathcal{M}_{\text{orig}}$. Sei $\mathcal{M}$ eine Menge von Kanten mit

$$\mathcal{M} = (\mathcal{M}_1 \cap \mathcal{M}_2) \cup \{\{v_t, v_s\} \in \mathcal{M}_{\text{orig}} \mid \exists \{v_t, v_{s_1}\} \in \mathcal{M}_1, \{v_t, v_{s_2}\} \in \mathcal{M}_2 \colon v_{s_1} \neq v_{s_2}\}.$$

$\mathcal{M}$ enthält alle Kanten, die sowohl in $\mathcal{M}_1$ als auch in $\mathcal{M}_2$ vorhanden sind, und ist auf dem restlichen Graphen mit dem Originalmatching identisch. Aus der Wahl von $\mathcal{M}_1$ und $\mathcal{M}_2$ folgt, dass $\mathcal{M}$ ein perfektes Matching in $\mathcal{G}$ ist.[1]

[1] Offensichtlich wird jeder Tupelknoten von $\mathcal{M}$ überdeckt. Wenn es in $\mathcal{M}$ zwei Kanten e_1 und e_2 gibt, die einen gemeinsamen SA-Knoten haben, muss $e_1 \in \mathcal{M}_1 \cap \mathcal{M}_2$ und $e_2 \in \mathcal{M}_{\text{orig}}$ gelten. Da $\mathcal{M}_1$ und $\mathcal{M}_2$ jeweils Δ2-Matchings bezüglich $\mathcal{M}_{\text{orig}}$ sind, muss es eine Kante $e_2' \in \mathcal{M}_1 \cap \mathcal{M}_2$ geben, die denselben Tupelknoten enthält wie e_2. Das widerspricht der Annahme $e_2 \in \mathcal{M}$.

Wenn $\mathcal{M}$ nicht valid ist, gibt es ein Tupel, deren dazugehörige Knoten in zwei Teilgraphen von $\mathcal{G}$ mit SA-Knoten mit unterschiedlichem Label matchen. Seien e_1 und e_2 die entsprechenden Kanten, dann muss $e_1 \in \mathcal{M}_1 \cap \mathcal{M}_2$ und $e_2 \in \mathcal{M}_{\text{orig}}$ gelten. Da $\mathcal{M}_1$ und $\mathcal{M}_2$ valid sind, gibt es zwei Kanten $e_2' \in \mathcal{M}_1$ und $e_2'' \in \mathcal{M}_2$, die denselben Tupelknoten beinhalten wie e_2 und jeweils einen SA-Knoten mit demselben Label wie der SA-Knoten in e_1. Da $\mathcal{G}$ SA-eindeutig ist, folgt $e_2' = e_2''$ und nach Konstruktion $e_2' \in \mathcal{M}$. Das widerspricht der Annahme $e_2 \in \mathcal{M}$, wodurch die Validität von $\mathcal{M}$ bewiesen ist.

Da $\mathcal{M}_1 \neq \mathcal{M}_2$ gilt, muss $\mathcal{M} \cap \mathcal{M}_{\text{orig}} \subset \mathcal{M}_1 \cap \mathcal{M}_{\text{orig}}$ oder $\mathcal{M} \cap \mathcal{M}_{\text{orig}} \subset \mathcal{M}_2 \cap \mathcal{M}_{\text{orig}}$ gelten. Nach Definition 4.4 ist damit $\mathcal{M}_1$ oder $\mathcal{M}_2$ nicht Δ-minimal bezüglich des Originalmatchings, was ein Widerspruch zur Annahme ist. □

7.3 Erweiterung von Matchings

Das folgende Lemma bildet die Grundlage dafür, dass Erweiterungen von Matchings in SA-eindeutigen Anfragegraphen ohne weitere Einschränkungen durchgeführt werden können.

Lemma 7.4 *Seien $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ eine SA-eindeutige Folge von Anfragegraphen, t_1 und t_2 zwei Tupel mit den originalen SA-Werten s_1 beziehungsweise s_2 sowie $\mathcal{M}_1^{(n)}$ und $\mathcal{M}_2^{(n)}$ zwei Δ-top Matchings in $\mathcal{G}^{(n)}$ mit $(t_1, s_2), (t_2, s_2) \in \mathcal{M}_1^{(n)}$ und $(t_1, s_1), (t_2, s_1) \in \mathcal{M}_2^{(n)}$. Sei G_{n+1} ein weiterer SA-eindeutiger Anfragegraph, der Knoten für t_1, t_2, s_1 und s_2 enthält. Dann existiert ein Δ-top Matching $\mathcal{M}^{(n+1)}$ in $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$, das die Kanten (t_1, s_2) und (t_2, s_1) enthält.*

In Abbildung 7.2 ist die in Lemma 7.4 beschriebene Situation dargestellt. Es bleibt zu zeigen, dass das in (d) angegebene Matching $\mathcal{M}_{1+2}^{(n+1)}$ immer existiert und Δ-top ist.

Beweis: Seien $\mathcal{M}_{\text{orig}}^{(n)}$ das Originalmatching und $\mathcal{M}^{(n)}$ eine Menge von Kanten, die wie folgt definiert ist:

$$\mathcal{M}^{(n)} = (\mathcal{M}_1^{(n)} - \mathcal{M}_{\text{orig}}^{(n)}) \cup (\mathcal{M}_2^{(n)} - \mathcal{M}_{\text{orig}}^{(n)}) \cup (\mathcal{M}_1^{(n)} \cap \mathcal{M}_2^{(n)}).$$

$\mathcal{M}^{(n)}$ besteht aus allen Kanten, die in $\mathcal{M}_1^{(n)}$ beziehungsweise $\mathcal{M}_2^{(n)}$ vorhanden, aber keine Originalkanten sind. Dazu kommen alle Kanten, die sowohl in $\mathcal{M}_1^{(n)}$ als auch in $\mathcal{M}_2^{(n)}$ vorkommen. Als Erstes ist zu zeigen, dass $\mathcal{M}^{(n)}$ ein valides Matching in $\mathcal{G}^{(n)}$ ist.

Sei v_t ein Tupelknoten mit den Matchingpartnern v_s in $\mathcal{M}_1^{(n)}$ und $v_{s'}$ in $\mathcal{M}_2^{(n)}$. Wenn $v_s = v_{s'}$ gilt, wird v_t in $\mathcal{M}^{(n)}$ nur von der Kante $\{v_t, v_s\}$ überdeckt. Wenn die beiden SA-Knoten ungleich sind und einer auch der Matchingpartner von v_t im Originalmatching ist, gibt es ebenfalls nur eine Kante in $\mathcal{M}^{(n)}$, die v_t enthält.[2] Wenn weder v_s noch $v_{s'}$ der Partner von v_t im Originalmatching ist, müssen auch die SA-Werte s von v_s, s' von $v_{s'}$ und der originale SA-Wert des Tupels t von v_t verschieden sein, denn der Anfragegraph ist SA-eindeutig. Da in der symmetrischen Differenz von $\mathcal{M}_1^{(n)}$ beziehungsweise $\mathcal{M}_2^{(n)}$ und des Originalmatchings bereits jeweils die SA-Werte s_1 und s_2 vorkommen, müssen in mindestens einer der beiden Differenzen drei SA-Werte vorhanden sein. Nach Proposition 4.10 ist dieser Fall für Δ-top Matchings ausgeschlossen.

[2] Falls $\{v_t, v_s\} \in \mathcal{M}_{\text{orig}}^{(n)}$, ist nur $\{v_t, v_{s'}\}$ in $\mathcal{M}^{(n)}$ enthalten und falls $\{v_t, v_{s'}\} \in \mathcal{M}_{\text{orig}}^{(n)}$, analog nur $\{v_t, v_s\}$.

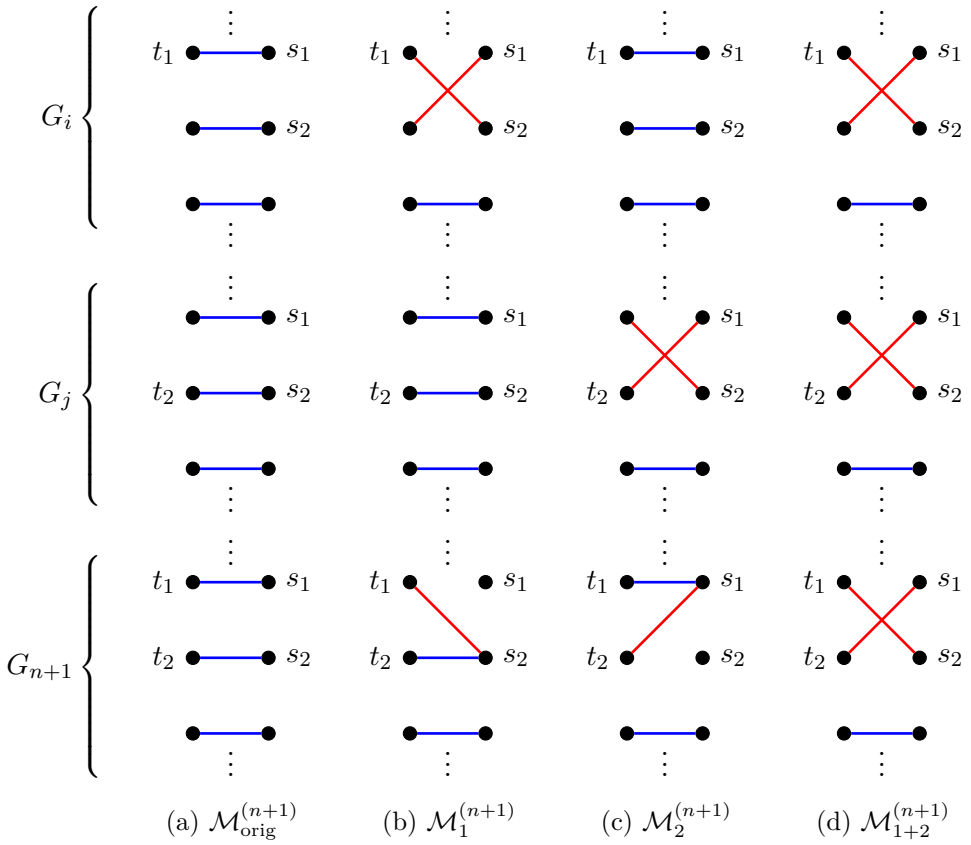


Abbildung 7.2: Vereinigung von Matchings in SA-eindeutigen Anfragegraphen (vgl. Lemma 7.4)

Sei v_s ein SA-Knoten mit den Matchingpartnern v_t in $\mathcal{M}^{(n)}_1$ und $v_{t'}$ in $\mathcal{M}^{(n)}_2$. Wenn $v_t = v_{t'}$ gilt, wird v_s in $\mathcal{M}^{(n)}$ nur von der Kante $\{v_t, v_s\}$ überdeckt. Wenn die beiden Tupelknoten ungleich sind und einer auch der Matchingpartner von v_s im Originalmatching ist, gibt es ebenfalls nur eine Kante in $\mathcal{M}^{(n)}$, die v_s enthält. Der Fall, dass beide Tupelknoten nicht der Matchingpartner von v_s im Originalmatching sind, ist aus einem analogen Argument wie davor ausgeschlossen.

Folglich ist $\mathcal{M}^{(n)}$ ein perfektes Matching in $\mathcal{G}^{(n)}$, das aufgrund der Konstruktion ebenfalls valid sein muss. Das für Lemma 7.4 gesuchte Matching $\mathcal{M}^{(n+1)}$ in $\mathcal{G}^{(n+1)}$ entsteht aus $\mathcal{M}^{(n)}$, indem im zusätzlichen Graphen G_{n+1} die Labelkanten (t_1, s_2) und (t_2, s_1) sowie für alle anderen Knoten deren Originalkante hinzufügt werden. Da auch G_{n+1} SA-eindeutig ist, kann es nach Proposition 4.10 außer t_1 und t_2 in G_{n+1} kein weiteres Tupel geben, das in $\mathcal{M}^{(n)}_1$ oder $\mathcal{M}^{(n)}_2$ nicht mit seinem originalen SA-Wert matcht. Somit ist auch $\mathcal{M}^{(n+1)}$ ein valides Matching. Da $\mathcal{M}^{(n)}_1$ und $\mathcal{M}^{(n)}_2$ Δ-minimal bezüglich des Originalmatchings sind, muss nach Konstruktion auch $\mathcal{M}^{(n+1)}$ Δ-minimal und offensichtlich auch ein $\Delta 2$-Matching sein. Das bedeutet, dass $\mathcal{M}^{(n+1)}$ das gesuchte Δ-top Matching ist. □

Lemma 7.4 liefert die Grundlage für die Erweiterung von Matchings mit alten Tupeln. Sei $\mathcal{M}^{(n)}_1$ ein Δ-top Matching in einer SA-eindeutigen Folge von Anfragegraphen $\mathcal{G}^{(n)}$ und

$(t_1, s_2) \in \mathcal{M}_1^{(n)}$ eine Kante in einem weiteren Graphen G_{n+1}, die nicht im Originalmatching vorkommt. Wenn es auch eine Kante $(t_2, s_1) \in \mathcal{M}_1^{(n)}$ gibt, wobei s_i für $i = 1, 2$ jeweils der originale SA-Wert von t_i ist, muss $\mathcal{M}_1^{(n)}$ nicht erweitert werden und bleibt in $\mathcal{G}^{(n+1)}$ erhalten (vgl. Definition 5.1 auf Seite 133). $\mathcal{M}_{1+2}^{(n+1)}$ in Abbildung 7.2d ist ein Beispiel für diesen Fall. Wenn diese Kante nicht existiert, muss t_2 ein neues Tupel sein oder in $\mathcal{M}_1^{(n)}$ mit seinem originalen SA-Wert s_2 matchen, da nach Proposition 4.10 ein anderer SA-Wert als Matchingpartner nicht in Frage kommt (siehe Abbildung 7.2b). Falls t_2 ein altes Tupel ist, kann $\mathcal{M}_1^{(n)}$ nur erweitert werden, wenn es ein Matching $\mathcal{M}_2^{(n)}$ mit der Kante (t_2, s_1) gibt. In diesem muss offensichtlich t_1 mit seinem originalen SA-Wert s_1 matchen (siehe Abbildung 7.2c). Während diese Bedingung für beliebige Anfragegraphen nicht ausreichend ist,[3] ist sie nach Lemma 7.4 für SA-eindeutige Anfragegraphen hinreichend.

Eine Folgerung daraus ist, dass eine Δ-top Matchingkante $e_\Delta = (t, s'(S_T))$ immer genau dann mit einem Tupel t' erweitert werden kann, wenn eine Δ-top Matchingkante $e'_\Delta = (t', s(S'_T))$ existiert, wobei s' der originale SA-Wert von t' und s der von t ist. Insbesondere können damit alte Tupel mit anderen alten Tupeln erweitert werden, was bedeutet, dass die zusätzliche Einschränkung wie im allgemeinen Fall für SA-eindeutige Anfragegraphen überflüssig ist.

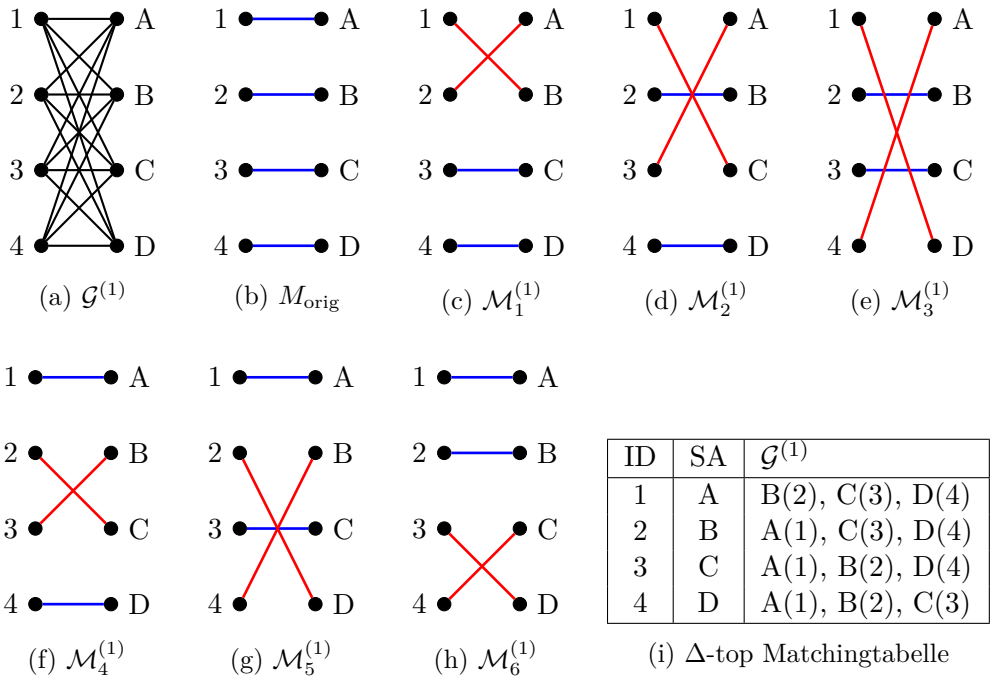
7.4 Konstruktion von Matchings

Mithilfe der bisherigen Erkenntnissen kann ein Algorithmus angegeben werden, der Δ-top Matchings in SA-eindeutigen Anfragegraphen berechnet. Dabei wird der erste Graph genauso behandelt wie im allgemeinen Fall. Das heißt, für jeden Anfragegraphen G_1 werden alle Kombinationen von Tupel- zu SA-Knoten gespeichert. Diese entsprechen allen Δ-top Matchings in G_1. Besteht der Graph aus n Tupel- beziehungsweise SA-Knoten, werden somit $\binom{n}{2}$ Matchings beziehungsweise $n \cdot (n-1)$ Kanten abgebildet.

Beispiel 7.2 Abbildung 7.3 zeigt in (a) einen Anfragegraphen $\mathcal{G}^{(1)}$, der aus vier Tupel- beziehungsweise SA-Knoten besteht. Zu dem in (b) dargestellten Originalmatching existieren demzufolge $\binom{4}{2} = 6$ Δ-top Matchings, die in (c) bis (h) angegeben sind. Alle darin enthaltenen Kanten werden in der Δ-top Matchingtabelle in (i) aufgeführt.

Matchings für eine Folge von Anfragegraphen $\mathcal{G}^{(n+1)}$ werden analog zum Fall der beliebigen Graphen aus den Matchings für $\mathcal{G}^{(n)}$ berechnet. Das bedeutet, es werden alle für $\mathcal{G}^{(n)}$ gespeicherten Δ-top Matchingkanten $e_\Delta = (t, s'(S_T))$ getestet, ob sie erweitert werden können, gelöscht werden müssen oder einfach erhalten bleiben. Ein Matching mit e_Δ muss gelöscht werden, wenn in G_{n+1} kein SA-Knoten mit Label s' vorkommt. In diesem Fall kann es kein valides Matching in $\mathcal{G}^{(n+1)}$ geben, in dem Tupel t mit s' matcht. Kommt genau ein Tupel aus S_T in $\mathcal{G}^{(n+1)}$ vor, bleibt e_Δ in der aktuellen Form erhalten. e_Δ kann erweitert werden, wenn kein Tupel aus S_T in $\mathcal{G}^{(n+1)}$ vorkommt. Dazu muss es aber ein Tupel t' in $\mathcal{G}^{(n+1)}$ geben, das den originalen SA-Wert s' besitzt und dem s (originaler SA-Wert von t) zugeordnet werden kann. Das ist der Fall, wenn eine Δ-top Matchingkante $e'_\Delta = (t', s(S'_T))$ existiert oder t' ein neues Tupel ist. Für neue Tupel existieren noch keine Matchings und demzufolge können ihnen prinzipiell alle SA-Werte zugeordnet werden, die im aktuellen Graphen vorkommen.

[3] Vgl. Kapitel 5.4.1.



(a) $\mathcal{G}^{(1)}$ (b) M_{orig} (c) $\mathcal{M}_1^{(1)}$ (d) $\mathcal{M}_2^{(1)}$ (e) $\mathcal{M}_3^{(1)}$

(f) $\mathcal{M}_4^{(1)}$ (g) $\mathcal{M}_5^{(1)}$ (h) $\mathcal{M}_6^{(1)}$

ID	SA	$\mathcal{G}^{(1)}$
1	A	B(2), C(3), D(4)
2	B	A(1), C(3), D(4)
3	C	A(1), B(2), D(4)
4	D	A(1), B(2), C(3)

(i) Δ-top Matchingtabelle

Abbildung 7.3: Δ-top Matchings für $\mathcal{G}^{(1)} = (G_1)$

Da alle Graphen SA-eindeutig sind, ist es nicht möglich, dass mehr als ein Tupel aus S_T in G_{n+1} vorkommt. Ein weiterer Unterschied zur Erweiterung in beliebigen Anfragegraphen ist, dass es keine zusätzlichen Einschränkungen gibt. Sind die Voraussetzungen einer Erweiterung vorhanden, werden Kanten auf jeden Fall erweitert.

Zusammenfassend gilt für jede Δ-top Matchingkante $e_\Delta = (t, s'(S_T))$:

1. e_Δ wird gelöscht, wenn
   - in G_{n+1} kein SA-Knoten mit Label s' vorkommt oder
   - das Tupel t' in G_{n+1} mit SA-Wert s' nicht mit s erweitert werden kann (d. h., t' ist ein altes Tupel und es existiert keine Matchingkante $e'_\Delta = (t', s''(S'_T))$ mit $s'' = s$).
2. e_Δ bleibt erhalten, wenn
   - genau ein Tupel aus S_T in G_{n+1} vorkommt.
3. e_Δ wird erweitert, wenn
   - kein Tupel aus S_T in G_{n+1} vorkommt und
   - ein Tupel t' in G_{n+1} mit SA-Wert s' existiert, das mit s erweitert werden kann (d. h. t' ist ein neues Tupel oder es existiert eine Matchingkante $e'_\Delta = (t', s''(S'_T))$ mit $s'' = s$).

Algorithmus 7.1 zeigt die Konstruktion von Matchings für eine SA-eindeutige Folge von Anfragegraphen $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$. Als Erstes wird die Δ-top Matchingtabelle als $\mathcal{T}_\Delta^{(n+1)}$ übernommen und es werden nur darin Änderungen durchgeführt. Dazu werden

Algorithmus 7.1 Berechne Δ-top Matchings (für SA-eindeutige Anfragegraphen)

Eingabe: $\mathcal{T}_\Delta^{(n)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$, G_{n+1}: Graph
Ausgabe: $\mathcal{T}_\Delta^{(n+1)}$: Δ-top Matchingtabelle für $\mathcal{G}^{(n+1)} = (G_1, \ldots, G_n, G_{n+1})$

```
1: T_Δ^(n+1) ← T_Δ^(n)
2: for all alte Tupel t (mit originalem SA-Wert s) in G_{n+1} do
3:   for all Δ-top Matchingkanten e_Δ = (t, s'(S_T)) in T_Δ^(n+1) do
4:     if s' kommt nicht in G_{n+1} vor then
5:       Lösche Matching mit e_Δ
6:     end if
7:     Vergleiche S_T mit Menge aller Tupel aus G_{n+1}
8:       Fall 1: genau ein Tupel aus S_T in G_{n+1} ⇒ e_Δ bleibt erhalten
9:       Fall 2: kein Tupel aus S_T in G_{n+1}
10:        Fall 2.1: Tupel t' in G_{n+1} mit SA-Wert s' ist neu oder es existiert eine Δ-top
           Matchingkante e'_Δ = (t', s(S'_T)) ⇒ erweitere t mit t' und umgekehrt
11:        Fall 2.2: andernfalls ⇒ lösche Matching mit e_Δ
12:  end for
13: end for
14: for all neue Tupel t, t' mit verschiedenen SA-Werten do
15:   Erzeuge neues Matching mit Kanten (t, s'(t')) und (t', s(t)), wobei s SA-Wert von t
      und s' SA-Wert von t'
16: end for
17: return T_Δ^(n+1)
```

alle gespeicherten Δ-top Matchingkanten $e_\Delta = (t, s'(S_T))$ der alten Tupel untersucht und deren SA-Werte beziehungsweise Tupel mit denen aus G_{n+1} verglichen. e_Δ wird gelöscht, wenn s' nicht in G_{n+1} vorkommt. Nach Proposition 7.3 existiert genau ein Matching, das e_Δ enthält und in diesem Fall komplett gelöscht wird (vgl. Algorithmus 7.2). Wenn s' in G_{n+1} vorkommt, wird die Tupelmenge S_T der Kante mit den Tupeln aus G_{n+1} verglichen. Dazu gibt es zwei verschiedene Fälle. Als Erstes kann genau ein Tupel aus S_T in G_{n+1} vorkommen (Fall 1), wobei das Matching beziehungsweise die Kante erhalten bleibt. Da bereits alle Kanten nach $\mathcal{T}_\Delta^{(n+1)}$ übertragen wurden, ist in diesem Fall nichts weiter zu tun. Die zweite Möglichkeit ist, dass kein Tupel aus S_T in G_{n+1} vorhanden ist. Dann wird das Tupel t' aus G_{n+1} gesucht, welches den SA-Wert s' hat. Wenn t' ein neues Tupel ist oder eine Δ-top Matchingkante besitzt, in der der originale SA-Wert s von t vorkommt, wird t mit t' erweitert. In Algorithmus 7.3 ist dieser Schritt angegeben.

Als Letztes werden mithilfe von Algorithmus 7.1 sogenannte neue Matchings berechnet, welche Erweiterungen des Originalmatchings darstellen. Dazu wird für jedes Paar von neuen Tupeln, die jeweils einen verschiedenen SA-Wert haben, ein Matching erstellt. Dieses besteht aus genau zwei Δ-top Matchingkanten und unterscheidet sich nur in G_{n+1} vom Originalmatching. Im ersten Anfragegraph G_1, in dem es nur neue Tupel gibt, werden Matchings nur mithilfe dieser Zeilen konstruiert.

Algorithmus 7.2 zeigt den Löschschritt von Algorithmus *Berechne Δ-top Matchings*. Der einzige Unterschied zum Algorithmus für beliebige Anfragegraphen (vgl. Algorithmus 5.2) ist, dass in SA-eindeutigen Graphen das Tupel t' in keiner anderen Matchingkante von t

Algorithmus 7.2 Lösche (für SA-eindeutige Anfragegraphen)

Eingabe: $e_\Delta = (t, s'(S_T))$: Matchingkante
Ausgabe: Löscht alle Matchings mit e_Δ aus der Matchingtabelle $\mathcal{T}_\Delta$
1: Lösche Kante e_Δ aus $\mathcal{T}_\Delta$
2: **for all** Tupel $t' \in S_T$ **do**
3: **if** es existiert eine Matchingkante $e'_\Delta = (t', s(S'_T))$ von t', in der t vorkommt (d. h. $t \in S'_T$) **then**
4: `Lösche(`e'_Δ`)`
5: **end if**
6: **end for**
7: **return**

als e_Δ vorkommen kann und andersherum es auch nur eine Kante e'_Δ von t' gibt, die t enthält.

Algorithmus 7.3 Erweitere (für SA-eindeutige Anfragegraphen)

Eingabe: t, t' Tupel
Ausgabe: Erweiterung aller möglicher Kanten von t mit t' und umgekehrt
1: Bestimme Matchingkante $e_\Delta = (t, s'(S_T))$, wobei s' SA-Wert von t'
2: Erzeuge erweiterte Kante zu e_Δ durch das Hinzufügen von t' zur Menge S_T
3: **if** t' ist neues Tupel **then**
4: Erstelle neue Kante $(t', s(\{t\}))$, wobei s SA-Wert von t ist
5: **else**
6: Bestimme Matchingkante $e'_\Delta = (t', s(S'_T))$, wobei s SA-Wert von t
7: Erzeuge erweiterte Kante zu e'_Δ durch das Hinzufügen von t zur Menge S'_T
8: **end if**
9: Füge beide Kanten in aktuelle Matchingtabelle ein

Erweiterungen von Tupeln erfolgen nach Algorithmus 7.3. Während der Algorithmus für beliebige Anfragegraphen (vgl. Algorithmus 6.2) nur Erweiterungen mit neuen Tupeln zulässt, werden bei SA-eindeutigen Anfragegraphen auch alte Tupel miteinander erweitert.

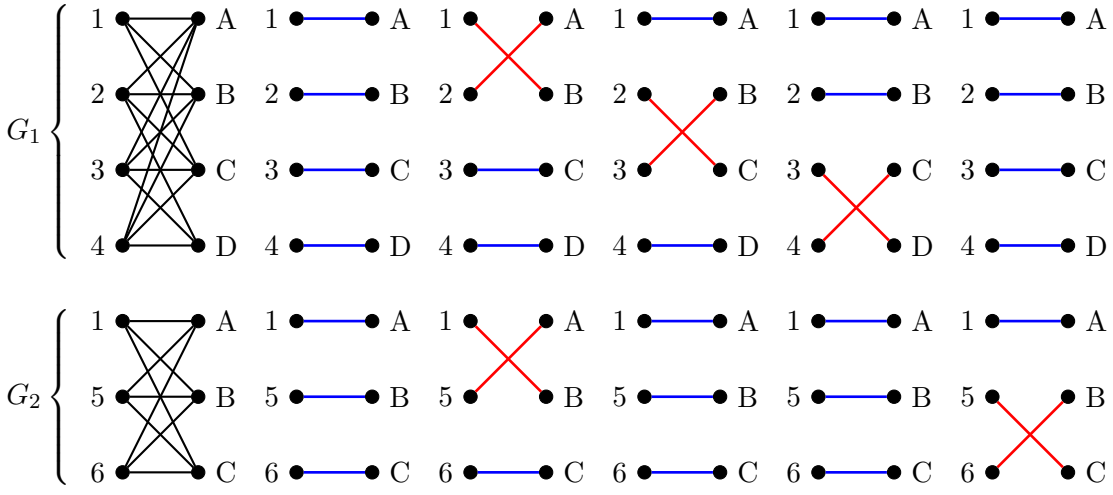
Theorem 7.5 (Korrektheit von Algorithmus 7.1) *Seien $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ eine SA-eindeutige Folge von Anfragegraphen und G_{n+1} ein weiterer SA-eindeutiger Anfragegraph. Dann berechnet Algorithmus 7.1 Δ-top Matchings für $\mathcal{G}^{(n+1)}$ in Laufzeit $\mathcal{O}(N^4)$, wobei N die Anzahl aller Tupel in der Datentabelle ist.*

Beweis: Die Korrektheit von Satz 7.5 wird mithilfe einer Induktion über n bewiesen. Offensichtlich werden alle möglichen Δ-top Matchings in $\mathcal{G}^{(1)}$ durch die Zeilen 14 bis 16 berechnet. Man nehme nun an, für $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ seien alle Δ-top Matchings in der Tabelle $\mathcal{T}_\Delta^{(n)}$ gespeichert und G_{n+1} sei ein weiterer SA-eindeutiger Anfragegraph. Zu zeigen ist, dass die erweiterten Matchings Δ-top in $\mathcal{G}^{(n+1)}$ sind und es keine weiteren Δ-top Matchings gibt, die nicht gespeichert werden. Matchings werden gelöscht, wenn keine Erweiterungen existieren, die $\Delta 2$-Matchings bezüglich des Originalmatchings sind (vgl. Zeilen 5 und 11). Erweiterungen nach Zeile 10 führen zu Δ-top Matchings, was für neue Tupel offensichtlich ist und für alte Tupel aus Lemma 7.4 folgt. Da Matchings mit

allen möglichen Kanten gebildet werden, ist ebenfalls sichergestellt, dass auch alle Δ-top Matchings in $\mathcal{G}^{(n+1)}$ gefunden werden.

In der Matchingtabelle existiert für jedes Tupel und jeden SA-Wert höchstens eine Δ-top Matchingkante, die wiederum höchstens $\mathcal{O}(N)$ Tupel enthält. Demzufolge kann die Größe von $\mathcal{T}_{\Delta}^{(n)}$ für jedes n mit $\mathcal{O}(N^3)$ abgeschätzt werden. Da in jedem Teilgraphen ebenfalls höchstens $\mathcal{O}(N)$ Tupel vorkommen, ist die Anzahl der Tupelvergleiche in Algorithmus 7.1 durch $\mathcal{O}(N^4)$ beschränkt.[4] Während das Einfügen von Kanten in die Tabelle nur konstante Zeit benötigt, kann sich das Löschen von Kanten rekursiv fortsetzen. Allerdings kann für jedes Tupel und jeden SA-Wert nur höchstens eine Matchingkante entfernt werden, wodurch das Löschen aller Kanten nur $\mathcal{O}(N^3)$ benötigt. □

Erweiterungen von Matchings in SA-eindeutigen Anfragegraphen nach Algorithmus *Berechne Δ-top Matchings* werden mithilfe der bereits in Abbildung 7.1 eingeführten Graphen in den beiden folgenden Beispielen veranschaulicht.



(a) $\mathcal{G}^{(2)}$ (b) $\mathcal{M}^{(2)}_{\text{orig}}$ (c) $\mathcal{M}^{(2)}_{1}$ (d) $\mathcal{M}^{(2)}_{4}$ (e) $\mathcal{M}^{(2)}_{6}$ (f) $\mathcal{M}^{(2)}_{7}$

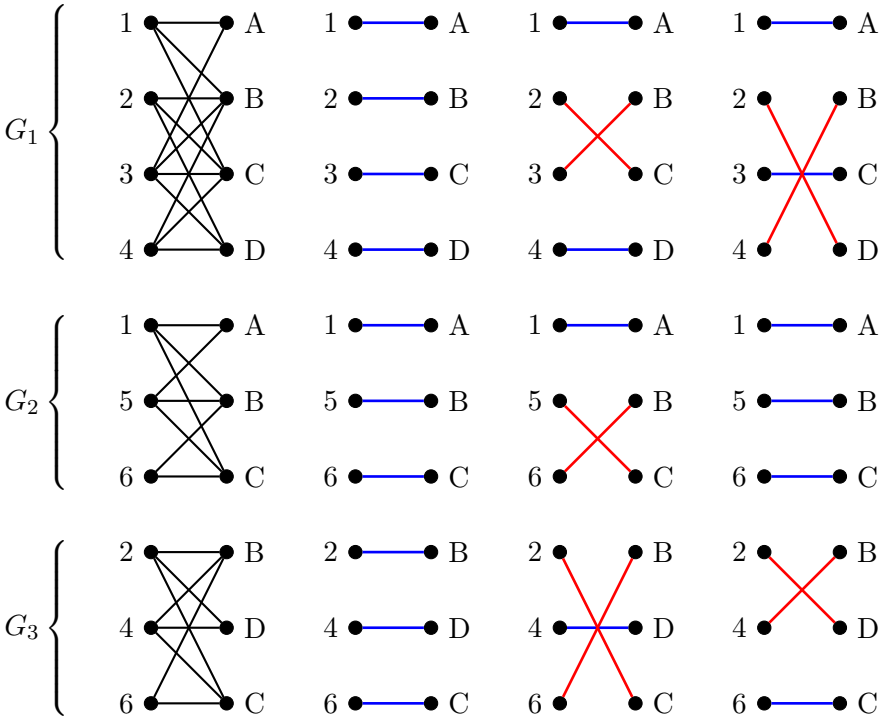
ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$
1	A	B(2), C(3), D(4)	B(2, 5), C(3, 6)
2	B	A(1), C(3), D(4)	A(1), C(3), D(4)
3	C	A(1), B(2), D(4)	A(1), B(2), D(4)
4	D	A(1), B(2), C(3)	B(2), C(3)
5	B		A(1), C(6)
6	C		A(1), B(5)

(g) Δ-top Matchingtabelle

Abbildung 7.4: Δ-top Matchings für $\mathcal{G}^{(2)} = (G_1, G_2)$. In (c) bis (f) sind exemplarisch nur vier der in der Δ-top Matchingtabelle gespeicherten Matchings dargestellt.

[4] Diese Abschätzung kann verschärft werden, indem zum Beispiel die Tupel in den Matchingkanten und Graphen zuvor sortiert werden.

Beispiel 7.3 Abbildung 7.4 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G}^{(2)}$, die sich aus dem bereits in Abbildung 7.3 eingeführten Graphen G_1 und dem Graphen G_2 aus Abbildung 7.1 zusammensetzt. Δ-top Matchings für $\mathcal{G}^{(2)}$ werden aus den Matchings für $\mathcal{G}^{(1)}$ (siehe Abbildung 7.3 bzw. Spalte $\mathcal{G}^{(1)}$ in (g)) berechnet. $\mathcal{M}_1^{(1)}$ kann sehr einfach erweitert werden, indem in G_2 Tupel 1 mit B(5) und 5 mit A(1) matchen. Dafür wird die dazugehörige Matchingkante $(1, \mathrm{B}(2))$ mit B(5) zu $(1, \mathrm{B}(2,5))$ erweitert. Gleichzeitig entsteht die Gegenkante $(5, \mathrm{A}(1))$. Analoges gilt für $\mathcal{M}_2^{(1)}$, während $\mathcal{M}_3^{(1)}$ gelöscht werden muss, da in G_2 kein SA-Wert D vorkommt. Dementsprechend wird die Kante $(1, \mathrm{D}(4))$ zusammen mit der Gegenkante $(4, \mathrm{A}(1))$ aus der Tabelle entfernt. In (c) bis (f) sind exemplarisch nur vier der in der Δ-top Matchingtabelle gespeicherten Matchings dargestellt.



(a) $\mathcal{G}^{(3)}$ (b) M_{orig} (c) $\mathcal{M}_4^{(3)}$ (d) $\mathcal{M}_5^{(3)}$

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$
1	A	B(2), C(3), D(4)	B(2,5), C(3,6)	–
2	B	A(1), C(3), D(4)	A(1), C(3), D(4)	C(3,6), D(4)
3	C	A(1), B(2), D(4)	A(1), B(2), D(4)	B(2)
4	D	A(1), B(2), C(3)	B(2), C(3)	B(2)
5	B		A(1), C(6)	C(6)
6	C		A(1), B(5)	B(5,2)

(e) Δ-top Matchingtabelle

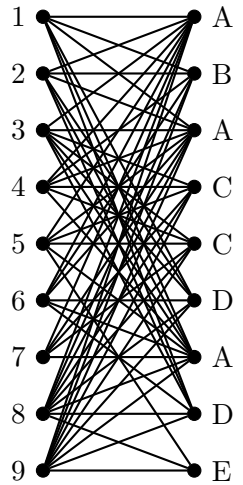
Abbildung 7.5: Δ-top Matchings für $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$. $\mathcal{M}_4^{(3)}$ aus (c) entsteht durch das Vereinigen von $\mathcal{M}_4^{(2)}$ und $\mathcal{M}_7^{(2)}$ in $\mathcal{G}^{(2)}$ (vgl. Abbildung 7.4).

Beispiel 7.4 Abbildung 7.5 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G}^{(3)}$, die sich aus dem bereits in Abbildung 7.4 eingeführten Anfragegraphen $\mathcal{G}^{(2)}$ und dem Graphen G_3 aus Abbildung 7.1 zusammensetzt. In $\mathcal{G}^{(3)}$ existieren nur noch zwei Δ-top Matchings, die in (c) und (d) angegeben sind.

Beispielsweise muss Matching $\mathcal{M}_1^{(2)}$ gelöscht werden, da in G_3 kein SA-Label A vorkommt. Dazu müssen, nachdem die entsprechende Kante $(2, \mathrm{A}(1))$ entfernt wurde, auch die Kanten $(1, \mathrm{B}(2,5))$ und $(5, \mathrm{A}(1))$ aus der Tabelle entfernt werden. Kante $(2, \mathrm{C}(3))$ des Matchings $\mathcal{M}_4^{(2)}$ kann mit C(6) erweitert werden, denn Tupel 6 kann mit B(2) erweitert werden, da eine Kante $(6, \mathrm{B}(5))$ existiert. Das entstehende Matching $\mathcal{M}_4^{(3)}$ ist eine Vereinigung der Matchings $\mathcal{M}_4^{(2)}$ und $\mathcal{M}_7^{(2)}$ aus Abbildung 7.4. Im Gegensatz dazu muss das Matching $\mathcal{M}_6^{(2)}$ gelöscht werden. Die darin enthaltene Kante $(4, \mathrm{C}(3))$ könnte zwar mit C(6) erweitert werden, 6 allerdings nicht mit D(4). Das gilt, da 6 ein altes Tupel ist und keine Matchingkante besitzt, die den SA-Wert D enthält.

7.5 Beliebige Anfragegraphen

Für SA-eindeutige Anfragegraphen ist in Algorithmus 7.1 ein Verfahren angegeben, mit dem Matchings und damit Zuordnungen von sensiblen Werten zu Tupeln berechnet werden können. Zusätzliche Erweiterungsalgorithmen werden dabei nicht benötigt. Allerdings bleibt die starke Einschränkung, dass jeder SA-Wert nur einmal pro Graph vorkommen darf. Um diesen Ansatz auch auf beliebige Graphen auszudehnen, wird in diesem Abschnitt ein Vorverarbeitungsschritt eingeführt, der aus beliebigen Anfragegraphen eine Menge von SA-eindeutigen Graphen erstellt.



(a) G_{n+1}

ID	SA	*Sig*
1	A	ABD
2	B	ABD
3	A	ACD
4	C	ABC
5	C	ACE
6	D	ABD
7	A	AC
8	D	neu
9	E	neu

(b) G_{n+1}

ID	SA	*Sig*
1	A	ABD
5	C	ACE
8	D	neu
2	B	ABD
3	A	ACD
9	E	neu
4	C	ABC
6	D	ABD
7	A	AC

(c) $G_{n+1_{v1}}$

ID	SA	*Sig*
1	A	ABD
2	B	ABD
6	D	ABD
3	A	ACD
4	C	ABC
8	D	neu
5	C	ACE
7	A	AC
9	E	neu

(d) $G_{n+1_{v2}}$

Abbildung 7.6: Partitionierung eines Anfragegraphen

Abbildung 7.6 zeigt in (a) einen Anfragegraphen G_{n+1}, der aus je neun Tupel- und SA-Knoten besteht. Da die Darstellung solch großer Graphen recht unübersichtlich ist und in diesem Abschnitt keine Matchings veranschaulicht werden müssen, wird in (b) eine alternative Darstellung gewählt. Dabei werden alle Tupel mit den dazugehörigen SA-

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}_{v1}^{(n+1)}$	$a(t)_{v1}$	$\mathcal{G}_{v2}^{(n+1)}$	$a(t)_{v2}$
1	A	B($x_{1.1}$), D($x_{1.2}$)	D($x_{1.2}$, 8)	2	B($x_{1.1}$, 2), D($x_{1.2}$, 6)	3
2	B	A($x_{2.1}$), D($x_{2.2}$)	–	1	A($x_{2.1}$, 1), D($x_{2.2}$, 6)	3
3	A	C($x_{3.1}$), D($x_{3.2}$)	–	1	C($x_{3.1}$, 4), D($x_{3.2}$, 8)	3
4	C	A($x_{4.1}$), B($x_{4.2}$)	A($x_{4.1}$, 7)	2	A($x_{4.1}$, 3)	2
5	C	A($x_{5.1}$), E($x_{5.2}$)	–	1	A($x_{5.1}$, 7), E($x_{5.2}$, 9)	3
6	D	A($x_{6.1}$), B($x_{6.2}$)	–	1	A($x_{6.1}$, 1), B($x_{6.2}$, 2)	3
7	A	C($x_{7.1}$)	C($x_{7.1}$, 4)	2	C($x_{7.1}$, 5)	2
8	D	neu	A(1)	2	A(3)	2
9	E	neu	–	1	A(5)	2

Tabelle 7.1: Δ-top Matchingtabelle für beide Partitionierungen aus Abbildung 7.6

Werten in Tabellenform aufgelistet. Anstelle der Kanten werden die Signaturen der Tupel aus der Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ angegeben. Da die Signatur eines Tupels die Menge aller SA-Werte aus Matchingkanten des Tupels ist,[5] sind daraus die Kanten des Graphen leicht ableitbar. Beispielsweise haben Tupel 1 und 2 jeweils Kanten zu allen SA-Knoten mit Label A, B oder D sowie Tupel 3 zu allen Knoten mit Label A, C oder D. Die neuen Tupel (hier 8 und 9) haben immer Kanten zu allen SA-Knoten.

Offensichtlich ist G_{n+1} kein SA-eindeutiger Anfragegraph, woraus folgt, dass die im vorigen Abschnitt eingeführten Algorithmen nicht angewendet werden können. Allerdings gilt, dass G_{n+1} das Ergebnis einer Anfrage darstellt und für die Tupel (bzw. Individuen, die zu den Tupeln gehören) ein vorgegebenes Schutzkriterium gelten soll. Wenn beispielsweise 2-assign Anonymität die Vorgabe war, können anstelle eines Ergebnisses mit neun Tupeln auch mehrere Ergebnisse mit viel weniger Tupeln zurückgegeben werden. Die Vereinigung dieser Ergebnisse (auch Partitionen genannt) entspricht dann der Gesamtergebnismenge. Zwei mögliche Partitionierungen sind in (c) und (d) dargestellt und haben die Eigenschaft, dass in jeder Partition kein SA-Wert mehrfach vorkommt. Daraus können wiederum Anfragegraphen erzeugt werden, die SA-eindeutig und Teilgraphen von G_{n+1} sind. Um im Allgemeinen Algorithmen für SA-eindeutige Anfragegraphen auch für beliebige Graphen zu verwenden, werden letztere in Teilgraphen partitioniert, sodass jeder entstehende Graph SA-eindeutig ist. Anstelle des Gesamtgraphen werden dann die einzelnen Teilgraphen bearbeitet.

Durch das Partitionieren von Graphen kann es allerdings vorkommen, dass Matchings gelöscht werden, die im Gesamtgraphen hätten erweitert werden können. Zusätzliche Matchings können offensichtlich nicht entstehen. Für die beiden Partitionierungen in (c) und (d) sind die entstehenden Δ-top Matchingtabellen in Tabelle 7.1 berechnet worden. Dabei stehen alle Variablen x_i für Tupel oder Tupelmengen, die hier nicht dargestellt sind. In Spalte $\mathcal{G}^{(n)}$ ist die Tabelle vor der Bearbeitung des Graphen G_{n+1} gegeben, aus der sich die Signaturen der Tupel ablesen lassen. Die Tabelle, die entsteht, wenn Algorithmus *Berechne Δ-top Matchings* auf die drei SA-eindeutigen Graphen aus Abbildung 7.6c angewendet wird, ist in Spalte $\mathcal{G}_{v1}^{(n+1)}$ dargestellt. In diesem Fall fallen sehr viele Matchingkanten beziehungsweise SA-Werte weg, sodass der minimale Anonymitätsgrad nur noch 1 beträgt. Das bedeutet, dass hier eine Verletzung des Schutzkriteriums vorliegt. Bei der Partitionierung nach Abbildung 7.6d wird insgesamt nur eine Matchingkante gelöscht und der minimale

[5] Vgl. Definition 4.10.

Anonymitätsgrad beträgt 2. Die erste Partitionierung ist damit sehr viel schlechter als die zweite, da viel mehr Zuordnungen von Tupeln zu SA-Werten wegfallen als nötig. Die Herausforderung bei diesem Ansatz ist demnach, eine Partionierung zu wählen, durch die möglichst wenige Verletzungen des Schutzkriteriums entstehen.

7.5.1 Der erste Graph

Ein einfacher Fall liegt vor, wenn in einem Anfragegraphen nur neue Tupel vorhanden sind, wie zum Beispiel bei der ersten Anfrage G_1. Neue Tupel haben noch keine Signatur und können prinzipiell mit jedem SA-Wert matchen. Demzufolge muss bei einer Partitionierung von neuen Tupeln nur darauf geachtet werden, dass die einzelnen Partitionen groß genug sind, um das Schutzkriterium zu erfüllen.

ID	SA	*Sig*
1	A	neu
2	B	neu
3	A	neu
4	C	neu
5	C	neu
6	D	neu
7	A	neu
8	D	neu
9	E	neu

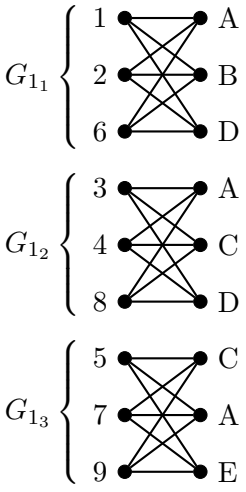
(a) G_1

ID	SA	*Sig*
1	A	neu
4	C	neu
9	E	neu
3	A	neu
5	C	neu
6	D	neu
7	A	neu
2	B	neu
8	D	neu

(b) $G_{1_{k=2}}$

ID	SA	*Sig*
1	A	neu
2	B	neu
6	D	neu
3	A	neu
4	C	neu
8	D	neu
5	C	neu
7	A	neu
9	E	neu

(c) $G_{1_{k=3}}$



(d) $G_{1_{k=3}}$

Abbildung 7.7: Partitionierung des ersten Anfragegraphen

ID	SA	$\mathcal{G}^{(1)}_{k=2}$	$a(t)_{k=2}$	$\mathcal{G}^{(1)}_{k=3}$	$a(t)_{k=3}$
1	A	C(4), E(9)	3	B(2), D(6)	3
2	B	D(8)	2	A(1), D(6)	3
3	A	C(5)	2	C(4), D(8)	3
4	C	A(1), E(9)	3	A(3), D(8)	3
5	C	A(3)	2	A(7), E(9)	3
6	D	A(7)	2	A(1), B(2)	3
7	A	D(6)	2	C(5), E(9)	3
8	D	B(2)	2	A(3), C(4)	3
9	E	A(1), C(4)	3	C(5), A(7)	3

Tabelle 7.2: Δ-top Matchingtabelle für beide Partitionierungen aus Abbildung 7.7

In Abbildung 7.7 ist in (a) ein Anfragegraph G_1 dargestellt, der aus neun neuen Tupeln und SA-Werten besteht und nicht SA-eindeutig ist. Für das Partitionieren in SA-eindeutige Anfragegraphen gibt es mehrere Möglichkeiten. Die Anzahl entstehender Partitionen wird

dabei durch das Vorkommen des häufigsten SA-Wertes bestimmt. Da es insgesamt drei Tupel mit SA-Wert A gibt, müssen mindestens drei Partitionen erstellt werden, die jeweils ein Tupel mit SA-Wert A enthalten. Damit keine Verletzungen der Privatsphäre entsteht, muss in jeder Partition das Schutzkriterium erfüllt sein. Angenommen, es soll k-assign Anonymität mit $k = 2$ gelten, dann muss jede Partition aus mindestens zwei Tupeln bestehen. Sei n die Anzahl aller Tupel im Graphen, dann können höchstens $\frac{n}{k}$ solcher Partitionen erstellt werden. Für $k = 2$ sind das $\lfloor \frac{9}{2} \rfloor = 4$ und für $k = 3$ höchstens $\lfloor \frac{9}{3} \rfloor = 3$ Partitionen. (b) und (c) zeigen für diese beiden Fälle mögliche Partitionierungen, die die genannten Eigenschaften erfüllen. Für $k = 3$ sind in (d) die SA-eindeutigen Anfragegraphen dargestellt, die aus den Partitionen konstruiert werden können. In Tabelle 7.2 sind für beide Möglichkeiten die entsprechenden Δ-top Matchingtabellen und jeweils der Anonymitätsgrad der einzelnen Tupel angegeben.

Wenn $k \geq 4$ gilt, müssen mindestens $\lfloor \frac{9}{4} \rfloor = 2$ Partitionen erstellt werden, damit das Schutzkriterium erfüllt wird. Da der SA-Wert A dreimal in G_1 vorkommt, wäre eine der beiden Partitionen dann aber nicht SA-eindeutig. Im Umkehrschluss würde aber eine Partitionierung in drei Teile wie zum Beispiel in Abbildung 7.7c zu einer Verletzung der Privatsphäre führen.

Algorithmus 7.4 Partitioniere Tupel

Eingabe: S_T: Menge von Tupeln, k: Wert für k-assign Anonymität
Ausgabe: $P_1, \ldots, P_m$: SA-eindeutige Mengen von Tupeln (Partitionierung)

1: $s_{\max} \leftarrow$ Anzahl Tupel aus S_T mit dem häufigsten SA-Wert
2: **if** $s_{\max} > \frac{|S_T|}{k}$ **then**
3: $\quad m \leftarrow s_{\max}$
4: **else**
5: $\quad m \leftarrow \left\lfloor \frac{|S_T|}{k} \right\rfloor$
6: **end if**
7: Sortiere S_T, sodass jeweils alle Tupel mit gleichem SA-Wert hintereinander stehen
8: $i \leftarrow 0$
9: **for all** Tupel $t \in S_T$ **do**
10: $\quad$ Füge t in P_i ein
11: $\quad i \leftarrow (i + 1) \mod m$
12: **end for**

Analog zum Begriff der SA-eindeutigen Anfragegraphen heißt eine Menge von Tupeln *SA-eindeutig*, wenn jeder enthaltene SA-Wert höchstens einmal vorkommt.[6] Algorithmus 7.4 zeigt ein Verfahren, das eine gegebene Menge S_T von Tupeln nach den vorgestellten Kriterien in SA-eindeutige Teilmengen partitioniert. Dabei wird die Anzahl der Partitionen m so gesetzt, dass möglichst viele Partitionen entstehen ohne k-assign Anonymität zu verletzen. Ist das nicht möglich, entstehen Tupelmengen, die zwar weniger als k Elemente beinhalten, dafür aber auf jeden Fall SA-eindeutig sind.

Beispiel 7.5 Die Partitionierungen in Abbildung 7.7b und 7.7c wurden mit Algorithmus *Partioniere Tupel* berechnet, wobei die SA-Werte jeweils alphabetisch sortiert wurden.

[6] Man beachte, dass SA-eindeutige Mengen von Tupeln nichts mit (streng) eindeutig augmentierenden Mengen von Pfaden zu tun haben (vgl. Definition 6.5 auf Seite 185).

Ist eine Folge von Anfragegraphen $\mathcal{G}^{(n)} = (G_1, \ldots, G_n)$ gegeben, kann aus G_1 nach dem vorgestellten Verfahren eine Menge von SA-eindeutigen Anfragegraphen erstellt werden. Dazu wird die Menge aller Tupel aus G_1 mithilfe von Algorithmus 7.4 in mehrere SA-eindeutige Mengen $P_1, \ldots, P_m$ partitioniert. Aus jeder dieser Mengen kann wiederum ein SA-eindeutiger Anfragegraph $G_{1_1}, \ldots, G_{1_m}$ konstruiert werden. Die Besonderheit ist, dass alle Tupel in allen Graphen G_{1_i} neue Tupel sind.

7.5.2 Clustern der alten Tupel

Ist ein Anfragegraph gegeben, der aus alten und eventuell neuen Tupeln besteht, können Partitionierungen schnell zu schlechten Ergebnissen führen. In Abbildung 7.6 sind zwei mögliche Partitionierungen angegeben, wobei in einer Variante viel mehr Verletzungen des Schutzkriteriums entstehen als in der anderen. Beide Varianten erfüllen aber die Eigenschaften, dass jeder SA-Wert höchstens einmal vorkommt, und haben identische Größen der einzelnen Teile. Da das Ziel lautet, möglichst wenige Verletzungen zu erhalten, muss für das Aufteilen von alten Tupeln ein anderer Ansatz gewählt werden als für neue Tupel.

ID	SA	*Sig*
1	A	ABD
5	C	ACE
6	D	ABD
7	A	AC

(a) G_{n+1}

ID	SA	*Sig*
1	A	ABD
5	C	ACE
6	D	ABD
7	A	AC

(b) $G_{n+1_{v1}}$

ID	SA	*Sig*
1	A	ABD
6	D	ABD
5	C	ACE
7	A	AC

(c) $G_{n+1_{v2}}$

ID	SA	*Sig*
5	C	ACE
6	D	ABD

(d) $G_{n+1_{v3}}$

Abbildung 7.8: Partitionierung eines Anfragegraphen

Aus dem bereits in Abbildung 7.6 eingeführten Anfragegraphen G_{n+1} soll beispielhaft das Partitionieren von vier Tupeln näher untersucht werden. Abbildung 7.8 zeigt in (a) die Tupel 1, 5, 6 und 7 mit ihren SA-Werten und Signaturen. Da die Signatur von 1 die SA-Werte A, B und D enthält, wobei A der originale SA-Wert von 1 ist, gibt es nur Δ-top Matchings, in denen 1 mit B oder D matcht. Analog gibt es Matchings, die die Kanten $(5, \mathrm{A})$, $(5, \mathrm{E})$, $(6, \mathrm{A})$, $(6, \mathrm{B})$ oder $(7, \mathrm{C})$ enthalten. Diese Matchings können nur erweitert werden, wenn die Tupel in einem Graphen beziehungsweise in einer Partition mit den jeweiligen SA-Werten vorkommen.

Eine mögliche Partitionierung ist in (b) angegeben, bei der die Tupel 1 und 5 sowie 6 und 7 jeweils eine Partition bilden. Hierbei gilt, dass das Matching mit der Kante $(5, \mathrm{A})$ auf jeden Fall gelöscht werden muss, obwohl der SA-Wert A in der Partition vorkommt. Das liegt daran, dass das Tupel 1, welches den originalen SA-Wert A hat, keine Matchingkante mit dem SA-Wert C besitzt.[7] Aus analogem Grund muss auch das Matching mit der Kante $(6, \mathrm{A})$ in der zweiten Partition gelöscht werden.

Wesentlich besser ist es, eine Partitionierung wie in (c) zu erstellen, in der mehrere Kanten beziehungsweise Matchings erweitert werden können. In diesem Fall kann Tupel 1 mit 6 sowie 5 mit 7 erweitert werden, was zu Matchings mit den Kanten $(1, \mathrm{D}(5))$ und $(5, \mathrm{A}(1))$ sowie $(5, \mathrm{A}(7))$ und $(7, \mathrm{C}(5))$ führt. Sind andererseits die Tupel 5 und 6 zusammen in einer Partition (siehe (d)), müssen nicht zwingend Kanten gelöscht werden. 5 hat kein

[7] Vgl. Algorithmus 7.1 auf Seite 204.

Matching mit dem SA-Wert von 6 und umgekehrt. Sofern hier noch weitere Tupel mit den SA-Werten A, B, D oder E hinzugefügt werden, können sogar eventuell alle Kanten erweitert werden.

Zusammenfassend kann man drei Fälle unterscheiden, wann zwei Tupel in einem Teilgraphen zusammen sein sollten und wann eher nicht. Seien t und t' zwei alte Tupel mit den originalen SA-Werten s beziehungsweise s'. Wenn $s \in Sig(t')$, aber $s' \notin Sig(t)$ gelten, muss ein Matching gelöscht werden und dementsprechend sollten t und t' in diesem Fall eher nicht im selben Teilgraphen vorkommen.[8] Das gleiche Ergebnis folgt natürlich auch, wenn beide SA-Werte identisch sind. Wenn $s \neq s'$, $s \in Sig(t')$ und $s' \in Sig(t)$ gelten, können zwei Δ-top Matchingkanten erweitert werden und t und t' können im selben Teilgraphen vorkommen.[9] Im letzten Fall gelten $s \neq s'$, $s \notin Sig(t')$ und $s' \notin Sig(t)$, was zu keinem Löschen, aber auch keinem Erweitern führt. Die Frage, ob t und t' im selben Teilgraphen vorkommen sollten, ist hierbei von anderen Tupeln abhängig.[10]

Aus diesen Beobachtungen wird ein Algorithmus abgeleitet, der Tupel entsprechend den drei vorgestellten Fällen zu Teilgraphen zuordnet. Dabei wird für jedes Tupel der Teilgraph berechnet, in dem das Tupel am nützlichsten ist. Ein Tupel hat einen *Nutzen* für ein anderes Tupel, wenn es mit diesem erweitert werden kann (und umgekehrt). Ein Tupel *schadet*[11] einem anderen Tupel, wenn Matchingkanten gelöscht werden müssen, sofern beide Tupel in einem Graphen vorkommen. Da der Algorithmus Entscheidungen mithilfe einer Nutzenfunktion trifft, spricht man von einem *Clustering*[12] anstelle von einer Partitionierung. Entsprechend heißen die entstehenden Teilmengen von Tupeln *Cluster*, wobei mit dem Begriff Clustering insbesondere auch die Menge aller Cluster bezeichnet wird. Verfahren, mit deren Hilfe beliebige Objekte in Cluster aufgeteilt werden können, werden unter dem Begriff der *Clusteranalyse* zusammengefasst. An dieser Stelle wird auf eine allgemeine Einführung in dieses Themengebiet verzichtet, da nur ein spezieller Algorithmus benötigt wird.[13]

Clusterkriterium Als Erstes wird ein Kriterium festgelegt, das in jedem Cluster gelten soll. Dabei wird ein pessimistischer Ansatz gewählt, das heißt, in keinem Cluster sollen Tupel vorkommen, die einander schaden. Folglich gilt in jedem Cluster, dass der SA-Wert eines Tupels genau dann in der Signatur eines anderen enthalten ist, wenn das gleiche auch umgekehrt gilt. Sei C ein Cluster von Tupeln, dann muss für alle enthaltenen Tupel t und t' mit den originalen SA-Werten s beziehungsweise s' gelten:

$$s \neq s' \tag{7.1}$$

$$s \in Sig(t') \Leftrightarrow s' \in Sig(t). \tag{7.2}$$

[8] Vgl. $G_{n+1_{v1}}$ aus Abbildung 7.8b.
[9] Vgl. $G_{n+1_{v2}}$ aus Abbildung 7.8c.
[10] Vgl. $G_{n+1_{v3}}$ aus Abbildung 7.8d.
[11] Hierbei kann auch vom negativen Nutzen gesprochen werden.
[12] Da sich der englische Begriff „Clustering“ in der deutschen Fachliteratur durchgesetzt hat, wird er in dieser Arbeit anstelle eines deutschen Äquivalents verwendet.
[13] Gute Einführungen in das Thema Clusteranalyse geben zum Beispiel die Bücher von Bacher et al. [4] sowie Kaufman und Rousseeuw [78].

Um den Nutzen zu berechnen, den ein Tupel zu einem anderen Tupel beziehungsweise einem Cluster hat, wird eine Funktion *sim* definiert.[14] Sie gibt an, wie viele SA-Werte einzelner Tupel erweitert werden oder auf jeden Fall erhalten bleiben. Wenn der SA-Wert eines Tupels t in der Signatur eines anderen Tupels t' vorkommt und umgekehrt, können beide Tupel miteinander erweitert werden. Dabei werden insgesamt zwei Δ-top Matchingkanten erweitert (von jedem Tupel eine) und somit bleiben zwei SA-Werte erhalten.[15] Folglich wird in diesem Fall $sim(t, t') = 2$ definiert. Sind die SA-Werte von t beziehungsweise t' jeweils nicht in der Signatur des anderen Tupels enthalten, werden Matchings weder gelöscht noch erweitert. Der Nutzen wird dementsprechend auf 0 gesetzt. In allen anderen Fällen müssen Matchings und damit SA-Werte gelöscht werden. Da hierbei auch das Clusterkriterium verletzt ist, wird *sim* auf $-\infty$ gesetzt.

Definition 7.2 (Nutzen- bzw. Ähnlichkeitsfunktion *sim* für alte Tupel) *Seien t und t' zwei alte Tupel mit den originalen SA-Werten s beziehungsweise s' sowie C ein Cluster (Menge) von Tupeln. Dann ist die Funktion sim wie folgt definiert:*

$$sim(t, t') = \begin{cases} 2, & \textit{falls } s \neq s',\ s \in Sig(t') \textit{ und } s' \in Sig(t) \\ 0, & \textit{falls } s \notin Sig(t') \textit{ und } s' \notin Sig(t) \\ -\infty, & \textit{andernfalls} \end{cases} \tag{7.3}$$

$$sim(t, C) = \sum_{t' \in C} sim(t, t'). \tag{7.4}$$

Algorithmus 7.5 Clustere alte Tupel

Eingabe: S_T: Menge von (alten) Tupeln
Ausgabe: $C_1, \ldots, C_m$: Clustering der Tupel aus S_T

1: Füge jedes Tupel $t \in S_T$, das den häufigsten SA-Wert aller Tupel hat, in ein neues Cluster ein und entferne t aus S_T
2: **for all** Tupel $t \in S_T$ **do**
3: Bestimme Cluster C_i mit größtem Wert für $sim(t, C_i)$
4: Fall 1: $sim(t, C_i) > 0 \rightarrow$ füge t in Cluster C_i ein
5: Fall 2: $sim(t, C_i) = -\infty \rightarrow$ füge t in ein neues Cluster ein
6: Fall 3: $sim(t, C_i) = 0 \rightarrow$ ordne t noch nicht zu
7: **end for**
8: Wiederhole die For-Schleife für alle noch nicht zugeordneten Tupel, wobei diesmal t auch im Fall 3 in C_i eingefügt wird

Ein Verfahren, um alte Tupel mithilfe der *sim*-Funktion zu clustern, ist in Algorithmus 7.5 angegeben. Es basiert auf dem *k-means-Algorithmus* von Lloyd [96][16] und dem *Leader-Algorithmus* von Hartigan [63]. Zuerst wird für jedes Tupel, das den am häufigsten vorkommenden SA-Wert aller Tupel hat, jeweils ein Cluster erstellt. Danach wird für jedes Tupel t das Cluster C_i berechnet, zu dem t den größten Nutzen hat. Je nachdem

[14] Beim Clustern wird häufig von Ähnlichkeit (engl. *similarity*) gesprochen. Da hier in gewisser Weise die Ähnlichkeit von Signaturen untersucht wird, wird die Funktion mit *sim* benannt.
[15] Falls es bereits ein Matching mit t und t' gibt, bleibt das Matching und damit zwei Δ-Matchingkanten erhalten.
[16] Der Begriff k-means wurde allerdings zuerst von MacQueen [103] verwendet.

ob der Wert der dafür verwendeten *sim*-Funktion positiv oder negativ ist, wird t in C_i eingefügt oder bildet ein neues Cluster. Ein Spezialfall liegt vor, wenn der Nutzen Null ist. Dann wird t zunächst zurückgestellt und in einem weiteren Durchlauf erneut betrachtet. Durch das Einfügen oder Erstellen von Clustern kann sich bei der zweiten Berechnung der maximale *sim*-Wert verändern, sodass jetzt eine Entscheidung über die Zugehörigkeit von t getroffen werden kann. Falls der Wert erneut Null ist, wird t diesmal in ein Cluster eingefügt, sodass jedes Tupel höchstens zweimal betrachtet wird.

ID	SA	*Sig*
1	A	ABD
2	B	ABD
3	A	ACD
4	C	ABC
5	C	ACE
6	D	ABD
7	A	AC

Tabelle 7.3: Menge alter Tupel

	Init	2		4		5		6	
	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i
C_1	$\{1\}$	2	$\{1,2\}$	$-\infty$	$\{1,2\}$	$-\infty$	$\{1,2\}$	4	$\{1,2,6\}$
C_2	$\{3\}$	$-\infty$	$\{3\}$	2	$\{3,4\}$	$-\infty$	$\{3,4\}$	2	$\{3,4\}$
C_3	$\{7\}$	$-\infty$	$\{7\}$	2	$\{7\}$	2	$\{5,7\}$	$-\infty$	$\{5,7\}$

Tabelle 7.4: Clustern von alten Tupeln nach Algorithmus 7.5

Beispiel 7.6 In Tabelle 7.3 sind die alten Tupel des bereits eingeführten Anfragegraphen aus Abbildung 7.6 aufgelistet. Mithilfe von Algorithmus 7.5 soll ein Clustering erstellt werden, wobei in jedem Cluster kein SA-Wert mehrfach vorkommen darf. Da der insgesamt am häufigsten vorkommende SA-Wert das A ist, wird zu Beginn für jedes Tupel mit SA-Wert A ein Cluster erstellt. Dabei entstehen die Cluster $C_1 = \{1\}$, $C_2 = \{3\}$ und $C_3 = \{7\}$. Der weitere Ablauf des Algorithmus ist in Tabelle 7.4 skizziert. Seien die restlichen Tupel entsprechend ihren IDs sortiert, dann wird zunächst Tupel 2 betrachtet und das Cluster mit dem größten *sim*-Wert gesucht (vgl. Spalte „2“). Da $sim(2, C_1) = sim(2, \{1\}) = 2$ und $sim(2, C_2) = sim(2, \{3\}) = sim(2, C_3) = sim(2, \{7\}) = -\infty$ gelten, wird 2 in das Cluster mit 1 eingefügt. Für Tupel 4 gibt es zwei Cluster mit maximalem *sim*-Wert 2 (vgl. Spalte „4“). Angenommen, Tupel 4 wird in C_2 eingefügt, dann muss Tupel 5 danach in C_3 eingefügt werden. Für das letzte Tupel 6 gibt es zwei Cluster mit positivem *sim*-Wert, wobei $sim(6, C_1) > sim(6, C_2)$ gilt. Folglich wird 6 in das erste Cluster eingefügt und es ergibt sich das Clustering $C_1 = \{1, 2, 6\}$, $C_2 = \{3, 4\}$ und $C_3 = \{5, 7\}$.

Ein komplexeres Clustering wird im Anhang in Beispiel A.22 auf Seite 298 berechnet. Dabei werden insbesondere auch Tupel zurückgestellt und während der Bearbeitung entstehen weitere Cluster.

Ist ein Anfragegraph G_{n+1} gegeben, in dem mindestens ein SA-Wert mehrfach vorkommt, werden die Tupel aus G_{n+1} in alte und neue Tupel unterteilt. Alte Tupel werden

mithilfe von Algorithmus 7.5 geclustert, sodass in jedem Cluster jeder SA-Wert höchstens einmal vorkommt. Das nächste Kapitel zeigt, wie die neuen Tupel den bereits vorhandenen Clustern zugeordnet werden.

7.5.3 Clustern der neuen Tupel

Enthält ein Anfragegraph beziehungsweise eine Menge neue Tupel, können diese analog zu den alten Tupeln geclustert werden. Allerdings haben sie noch keine Signatur, wodurch die Definition des Nutzens der Tupel angepasst werden muss. Ein neues Tupel t_{neu} hat einen Nutzen für ein altes t_{alt}, wenn t_{alt} mit t_{neu} erweitert werden kann. Das ist der Fall, wenn der SA-Wert von t_{neu} in der Signatur von t_{alt} enthalten ist.

ID	SA	*Sig*
1	A	ABD
2	C	ABC
3	D	neu
4	E	neu

(a) G_{n+1}

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B($x_{1.1}$), D($x_{1.2}$)	D($x_{1.2}, 3$)
2	C	A($x_{2.1}$), B($x_{2.2}$)	–
3	D		A(1), E(4)
4	E		D(3)

(b) Δ-top Matchingtabelle

Abbildung 7.9: Clustern von alten Tupeln nach Algorithmus 7.5

Abbildung 7.9 zeigt in (a) einen Anfragegraphen, der aus den alten Tupeln 1 und 2 sowie den neuen Tupeln 3 und 4 besteht. In (b) ist die entsprechende Δ-top Matchingtabelle dargestellt. Kante $(1, \text{D}(x_{1.2}))$ kann mit dem neuen Tupel 3 erweitert werden, da D der SA-Wert von 3 ist. Das ist auch daran erkennbar, dass D in der Signatur von 1 enthalten ist. Dagegen müssen alle anderen Kanten gelöscht werden, da sie nicht erweitert werden können. Aus beiden neuen Tupeln 3 und 4 entsteht aber auch ein neues Matching, welches die Kanten $(3, \text{E}(4))$ und $(4, \text{D}(3))$ enthält. Demzufolge nützen sich neue Tupel auch gegenseitig. Diese Zusammenhänge werden durch die *sim*-Funktion genauso wie für alte Tupel abgebildet.

Definition 7.3 (Nutzen- bzw. Ähnlichkeitsfunktion *sim* für neue Tupel) *Seien t_{neu}, t'_{neu} zwei neue und t_{alt} ein altes Tupel mit den originalen SA-Werten s_{neu}, s'_{neu} beziehungsweise s_{alt} sowie C ein Cluster (Menge) von Tupeln. Dann ist die Funktion sim wie folgt definiert:*

$$sim(t_{neu}, t_{alt}) = \begin{cases} 2, & \textit{falls } s_{neu} \neq s_{alt} \textit{ und } s_{neu} \in Sig(t_{alt}) \\ 0, & \textit{falls } s_{neu} \notin Sig(t_{alt}) \\ -\infty, & \textit{andernfalls (d. h. } s_{neu} = s_{alt}) \end{cases} \tag{7.5}$$

$$sim(t_{neu}, t'_{neu}) = \begin{cases} 2, & \textit{falls } s_{neu} \neq s'_{neu} \\ -\infty, & \textit{andernfalls (d. h. } s_{neu} = s_{alt}) \end{cases} \tag{7.6}$$

$$sim(t_{neu}, C) = \sum_{t' \in C} sim(t_{neu}, t'). \tag{7.7}$$

Definition 7.3 gewährleistet, dass die *sim*-Funktion weiterhin die Anzahl der SA-Werte angibt, die durch das Hinzufügen eines Tupels zu einem anderen Tupel beziehungsweise Cluster auf jeden Fall erhalten bleiben. Werden neue Tupel allerdings mithilfe dieser

Funktion und Algorithmus 7.5 geclustert, können Clusterings entstehen, die zwar zu vielen Matchings führen, aber offensichtlich auch mehr Verletzungen des Schutzkriteriums verursachen, als nötig wären.

ID	SA	*Sig*
1	A	ABC
2	A	ADE
3	B	neu
4	C	neu
5	D	neu
6	E	neu

Tabelle 7.5: Menge alter Tupel

	Init	3		4		5		6	
	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i
C_1	$\{1\}$	2	$\{1,3\}$	4	$\{1,3,4\}$	6	$\{1,3,4,5\}$	8	$\{1,3,4,5,6\}$
C_2	$\{2\}$	0	$\{2\}$	0	$\{2\}$	2	$\{2\}$	2	$\{2\}$

Tabelle 7.6: Clustern von neuen Tupeln nach Algorithmus 7.5

Tabelle 7.5 zeigt einen Anfragegraphen, der geclustert werden soll. Dieser Vorgang ist in Tabelle 7.6 skizziert. Die beiden alten Tupel 1 und 2 führen zunächst zu den beiden Clustern $C_1 = \{1\}$ und $C_2 = \{2\}$. Das neue Tupel 3 hat den maximalen *sim*-Wert von 2 zu C_1, da der SA-Wert von 3 in der Signatur von 1 enthalten ist. Das zweite neue Tupel 4 wird ebenfalls in C_1 eingefügt, da es dem alten Tupel 1 und dem eben eingefügten neuen Tupel 3 nützlich ist. Tupel 5 hat zu beiden Clustern einen positiven *sim*-Wert. In C_2 würde es dem alten Tupel 2 nutzen und in C_1 den beiden neuen Tupeln 3 und 4. Folglich wird auch dieses Tupel in C_1 eingefügt. Da das letzte neue Tupel 6 aus analogem Grund auch in C_1 einsortiert wird, müssen die Matchings mit den Kanten $(2, \mathrm{D})$ und $(2, \mathrm{E})$ gelöscht werden. Offensichtlich wäre es besser, Tupel 5 und 6 in C_2 einzufügen, denn dann müsste kein Matching gelöscht werden.

Wenn in einem Cluster mehrere neue Tupel eingefügt wurden, hat dieses Cluster immer auch einen positiven Nutzen zu anderen neuen Tupeln. Je mehr Tupel bereits vorhanden sind, um so wahrscheinlicher ist es daher, dass noch weitere (mit jeweils anderen SA-Werten) hinzukommen. Es ist also die Tendenz vorhanden, dass ein großes Cluster entsteht, in dem alle SA-Werte einmal vorkommen. Matchingkanten von alten Tupeln in anderen kleineren Clustern werden somit viel häufiger gelöscht als nötig.

Die Lösung dieses Problems ist, beim Clustern von neuen Tupeln alte Tupel bevorzugt zu behandeln. Beispielsweise ist es bei k-assign Anonymität egal, wie der SA-Wert weiterer neuer Tupel ist, die zusätzlich in C_1 aus dem vorigen Beispiel eingefügt werden. Die vorhandenen neuen Tupel können Matchings mit jedem beliebigen SA-Wert bilden. Tupel 2 in Cluster C_2 benötigt aber auf jeden Fall ein Tupel mit dem SA-Wert D oder E. Das führt zu der Idee eines zweistufigen Clusterings. Zunächst wird in der ersten Stufe beziehungsweise Phase der *sim*-Wert eines neuen Tupels und eines Clusters nur mithilfe der im Cluster vorhandenen alten Tupel berechnet. Neue Tupel werden somit nur in die Cluster einsortiert, in denen sie einen positiven Nutzen zu alten Tupeln haben. Wenn sie keinem

alten Tupel etwas nutzen, werden sie zunächst zurückgestellt und noch nicht zugeordnet. In der zweiten Phase werden beim Berechnen der *sim*-Funktion auch andere neue Tupel beachtet, allerdings mit einer weiteren Einschränkung. Ist beispielsweise 3-assign Anonymität gefordert, muss in Cluster C_1 aus Tabelle 7.6 außer 3 und 4 kein weiteres neues Tupel eingefügt werden, denn alle Tupel erfüllen bereits das Kriterium. In solchen Fällen erfolgt auch in der zweiten Phase die Berechnung des *sim*-Wertes nur mithilfe der alten Tupel.

Für die erste Phase wird eine andere *sim*-Funktion benötigt, bei der der Nutzen von zwei neuen Tupeln mit Null angeben wird.

Definition 7.4 (Nutzen- bzw. Ähnlichkeitsfunktion sim_0 für neue Tupel) *Seien t und t' zwei Tupel mit den originalen SA-Werten s beziehungsweise s' sowie C ein Cluster (Menge) von Tupeln. Dann ist die Funktion sim_0 wie folgt definiert:*

$$sim_0(t, t') = \begin{cases} 0, & \textit{falls } t, t' \textit{ neue Tupel und } s \neq s' \\ sim(t, t'), & \textit{andernfalls} \end{cases} \tag{7.8}$$

$$sim_0(t, C) = \sum_{t' \in C} sim_0(t, t'). \tag{7.9}$$

Algorithmus 7.6 Clustere neue Tupel

Eingabe: $C_1, \ldots, C_m$: Clustering von alten Tupeln, S_T: Menge von neuen Tupeln
Ausgabe: $C'_1, \ldots, C'_m$: Clustering der neuen und alten Tupel

```
1: for all Tupel t ∈ S_T do                                   # Phase 1
2:    Bestimme Cluster C_i mit größtem Wert für sim_0(t, C_i)
3:    if sim_0(t, C_i) > 0 then
4:       Füge t in Cluster C_i ein
5:    end if
6: end for
7: for all noch nicht zugeordnete Tupel t ∈ S_T do            # Phase 2
8:    Bestimme Cluster C_i mit größtem Wert für sim(t, C_i), beachte dabei nur Cluster
      C_i, in denen das Schutzkriterium bei mind. einem neuen Tupel nicht erfüllt ist
9:    if sim(t, C_i) > 0 then
10:      Füge t in Cluster C_i ein
11:   end if
12: end for
13: Clustere alle noch nicht zugeordneten Tupel mithilfe von Algorithmus 7.4
```

Algorithmus 7.6 erstellt ein Clustering der neuen Tupel nach dem beschriebenen Verfahren. In Phase 1 wird der Nutzen eines neuen Tupels zu einem Cluster mittels sim_0 berechnet, das heißt, der Nutzen von zwei neuen Tupeln zueinander ist Null. Alle Tupel, die keinen positiven Nutzen zu irgendeinem Cluster haben, werden zunächst nicht zugeordnet und später erneut betrachtet. In Phase 2 wird der Nutzen mit der ursprünglichen *sim*-Funktion bestimmt. Es werden dabei aber nicht mehr alle Cluster betrachtet, sondern nur noch die, in denen das Schutzkriterium bei mindestens einem neuen Tupel nicht erfüllt ist. Das hat den Vorteil, dass in Clustern, in denen alle neuen Tupel das Schutzkriterium erfüllen, keine weiteren neuen Tupel eingefügt werden. Das Cluster wird damit nicht mehr

unnötig groß. Tupel, die auch nach dem zweiten Durchlauf keinem Cluster zugeordnet wurden, werden mithilfe des Algorithmus 7.4 (*Partitioniere Tupel*) auf neue Partitionen verteilt.

	Init	3		4		5		6	
	C_i	sim_0	C_i	sim_0	C_i	sim_0	C_i	sim_0	C_i
C_1	$\{1\}$	2	$\{1,3\}$	2	$\{1,3,4\}$	0	$\{1,3,4\}$	0	$\{1,3,4\}$
C_2	$\{2\}$	0	$\{2\}$	0	$\{2\}$	2	$\{2,5\}$	2	$\{2,5,6\}$

Tabelle 7.7: Clustern von neuen Tupeln nach Algorithmus 7.6

Aus dem Beispiel in Tabelle 7.5 wird durch Algorithmus 7.6 ein anderes Clustering erstellt als das in Tabelle 7.6. Der größte Unterschied besteht darin, dass für die neuen Tupel der sim_0-Wert berechnet wird. Das führt dazu, dass die Tupel 3 und 4 in Cluster C_1 sowie 5 und 6 in C_2 eingefügt werden (siehe Tabelle 7.7). Die Phase 2 entfällt in diesem Beispiel.

Angenommen, es gibt ein weiteres neues Tupel 7 mit SA-Wert F, dann gilt $sim_0(7, C_1) = sim_0(7, C_2) = 0$ und 7 wird zunächst zurückgestellt. In Phase 2 von Algorithmus 7.6 müsste für beide Cluster getestet werden, ob das Schutzkriterium bei allen neuen Tupeln erfüllt ist. Beispielsweise führt Cluster C_1 zu einer Δ-top Matchingtabelle, in der Tupel 3 die beiden Kanten $(3, \mathrm{A}(1))$ und $(3, \mathrm{C}(4))$ sowie Tupel 4 die Kanten $(4, \mathrm{A}(1))$ und $(4, \mathrm{B}(3))$ besitzt. Das bedeutet, der aktuelle Anonymitätsgrad a^* beider Tupel ist 3, wobei der Begriff des aktuellen Anonymitätsgrades in diesem Kapitel analog zu Definition 6.2 verwendet wird. Wenn 3-assign Anonymität verlangt ist, erfüllen beide neuen Tupel in C_1 das Schutzkriterium. Da für die neuen Tupel in C_2 analoge Werte gelten, wird 7 in keines der beiden Cluster eingefügt. Ist hingegen 4-assign Anonymität gefordert, erfüllen jeweils beide neuen Tupel in C_1 und C_2 nicht das Kriterium und es wird ein Nutzen mittels sim-Funktion berechnet. Da $sim(7, C_1) = sim(7, C_2) = 4$, kann 7 diesmal in C_1 oder C_2 eingefügt werden.

Ob das Schutzkriterium bei neuen Tupeln während des Clusterns erfüllt ist, kann bei k-assign Anonymität recht einfach und effizient überprüft werden. Dazu wird im Algorithmus 7.6 für jedes Cluster C der kleinste vorkommende aktuelle Anonymitätsgrad der neuen Tupel gespeichert. Wenn dieser Wert mindestens k ist, erfüllen alle neuen Tupel in C k-assign Anonymität. Bezeichne $C_{\text{alt}} \subseteq C$ die Menge aller alten und $C_{\text{neu}} \subseteq C$ die Menge aller neuen Tupel. Dann ist der minimale aktuelle Anonymitätsgrad

$$a^*_{\min}(C_{\text{neu}}) = \min_{t \in C_{\text{neu}}} a^*(t). \tag{7.10}$$

Sei t ein neues Tupel, das in Cluster C eingefügt werden soll. Kommt noch kein anderes neues Tupel in C vor, erhält t in diesem Fall eine Matchingkante zu den SA-Werten aller alten Tupel, in deren Signatur der SA-Wert von t vorkommt. Die Anzahl dieser Tupel ist nach Definition 7.3 $\frac{sim(t,C)}{2}$ und der aktuelle Anonymitätsgrad von t ist um 1 größer. Somit ist der minimale aktuelle Anonymitätsgrad der neuen Tupel nach dem Einfügen

$$a^*_{\min}((C \cup \{t\})_{\text{neu}}) = \frac{sim(t, C)}{2} + 1. \tag{7.11}$$

Wenn in C bereits mindestens ein neues Tupel vorhanden ist, gibt es zwei Möglichkeiten, welches neue Tupel in $(C \cup \{t\})_{\text{neu}}$ den minimalen aktuellen Anonymitätsgrad hat: entweder das Tupel, dessen aktueller Anonymitätsgrad bereits in C_{neu} minimal war oder das neu eingefügte Tupel t. Da sich der Grad von allen neuen Tupeln in C durch das Einfügen von t um 1 vergrößert, folgt

$$a^*_{\min}((C \cup \{t\})_{\text{neu}}) = \min\{a^*_{\min}(C_{\text{neu}}) + 1, \frac{sim(t, C)}{2} + 1\}. \tag{7.12}$$

Im Anhang wird in Beispiel A.23 auf Seite 299 ein sehr umfangreiches Clustering der neuen Tupel erstellt. Dabei wird insbesondere auch der aktuelle minimale Anonymitätsgrad der neuen Tupel berechnet. Ein etwas kürzeres Beispiel folgt im nächsten Abschnitt.

7.5.4 Konstruktion von SA-eindeutigen Anfragegraphen

Mithilfe der vorgestellten Algorithmen 7.4, 7.5 und 7.6 kann aus einem gegebenen Anfragegraphen G_i eine Menge von jeweils SA-eindeutigen Anfragegraphen erstellt werden. Dazu werden zuerst die alten Tupel aus G_i mithilfe von Algorithmus 7.5 geclustert. Danach wird Algorithmus 7.6 verwendet, um den entstandenen Clustern neue Tupel zuzuweisen. Aus allen neuen Tupeln, die in kein Cluster einsortiert wurden, werden mithilfe von Algorithmus 7.4 weitere Cluster erstellt. Zum Schluss können aus allen Clustern SA-eindeutige Anfragegraphen konstruiert werden. Dieses Vorgehen wird in Algorithmus 7.7 zusammengefasst.

Algorithmus 7.7 Teile Anfragegraph

Eingabe: G_i: Anfragegraph
Ausgabe: $G_{i.1}, \ldots, G_{i.m}$: SA-eindeutige Anfragegraphen mit $G_{i.j} \subseteq G_i$ für alle j

```
1: if G_i ist SA-eindeutig then
2:    return G_i
3: end if
4: S_alt ← Menge aller alten Tupel aus G_i
5: S_neu ← Menge aller neuen Tupel aus G_i
6: Erstelle Clustering von S_alt mithilfe von Algorithmus 7.5
7: Clustere neue Tupel aus S_neu mithilfe von Algorithmus 7.6
8:    Dabei werden aus nicht einsortierten neuen Tupeln mithilfe von Algorithmus 7.4
      neue Cluster erstellt
9: Erstelle aus allen Clustern jeweils einen Anfragegraphen G_i.j
```

Die Transformation eines beliebigen Anfragegraphen in eine Menge von SA-eindeutigen Graphen wird im folgenden Beispiel erläutert.

Beispiel 7.7 Ein Anfragegraph G_{n+1} ist in Abbildung 7.10 gegeben und besteht aus vier alten und acht neuen Tupeln. Da G_{n+1} nicht SA-eindeutig ist, werden mithilfe von Algorithmus 7.7 Teilanfragegraphen erstellt, in denen kein SA-Wert mehrfach vorkommt. Das Schutzkriterium sei k-assign Anonymität mit $k = 3$.

Tabelle 7.8 gibt einen Überblick über den gesamten Clusteringprozess. Als Erstes werden die alten Tupel nach Algorithmus 7.5 geclustert. Dabei entstehen zwei Cluster C_1 und C_2 mit jeweils zwei alten Tupeln (siehe Spalten für Tupel 3 und 4). Danach werden die neuen

ID	SA	*Sig*
1	A	ABDE
2	A	ACF
3	B	ABEG
4	C	ACE

(a) Menge alter Tupel in G_{n+1}

ID	SA	*Sig*
5	A	neu
6	B	neu
7	B	neu
8	D	neu
9	E	neu
10	F	neu
11	F	neu
12	G	neu

(b) Menge neuer Tupel in G_{n+1}

Abbildung 7.10: Anfragegraph G_{n+1}

	Init	3		4		5			6/7		
	C_i	sim	C_i	sim	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i
C_1	$\{1\}$	2	$\{1,3\}$	$-\infty$	$\{1,3\}$	0	$-\infty$	$\{1,3\}$	0	$-\infty$	$\{1,3\}$
C_2	$\{2\}$	$-\infty$	$\{2\}$	2	$\{2,4\}$	0	$-\infty$	$\{2,4\}$	0	0	$\{2,4\}$

	8			9			10		
	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i
C_1	0	2	$\{1,3,8\}$	2	4	$\{1,3,8,9\}$	3	0	$\{1,3,8,9\}$
C_2	0	0	$\{2,4\}$	0	2	$\{2,4\}$	0	2	$\{2,4,10\}$

	11			12			6		
	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim	C_i
C_1	3	0	$\{1,3,8,9\}$	3	2	$\{1,3,8,9,12\}$	4	–	$\{1,3,8,9,12\}$
C_2	2	$-\infty$	$\{2,4,10\}$	2	0	$\{2,4,10\}$	2	2	$\{2,4,6,10\}$

Tabelle 7.8: Clustern von Tupeln nach Algorithmus 7.7

Tupel den Clustern zugeordnet, wobei mit sim_0 zunächst der Nutzen nur zu alten Tupeln berechnet wird. Die Tupel 5, 6 und 7 haben jeweils keinen positiven Nutzen und werden vorerst zurückgestellt (vgl. Algorithmus 7.6). Da $sim_0(8, C_1) = sim(8, \{1,3\}) = 2$ gilt, wird Tupel 8 C_1 zugeordnet und hat dort einen aktuellen Anonymitätsgrad von $\frac{sim(8,C_1)}{2} + 1 = 2$. Tupel 9 hat zu beiden Clustern einen positiven sim_0-Wert, wird aber ebenfalls in C_1 eingefügt, da $sim_0(9, C_1) > sim_0(8, C_2)$ gilt. Der Anonymitätsgrad der neuen Tupel ist danach $a^*_{\min}((C_1 \cup \{9\})_{\text{neu}}) = \min\{a^*_{\min}(C_{1\text{neu}}) + 1, \frac{sim(9,C_1)}{2} + 1\} = \min\{3, 4\} = 3$. Die weiteren Tupel 10 (in C_2), 11 (zurückgestellt) und 12 (in C_1) werden ähnlich behandelt.

In einem zweiten Schritt werden die vier neuen Tupel 5, 6, 7 und 11, die noch nicht eingefügt wurden, erneut betrachtet. Da der minimale aktuelle Anonymitätsgrad der neuen Tupel in C_1 größer und in C_2 kleiner als k ist, können Tupel nur in C_2 eingefügt werden. Es gilt $sim(6, C_2) = sim(6, 10) = 2$ und Tupel 6 wird nun C_2 zugeordnet.

Tupel 5, 6 und 11 können auch im zweiten Schritt in kein Cluster eingefügt werden und werden zum Schluss mithilfe von Algorithmus 7.6 partitioniert. Dadurch entsteht eine weitere Menge von Tupeln. Das resultierende Clustering ist in Abbildung 7.11 dargestellt. Werden daraus die Anfragegraphen erstellt und Δ-top Matchings berechnet, muss die

Labelkante $(4, \mathrm{E})$ gelöscht werden. Somit entsteht eine Verletzung des Schutzkriteriums, denn Tupel 4 hat nur noch eine Δ-top Matchingkante mit SA-Wert A ($a(4) = 2 < k$). Auch der Anonymitätsgrad des neuen Tupels 6 ist nur 2, denn für 6 wird nur die Kante $(6, \mathrm{F}(10))$ gespeichert.

ID	SA	*Sig*
1	A	ABDE
3	B	ABEG
8	D	neu
9	E	neu
12	G	neu

(a) C_1/G_{n+1_1}

ID	SA	*Sig*
2	A	ACF
4	C	ACE
6	B	neu
10	F	neu

(b) C_2/G_{n+1_2}

ID	SA	*Sig*
5	A	neu
7	B	neu
11	F	neu

(c) C_3/G_{n+1_3}

Abbildung 7.11: Clustering von G_{n+1}

Wird ein Anfragegraph in mehrere SA-eindeutige Teilanfragegraphen zerlegt, werden diese wie einzelne Anfragegraphen behandelt. Das bedeutet, die Berechnung von Δ-top Matchings kann auf jeden einzelnen Teilanfragegraphen durchgeführt werden. Somit können auch die Ideen und Algorithmen dieses Kapitels, wie insbesondere Algorithmus 7.1, auf Anfragegraphen verwendet werden, die gegen die SA-Eindeutigkeit verstoßen. Algorithmus 7.7 ist nur ein zusätzlicher Vorverarbeitungsschritt für Algorithmus 7.1 (*Berechne Δ-top Matchings für SA-eindeutige Anfragegraphen*). Damit ist der Ansatz in diesem Kapitel eine echte Alternative zu den im Kapitel 5 vorgestellten Ideen und Algorithmen.

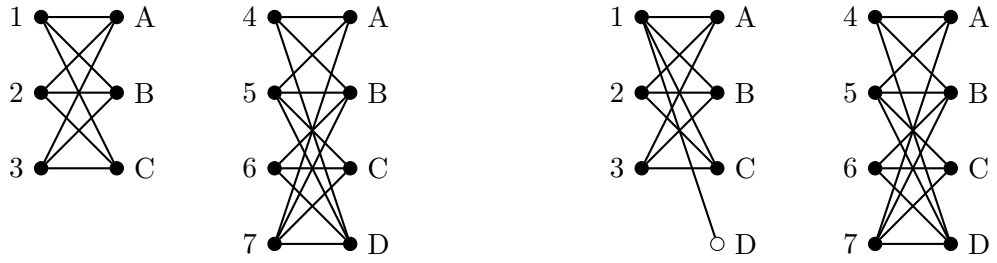
7.6 k-assign Anonymität durch Hinzufügen künstlicher sensibler Werte

Trotz des Auftretens von Verletzungen des Schutzkriteriums gibt es auch bei SA-eindeutigen Graphen Möglichkeiten, jede dazugehörige Anfrage zu beantworten. Ähnlich zu dem Ansatz für allgemeine Anfragegraphen in Kapitel 5.6 werden dazu bei Bedarf künstliche SA-Knoten eingefügt. Die entsprechenden resultierenden Ergebnisse enthalten neben allen realen auch zusätzliche SA-Werte. Um das zu erreichen, wird die *Lösche*-Routine (vgl. Algorithmus 7.2) so verändert, dass Kanten nur genau dann entfernt werden, wenn danach keine Verletzung von k-assign Anonymität vorliegt. Falls ein Tupel existiert, dessen Anonymitätsgrad kleiner als k ist, wird der komplette Löschschritt rückgängig gemacht und ein künstlicher SA-Knoten in den Graphen eingefügt. Die Details orientieren sich an denen aus Kapitel 5.6, sodass an dieser Stelle nur auf Unterschiede eingegangen wird.

In SA-eindeutigen Anfragegraphen gibt es keine Verletzungen von Δ-Pfadkonformität und das Erweitern von Matchings ist wesentlich einfacher als in beliebigen Graphen. In Algorithmus 7.1 (*Berechne Δ-top Matchings*) existieren daher nur zwei Stellen, an denen Δ-top Matchingkanten gelöscht werden. Kommt der gewünschte SA-Wert s' nicht im Graphen vor (Zeile 5), kann durch das Hinzufügen eines entsprechenden SA-Knotens die gerade betrachtete Matchingkante erhalten bleiben. Dieser Schritt ist analog zum Verfahren in Kapitel 5.6. Einen Unterschied dazu bildet hingegen der Löschschritt in Zeile 11. In diesem Fall kann eine übergebene Matchingkante nicht mit einem Tupel erweitert werden,

der enthaltene SA-Wert kommt allerdings bereits im Graphen vor.[17] Das Einfügen eines entsprechenden SA-Knotens würde demnach die SA-Eindeutigkeit verletzen.

Abhilfe schafft das in Kapitel 7.5.2 eingeführte Clusterkriterium. Es besagt, dass für jeweils zwei alte Tupel in jedem Graphen gilt, dass der originale SA-Wert des einen genau dann in der Signatur des anderen ist, wenn das auch umgekehrt der Fall ist. Solange es in einem Anfragegraphen Gültigkeit hat, kann der beschriebene Löschfall nicht eintreten und das Einfügen von SA-Knoten funktioniert analog zu den allgemeinen Anfragegraphen. Die Herausforderung besteht demnach darin, künstliche SA-Knoten bereits dann einzufügen, wenn das Clusterkriterium in einem Cluster nicht erfüllt ist. Das ist sogar unabhängig davon, ob das Schutzkriterium gilt oder nicht.



(a) G_{n+1_1} (b) G_{n+1_2} (c) G'_{n+1_1} (d) G'_{n+1_2}

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}'^{(n+1)}$
1	A	B($x_{1.1}$), C($x_{1.2}$), D(7)	B($x_{1.1}, 2$), C($x_{1.2}, 3$), D(7)
2	B	A($x_{2.1}$), C($x_{2.2}$)	A($x_{2.1}, 1$), C($x_{2.2}, 3$)
3	C	A($x_{3.1}$), B($x_{3.2}$)	A($x_{3.1}, 1$), B($x_{3.2}, 2$)
4	A	B($x_{4.1}$), D($x_{4.2}$)	B($x_{4.1}, 5$), D($x_{4.2}, 7$)
5	B	A($x_{5.1}$), C($x_{5.2}$), D($x_{5.3}$)	A($x_{5.1}, 4$), C($x_{5.2}, 6$), D($x_{5.3}, 7$)
6	C	B($x_{6.1}$), D($x_{6.2}$)	B($x_{6.1}, 5$), D($x_{6.2}, 7$)
7	D	A(1), B($x_{7.1}$), C($x_{7.2}$)	A(1), B($x_{7.1}, 5$), C($x_{7.2}, 6$)

(e) Δ-top Matchingtabelle

Abbildung 7.12: Erweiterung inkl. künstlicher SA-Knoten

Sei G_{n+1} ein Anfragegraph, der aus den alten Tupeln 1 bis 7 bestehe und nicht SA-eindeutig sei. Das in Kapitel 7.5 beschriebene Verfahren clustere die Tupel und erzeuge zwei Anfragegraphen, die in Abbildung 7.12 in (a) und (b) dargestellt sind. G_{n+1_1} enthalte die Tupel 1 bis 3 und G_{n+1_2} die Tupel 4 bis 7. Sei eine Δ-top Matchingtabelle für eine Folge von n Anfragegraphen in (e) gegeben (Spalte $\mathcal{G}^{(n)}$). Die aufgelisteten Matchings sollen mithilfe von Algorithmus 7.1 erweitert werden, wobei jedes $x_{i.j}$ für eine Menge (bzw. Liste) von Tupeln stehe, die hier nicht angegeben sind.

Bei der Erweiterung der Matchings aus G_{n+1_1} stellt sich heraus, dass die Δ-top Matchingkante $(1, \mathrm{D}(7))$ gelöscht werden muss, da kein Knoten mit SA-Wert D vorhanden ist (Zeile 5). Dadurch muss auch die Gegenkante $(7, \mathrm{A}(1))$ entfernt werden. Ist 3-assign Anonymität gefordert, stellt das sowohl für Tupel 1 als auch für 7 kein Problem dar, denn beide haben nach dem Entfernen einen Anonymitätsgrad von 3. Allerdings ist das Clusterkriterium in G_{n+1_2} (bzw. im dazugehörigen Cluster) nicht mehr erfüllt. Tupel 4 hat den

[17] Vgl. z. B. Kante $(4, \mathrm{C}(3))$ in Beispiel 7.4 auf Seite 208.

originalen SA-Wert A und ein D in seiner Signatur. Nach dem Löschen enthält die Signatur von Tupel 7 aber kein A mehr, wodurch Gleichung 7.2 verletzt ist. Das hat zur Folge, dass später bei der Erweiterung der Matchings aus G_{n+1_2} die Kante $(4, \mathrm{D}(x_{4.2}))$ gelöscht werden muss (Zeile 11). Im dargestellten Beispiel würde damit 3-assign Anonymität bei Tupel 4 verletzt werden. Wird hingegen ein künstlicher SA-Knoten mit Label D in G_{n+1_2} eingefügt, um $(4, \mathrm{D}(x_{4.2}))$ erweitern zu können, wäre der entstehende Graph nicht mehr SA-eindeutig.

Dieses Problem kann umgangen werden, indem die Kante $(1, \mathrm{D}(7))$, deren Löschen zur Verletzung des Clusterkriteriums führte, nicht entfernt wird. Folglich muss bereits in den Graphen G_{n+1_1} ein künstlicher SA-Knoten mit Label D eingefügt werden. Abbildung (c) stellt den resultierenden Anfragegraphen G'_{n+1_1} dar.

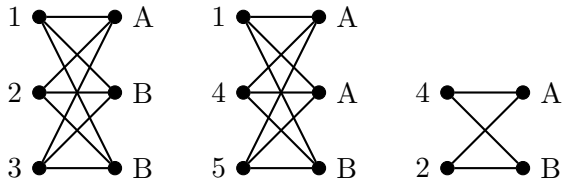
Im Allgemeinen wird eine Δ-top Matchingkante aus einem Graphen $G_{i.j}$ nicht gelöscht, wenn ein Graph $G_{i.k}$ existiert, in dem durch das Löschen eine Verletzung des Clusterkriteriums hervorgerufen wird. Dabei werden alle noch nicht bearbeiteten Graphen $G_{i.k}$ des Clusterings, welches für den Ursprungsgraphen G_i erzeugt wurde, überprüft. Die Umsetzung dieser Idee erfolgt in Algorithmus 7.2 (*Lösche*). Für das in der übergebenen Matchingkante enthaltene Tupel t werden alle Cluster getestet, die t enthalten. Bei Bedarf wird das Löschen durch das Einfügen eines künstlichen SA-Knotens in $G_{i.j}$ verhindert.

Als Letztes sei erwähnt, dass bei einer Verletzung bei einem neuen Tupel darauf geachtet werden muss, nur SA-Werte hinzuzufügen, die noch nicht im Anfragegraphen vorhanden sind. Ein Ziel dabei lautet, Werte nicht willkürlich auszuwählen, sondern möglichst Werte zu nehmen, die mit dem originalen SA-Wert des neuen Tupels in Beziehung stehen. Dazu können beispielsweise Werte aus gerade gelöschten Matchingkanten alter Tupel oder Werte aus anderen Clustern verwendet werden. Letzteres sind sensible Werte, die im Ergebnis zu der entsprechenden Anfrage bereits vorkommen.

7.7 Kombination verschiedener Approximationsalgorithmen

Da die Algorithmen für SA-eindeutige Graphen eine Alternative zu denen für beliebige Graphen sind, stellt sich die Frage, ob beide Varianten zusammen kombiniert werden können. Hierbei ist insbesondere der Fall von Interesse, wenn ein Anfragegraph SA-eindeutig ist, vorige Graphen das aber nicht waren.

Abbildung 7.13 zeigt eine Folge von Anfragegraphen $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$. Die ersten beiden enthalten jeweils doppelte SA-Werte und Matchings werden mit den Algorithmen aus Kapitel 5 berechnet. Die entstehende Δ-top Matchingtabelle ist in (d) angegeben (Spalten $\mathcal{G}^{(1)}$ und $\mathcal{G}^{(2)}$). Da G_3 SA-eindeutig ist, könnten auch die Algorithmen aus diesem Kapitel verwendet werden. Demnach können die Tupel 2 und 4 jeweils miteinander erweitert werden. In der vierten Spalte der Matchingtabelle sind die erzeugten Kanten aufgelistet. Werden hingegen alle Teilgraphen zusammen betrachtet, ist schnell offensichtlich, dass kein valides Matching mit der Kante $(2, \mathrm{A})$ existieren kann. Ein solches Matching müsste nämlich in G_1 auch die Kante $(1, \mathrm{B})$ und demnach in G_2 die Kante $(4, \mathrm{A})$ enthalten. In G_3 können aber die Tupel 2 und 4 nicht gleichzeitig mit A matchen, da nur ein Knoten mit Label A vorkommt. Die Erweiterung, die die SA-Eindeutigkeit ausnutzt, ist in diesem Fall falsch. Eine Erweiterung analog zu beliebigen Anfragegraphen würde die Kanten $(2, \mathrm{A}(1))$ und $(4, \mathrm{B}(5))$ löschen, denn alte Tupel dürfen nicht miteinander erweitert werden. Damit



(a) G_1 (b) G_2 (c) G_3

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$ (SA-eindeutig)	$\mathcal{G}^{(3)}$ (beliebig)
1	A	B(2), B(3)	B(2, 5), B(3, 5)	B(2, 5), B(3, 5)	B(3, 5)
2	B	A(1)	A(1)	A(1, 4)	–
3	B	A(1)	A(1)	A(1)	A(1)
4	A		B(5)	B(5, 2)	–
5	B		A(1), A(4)	A(1), A(4)	A(1)

(d) Δ-top Matchingtabelle

Abbildung 7.13: Kombination von beliebigen und SA-eindeutigen Graphen

bleibt nur noch ein Matching erhalten, in dem Tupel 1 mit B und die Tupel 3 sowie 5 jeweils mit A matchen. Dieses Matching ist auch tatsächlich valid.

Die Konsequenz dieser Überlegungen ist, dass die Algorithmen der Kapitel 5 und 7 nicht ohne Weiteres kombiniert werden können. Insbesondere muss das Verfahren für beliebige auch für SA-eindeutige Anfragegraphen verwendet werden, wenn es in vorigen Graphen bereits benutzt wurde. Andersherum ist es aber möglich, zunächst mit Algorithmen für SA-eindeutige Graphen zu arbeiten und dann auf die Algorithmen für beliebige Anfragegraphen zu wechseln. Das gilt, da die weiteren Einschränkungen der letztgenannten Verfahren sicherstellen, dass weiterhin nur valide Matchings erzeugt werden.[18]

Zusammenfassung

Die Algorithmen in diesem Kapitel stellen eine Alternative zu denen aus Kapitel 5 dar. Durch die Beschränkung auf SA-eindeutige Anfragegraphen können Erweiterungen von Matchings effizienter und ohne die Einschränkungen wie bei beliebigen Graphen berechnet werden. Insbesondere ist es möglich, alte Tupel mit anderen alten Tupeln zu erweitern. Dadurch kann auf spezielle Erweiterungsalgorithmen wie die in Kapitel 6 verzichtet werden. Die Speicherung der Matchings in Tabellen wird übersichtlicher, denn einzelne Kanten charakterisieren bereits komplette Matchings. Sind Ergebnisse zu Anfragen gegeben, in denen sensible Werte nicht mehrfach vorkommen, ist dieser Ansatz dem aus Kapitel 5 vorzuziehen. Ist das nicht der Fall, ermöglicht ein zusätzliches Clustering, aus beliebigen Anfragegraphen SA-eindeutige Teilgraphen zu erstellen. Damit können die Algorithmen aus diesem Kapitel auch mit beliebigen Anfragegraphen verwendet werden. Ein Nachteil entsteht jedoch dadurch, dass die durch den Clusteransatz erzeugten Graphen recht klein werden können, insbesondere wenn die enthaltenen Tupel verschiedene Signaturen besitzen. Somit werden Verletzungen des Schutzkriteriums eher prognostiziert und mit

[18] Vgl. Kapitel 5.4.

künstlichen Werten kompensiert, als es beim Ansatz für beliebige Anfragegraphen der Fall ist. In Kapitel 8 werden beide Verfahren experimentell miteinander verglichen.

8 Evaluation

In diesem Kapitel werden die vorgestellten Algorithmen evaluiert. Zunächst wird kurz zusammengefasst, welche Algorithmen untersucht werden, ehe die Testumgebung vorgestellt wird (Kapitel 8.1). Danach wird experimentell die Güte der Approximationen abgeschätzt (Kapitel 8.2). Da dazu die Eingabegrößen stark beschränkt werden, werden bei Tests mit großen Datenmengen nur die Approximationsalgorithmen miteinander verglichen (Kapitel 8.3). Es wird gezeigt, dass es Eingabedaten gibt, bei denen die Verfahren aus Kapitel 7 besser geeignet sind als die aus Kapitel 5, während meistens eher das Umgekehrte der Fall ist (Kapitel 8.4). Als Letztes werden Tests mit Daten aus der realen Welt durchgeführt (Kapitel 8.5), ehe ein abschließendes Fazit gezogen wird (Kapitel 8.6).

8.1 Testaufbau und Ziele

Im Rahmen der vorliegenden Arbeit werden die in den Kapiteln 3, 5 und 7 vorgestellten Algorithmen evaluiert (siehe Tabelle 8.1). Für die exakte Überprüfung von k-assign Anonymität dient Algorithmus 3.2 (*Teste k-assign Anonymität*), der so verändert wird, dass er anstelle von „true“ oder „false“ die berechneten Signaturen und Anonymitätsgrade der Tupel ausgibt. Ist k-assign Anonymität verletzt, werden dem Ergebnis ähnlich wie bei den anderen Algorithmen künstliche sensible Werte hinzugefügt. Damit die Güte der Approximationen abgeschätzt werden kann, muss garantiert werden, dass die Größe der Menge zusätzlicher SA-Werte minimal ist. Folglich kann hierfür nicht die im Kapitel 5.6 vorgestellte Heuristik benutzt werden. Allerdings stellt das Bestimmen einer minimalen Menge ein komplexes Optimierungsproblem dar, welches in dieser Arbeit nicht näher untersucht wird. Es wird vielmehr ein einfacher Brute-Force-Ansatz verwendet, der mögliche Mengen sukzessive testet, bis er die kleinste gefunden hat. Die größte zu testende Menge stellt dabei die Vereinigung aller SA-Werte dar, die in den Signaturen der Tupel auftreten, welche im aktuellen Ergebnis vorkommen. Entsprechend ihrer Häufigkeit können SA-Werte in der besagten Menge auch mehrfach vorhanden sein. Der resultierende Algorithmus wird mit *ILP* bezeichnet, da er auf ganzzahliger linearer Programmierung (engl. *integer linear programming*) beruht.

Als Approximationsalgorithmen werden die Verfahren aus den Kapiteln 5 und 7 getestet. In Algorithmus 5.3 (*Berechne Δ-top Matchings*) werden die vier in Kapitel 6 eingeführten Erweiterungsalgorithmen *Maximale Kanten*, *Maximaler Anonymitätsgrad*, *Einfaches Matching* und *Klonmatching* eingesetzt und die entstehenden Algorithmen dementsprechend *MaxKanten* (MK), *MaxAno* (MA), *SimpelMatching* (SM) beziehungsweise *KlonMatching* (KM) genannt. Für den Spezialfall der SA-eindeutigen Ergebnisse von Anfragen wird Algorithmus 7.1 (*Berechne Δ-top Matchings (für SA-eindeutige Anfragegraphen)*) verwendet, welcher zusammen mit Algorithmus 7.7 (*Teile Anfragegraph*) auch mit beliebigen Ergebnissen benutzt werden kann. Diese Kombination wird in der Evaluation mit *SA-Ein* (SAE) bezeichnet. Als Letztes wird ein Algorithmus getestet, der eine Erweiterung des Prinzips der m-Invarianz darstellt und in Kapitel 7.1 beschrieben wurde. Er wird mit *M-Inv* (MI)

Name		Nummer	Seite
ILP	ILP	Algorithmus 3.2	95
MaxKanten	MK	Algorithmus 5.3 mit Erweiterungsalgorithmus 6.1	150, 162
MaxAno	MA	Algorithmus 5.3 mit Erweiterungsalgorithmus 6.3	150, 164
SimpelMatching	SM	Algorithmus 5.3 mit Erweiterungsalgorithmus 6.4	150, 170
KlonMatching	KM	Algorithmus 5.3 mit Erweiterungsalgorithmus 6.10	150, 194
SA-Ein	SAE	Algorithmus 7.1 mit Vorverarbeitungsalgorithmus 7.7	204, 220
M-Inv	MI	Beschreibung in Kapitel 7.1 in Verbindung mit 2.5	197, 58

Tabelle 8.1: Evaluierte Algorithmen

abgekürzt und steht für bereits existierende Methoden aus dem Bereich der Veröffentlichung von Mikrodaten, die auf das Szenario der Anfrageauditierung übertragen werden können. Ein essenzielles Ziel der vorliegenden Arbeit besteht darin, Algorithmen zu entwickeln, die für das grundlegende Problem der Anfragebearbeitung besser geeignet sind als bereits existierende Methoden. Daher stellt *M-Inv* eine Art Referenzalgorithmus dar, dessen Güte von den eingeführten Approximationen unterboten werden soll.

Als Testumgebung wird ein PC mit zwei Intel-Quad-Core-Prozessoren Xeon E5520 mit 2,26 GHz Taktfrequenz und 12 GB RAM verwendet. Die Algorithmen wurden in C# implementiert und laufen unter dem Betriebssystem Windows 7 Professional. Für das Lösen der ganzzahligen linearen Programme werden Bibliotheken der Microsoft Solver Foundation [107] benutzt.

Im Folgenden werden die getesteten Szenarien vorgestellt. Dazu wird jeweils eine Tabelle mit personenbezogenen (sensiblen) Daten und eine Folge von Anfragen erstellt. Die Privatsphäre der Individuen gilt als ausreichend geschützt, wenn die Folge von dazugehörigen Ergebnissen k-assign Anonymität für ein gegebenes k nicht verletzt. Bei Bedarf ist das Hinzufügen künstlicher SA-Werte möglich, um das Schutzkriterium zu erfüllen. Die wichtigsten Vergleichskriterien für die Algorithmen sind die Anzahl künstlicher SA-Werte und die Laufzeit. Je schneller ein Algorithmus die Anfrage bearbeitet und je weniger künstliche Werte erzeugt werden, desto geeigneter ist er für das getestete Szenario.

8.2 Vergleich exakter mit approximierenden Algorithmen

Zunächst wird die Güte der Approximationsalgorithmen experimentell abgeschätzt. Dazu werden sie mit dem Algorithmus *ILP* verglichen, der auf ganzzahliger linearer Programmierung beruht und die exakte Berechnung von k-assign Anonymität garantiert. Aufgrund seiner exponentiellen Laufzeit müssen die Eingabewerte, wie zum Beispiel die Anzahl der Tupel in der Datentabelle, stark beschränkt werden.

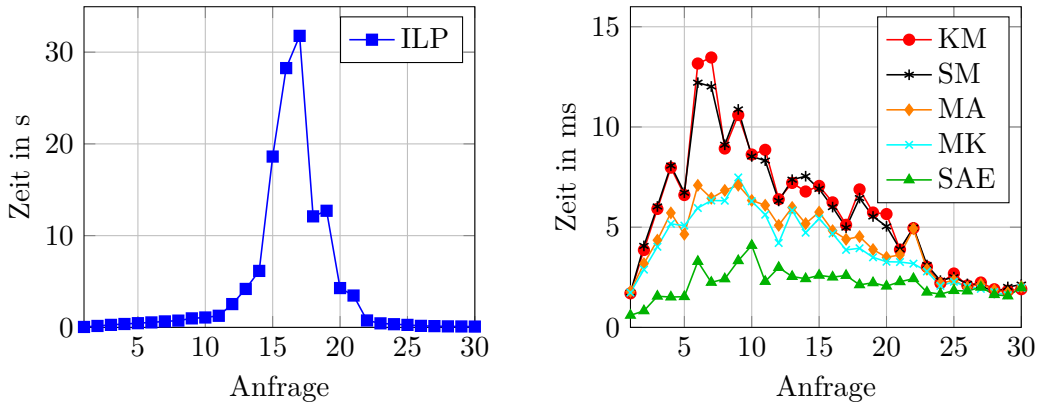
8.2.1 Ohne Einhaltung eines Schutzkriteriums

Im ersten Test wird $k = 1$ gesetzt, wodurch kein Schutz der Privatsphäre gefordert ist und nur die Veränderungen der Anonymitätsgrade von Tupeln verglichen werden können. Dazu werden eine Datentabelle und eine Folge von Anfragen ausgewählt, die mit den Algorithmen ausgewertet werden. Die verwendeten Eingabegrößen sind in Tabelle 8.2 angegeben. Aus einer Tabelle, die 100 Tupel enthält, werden zufällig gleichverteilt acht Tupel

pro Anfrage beziehungsweise Ergebnis ausgewählt. Auf diese Weise werden insgesamt 30 Ergebnisse konstruiert. Da auch in der Datentabelle acht verschiedene SA-Werte vorkommen, ist der maximale Anonymitätsgrad eines Tupels acht. Dieser und alle folgenden Tests werden jeweils mehrfach wiederholt, wodurch in allen Diagrammen im aktuellen Kapitel Durchschnittswerte angegeben werden können.

Datenbankgröße	100 Tupel
Anzahl der SA-Werte	8
Ergebnisgröße	8 Tupel
Anzahl der Anfragen	30
Schutzkriterium	keins

Tabelle 8.2: Testsetup 1 (Auswertung in Abbildungen 8.1 und 8.2)

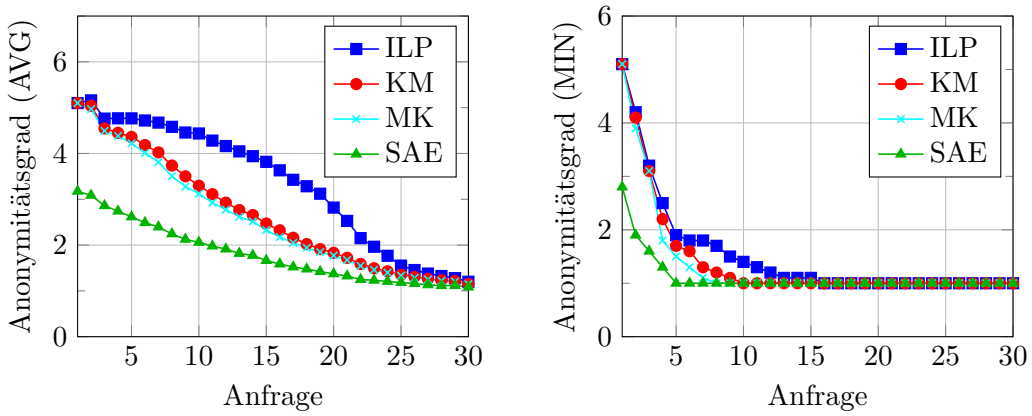


(a) *ILP* (b) Approximationsalgorithmen

Abbildung 8.1: Laufzeit (Testsetup siehe Tabelle 8.2)

Abbildung 8.1 stellt die Laufzeit der verwendeten Algorithmen im beschriebenen Testfall dar. Zu jeder der 30 Anfragen ist auf der Y-Achse die Zeit angegeben, die die Bearbeitung dieser Anfragen benötigt. Dabei werden nur die tatsächlichen Berechnungen der Algorithmen und keine Operationen des DBMS beachtet. Die Zahlen entsprechen damit genau der Zeit, die zusätzlich zur normalen Auswertung der Anfrage nötig ist, um den Schutz der Privatsphäre sicherzustellen. Es ist vorhersehbar, dass der exakte Algorithmus *ILP* wesentlich mehr Zeit benötigt als die getesteten Approximationsalgorithmen. Einen Höchstwert von knapp 32 s erreicht *ILP* bei der 17. Anfrage und ist davor und danach zum Teil deutlich schneller. Die Approximationsalgorithmen erreichen ihr jeweiliges Maximum bereits zwischen der 6. und 10. Anfrage, wobei die größte Laufzeit ca. 13 ms beträgt. Damit sind sie in etwa um den Faktor 2460 schneller als *ILP*. Insgesamt ist *KlonMatching* am langsamsten und *SA-Ein* am schnellsten. Da *M-Inv* nur korrekt funktioniert, wenn auch künstliche SA-Werte eingefügt werden dürfen, kann dieser Algorithmus im beschriebenen Szenario nicht verwendet werden.

Eine Erklärung für den Verlauf der angegebenen Laufzeiten liefert ein Vergleich der Anonymitätsgrade aller Tupel in Abbildung 8.2. Anfangs existieren nur wenige Tupel, für die das Schutzkriterium überprüft werden muss, wodurch alle Algorithmen schnell



(a) Durchschnittlicher Anonymitätsgrad (b) Minimaler Anonymitätsgrad

Abbildung 8.2: Sinkende Anonymitätsgrade (Testsetup siehe Tabelle 8.2)

arbeiten. Da in einem Ergebnis, welches aus acht Tupeln besteht, oft fünf verschiedene SA-Werte vorkommen, ist der durchschnittliche Anonymitätsgrad fünf. Damit existieren Werte aber auch mehrfach und der Algorithmus *SA-Ein* partitioniert das Ergebnis in zwei oder drei Teile. Es entsteht hierbei ein durchschnittlicher Anonymitätsgrad von nur 3,2.

Während weitere Anfragen gestellt und Ergebnisse verarbeitet werden, entfallen Zuordnungen von SA-Werten zu Tupeln, wenn sie nicht mehr möglich sind (*ILP*) oder nicht mehr modelliert werden (Approximationen). Demzufolge sinkt der durchschnittliche und minimale Anonymitätsgrad mit jeder Anfrage. Da jedem Tupel immer mindestens sein originaler (realer) SA-Wert zugeordnet werden kann, fällt der Anonymitätsgrad niemals unter 1. Nach der 30. Anfrage muss das Schutzkriterium zwar für viele Tupel überprüft werden, allerdings existiert für jedes Tupel meist nur noch eine Zuordnung, wodurch erneut alle Algorithmen eine schnelle Laufzeit haben. *ILP* muss bei der 17. Anfrage die meisten Zuordnungen überprüfen und hat daher an dieser Stelle seine größte Laufzeit. Die anderen Algorithmen sind nur Approximationen, ihr durchschnittlicher und minimaler Anonymitätsgrad ist jeweils geringer und sie erreichen ihr Laufzeitmaximum bereits früher als *ILP*. Der minimale Anonymitätsgrad liegt bei allen Algorithmen spätestens nach der 16. Anfrage bei 1, sodass zu diesem Zeitpunkt einem Individuum in jedem Fall sein sensibler Wert zugeordnet werden kann. Dessen Privatsphäre wäre demnach verletzt.

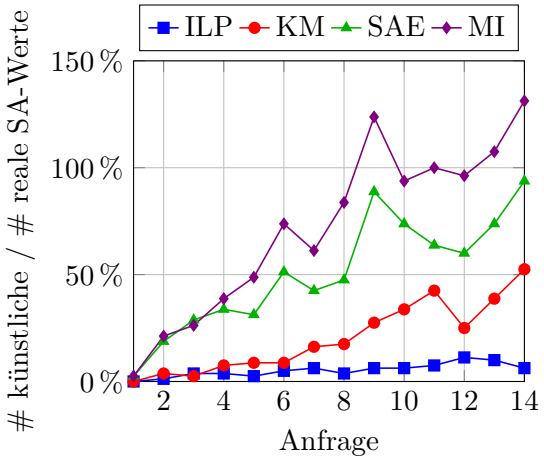
Zusammenfassend gilt, dass der Algorithmus *KlonMatching* von den Approximationen zwar die größte Laufzeit hat, den Anonymitätsgrad hingegen am besten abschätzt. Die Entwicklung der Anonymitätsgrade der nicht in Abbildung 8.2 dargestellten Algorithmen *SimpelMatching* und *MaxAno* sind nahezu identisch mit den Verläufen von *KlonMatching*.

8.2.2 Mit Einhaltung eines Schutzkriteriums

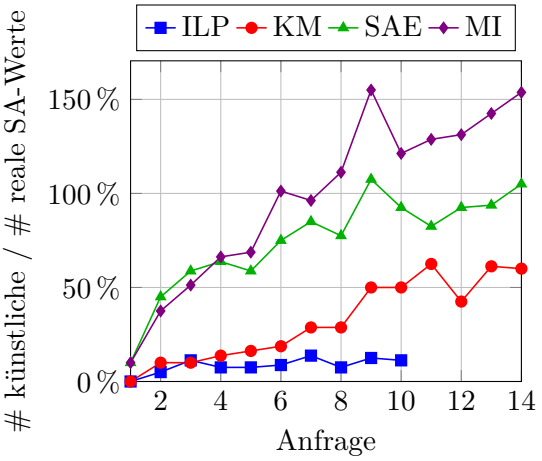
Beim folgenden Test und allen weiteren Untersuchungen wird ein Schutzkriterium in Form von k-assign Anonymität mit verschiedenen Werten für $k > 1$ gewählt. Gleichzeitig werden dem Ergebnis künstliche Werte hinzugefügt, wenn das Kriterium und damit die Privatsphäre verletzt ist. Die Eingabegrößen entsprechen zunächst denselben des ersten Tests, um eine größtmögliche Vergleichbarkeit zu erzielen. Als Wert für k wird 3 beziehungsweise 4 gewählt (vgl. Abbildung 8.3a).

Datenbankgröße	100 Tupel
Anzahl der SA-Werte	8
Ergebnisgröße	8 Tupel
Anzahl der Anfragen	14
Schutzkriterium	$k = 3; 4$

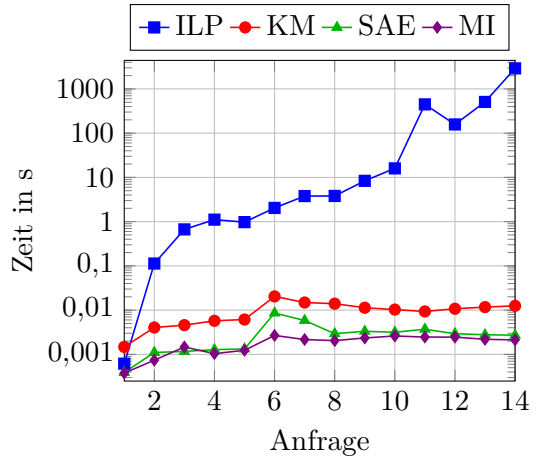
(a) Testsetup 2



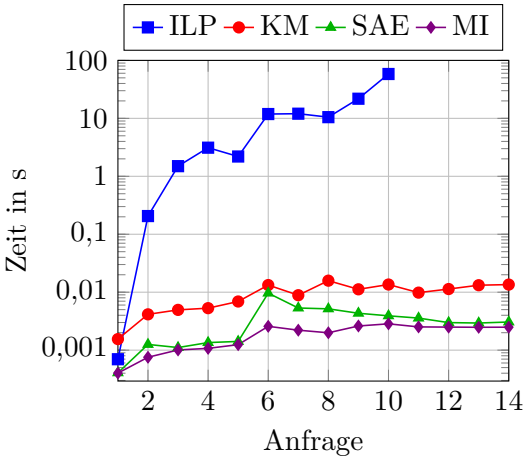
(b) Anteil künstlicher SA-Werte, $k = 3$



(c) Anteil künstlicher SA-Werte, $k = 4$



(d) Laufzeit, $k = 3$



(e) Laufzeit, $k = 4$

Abbildung 8.3: Testsetup, Anteil künstlicher SA-Werte und Laufzeit

Die Abbildungen 8.3b und 8.3c zeigen, wie viele künstliche SA-Werte zu jeder Anfrage hinzugefügt werden, um k-assign Anonymität zu gewährleisten. Dabei ist auf der Y-Achse jeweils das Verhältnis der Anzahl künstlicher zur Anzahl realer SA-Werte abgebildet. 100 % bedeuten, dass das an den Nutzer zurückgegebene Ergebnis aus genauso vielen künstlichen wie realen SA-Werten besteht. Bei 50 % beträgt das Verhältnis 1:2 und bei 200 % 2:1. Je mehr künstliche Werte erzeugt werden, desto weniger Nutzen und Aussagekraft hat das Ergebnis für weitere Analysen.

Innerhalb der ersten 14 Anfragen fügt *ILP* bei $k = 3$ nicht mehr als 12 % künstliche Werte hinzu, bei *KlonMatching* sind es höchstens 53 %, bei *SA-Ein* 94 % und bei *M-Inv* über 130 %. Für die Approximationsalgorithmen ergeben sich gemessene Güten von ca. $\frac{53}{12} = 4{,}4$ (*KlonMatching*), 7,8 (*SA-Ein*) und 10,8 (*M-Inv*). Für die anderen nicht dargestellten Algorithmen *MaxAno* und *SimpelMatching* gilt, dass sie nahezu identisch viele künstliche SA-Werte erzeugen wie *KlonMatching*. Bei *MaxKanten* liegt die Kurve insgesamt etwas darüber, jedoch noch deutlich unter der von *S-Ein*. Um Unterschiede zwischen diesen Ansätzen empirisch zu untersuchen, reichen die Größen der Eingabewerte nicht aus. Erst durch Tests mit umfangreicheren Daten wird später erkennbar, welcher Algorithmus die wenigsten künstlichen Werte erzeugt und damit die beste Güte hat. Aus diesem Grund wird auf die Darstellung von *MaxAno*, *SimpelMatching* und *MaxKanten* zunächst verzichtet.

Im Fall $k = 4$ sind die gemessenen Werte insgesamt größer. Allerdings ist die Laufzeit von *ILP* nach der 10. Anfrage bereits so hoch, dass keine weiteren Anfragen in einer vorgegebenen Zeit von 3 h beantwortet werden. Die Abbildungen 8.3d und 8.3e zeigen die Entwicklung der Laufzeiten der Algorithmen. Wie zuvor werden nur die Zeiten angegeben, die die Auswertung jedes Ergebnisses durch die Verfahren benötigt. Während die Approximationsalgorithmen nur wenige Millisekunden brauchen, liegt die Laufzeit von *ILP* im Sekundenbereich. Die logarithmische Skalierung verschleiert zwar etwas die exponentielle Steigung, ermöglicht aber auch Unterschiede im Millisekundenbereich zu erkennen. Demzufolge ist *M-Inv* der schnellste Algorithmus, gefolgt von *SA-Ein* und *KlonMatching*.

Bei Tests mit größeren Datenmengen überschreitet die Laufzeit von *ILP* oft nach nur wenigen Anfragen eine vorgegebene Zeitschranke von 3 h. Der mit Abstand größte Teil dieser Zeit wird dabei für das Lösen der ganzzahligen linearen Programme verwendet, wofür Bibliotheken der Microsoft Solver Foundation [107] benutzt wurden. Auf die Auswertung mithilfe anderer Solver wird im Rahmen der vorliegenden Arbeit verzichtet. Der extreme Anstieg der Laufzeit von einer auf die nächste Anfrage (z. B. bei $k = 3$ von 16 s bei der 10. Anfrage zu 2900 s bei der 14. Anfrage) lässt vermuten, dass auch andere Methoden nicht entscheidend größere Probleminstanzen lösen können. Um dennoch weitere Abschätzungen der Güte der Approximationen zu erreichen, wird im folgenden Abschnitt eine bestimmte Eigenschaft der Ergebnisse festgehalten.

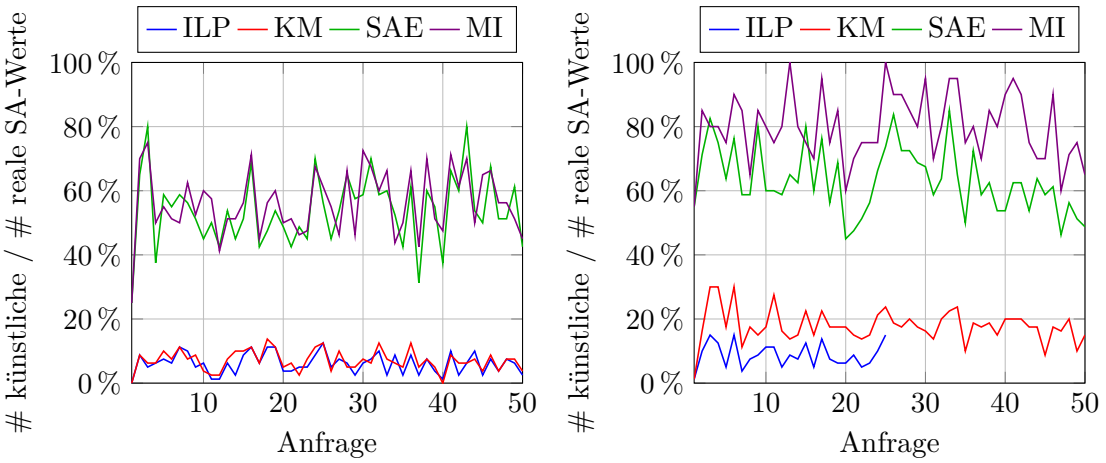
8.2.3 Fester Anteil neuer Tupel

Die in der vorliegenden Arbeit entwickelten Approximationsalgorithmen unterscheiden die Tupel im Ergebnis in alte und neue Tupel. Im Gegensatz zu den zuerst genannten kommen die letzteren in keinem vorangegangenen Ergebnis vor. Besonders viele Zuordnungen von SA-Werten zu Tupeln werden gespeichert, wenn der Anteil neuer Tupel im Ergebnis besonders hoch ist. Daher kann vermutet werden, dass in diesem Fall der Fehler der Approximation besonders klein ist.

Für den nächsten Test wird daher die Anzahl neuer Tupel in jedem Ergebnis fest vorgegeben. Abbildung 8.4a listet die jeweiligen Eingabegrößen auf. In jedem Ergebnis, welches erneut aus acht Tupeln besteht, kommen in einem Fall sieben und im zweiten Fall sechs neue Tupel vor. Im Gegensatz zu den ersten beiden Tests können auf diese Weise mit *ILP* 50 Anfragen beantwortet werden. Daraus folgt, dass auch die Gesamtzahl aller Tupel in der Datentabelle erhöht werden muss.[1]

Datenbankgröße	1000 Tupel
Anzahl der SA-Werte	8
Ergebnisgröße	8 Tupel
davon neue Tupel	7; 6
Anzahl der Anfragen	50
Schutzkriterium	$k = 4$

(a) Testsetup 3



(b) 7 neue Tupel pro Anfrage

(c) 6 neue Tupel pro Anfrage

Abbildung 8.4: Testsetup und Anteil künstlicher SA-Werte

Abbildung 8.4 zeigt, wie viele künstliche SA-Werte dem Ergebnis hinzufügt werden. Es fällt auf, dass die Kurven nicht steigen, was auf die konstante Anzahl neuer Tupel in jedem Ergebnis zurückzuführen ist. Bei sieben neuen Tupeln beträgt das Verhältnis der künstlichen zu den realen SA-Werten bei *ILP* durchschnittlich über alle 50 Anfragen hinweg 6 % und bei *KlonMatching* 7 %. Die beiden Algorithmen, die das Ergebnis partitionieren, fügen wesentlich mehr zusätzliche Werte ein. So werden bei *SA-Ein* 54 % und bei *M-Inv* 57 % künstliche SA-Werte erzeugt. Enthält jedes Ergebnis sechs neue Tupel, sind die Werte insgesamt größer und auch die Güte der Approximationen ist schlechter (z. B. *KlonMatching* 18 % zu *ILP* 9 %). Allerdings steigt die Laufzeit von *ILP* so stark an, dass die Bearbeitung der 25. Anfrage ca. 30 min benötigt und weitere Anfragen nicht in der vorgegebenen Zeit von 3 h ausgewertet werden können. Auch in diesem Fall gilt, dass sich *MaxAno* und *SimpelMatching* in etwa identisch zu *KlonMatching* verhalten und *MaxKanten* etwas mehr künstliche SA-Werte erstellt.

[1] Bei sieben neuen Tupeln pro Ergebnis und 50 Anfragen werden mindestens $8 + 49 \cdot 7 = 351$ Tupel benötigt.

8.3 Große synthetische Datenmengen

8.3.1 Gleichverteilte sensible Werte

Bei Tests mit größeren Datenbeständen steigt die Laufzeit des exakten Algorithmus *ILP* so stark, dass nur eine Handvoll Anfragen innerhalb weniger Stunden beantwortet werden können. Daher werden in den weiteren Beispielen nur die Approximationsalgorithmen ausgewertet. Im Testsetup aus Abbildung 8.5a sind die Eingabegrößen im Vergleich zu den bisherigen Szenarien deutlich erhöht worden. Beispielsweise werden nun Datenbankgrößen von 10000 und 100000 Tupeln mit bis zu 200 verschiedenen SA-Werten verwendet. Zunächst wird nur der Algorithmus *KlonMatching* bei einer gegebenen Gleichverteilung der SA-Werte untersucht.

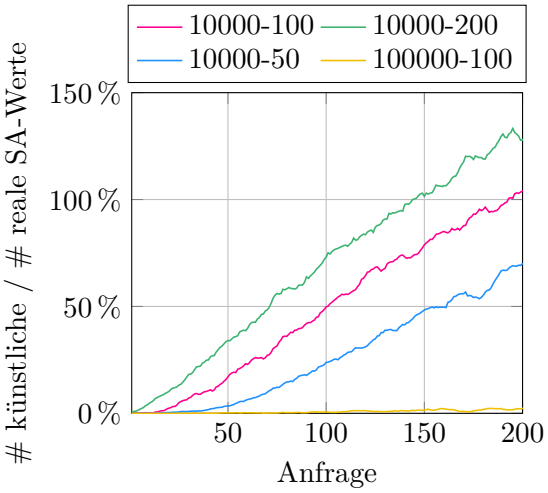
In den Abbildungen 8.5b und 8.5c ist der Anteil hinzugefügter künstlicher SA-Werte für 200 Anfragen dargestellt. Enthält die Datenbank 10000 Tupel, werden bei 200 vorhandenen SA-Werten mehr künstliche Werte ins Ergebnis eingefügt als bei 100 oder 50 SA-Werten. Der Grund dafür ist die Tatsache, dass bei insgesamt weniger verschiedenen SA-Werten die Wahrscheinlichkeit, dass ein bestimmter Wert im Ergebnis vorhanden ist, größer ist. Somit ist auch die Chance höher, dass Matchingkanten erweitert werden können, und es müssen weniger Zuordnungen von SA-Werten zu Tupeln entfernt werden. Während im Fall $k = 10$ die Kurve für 50 SA-Werte am niedrigsten verläuft, schneidet sie im Fall $k = 25$ die beiden anderen Kurven und liegt nach 200 Anfragen sogar am höchsten. Hier kann ein anderer Zusammenhang beobachtet werden. Bei 50 SA-Werten sind die Signaturen und Anonymitätsgrade der Tupel potentiell kleiner als bei 100 oder 200 SA-Werten. Demzufolge können auch weniger Matchingkanten beziehungsweise Zuordnungen von SA-Werten zu Tupeln gelöscht werden, um das Schutzkriterium noch zu erfüllen. Da der Anteil neuer Tupel mit zunehmender Anzahl von Anfragen sinkt, existieren für den Algorithmus *KlonMatching* weniger Möglichkeiten, Matchingkanten entsprechend zu erweitern. Die Folge ist, dass mehr künstliche SA-Werte hinzugenommen werden müssen, um das Löschen von Kanten zu verhindern. Dieser Effekt tritt bei einem hohen Wert für k stärker in Erscheinung als bei einem niedrigen Wert.

Weiterhin liegt der Anteil neuer Tupel in einem Ergebnis bei einer Datenbankgröße von 10000 Tupeln durchschnittlich bei 63 %, während er bei 100000 Tupeln 95 % beträgt. Deshalb müssen bei der kleineren Datenbank deutlich mehr künstliche SA-Werte erzeugt werden (vgl. Kapitel 8.2.3). Durchschnittlich über alle 200 Anfragen werden im Fall $k = 10$ bei 10000 Tupeln und 200 SA-Werten (10000-200) 69 %, bei 10000-100 49 %, bei 10000-50 27 % und bei 100000-100 nur 0,8 % zusätzliche SA-Werte benötigt. Für $k = 25$ liegen diese Werte aus offensichtlichen Gründen jeweils höher. Insgesamt sehen alle dargestellten Verläufe nach einer anfänglichen leichten Linkskrümmung in etwa linear aus. Tests mit mehr als 200 Anfragen werden allerdings zeigen, dass eine leichte Rechtskrümmung vorliegt.

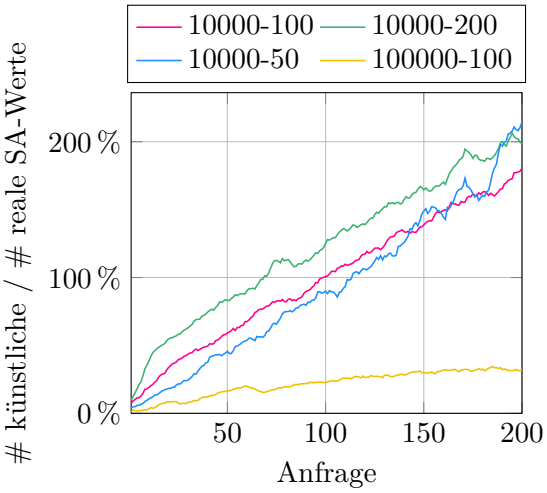
Abbildungen 8.5d und 8.5e zeigen die Laufzeit, die für die Auswertung jeder Anfrage benötigt wurde. Interessanterweise liegen alle drei Kurven der kleineren Datenbankgröße fast übereinander. Nur im Fall $k = 25$ ist am Ende zu erkennen, dass die Kurve für 50 SA-Werte weiter steigt, während die anderen beiden Kurven etwas abflachen. Sind 100000 Tupel gegeben, ist die Laufzeit wesentlich geringer. Eine Ursache dafür ist die Tatsache, dass in diesem Fall auch weniger künstliche SA-Werte konstruiert werden müssen. Weiterhin ist der Verlauf deutlich links gekrümmt.

Datenbankgröße	10000; 100000 Tupel
Anzahl der SA-Werte	50; 100; 200
Verteilung	Gleichverteilung
Ergebnisgröße	50 Tupel
Anzahl der Anfragen	200
Schutzkriterium	$k = 10$; 25

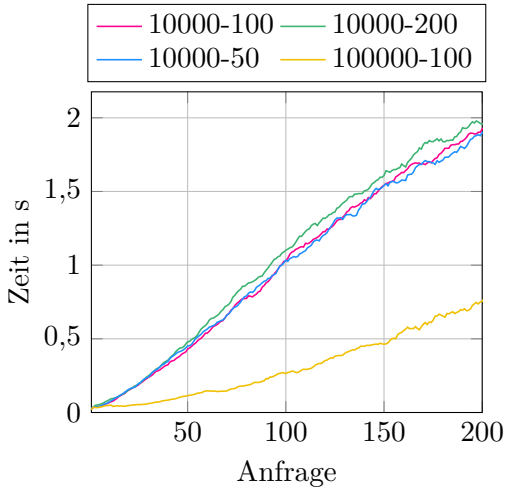
(a) Testsetup 4



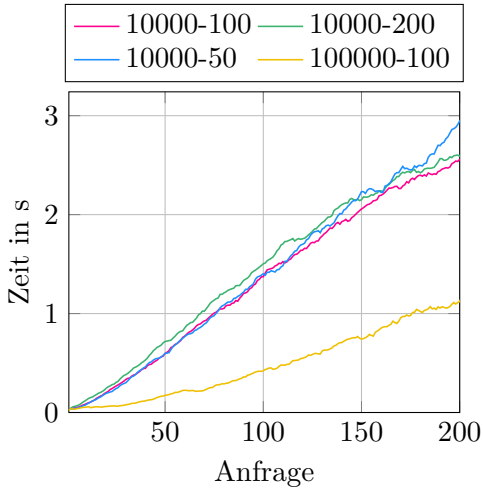
(b) Anteil künstlicher SA-Werte, $k = 10$



(c) Anteil künstlicher SA-Werte, $k = 25$



(d) Laufzeit, $k = 10$



(e) Laufzeit, $k = 25$

Abbildung 8.5: Testsetup, Anteil künstlicher SA-Werte und Laufzeit. Bei der Bezeichnung „x-y“ steht x für die Größe der Datenbank und y für die Anzahl verschiedener SA-Werte.

Für die hier nicht dargestellten Algorithmen *MaxKanten*, *MaxAno* und *SimpelMatching* folgen ähnliche Ergebnisse, wie bereits bei den Tests mit kleineren Datenmengen zu beobachten waren. Im Vergleich zu *KlonMatching* erzeugt *MaxKanten* deutlich mehr SA-Werte. Bei der Eingabe 10000-100 sind es beispielsweise mit durchschnittlich 73 % 24 Prozentpunkte mehr als mit *KlonMatching*. Die Kurven von *MaxAno* und *SimpelMatching* sind hingegen nahezu identisch mit den Kurven von *KlonMatching*. Auch bei der Laufzeit sind die Unterschiede zwischen *MaxAno*, *SimpelMatching* und *KlonMatching* nur marginal, während *MaxKanten* in etwa 40 % mehr Zeit benötigt. In den weiteren Tests wird daher nur noch *KlonMatching* ausgewertet. Die Algorithmen *SA-Ein* und *M-Inv* werden separat in Abschnitt 8.4 evaluiert.

Nach 200 Anfragen mit dazugehörigen Ergebnissen, die jeweils 50 Tupel enthalten, sind insgesamt (nicht notwendigerweise unterschiedliche) 10000 Tupel zurückgegeben worden. Diese Anzahl entspricht der Größe der einen vorgegebenen Datenbank. Werden mehr Anfragen gestellt, werden Tupel in jeden Fall mehrfach angefragt und sind in unterschiedlichen Ergebnissen enthalten. Offensichtlich wird in diesem Fall auch der Anteil hinzugefügter künstlicher SA-Werte ansteigen, was den Nutzen der Ergebnisse für weitere Analysen beeinträchtigt. Es ist allerdings fragwürdig, ob es ein reales Szenario gibt, in dem Nutzer so viele Anfragen stellen, dass alle Tupel mehrfach in den entsprechenden Ergebnissen vorkommen. Für die Analyse der Algorithmen ist es jedoch wünschenswert, auch diese Fälle zu evaluieren. Daher wird die Anzahl der Anfragen erneut erhöht. Da damit aber auch die Gesamtlaufzeit eines Testlaufs schnell steigt, wird nur der Fall untersucht, bei dem die Datenbank aus 5000 und 10000 Tupeln besteht.

In den Abbildungen 8.6b und 8.6c sind der Anteil künstlicher SA-Werte bei insgesamt 1000 Anfragen dargestellt. Es ist zu erkennen, dass ab einem gewissen Wert nicht noch mehr künstliche SA-Werte ins Ergebnis eingefügt werden und deren Anteil relativ konstant bleibt. Im Fall $k = 10$ liegt diese Sättigung aus offensichtlichen Gründen deutlich unter der für $k = 25$. Allerdings fällt auf, dass die Höhe dieser Grenze unabhängig von der Datenbankgröße ist. Einen Einfluss haben hingegen der Wert von k, die Anzahl verschiedener SA-Werte in der Datenbank und die hier konstant gehaltene Ergebnisgröße.[2] Auch liegt die Sättigung im Fall $k = 10$ für alle vier Kurven bei einem ähnlichen Wert, während es bei $k = 25$ große Unterschiede gibt. Die Ursache ist der bereits angesprochene Effekt, wenn das Verhältnis von k zur Ergebnisgröße relativ hoch ist.

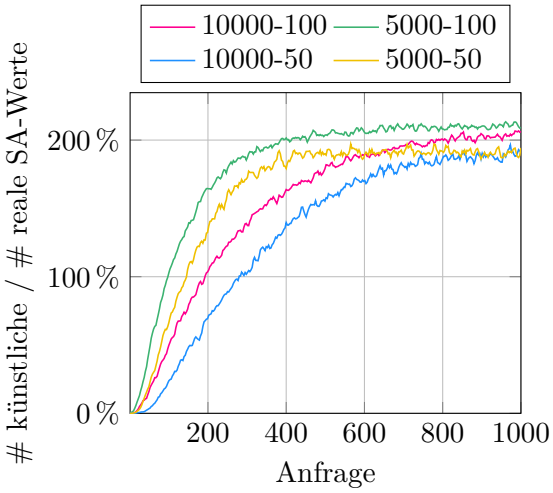
200 % zusätzliche künstliche SA-Werte bei $k = 10$ vergrößern das Ergebnis auf 150 SA-Werte, von denen nur 50 der Datenbank entnommen sind. Das erscheint zwar wenig wünschenswert, ist allerdings noch wesentlich besser als eine mögliche Alternative. Diese bestünde nämlich im einmaligen Veröffentlichen der gesamten Datentabelle (vgl. Kapitel 2.3). Dabei kann es passieren, dass die 50 angefragten Tupel in 50 verschiedenen QI-Gruppen der Größe mindestens $k = 10$ vorkommen. Da sie innerhalb dieser Gruppen nicht identifizierbar sind, würde das Ergebnis der Anfragen aus mindestens $50 \cdot 10 = 500$ Tupeln bestehen, was einem Anteil zusätzlicher SA-Werte von 1000 % entspräche.

Die Laufzeiten des Algorithmus sind in den Abbildungen 8.6d und 8.6e angegeben. Interessanterweise sinken die Kurven ab einer bestimmten Anfrage. Der Grund hierfür liegt darin, dass zu diesem Zeitpunkt bereits die meisten Tupel angefragt wurden und somit nur noch wenige neue Tupel für die Erweiterung zur Verfügung stehen. Ebenfalls sind schon so viele Kanten aus der Matchingtabelle gestrichen worden, dass das Überprüfen, ob eine

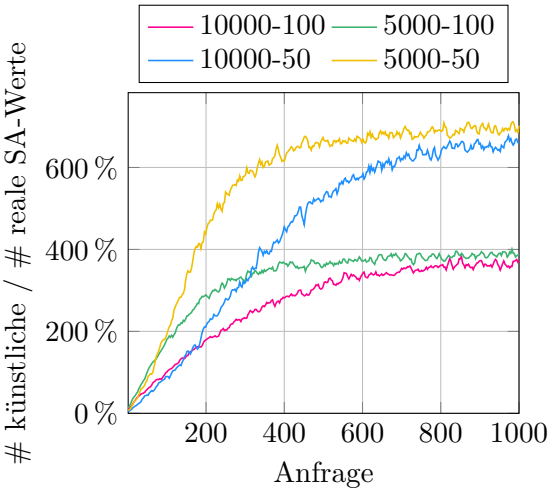
[2] Bei einem größeren Ergebnis werden weniger und bei einem kleineren mehr künstliche SA-Werte benötigt.

Datenbankgröße	5000; 10000 Tupel
Anzahl der SA-Werte	50; 100
Verteilung	Gleichverteilung
Ergebnisgröße	50 Tupel
Anzahl der Anfragen	1000
Schutzkriterium	$k = 10$; 25

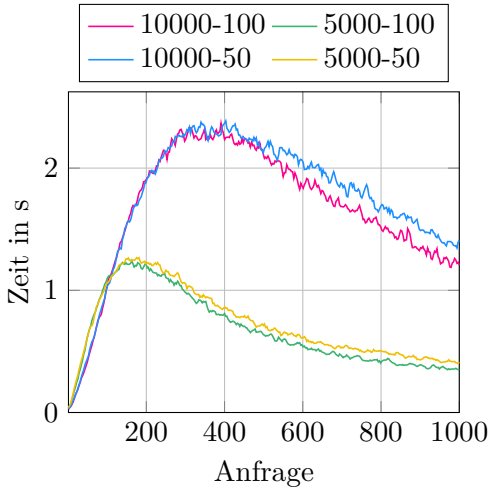
(a) Testsetup 5



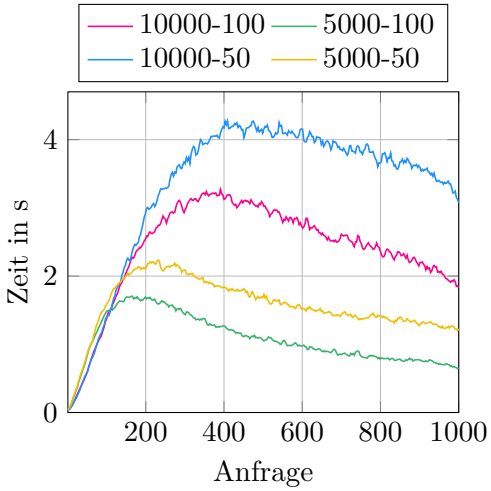
(b) Anteil künstlicher SA-Werte, $k = 10$



(c) Anteil künstlicher SA-Werte, $k = 25$



(d) Laufzeit, $k = 10$



(e) Laufzeit, $k = 25$

Abbildung 8.6: Testsetup, Anteil künstlicher SA-Werte und Laufzeit. Bei der Bezeichnung „x-y“ steht x für die Größe der Datenbank und y für die Anzahl verschiedener SA-Werte.

weitere Kante gelöscht werden kann, ohne das Schutzkriterium zu verletzen, in kürzerer Zeit möglich. Die Laufzeitkurven haben insgesamt drei verschiedene Krümmungen. Nach einer anfänglichen kurzen Linkskrümmung sind die Kurven zunächst leicht und später stark rechts gekrümmt, bis sie fallen und sich auf ein bestimmtes Niveau einpegeln. An diesem Punkt entsteht wieder eine Linkskrümmung. Auch in diesem Diagramm fallen die Unterschiede im Fall $k = 25$ wesentlich höher aus als bei $k = 10$. Das liegt erneut an der größeren Anzahl hinzugefügter künstlicher SA-Werte.

8.3.2 Schrägverteilte sensible Werte

Um auch reale Daten zu simulieren, werden verschiedene Verteilungen der SA-Werte untersucht. Als Datenbank- und Ergebnisgröße werden dieselben Werte wie beim vorigen Test gewählt (siehe Abbildung 8.8a). Als Verteilungen werden die Gleich-, Normal- und Exponentialverteilung verwendet. Die dazugehörigen Dichtefunktionen sind

$$\text{Gleichverteilung } (\mathcal{U}(a,b)): \quad f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & \text{sonst} \end{cases},$$

$$\text{Normalverteilung } (\mathcal{N}(\mu,\sigma^2)): \quad f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{1}{2}(\frac{x-\mu}{\sigma})^2},$$

$$\text{Exponentialverteilung } (Exp(\lambda)): \quad f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0 \\ 0, & x < 0 \end{cases}$$

und in Abbildung 8.7 graphisch dargestellt. Bei der Normalverteilung werden die Parameter so gewählt, dass die seltensten SA-Werte mindestens einmal und die häufigsten maximal oft vorkommen. Für die Exponentialverteilung gilt, dass je größer λ ist, desto größer sind die Unterschiede der Häufigkeiten einzelner SA-Werte. Es werden zwei verschiedene Werte für λ evaluiert. Während sich die Häufigkeiten bei $\lambda = 0{,}06$ sehr stark unterscheiden, wird mit $\lambda = 0{,}03$ nur eine leichte Schiefverteilung erzeugt.

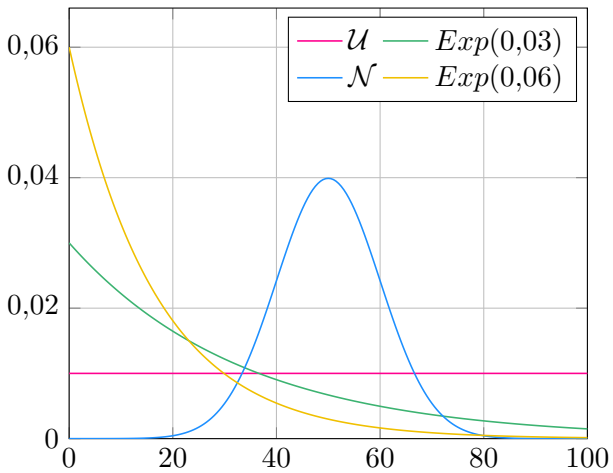
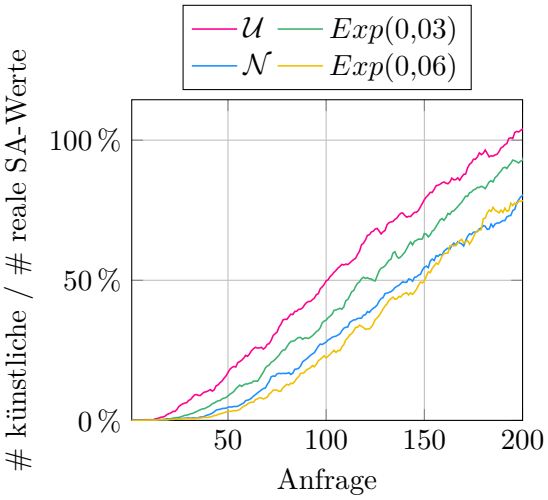


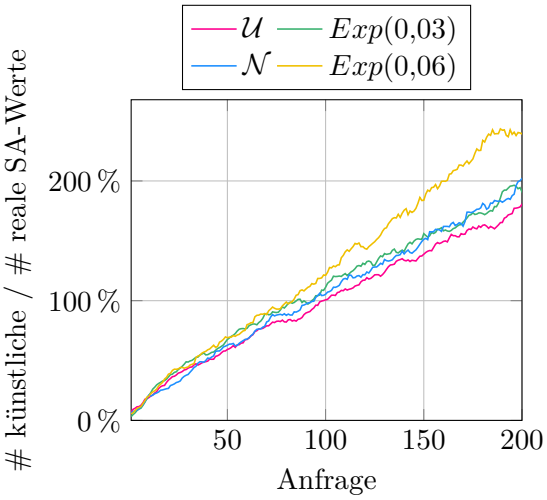
Abbildung 8.7: Dichtefunktionen der stetigen Gleich- ($\mathcal{U}$), Exponential- (Exp) und Normalverteilung ($\mathcal{N}$)

Datenbankgröße	10000 Tupel
Anzahl der SA-Werte	100
Verteilung	$\mathcal{U}$; $\mathcal{N}$; Exp
Ergebnisgröße	50 Tupel
Anzahl der Anfragen	200
Schutzkriterium	$k = 10$; 25

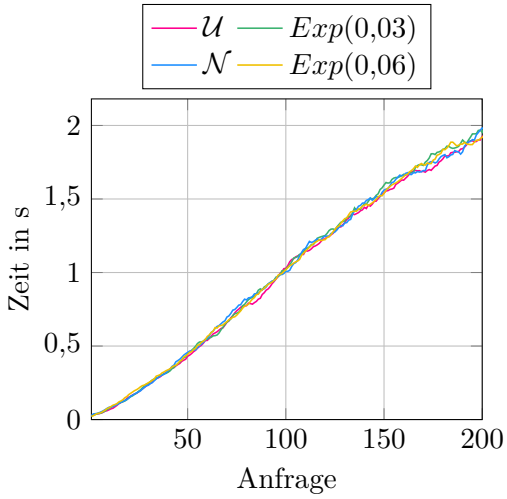
(a) Testsetup 6



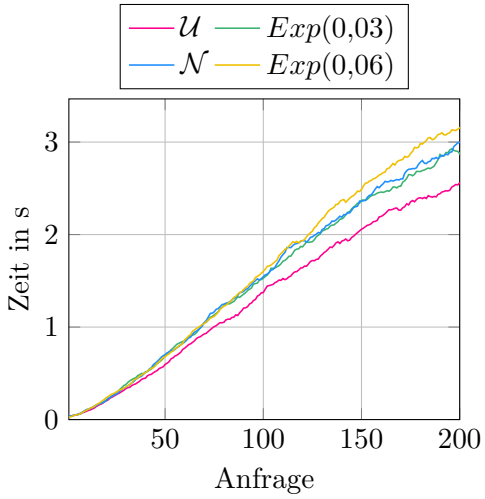
(b) Anteil künstlicher SA-Werte, $k = 10$



(c) Anteil künstlicher SA-Werte, $k = 25$



(d) Laufzeit, $k = 10$



(e) Laufzeit, $k = 25$

Abbildung 8.8: Testsetup, Anteil künstlicher SA-Werte und Laufzeit bei gleich- ($\mathcal{U}$), exponential- (Exp) und normalverteilten ($\mathcal{N}$) SA-Werten.

Der Anteil künstlicher SA-Werte in den Ergebnissen ist in den Abbildungen 8.8b und 8.8c sowie die dazugehörigen Laufzeiten in den Abbildungen 8.8d und 8.8e dargestellt. Die ersten Anfragen kann *KlonMatching* im Fall $k = 10$ bei allen Verteilungen ohne oder nur mit sehr wenigen künstlichen SA-Werten beantworten. Danach steigen alle Kurven annähernd linear an, wobei bei der Gleichverteilung die meisten SA-Werte erzeugt werden. Bei der Exponentialverteilung $Exp(0{,}06)$ werden bis zur ca. 170. Anfrage die wenigsten Werte produziert, danach liegt ihre Kurve etwas über der der Normalverteilung. Im Fall $k = 25$ steigen anfangs alle Kurven recht gleichmäßig, ehe die Kurve der Gleichverteilung am niedrigsten und die der Exponentialverteilung $Exp(0{,}06)$ am höchsten liegt. Die gemessenen Laufzeiten sind ebenfalls sehr ähnlich zueinander. Erneut benötigt $Exp(0{,}06)$ ab einer gewissen Anzahl von Anfragen die meiste und die Gleichverteilung vor allem bei $k = 25$ die wenigste Zeit.

Wird die Anzahl der Anfragen erneut auf 1000 erhöht, werden die weiteren Verläufe der Kurven sichtbar. In Abbildung 8.9 ist zu erkennen, dass ab einer bestimmten Anzahl von Anfragen die wenigsten künstlichen SA-Werte bei der Gleichverteilung benötigt werden. Der Unterschied zu anderen Verteilungen ist bei $k = 10$ wesentlich geringer als bei $k = 25$. Interessant ist auch, dass sich die Reihenfolge der Kurven insbesondere im Fall $k = 10$ verändert. Werden anfangs bei $Exp(0{,}06)$ die wenigsten künstlichen SA-Werte konstruiert, sind es dort später die meisten.

Die Kurven der Laufzeiten sind ähnlich zu den bereits bekannten aus Abbildung 8.6. Gleichverteilte Daten können am schnellsten verarbeitet werden, bei einer besonders schiefen Verteilung wie $Exp(0{,}06)$ dauert es hingegen bis zu 2 s länger.

Das Fazit dieses Abschnitts lautet, dass bei sehr vielen Anfragen weniger künstliche SA-Werte benötigt werden, wenn die SA-Werte gleichmäßig verteilt sind. Je schiefer die Verteilung ist, desto mehr künstliche Werte werden erzeugt. Sind allerdings nur wenige Anfragen gegeben, kann sich dieser Grundsatz auch schnell umkehren.

8.4 Untersuchung des Algorithmus für SA-eindeutige Anfragen

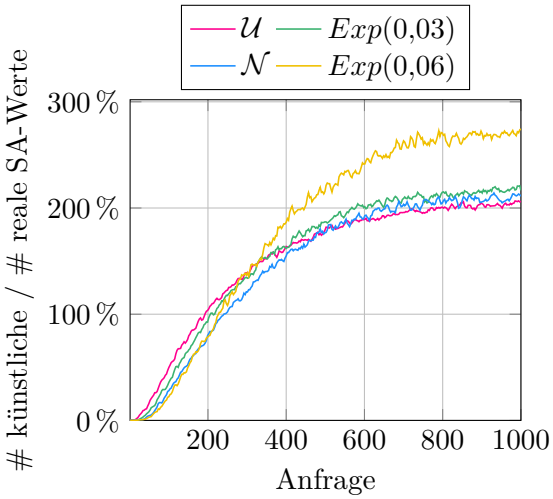
Die Algorithmen *SA-Ein* und *M-Inv* erzeugen in allen bisherigen Tests mehr künstliche SA-Werte als die anderen Approximationsalgorithmen. Das hängt vor allem damit zusammen, dass nur Ergebnisse zurückgegeben werden, in denen kein SA-Wert mehrfach vorkommt. Werden aber durch die Anfragen Tupel mit vielen identischen SA-Werten selektiert, muss das Ergebnis zunächst in kleinere Teilergebnisse partitioniert werden. Folglich ist die Wahrscheinlichkeit, dass bestimmte SA-Werte in den entstehenden Teilen fehlen und durch künstliche ersetzt werden müssen, wesentlich größer als zum Beispiel bei *KlonMatching*. Ein Vorteil von *SA-Ein* besteht aber darin, dass die zurückgegebenen Teilergebnisse potentiell weniger Tupel enthalten als das ursprüngliche Gesamtergebnis. Werden kleinere Tupelmengen zu einer QI-Gruppe zusammengefasst, müssen quasi-identifizierende Attributwerte oftmals weniger stark anonymisiert werden (vgl. Kapitel 2.2.3). Die Informationen, die aus diesen Daten gewonnen werden, können für den Nutzer bei weiteren Analysen wertvoller sein.[3]

Aus den genannten Gründen werden *SA-Ein* und *M-Inv* separat evaluiert. Dabei werden insbesondere drei einzelne Schritte von *SA-Ein* untersucht, deren Auswirkungen durch

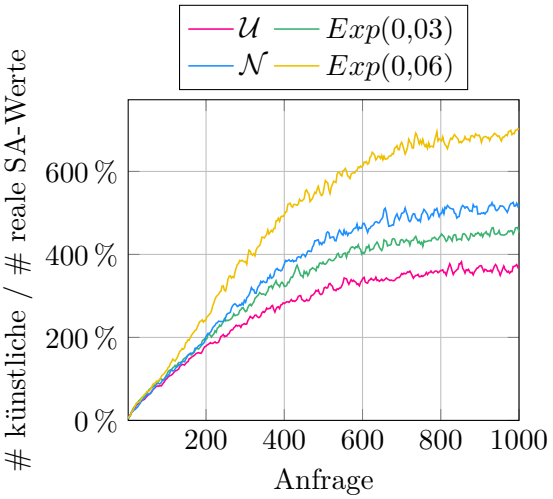
[3] Auf eine umfangreiche Untersuchung des Nutzens anonymisierter Daten wird in der vorliegenden Arbeit verzichtet. Einen Überblick über dieses Thema geben beispielsweise Fung et al. [54].

Datenbankgröße	10000 Tupel
Anzahl der SA-Werte	100
Verteilung	$\mathcal{U}$; $\mathcal{N}$; Exp
Ergebnisgröße	50 Tupel
Anzahl der Anfragen	1000
Schutzkriterium	$k = 10$; 25

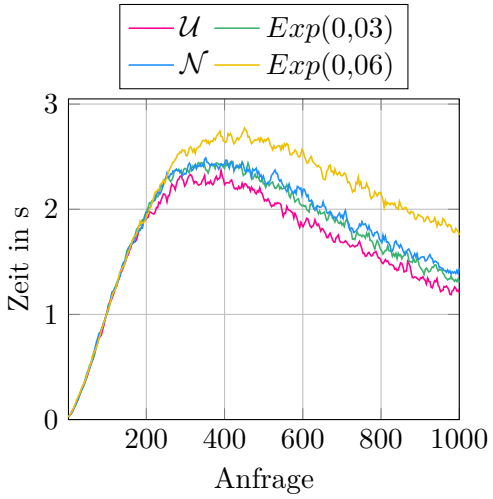
(a) Testsetup 7



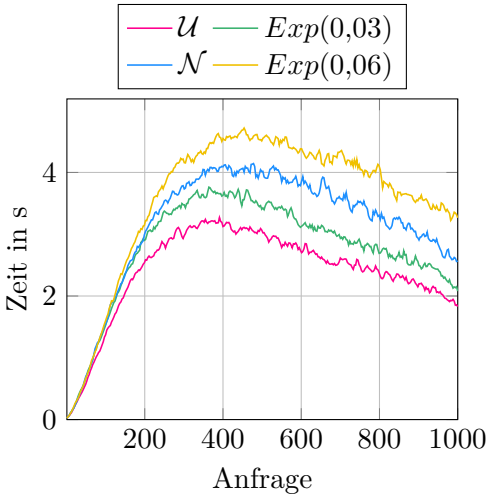
(b) Anteil künstlicher SA-Werte, $k = 10$



(c) Anteil künstlicher SA-Werte, $k = 25$



(d) Laufzeit, $k = 10$



(e) Laufzeit, $k = 25$

Abbildung 8.9: Testsetup, Anteil künstlicher SA-Werte und Laufzeit bei gleich- ($\mathcal{U}$), exponential- (Exp) und normalverteilten ($\mathcal{N}$) SA-Werten.

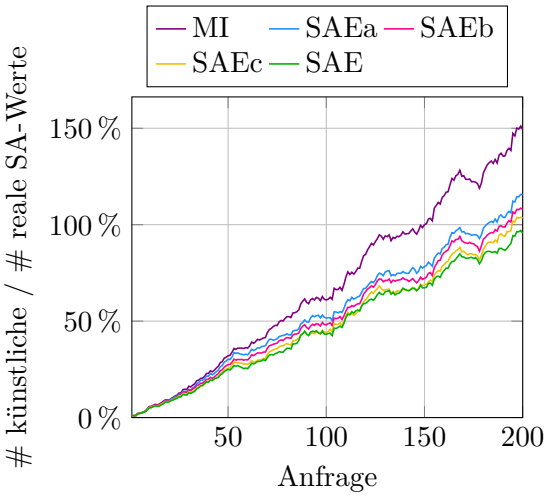
Tests aufgezeigt werden sollen. Die erste Stelle betrifft Algorithmus 7.5 auf Seite 214, welcher der erste Teil von *SA-Ein* ist. Statt das Tupel t im dort beschriebenen Fall 3 bei der ersten Betrachtung zurückzustellen und im zweiten Durchlauf ins Cluster C_i einzufügen, bildet es analog zu Fall 2 gleich ein neues Cluster. Das ist die zweite sinnvolle Möglichkeit, wenn $sim(t, C_i) = 0$ gilt (s. Zeile 6). Der entstehende Algorithmus wird mit *SA-Ein-a* (bzw. SAEa) bezeichnet. Eine weitere Alternative ist, auf Algorithmus 7.6 auf Seite 218 zu verzichten. Anstatt neue Tupel zu clustern, können diese wie bei *M-Inv* nur dann in ein Teilergebnis eingefügt werden, wenn dort eine Verletzung des Schutzkriteriums vorliegt. Dieser Ansatz wird *SA-Ein-b* (bzw. SAEb) genannt und ist analog zum Einfügen künstlicher SA-Werte, wie er in Kapitel 7.6 beschrieben wurde. Die letzte hier vorgestellte Variante besteht darin, nur auf Phase 2 von Algorithmus 7.6 zu verzichten. Dadurch werden neue Tupel nur in die Cluster eingefügt, in denen sie einen positiven Nutzen mit alten Tupeln erzeugen. *SA-Ein-c* (bzw. SAEc) steht für diese dritte Modifikation von *SA-Ein*.

Datenbankgröße	50000 Tupel
Anzahl der SA-Werte	100
Ergebnisgröße	50 Tupel
Anzahl der Anfragen	200
Schutzkriterium	$k = 10$

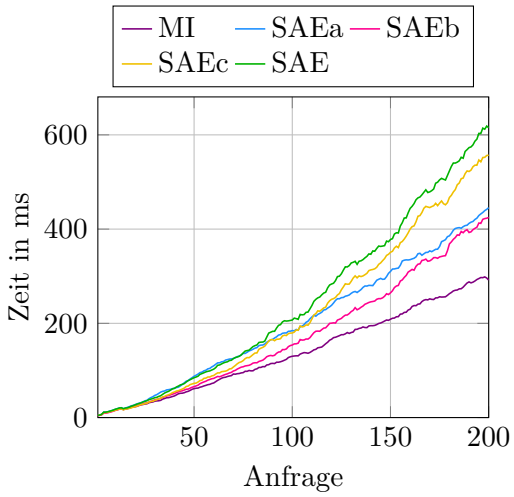
(a) Testsetup 8

Name	künstl. SA-Werte	Laufzeit
MI	68,9 %	140 ms
SAEa	55,4 %	202 ms
SAEb	52,3 %	177 ms
SAEc	49,0 %	223 ms
SAE	47,3 %	248 ms

(b) Durchschnittswerte über 200 Anfragen



(c) Anteil künstlicher SA-Werte



(d) Laufzeit

Abbildung 8.10: Testsetup, Anteil künstlicher SA-Werte und Laufzeit, $k = 10$

Abbildung 8.10 fasst in (a) die Eingabegrößen und in (b) bis (d) die Evaluation zusammen. *SA-Ein* erzeugt durchschnittlich über alle 200 Anfragen 47,3 % künstliche SA-Werte und ist damit in dieser Kategorie der beste der hier getesteten Algorithmen. Bei *M-Inv* sind es hingegen 68,9 %, was dem größten gemessenen Wert entspricht. In (b) sind die Durchschnittswerte der drei Modifikationen angegeben, die jeweils zwischen den genannten Werten liegen. Auch für die Verläufe der Kurven in (c) gilt, dass der vorgestellte Algorithmus *SA-Ein* bei jeder Anfrage weniger künstliche SA-Werte hinzufügt als die

Modifikationen. Dieses Ergebnis zeigt die Effektivität der drei angesprochenen Schritte innerhalb von *SA-Ein*.

Für die Laufzeiten in (d) gilt annähernd die umgekehrte Reihenfolge, das heißt, *M-Inv* ist der schnellste und *SA-Ein* der langsamste Algorithmus. Nur die Reihenfolge von *SA-Ein-a* und *SA-Ein-b* ist identisch zu der aus dem Vergleich der hinzugefügten künstlichen SA-Werte. Insgesamt liegen die Laufzeiten aber deutlich unter der, die *KlonMatching* in diesem Testfall benötigt (durchschnittlich 476 ms).

Als letzten Test mit synthetischen Daten werden die Eingaben so gewählt, dass in jedem Ergebnis viele identische SA-Werte vorkommen. Gleichzeitig sollen möglichst wenige neue Tupel vorhanden sein. Dahinter steht das Ziel, dass der Algorithmus *SA-Ein* in diesem Fall eine bessere Güte haben könnte als *KlonMatching*, da letzterer alte Tupel nicht mit anderen alten Tupeln erweitert (vgl. Kapitel 5.4.1).

In Teil (a) von Abbildung 8.11 sind die entsprechenden Eingabegrößen angegeben, wobei k auf 4 beziehungsweise 6 gesetzt wird. Die hinzugefügten künstlichen SA-Werte sind in (b) und (c) angegeben. Interessant ist, dass nach jeweils 200 Anfragen *SA-Ein* tatsächlich weniger Werte konstruiert als *KlonMatching*. In beiden Fällen werden zwar anfangs durch *KlonMatching* weniger SA-Werte erzeugt, nach einer bestimmten Anzahl von Anfragen ändert sich jedoch diese Reihenfolge. Bei $k = 4$ liegt die Kurve von *SA-Ein* ab der 60. Anfrage und bei $k = 6$ bereits ab der 40. Anfrage unter der von *KlonMatching*. Weiterhin fügt *SA-Ein* mit größerem k nur weniger SA-Werte mehr hinzu als das bei *KlonMatching* der Fall ist. Während es ab der 150. Anfrage für *SA-Ein* bei $k = 4$ durchschnittlich 140 % und bei $k = 6$ knapp 170 % sind, steigen diese Werte für *KlonMatching* deutlich von 180 % ($k = 4$) auf 310 % ($k = 6$).

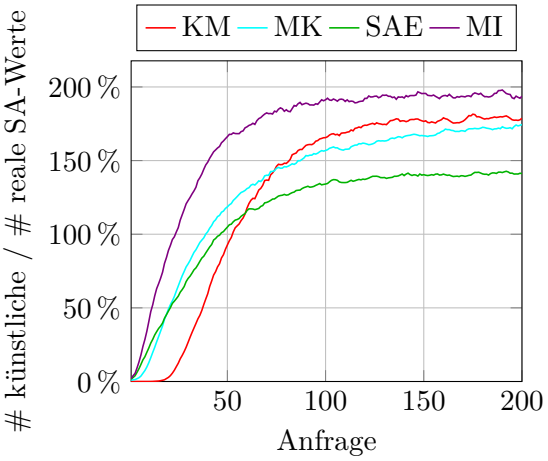
Für *M-Inv* folgen ähnliche Erkenntnisse wie für *SA-Ein* mit der Ausnahme, dass in beiden Fällen mehr künstliche SA-Werte erzeugt werden. Die Kurve von *MaxKanten* ist wiederum ähnlich zu der von *KlonMatching*, wobei sie anfangs deutlich darüber und später auf einem ähnlichen Niveau liegt. Nicht dargestellt sind erneut *MaxAno* und *SimpelMatching*, deren Verläufe fast identisch zu dem von *KlonMatching* sind. Insgesamt lässt sich aber feststellen, dass in allen Fällen bereits nach wenigen Anfragen die Ergebnisse mehr künstliche als reale SA-Werte enthalten und damit für weitere statische Analysen nur beschränkt von Nutzen sind. Andererseits sind nach 25 Anfragen mit jeweils 40 Elementen bereits 1000 Tupel zurückgegeben worden, was der Größe der gesamten Datenbank entspricht. Werden mehr Anfragen gestellt, ist es auch offensichtlich, dass durch die zufällige Konstruktion des Ergebnisses mehr künstliche SA-Werte hinzugefügt werden müssen.

Die Entwicklung der Laufzeit der Algorithmen ist in (d) und (e) dargestellt. Es ist deutlich zu erkennen, dass die Laufzeit nach einer gewissen Anzahl von Anfragen annähernd konstant bleibt (*SA-Ein* und *M-Inv*) beziehungsweise auf ein einheitliches Niveau sinkt (*KlonMatching* und *MaxKanten*). Die dazugehörigen Zeitpunkte entsprechen jeweils in etwa denen, bei welchen auch die Kurven der hinzugefügten künstlichen SA-Werte rechtsgekrümmt sind und demnach nicht mehr so stark steigen.

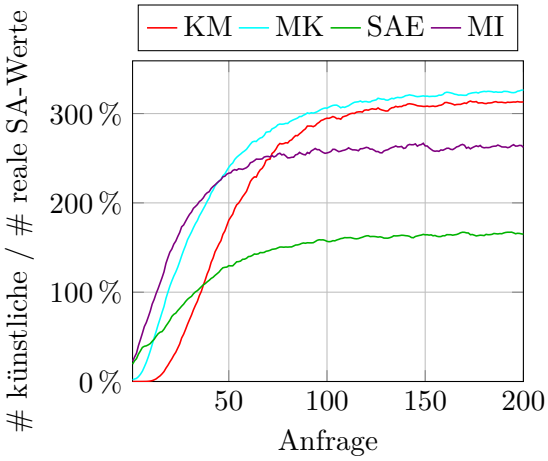
In allen anderen in diesem Kapitel vorgestellten Szenarien erstellt *KlonMatching* jeweils weniger künstliche SA-Werte und hat dadurch eine bessere Güte als *SA-Ein*. Das Beispiel aus Abbildung 8.11 zeigt, dass es auch Fälle gibt, bei denen *SA-Ein* für die Anfragebearbeitung insgesamt geeigneter ist. Wird das Verhältnis von Ergebnisgröße und Anzahl verschiedener SA-Werte erhöht, fällt dieser Unterschied sogar noch deutlicher aus.

Datenbankgröße	1000 Tupel
Anzahl der SA-Werte	10
Ergebnisgröße	40 Tupel
Anzahl der Anfragen	200
Schutzkriterium	$k = 4$; 6

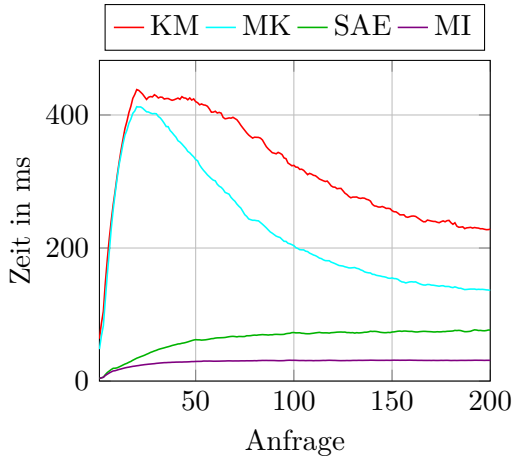
(a) Testsetup 9



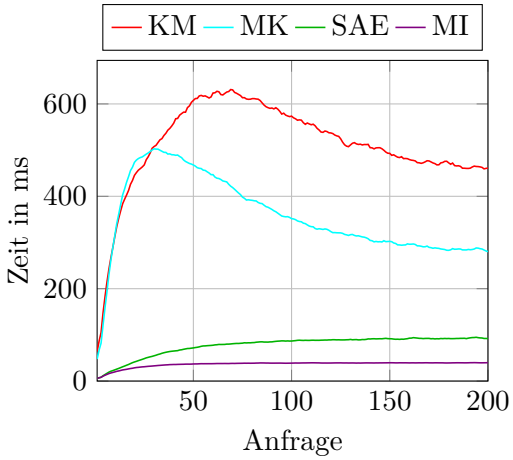
(b) Anteil künstlicher SA-Werte, $k = 4$



(c) Anteil künstlicher SA-Werte, $k = 6$



(d) Laufzeit, $k = 4$



(e) Laufzeit, $k = 6$

Abbildung 8.11: Testsetup, Anteil künstlicher SA-Werte und Laufzeit

8.5 Verwendung realer Daten

Als Abschluss der Evaluation werden Tests mit realen Daten durchgeführt. Dazu wird die Adult-Tabelle des UCI Machine Learning Repository [52] verwendet, die bereits in mehreren anderen Arbeiten zum Schutz der Privatsphäre genutzt wurde [89, 56, 101, 159, 93, 53]. Sie enthält demographische Daten einer Volkszählung in den USA und besteht aus 32561 Einträgen mit Werten in jeweils 15 Attributen. Mögliche sensible Attribute sind Angaben zum Gehalt (`salary`) oder zur Beschäftigung (`occupation`). Da allerdings `salary` nur die beiden Ausprägungen „≤ 50“ und „> 50“ besitzt, während in `occupation` insgesamt 14 verschiedene Werte vorkommen, wird als sensibles Attribut das Letztgenannte gewählt. Werden alle Tupel gelöscht, die keine Angaben über die Beschäftigung enthalten, besteht der Datensatz aus 30718 Tupeln.

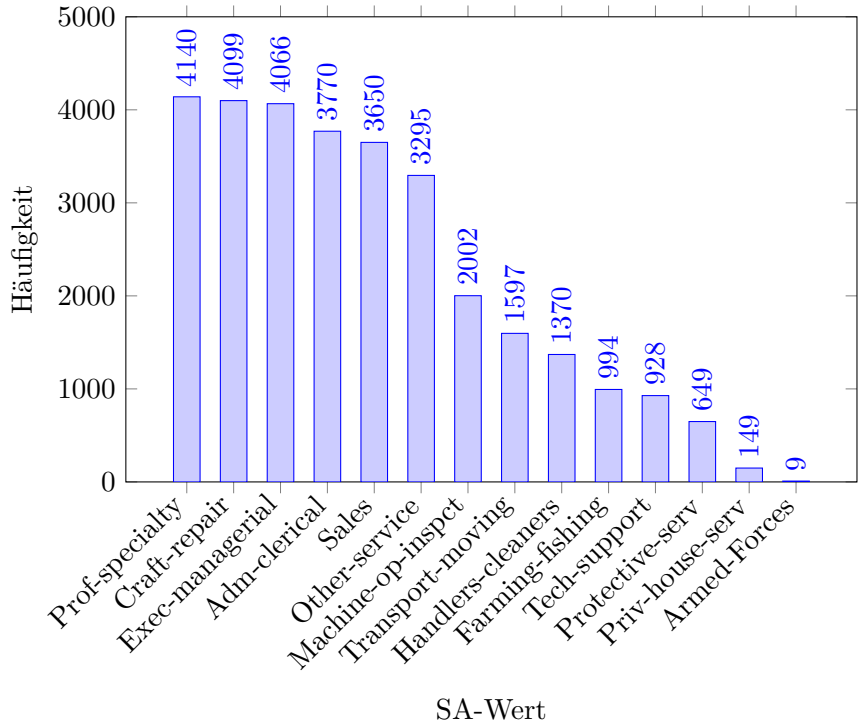


Abbildung 8.12: Verteilung der SA-Werte

Abbildung 8.12 zeigt die Verteilung der SA-Werte. Während Werte wie „Prof-specialty“, „Craft-repair“ oder „Exec-managerial“ recht häufig sind, kommt der seltenste Wert „Armed-Forces“ nur neunmal in den Daten vor. In Abbildung 8.13a sind die Eingabegrößen zusammengefasst. Eine Besonderheit bildet die variable Ergebnisgröße. Um möglichst reale Anfragen zu generieren, werden diesmal nicht beliebige Tupel zu einem Ergebnis zusammengefasst, sondern Anfragen mit Selektionskriterien konstruiert. Dafür wird zunächst ein Tupel zufällig aus der Tabelle gezogen, aus dessen Attributwerten die Selektionsprädikate erstellt werden. Sei beispielsweise ein Tupel von einer Person zufällig ausgewählt worden, die 50 Jahre alt (`age = 50`) und geschieden (`marital-status = 'divorced'`) ist. Dann kann eine mögliche dazugehörige Anfrage

```
Q: SELECT * FROM Adult WHERE age BETWEEN 49 AND 51
                        AND marital-status IN ('divorced', 'separated')
```

lauten. Durch Q werden alle Personen selektiert, deren Alter zwischen 49 und 51 Jahren liegt und die geschieden (`divorced`) sind oder von ihrem Partner getrennt leben (`separated`). Sowohl die Attribute, die in der WHERE-Klausel vorkommen, als auch deren angefragte Werte werden dabei zufällig ausgewählt. Die Voraussetzung besteht nur darin, dass die Werte des ausgewählten Tupels enthalten sein müssen. Eine auf diese Weise konstruierte Anfrage wird akzeptiert, wenn die Anzahl der Tupel im Ergebnis zu der Anfrage innerhalb der hier angegebenen Schranke zwischen 10 und 30 liegt. Besteht das Ergebnis aus weniger Tupeln, werden die Bereiche der Attributwerte in der Anfrage erweitert. Beispielsweise kann auch nach einem Alter zwischen 45 und 55 Jahren gefragt werden. Analog werden die Bereiche verkleinert, wenn zu viele Tupel zurückgegeben werden. Auf diese Weise werden zufällige Anfragen erzeugt, deren Ergebnisse immer zwischen 10 und 30 Tupeln enthalten.

Die Abbildungen 8.13b und 8.13c zeigen das Verhältnis künstlicher zu realen SA-Werten. Mit zunehmender Anfragenanzahl steigen alle dargestellten Kurven leicht an, wobei die Werte für $k = 7$ jeweils deutlich höher sind als die für $k = 5$. Der durchschnittliche Anteil künstlicher SA-Werte liegt für *KlonMatching* bei 3 % ($k = 5$) beziehungsweise 9 % ($k = 7$) und für *SA-Ein* bei 59 % ($k = 5$) beziehungsweise 117 % ($k = 7$). Es fällt auf, dass *SA-Ein* bereits ab der ersten Anfrage relativ viele künstliche SA-Werte produziert. Das liegt vor allem an der gegebenen Verteilung, bei der einige Werte besonders häufig vorkommen. Das hat zur Folge, dass in den Ergebnissen ein SA-Wert meistens ca. viermal vorhanden ist, sodass auch vier Teilanfragen konstruiert werden müssen. Um k-assign Anonymität mit $k = 5$ zu erfüllen, werden somit $4 \cdot 5 = 20$, bei $k = 7$ sogar $4 \cdot 7 = 28$ Tupel benötigt. Die durchschnittliche Ergebnisgröße liegt in diesem Szenario aber nur bei knapp 15 Tupeln, sodass immer entsprechend viele künstliche SA-Werte hinzugefügt werden müssen. Die Kurven für *M-Inv* sind aus diesem Grund auch nahezu identisch mit denen von *SA-Ein* und werden deshalb hier nicht dargestellt. Die Streuung der Werte ist größer als bei den synthetischen Tests, was mit der unterschiedlichen Anzahl von Tupeln in den einzelnen Ergebnissen erklärt werden kann. Größen im Bereich von 10 bis 30 Tupeln sind möglich.

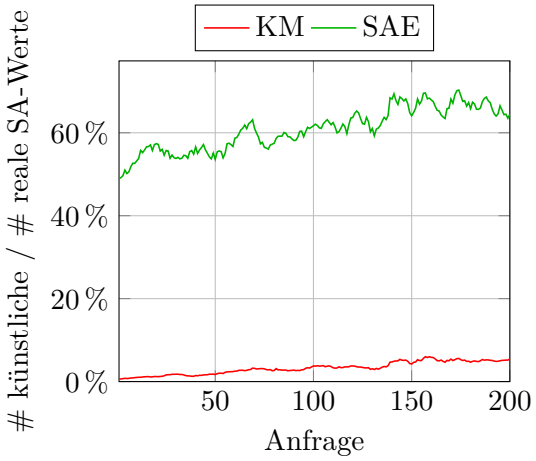
Interessanterweise sind die Kurven der Laufzeiten für $k = 5$ in Abbildung 8.13d sehr ähnlich zu denen für $k = 7$ in Abbildung 8.13e. Insbesondere liegen die Werte für $k = 7$ nur minimal über denen für $k = 5$, obwohl im ersten Fall deutlich mehr künstliche SA-Werte konstruiert werden. Der Gesamtumfang ist mit einigen Millisekunden deutlich unter dem der synthetischen Tests. Die Ursache dafür liegt im geringen Anteil künstlicher SA-Werte, der innerhalb von 200 Anfragen bei *KlonMatching* und $k = 7$ nicht über 16 % steigt. Insgesamt stehen diese Ergebnisse aber im Einklang mit den Erkenntnissen der Tests mit synthetischen Daten und zeigen, dass die Approximationsalgorithmen auch mit realen Daten und annähernd realen Anfragen gut funktionieren.

8.6 Fazit

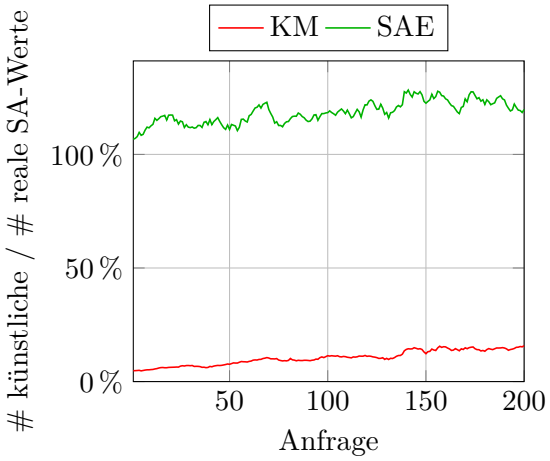
Die Eingaben für die durchgeführten Tests unterscheiden sich in der Datenbank- und Ergebnisgröße, der Anzahl und Verteilung der SA-Werte sowie der Werte für k, wenn k-assign Anonymität gefordert ist. Offensichtlich ist, dass mit zunehmender Größe der Datenbank oder des Ergebnisses immer weniger künstliche SA-Werte dem Ergebnis hinzugefügt wer-

Datenbankgröße	30718 Tupel
Anzahl der SA-Werte	14
Verteilung	siehe Abbildung 8.12
Ergebnisgröße	10–30 Tupel
Anzahl der Anfragen	200
Schutzkriterium	$k = 5; 7$

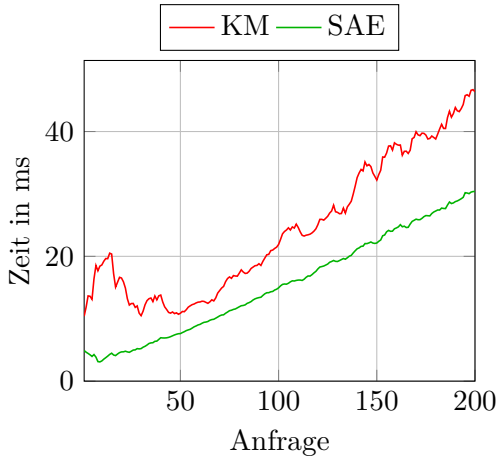
(a) Testsetup 10



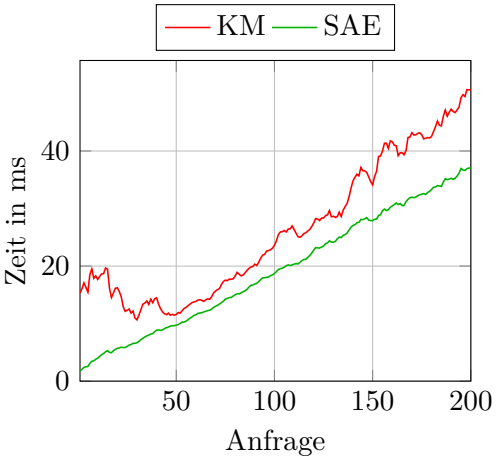
(b) Anteil künstlicher SA-Werte, $k = 5$



(c) Anteil künstlicher SA-Werte, $k = 7$



(d) Laufzeit, $k = 5$



(e) Laufzeit, $k = 7$

Abbildung 8.13: Testsetup, Anteil künstlicher SA-Werte und Laufzeit

den müssen, um das Schutzkriterium zu erfüllen. Demgegenüber wirkt sich ein großer Wert für k negativ auf das Ergebnis aus, das heißt, in diesem Fall müssen mehr künstliche SA-Werte erstellt werden. Eine größere Anzahl hinzugefügter Werte führt jeweils auch zu einer höheren Laufzeit der Algorithmen.

Wesentlich schwieriger ist es, allgemein gültige Aussagen über die Anzahl und Verteilung der SA-Werte zu treffen. Es gibt Fälle, in denen eine große Anzahl verschiedener SA-Werte zu weniger von den Algorithmen erkannten Verletzungen des Schutzkriteriums führt, und Fälle, in denen es genau umgekehrt ist. Oftmals spielt das Verhältnis zwischen der Anzahl der SA-Werte und der Ergebnisgröße eine zentrale Rolle. Sind nur wenige Anfragen gegeben, ist es häufig besser, wenn auch nur wenige verschiedene SA-Werte in den Daten vorkommen. Genau andersherum verhält es sich, wenn besonders viele Anfragen beantwortet werden müssen (vgl. Abbildungen 8.5 und 8.6). Ähnliche Erkenntnisse folgen bei unterschiedlichen Verteilungen der SA-Werte. Die Gleichverteilung führt meist nur bei vielen gegebenen Anfragen zu wenigen künstlichen SA-Werten. Müssen hingegen nur ein paar Anfragen bearbeitet werden, entstehen bei eher schieferen Verteilungen weniger künstliche SA-Werte (vgl. Abbildungen 8.8 und 8.9).

Für die untersuchten Algorithmen können ebenfalls einige Aussagen getroffen werden. *ILP* gewährleistet zwar eine exakte Überprüfung von k-assign Anonymität, hat jedoch exponentielle Laufzeit. Das macht sich bereits bei kleinen Datenmengen bemerkbar und sorgt dafür, dass die Antwortzeiten bereits nach wenigen Anfragen im Stundenbereich liegen (vgl. Abbildung 8.3). Damit ist dieser Algorithmus in der Praxis nicht verwendbar.

Über die getesteten Approximationsalgorithmen lässt sich insgesamt sagen, dass deren Güte besser ist, wenn im Ergebnis zu einer Anfrage nur wenig alte Tupel vorhanden sind (vgl. Abbildung 8.4). Kommen indes Tupel besonders häufig in verschiedenen Ergebnissen vor, kann es auch passieren, dass mehr künstliche SA-Werte erzeugt werden, als reale im Ergebnis sind. In allen Fällen erreicht die Anzahl hinzugefügter künstlicher SA-Werte mit zunehmender Anzahl von Anfragen eine bestimmte Grenze, ab der nicht noch mehr Werte erstellt werden. Ebenso existiert auch für die Laufzeit pro Anfrage eine Schranke, deren Höhe von den gewählten Eingabegrößen und Algorithmen abhängt.

Die beste Güte aller Approximationsalgorithmen hat in fast allen Fällen *KlonMatching*, wobei sich *SimpleMatching* und *MaxAno* sehr ähnlich zu ihm verhalten. Diese Erkenntnis erscheint auf den ersten Blick ungewöhnlich, da nur *KlonMatching* eine optimale Erweiterung von Matchings garantiert, während die anderen Verfahren diese nur näherungsweise bestimmen. Eine Ursache dafür stellen die Optimierungsschritte innerhalb der beiden Algorithmen dar, wie zum Beispiel die Unterteilung in mehrere Phasen. Sie wirken sich sehr positiv auf die Berechnung der optimalen Erweiterung aus. Auch die Laufzeit der Algorithmen *KlonMatching*, *SimpleMatching* und *MaxAno* unterscheidet sich nur gering.

Im Gegenzug dazu schneidet *MaxKanten* in allen untersuchten Beispielen zum Teil deutlich schlechter ab als *KlonMatching*. Einerseits erzeugt er mehr künstliche SA-Werte und andererseits liegt seine Laufzeit über der von *KlonMatching*. Das bestätigt die bereits in Kapitel 6 aufgestellte Vermutung, dass optimale Erweiterungen nach Problem 6.1 (*MaxKanten*) zu mehr künstlichen SA-Werten führen als optimale Erweiterungen nach Problem 6.2 (*KlonMatching*).

Der Vorteil von *SA-Ein* ist, dass er Antworten auf Anfragen in viel geringerer Zeit berechnet als die anderen Approximationen. Allerdings erstellt er im Allgemeinen auch weitaus mehr künstliche SA-Werte. Der Grund dafür liegt darin, dass die Ergebnisse zunächst in kleinere Teilergebnisse zerlegt werden, in denen jeder SA-Wert höchstens einmal

vorkommt. Kommt im ursprünglichen Ergebnis ein Wert besonders häufig vor, müssen dementsprechend viele Teilergebnisse gebildet werden. Diese müssen wiederum mit künstlichen SA-Werten aufgefüllt werden, um k-assign Anonymität zu gewährleisten. Abbildung 8.11 zeigt aber, dass es auch Fälle gibt, in denen *SA-Ein* eine bessere Güte hat als die anderen Approximationen. In diesem Beispiel liegt der Wert für k sehr dicht an der Anzahl aller verschiedener SA-Werte, die Ergebnismenge ist jedoch um einiges größer.

M-Inv ist ein Approximationsalgorithmus, der zum Vergleich von Ansätzen verwendet wurde. Er hat eine noch schnellere Laufzeit als *SA-Ein*, erzeugt allerdings viel mehr künstliche SA-Werte als die anderen Verfahren.

Zusammenfassung

In diesem Kapitel wurden die in der vorliegenden Arbeit eingeführten Algorithmen evaluiert. Es wurde zunächst die Güte der Approximationen abgeschätzt und danach wurden verschiedene Szenarien mit unterschiedlichen Einstellungen untersucht. Der Algorithmus *KlonMatching* erzeugt in fast allen Fällen die wenigsten künstlichen SA-Werte und hat damit die beste Güte. Insbesondere wenn viele neue Tupel in den Ergebnissen vorkommen, entstehen gute Resultate. Ist das nicht der Fall oder werden viele Tupel mehrfach angefragt, kann die Anzahl künstlicher SA-Werte mitunter auch die Anzahl realer SA-Werte übersteigen. Das ist für weitere Analysen der Daten meist nicht erwünscht. Die Zeit, die für die Bearbeitung einer Anfrage benötigt wird, liegt je nach Eingabegrößen im Millisekunden- oder niedrigen Sekundenbereich. Damit verlangsamt sich zwar merklich die Zeit der Anfragebearbeitung durch ein DBMS. Im Gegensatz zum Algorithmus *ILP*, der die Einhaltung von k-assign Anonymität in exponentieller Zeit exakt überprüft, können aber weitaus größere Datenmengen analysiert werden. Die niedrige Laufzeit der Approximationen erweist sich daher als großer Vorteil.

Teil III

Diskussion

9 Zusammenfassung und Ausblick

Dieses Kapitel stellt den Abschluss der vorliegenden Arbeit dar. Zunächst werden die wichtigsten Ergebnisse zusammengefasst (Kapitel 9.1) und diskutiert (Kapitel 9.2). Danach werden Angriffe besprochen, die zu einer Verletzung der Privatsphäre führen können, obwohl k-assign Anonymität erfüllt ist (Kapitel 9.3). Als Letztes wird gezeigt, unter welchen veränderten Voraussetzungen der eingeführte Ansatz weiterhin verwendet werden kann (Kapitel 9.4), und es werden weiterführende Forschungsthemen vorgestellt (Kapitel 9.5).

9.1 Zusammenfassung

In der vorliegenden Arbeit wurde der Schutz der Privatsphäre bei der Anfragebearbeitung in Datenbanksystemen untersucht. Ausgehend vom statischen Modell des Veröffentlichens von Mikrodaten wurde ein Ansatz entwickelt, Ergebnisse zu Anfragen herauszugeben. Die Privatsphäre der Individuen, zu denen die angefragten Daten gehören, wird dabei nicht verletzt.

Zunächst wurde in Kapitel 3 das Wissen eines Angreifers, der die Ergebnisse erhält, als Menge von SA-Werteverteilungen modelliert. Eine solche Verteilung entspricht einer möglichen Verknüpfung sensibler Werte mit Individuen, die der Angreifer durch Kenntnis der Ergebnisse ableiten kann. Das dazugehörige Problem SA-Werteverteilung beschreibt die Frage, ob einem gegebenen Tupel, welches zu einem Individuum gehört, ein bestimmter SA-Wert zugeordnet werden kann. Als Schutzkriterium für die Privatsphäre wurde k-assign Anonymität vorgeschlagen. Dessen Einhaltung garantiert, dass zu jedem Tupel mindestens k verschiedene SA-Werte existieren, unter denen der Angreifer den tatsächlichen Wert nicht herausfinden kann. k-assign Anonymität kann durch das Finden von Matchings in speziellen Graphen bestimmt werden. Dazu wird aus jedem Ergebnis zu einer Anfrage ein sogenannter Anfragegraph erstellt, in dem ein (valides) Matching einer möglichen SA-Werteverteilung entspricht. Das Problem QGM enthält die Frage, ob in einer gegebenen Folge von mehreren Anfragegraphen ein solches Matching existiert. Es wurde gezeigt, dass die Probleme QGM und SA-Werteverteilung NP-vollständig sind. Demzufolge kann k-assign Anonymität nicht in Polynomialzeit überprüft werden.[1] Ein mögliches Verfahren wandelt QGM in ein ganzzahliges lineares Programm um, welches mit Standardalgorithmen in exponentieller Zeit gelöst werden kann.

In den Kapiteln 4 bis 7 wurden Approximationsalgorithmen konstruiert, die k-assign Anonymität in Polynomialzeit garantieren. Als wichtige Anforderung an diese wurde in Kapitel 4 festgelegt, dass die Privatsphäre trotz der Näherung geschützt sein soll. Folglich müssen die Algorithmen weiterhin jede Verletzung von k-assign Anonymität finden. Als Vereinfachung werden nur bestimmte effizient zu berechnende Matchings verwendet, die unter dem Begriff der Δ-top Matchings zusammengefasst werden. Das Hauptaugenmerk liegt dabei auf Matchings, die sich nur wenig von dem Matching unterscheiden, welches

[1] wenn $\mathsf{P} \neq \mathsf{NP}$

der realen Zuordnung von SA-Werten zu Tupeln entspricht. In Kapitel 5 wurde gezeigt, wie Δ-top Matchings berechnet werden. Dazu wurde ein dynamischer Ansatz verwendet, der Matchings für eine Folge von n Anfragegraphen aus denen für $n-1$ Anfragegraphen konstruiert. Dieser Schritt wird Erweiterung von Matchings genannt. Kapitel 6 enthält vier verschiedene Algorithmen, die eine optimale Erweiterung berechnen.

Für einen Spezialfall wurde in Kapitel 7 ein vereinfachtes Verfahren angegeben. Hierbei enthält jedes Ergebnis zu einer Anfrage beziehungsweise jeder daraus erstellte Anfragegraph keine mehrfach vorkommenden sensiblen Werte. Ebenfalls wurde eine Möglichkeit aufgezeigt, beliebige Anfragegraphen in kleinere Teilgraphen zu zerlegen, sodass sensible Werte höchstens einmal pro Graph vorkommen. Der Approximationsalgorithmus in Kapitel 7 ist dadurch eine Alternative zu dem aus Kapitel 5.

Wird durch eines der vorgeschlagenen Verfahren eine Verletzung des Schutzkriteriums erkannt, kann die entsprechende Anfrage abgewiesen werden. Es ist aber auch möglich, trotzdem eine Antwort auszugeben. Dazu werden dem Ergebnis künstliche sensible Werte hinzugefügt, sodass k-assign Anonymität weiterhin erfüllt ist.

In Kapitel 8 wurden die beschriebenen Algorithmen evaluiert. Dabei wurde einerseits die Güte der Approximationen analysiert und andererseits die Laufzeiten der Verfahren miteinander verglichen. Insbesondere wurde in vielen Fällen die praktische Verwendbarkeit der in der vorliegenden Arbeit eingeführten Algorithmen aufgezeigt.

9.2 Diskussion der wichtigsten Ergebnisse dieser Arbeit

9.2.1 k-assign Anonymität

Im Bereich der Anfrageauditierung ist k-assign Anonymität ein neuartiges Modell. Ähnlich wie ℓ-Diversität stellt es sicher, dass für jedes Individuum mindestens k sensible Werte existieren, aus denen der Angreifer den tatsächlichen Wert nicht ableiten kann. Dadurch wird ein Schutz der Privatsphäre generiert, der weitaus größer ist, als nur den exakten Wert zu verschleiern.[2] Gleichzeitig ist das Kriterium wesentlich flexibler und für die Aspekte der Privatsphäre besser geeignet als das Prinzip der perfekten Privatsphäre.[3] Ein höherer Wert von k sorgt für einen größeren Schutz, während ein kleinerer Wert mehr Anfragen zulässt. Es ist somit möglich, für unterschiedliche sensible Daten andere Schutzkriterien zu definieren. Außerdem ist k-assign Anonymität wesentlich intuitiver als beispielsweise differentielle Privatsphäre, da der Wert von k eine klar verständliche Bedeutung hat.

Beim Veröffentlichen von Mikrodaten existieren weitere Modelle, die auf k-Anonymität und ℓ-Diversität aufbauen. Meist liegen ihnen bestimmte Angriffe zugrunde, vor denen sie schützen sollen. Es wäre denkbar, auch diese Modelle in die Anfrageauditierung zu integrieren. Ein Beispiel sind die probabilistische Attacke und die Ähnlichkeitsattacke aus Kapitel 2.3.2, vor denen k-assign Anonymität in der vorgestellten Form nicht schützt.

Bei der probabilistischen Attacke kommt ein sensibler Wert innerhalb einer QI-Gruppe wesentlich häufiger vor als andere. Dadurch steigt für einen Angreifer die Wahrscheinlichkeit, dass dieser Wert zu einem bestimmten Individuum gehört. Übertragen auf die Anfrageauditierung und die Abbildung des Wissens eines Angreifers bedeutet das, dass zu einem Tupel t viele SA-Werteverteilungen existieren, die t den sensiblen Wert s, und

[2] Vgl. die Modelle bei statistischen Anfragen aus Kapitel 2.6.1.
[3] Vgl. Kapitel 2.6.2.

nur wenige, die t einen anderen Wert s' zuordnen. k-assign Anonymität kann erweitert werden, sodass nicht nur das Vorhandensein von sensiblen Werten in SA-Werteverteilungen betrachtet wird, sondern auch die jeweilige Anzahl. Wie dieses Prinzip allerdings im Approximationsalgorithmus umgesetzt wird, ist nicht offensichtlich. Zwar werden in der Δ-top Matchingtabelle mehrere Kanten gespeichert, die Tupel mit SA-Werten verknüpfen. Allerdings ist unklar, ob beispielsweise die Einhaltung der Beschränkung auf eine maximale oder minimale Anzahl von Kanten mit bestimmten SA-Werten auch effizient realisiert werden kann.

Dagegen ist es leicht möglich, Ähnlichkeitsattacken vorzubeugen. Dazu muss nur gewährleistet werden, dass die sensiblen Werte, die zusammen mit einem bestimmten Tupel vorkommen, sich untereinander nicht zu sehr ähneln. Für das Schutzkriterium müssen demnach die Mengen aller sensibler Werte untersucht werden, die Tupeln durch SA-Werteverteilungen zugeordnet werden können. Analog kann leicht in der gespeicherten Matchingtabelle überprüft werden, welche sensiblen Werte, die einem Tupel durch Matchings zugeordnet werden können, Ähnlichkeiten zueinander aufweisen. Damit ist sogar die Überprüfung komplexerer Modelle, wie zum Beispiel t-closeness, möglich.

9.2.2 Δ-top Matchings und Approximationsalgorithmen

Die in den Kapiteln 5 und 7 vorgestellten Approximationsalgorithmen berechnen valide Matchings in Folgen von Anfragegraphen. Aus jedem Matching kann eine SA-Werteverteilung gewonnen werden, die wiederum dem Test auf k-assign Anonymität dient. Der Vorteil der Algorithmen ist, dass jede Verletzung des Schutzkriteriums tatsächlich gefunden wird. Die Approximation besteht demnach nur in einer zu pessimistischen Analyse, denn das Verfahren kann sich irren, wenn es eine Verletzung ausgibt.

Aus theoretischer Sicht ist in solchen Fällen die Angabe einer Güte wünschenswert. Dazu muss der Quotient aus exakter und approximierter Lösung eines Optimierungsproblems beschränkt werden.[4] Im vorliegenden Fall ist eine Abschätzung des minimalen Anonymitätsgrades gesucht, aus dem sich der Wert für k-assign Anonymität ableitet. Allerdings zeigt ein Beispiel in Kapitel 4.3.4, dass bereits durch die Beschränkung auf $\Delta 2$-Matchings keine feste Güte garantiert werden kann.[5] Die Approximation kann demnach beliebig schlecht ausfallen.

Eine Alternative ist, Δr-Matchings mit $r > 2$ zuzulassen. Das vergrößert die Anzahl der betrachteten Matchings und führt zu mehr abgebildeten SA-Werteverteilungen. Jedoch zeigt Beispiel A.4 im Anhang auf Seite 273, dass die Beschränkung auf Δr-Matchings für jedes feste r immer dazu führt, dass für die resultierende Approximation keine feste Güte angegeben werden kann. Somit ist nicht garantiert, dass ein höheres r immer zu einer besseren Approximation führt. Der Grund, warum in der vorliegenden Arbeit $r = 2$ gewählt wurde, liegt in der effizienten Berechnung der Matchings. Beispielsweise gibt es mehrere Situationen, in denen zu einer Kante schnell die Gegenkante gesucht werden muss (z. B. beim Löschen). Für $r = 2$ ist diese Kante eindeutig bestimmt und kann leicht aus der Matchingtabelle extrahiert werden. Sind größere Werte für r zugelassen, können Kanten in Matchings nicht so schnell aus der Matchingtabelle abgelesen werden. Insbesondere die Idee, Erweiterungen nur anhand gegebener Matchingkanten durchzuführen, ohne die

[4] Vgl. Kapitel 1.5.2.
[5] Vgl. auch Beispiel A.3 im Anhang auf Seite 270.

tatsächlichen Matchings zu erstellen, funktioniert nicht mehr für $r > 2$. Folglich müssen in diesem Fall andere Darstellungsformen und Algorithmen konzipiert werden.

Das vorgeschlagene Verfahren kann beim Erkennen einer Verletzung des Schutzkriteriums die entsprechende Anfrage verweigern oder zusätzliche sensible Werte im Ergebnis zurückgeben. Das Hinzufügen künstlicher Werte, zu denen kein reales Individuum gehört, ist in manchen Fällen nicht erwünscht. Zum Beispiel könnte es für ein pharmazeutisches Unternehmen wichtig sein, die exakten Reaktionen von Patienten auf bestimmte Medikamente zu erhalten. Gefälschte Werte würden ein Auswerten der Daten für die weitere medizinische Forschung erschweren.

Eine andere Vereinfachung in der vorliegenden Arbeit stellt die Behandlung des Ergebnisses einer Anfrage dar. Wie in den Kapiteln 3 und 5 vorgestellt, wird die Menge aller Tupel im Ergebnis als eine einzige QI-Gruppe betrachtet und anonymisiert. Gibt es beispielsweise 100 Tupel, die als Antwort zurückgegeben werden sollen, allerdings nur 3-assign Anonymität verlangt ist, könnte das Ergebnis eventuell auch in mehrere kleinere QI-Gruppen beziehungsweise Teilergebnisse aufgeteilt werden. Der Vorteil dabei ist, dass der Informationsgehalt einer kleineren QI-Gruppe größer und damit für den Anfragenden nützlicher sein kann als der einer einzigen großen Menge. Insbesondere müssen beim Generalisieren unter Umständen quasi-identifizierende Werte weniger stark verallgemeinert werden. Der Algorithmus in Kapitel 7 verwendet eine ähnliche Idee und clustert die Elemente der Ergebnisse. Dadurch entstehen kleinere Teilergebnisse, in denen der Informationsgehalt größer sein kann. Der Nachteil dieser Variante ist aber, dass das Schutzkriterium schneller verletzt wird, da weniger SA-Werteverteilungen erstellt werden können. Demzufolge müssen wiederum mehr künstliche Werte eingefügt werden, um Anfragen beantworten zu können, und der Nutzen der Daten sinkt. Auf eine genaue Untersuchung der Verbindungen zwischen dem Informationsgehalt der Daten und dem Erkennen von Verletzungen wird in dieser Arbeit verzichtet.

9.3 Angriffe

9.3.1 Verletzung der Privatsphäre trotz erfülltem Schutzkriterium

Inspiriert durch erweiterte Modelle von k-Anonymität wurde in der vorliegenden Arbeit k-assign Anonymität als Maß für den Schutz der Privatsphäre verwendet. Dabei wird das Wissen des Angreifers, welches aus den Daten der zurückgegebenen Ergebnisse gewonnen werden kann, abgebildet und beschränkt. Nicht modelliert werden Hintergrundwissen und logische Schlussfolgerungen, die ein Angreifer aus dem Ablauf des verwendeten Algorithmus ziehen kann. Insbesondere Letzteres kann unter bestimmten Bedingungen zu Verletzungen der Privatsphäre führen, obwohl das eigentliche Schutzkriterium erfüllt ist. Das gilt sowohl beim statischen Modell des einmaligen Veröffentlichens von Daten als auch beim dynamischen Modell, bei dem mehrere Anfragen an die Daten zugelassen sind.

Seien die Mikrodaten aus Abbildung 9.1a gegeben, die komplett veröffentlicht werden sollen. Ein Anonymisierungsalgorithmus generalisiere die Werte des quasi-identifizierenden Attributs `PLZ`, indem er pro Schritt die letzte Ziffer durch einen Stern ersetze.[6] Dazu sei eindeutige 2-Diversität gefordert und eine gewisse Art von Minimalität vorausgesetzt, das heißt, der Algorithmus anonymisiert nicht mehr als unbedingt nötig.

[6] analog zur Hierarchie in Abbildung 2.2 auf Seite 42

Name	PLZ	SA
Alina	10104	A
Bill	10109	A
Clive	10155	B
Doris	10156	C
Emily	10157	D

(a) Mikrodaten

	PLZ	SA
Alina	1010*	A
Bill	1010*	A
Clive	1015*	B
Doris	1015*	C
Emily	1015*	D

(b) Unzureichend anonymisierte Daten

	PLZ	SA
Alina	101**	A
Bill	101**	A
Clive	101**	B
Doris	101**	C
Emily	101**	D

(c) Anonymisierte Daten (2-divers)

Abbildung 9.1: Angriff auf eine 2-diverse Tabelle

Ein Angreifer kann aus den veröffentlichten Daten aus Abbildung 9.1c leicht ablesen, dass der Algorithmus alle `PLZ`-Werte zweimal generalisiert hat und demzufolge die Tabelle nach dem ersten Schritt noch nicht eindeutig 2-divers war. Kennt er außerdem die Postleitzahlen aller enthaltener Individuen, kann er ableiten, dass die Tupel für Alina und Bill sowie Clive, Doris und Emily nach dem einmaligen Generalisieren jeweils eine QI-Gruppe gebildet haben (siehe Abbildung 9.1b). Da mindestens eine dieser beiden Gruppen nicht eindeutig 2-divers gewesen sein muss und insgesamt nur der Wert A zweimal vorkommt, kann der Angreifer schlussfolgern, dass Alina und Bill den sensiblen Wert A haben müssen. Alle anderen Möglichkeiten würden nach dem ersten Generalisierungsschritt zu einer eindeutig 2-diversen Tabelle führen. Die Privatsphäre von Alina und Bill ist in diesem Fall offensichtlich verletzt, obwohl das Kriterium der eindeutigen ℓ-Diversität erfüllt ist.

Während es eine Vielzahl von Literatur über Restriktionen für veröffentlichte Tabellen wie k-Anonymität gibt, existieren nur wenige Ansätze, die sich mit dem Wissen auseinandersetzen, welches ein Angreifer aus der Kenntnis des Anonymisierungsalgorithmus erhält. Wong et al. [158] bezeichneten beispielsweise Angriffe wie im beschriebenen Fall als *Minimalitätsattacken* und stellten Varianten auch für weitere Anonymisierungsoperationen sowie für andere Modelle wie t-closeness oder (k, e)-Anonymität vor. Die von ihnen vorgeschlagene Lösung umfasst den Verzicht auf das Minimalitätsprinzip und das Einfügen zufälliger Generalisierungsschritte im verwendeten Algorithmus. Eine eher theoretische Untersuchung des Problems führten Zhang et al. [165] durch. Sie forderten, dass für jede veröffentlichte Tabelle T^* mehrere unterschiedliche Mikrodatentabellen existieren sollen, die durch den Anonymisierungsalgorithmus ebenfalls zu T^* generalisiert worden wären. Ein Angreifer kann aus der Kenntnis von T^* dadurch nicht schlussfolgern, welche der Mikrodatentabellen die Originaldaten repräsentiert. Während der von ihnen präferierte Algorithmus jedoch wenig praktikabel ist, schlugen Xiao et al. [164] ein ähnliches Modell vor, das allerdings wesentlich einfacher zu erfüllen ist. Alle drei Ansätze würden übrigens im obigen Beispiel die `PLZ`-Werte in jedem Fall zweimal generalisieren, auch wenn nach dem ersten Schritt bereits eindeutige 2-Diversität erfüllt wäre.

Für das anfragenbasierte Szenario untersuchten Kenthapadi et al. [80] das Wissen, welches Angreifer durch Kenntnis des Algorithmus erhalten. Sie zeigten insbesondere, dass bereits das Abweisen einer Anfrage zu einer Verletzung der Privatsphäre führen kann. Sei beispielsweise der Wert 15 die Antwort auf die gesuchte Summe dreier Zahlen. Als Nächstes soll das Maximum dieser Zahlen zurückgegeben werden. Wenn diese Anfrage abgewiesen wird, kann der Anfragende schlussfolgern, dass alle drei Zahlen den Wert 5 haben. Jede andere Kombination würde ein Ergebnis zur zweiten Anfrage liefern. Kent-

hapadi et al. schlugen vor, jede Anfrage abzuweisen, bei der es einen Fall gibt, in welchem die Antwort zu einer Verletzung des Schutzkriteriums führen kann. Somit wird die zweite Beispielanfrage nach dem Maximum auch dann nicht beantwortet, wenn die drei Zahlen unterschiedliche Werte besitzen (siehe Kapitel 2.6.1).

9.3.2 Angriffe auf k-assign Anonymität

Angriffe, wie sie im vorigen Abschnitt vorgestellt wurden, können auch gegen k-assign Anonymität erstellt werden. Anhand von drei Beispielen werden im Folgenden Szenarien vorgestellt, bei denen die Privatsphäre verletzt wird, obwohl das Schutzkriterium erfüllt ist. Eine wichtige Voraussetzung liegt jeweils darin, dass der Angreifer den kompletten Algorithmus im Detail kennt und Anfragen nur dann abgewiesen werden, wenn das Herausgeben ihres Ergebnisses zu einer Verletzung von k-assign Anonymität führen würde. Im Anschluss daran wird diskutiert, was im vorgestellten Algorithmus verändert werden muss, um auch vor solchen Angriffen zu schützen.

Seltenster SA-Wert

Im ersten Fall wird jede Anfrage beantwortet, das heißt, es werden dem Ergebnis bei Bedarf künstliche sensible Werte hinzugefügt. Wie in dieser Arbeit vorgestellt, sollen dabei so wenige Werte wie möglich verwendet werden.

Name	SA
Alina Clive Doris	A, B, C

(a) Ergebnis R_1

Name	SA
Alina Bill Emily Frank Gill	A, A, B, B, C, D

(b) Ergebnis R_2' (inkl. künstlichem SA-Wert)

Abbildung 9.2: Angriff auf den seltensten SA-Wert

Abbildung 9.2 zeigt die Ergebnisse zweier Anfragen an eine hier nicht dargestellte Tabelle. Das erste Ergebnis in (a) besteht aus drei Tupeln für Alina, Clive und Doris und enthält die sensiblen Werte A, B sowie C. Aus Gründen der Übersicht sind anstelle von quasi-identifizierenden Attributewerten in dem Beispiel die Namen der Individuen angegeben, zu denen die jeweiligen Tupel gehören. R_1 erfüllt k-assign Anonymität mit $k = 3$, denn jedes Individuum kann mit drei verschiedenen SA-Werten verknüpft werden. Eine zweite Anfrage selektiert die Tupel von Alina, Bill, Emily, Frank und Gill. Die zurückgegebenen SA-Werte umfassen je zwei A und B sowie ein C und ein D (siehe (b)). Da das Ergebnis insgesamt einen SA-Wert mehr als Tupel enthält, kann ein Angreifer schlussfolgern, dass ein künstlicher SA-Wert hinzugefügt wurde und demnach das korrekte Ergebnis zu einer Verletzung des Schutzkriteriums geführt hätte. Da Alina das einzige Individuum ist, das in beiden Ergebnissen vorkommt, und Tabelle (b) insgesamt vier verschiedene SA-Werte enthält, muss die Verletzung bei Alina aufgetreten sein.[7] Aus der ersten Anfrage folgt,

[7] Es kann leicht verifiziert werden, dass die Verletzung des Schutzkriteriums nicht bei einem der neuen Tupel aufgetreten sein kann.

dass Alina A, B oder C haben muss. Angenommen, sie hätte A oder B, dann muss der künstliche Wert C sein, denn andernfalls würde keine Verletzung von 3-assign Anonymität vorliegen. Wenn ihr originaler Wert hingegen C wäre, müsste kein künstlicher Wert hinzugefügt werden, denn es kommen mindestens ein reales A und ein reales B vor. Demzufolge können Alina mit A und B nur noch zwei verschiedene sensible Werte zugeordnet werden, was einer Verletzung ihrer Privatsphäre entspricht.

Im beschriebenen Fall hat k-assign Anonymität keinen ausreichenden Schutz erzeugt. Die Ursachen dafür sind zum einen, dass der Angreifer die Existenz eines künstlichen sensiblen Wertes ableiten konnte und dadurch weitere Informationen erhalten hat. Zum anderen konnte er durch Kenntnis des verwendeten Algorithmus nachvollziehen, welcher Wert in der beschriebenen Situation hinzugefügt wurde.

Zu diesem Angriff gibt es mehrere Abwehrmechanismen. Wenn die zweite Anfrage beispielsweise abgewiesen worden wäre, hätte der Angreifer keine konkreten Werte schlussfolgern können. Er hätte nur von einer Verletzung des Schutzkriteriums gewusst, wenn das Ergebnis zurückgegeben worden wäre. Eine weitere einfache Möglichkeit besteht darin, auf die Minimalität beim Erzeugen der künstlichen SA-Werte zu verzichten. Analog zu den von Wong et al. [158] beschriebenen Minimalitätsattacken könnten als Ergebnis auch die SA-Werte A, A, B, B, C, C, D ausgegeben werden. Dadurch kann der Angreifer nicht mehr zwischen den realen Werten A und B sowie dem künstlichen Wert C unterscheiden, denn diese Werte kommen jeweils zweimal vor. Ein dritter Ansatz bezieht sich auf die Darstellung des Ergebnisses. In Tabelle (b) kommen fünf Tupel und sechs SA-Werte vor, von daher ist es offensichtlich, dass genau ein künstlicher Wert hinzugefügt worden ist. Werden Daten eines fiktiven sechsten Individuums ebenfalls ausgegeben, weiß der Angreifer noch nicht einmal, dass die korrekte Ausgabe zu einer Verletzung des Schutzkriteriums führen würde. Diese Variante ist die effektivste bei diesem und den beiden folgenden Angriffen.

Doppelte SA-Werte

Der zweite Angriff kann sowohl bei der Verweigerung einer Anfrage als auch beim Hinzufügen von künstlichen SA-Werten durchgeführt werden. Seien R_1 und R_2 aus Abbildungen 9.3a und 9.3b die realen Ergebnisse zu zwei Anfragen und 2-assign Anonymität gefordert. Während R_1 ohne Einschränkungen zurückgegeben werden kann, würde R_2 offensichtlich zu einer Verletzung des Schutzkriteriums führen. Wird die zweite Anfrage abgewiesen, kann ein Angreifer trotzdem leicht auf die sensiblen Werte von Alina und Bill schließen. Dazu muss er durch Kenntnis von externen Daten nur herausfinden, dass die Tupel von Alina und Bill durch die zweite Anfrage selektiert wurden. Die Anfrage wird nämlich nur dann nicht beantwortet, wenn beide angefragten Tupel denselben SA-Wert haben. Da nur A zweimal in R_1 vorkommt und R_2 sozusagen eine Teilmenge von R_1 ist, müssen Alina und Bill den sensiblen Wert A haben.

Wird anstelle von R_2 das Ergebnis R_2' aus Abbildung 9.3c ausgegeben, entsteht erneut ein Bruch der Privatsphäre, wenn der Angreifer herausfindet, dass ein künstlicher Wert hinzugefügt wurde. Falls die beiden realen Werte aus R_2' nämlich A und D wären, wäre 2-assign Anonymität nicht verletzt und es müsste kein sensibler Wert ergänzt werden. Demzufolge ist D der künstliche SA-Wert.

Das Verhindern dieses Angriffes ist nur schwer möglich, denn selbst durch ein Hinzufügen weiterer künstlicher SA-Werte können die obigen Schlussfolgerungen gezogen werden. Verursacht wird dies dadurch, dass der Angreifer die Anzahl der künstlichen SA-Werte

Name	SA
Alina Bill Clive Doris Emily	A, A, B, C, D

(a) Ergebnis R_1

Name	SA
Alina Bill	A, A

(b) Reales Ergebnis R_2

Name	SA
Alina Bill	A, A, D

(c) Ergebnis R'_2 (inkl. künstlichem SA-Wert)

Abbildung 9.3: Angriff durch doppelte SA-Werte

kennt und die zweite Anfrage eine Teilmenge der ersten ist. Wenn beides nicht auszuschließen ist, kann zum Beispiel der Algorithmus so verändert werden, dass er in gewissen Situationen auch dann künstliche SA-Werte ausgibt, wenn diese für das Schutzkriterium nicht benötigt werden. Dann kann selbst durch R'_2 nicht ausgeschlossen werden, dass die realen Werte A und D sind und das zweite A nur zusätzlich eingefügt wurde.

Test auf SA-Werte

Im Gegensatz zu den beiden eben eingeführten Angriffen besteht das letzte hier vorgestellte Szenario aus Ergebnissen, in denen jeder SA-Wert höchstens einmal vorkommt. Damit ist es auch für den Algorithmus aus Kapitel 7 relevant. Sei als Schutzkriterium 3-assign Anonymität festgelegt. Anfragen sollen abgewiesen werden, wenn das Herausgeben ihres Ergebnisses das Kriterium verletzen würde. R_1 aus Abbildung 9.4a, welches Informationen zu Alina, Clive und Doris enthält, erfüllt 3-assign Anonymität und kann ausgegeben werden. Ein Angreifer stelle nun drei weitere Anfragen, die die Ergebnisse R_2, R_3 und R_4 erzeugen (siehe Abbildungen 9.4b, 9.4c und 9.4d). Das Herausgeben jeweils eines dieser Ergebnisse würde zu einer Verletzung des Schutzkriteriums führen. Beispielsweise würde der Angreifer durch das Veröffentlichen von R_2 wissen, dass Alina und Clive nicht den sensiblen Wert C haben können. Somit müssen die entsprechenden Antworten verweigert werden.

Name	SA
Alina Clive Doris	A, B, C

(a) Ergebnis R_1

Name	SA
Alina Clive Emily	A, B, D

(b) Ergebnis R_2

Name	SA
Alina Doris Emily	A, C, D

(c) Ergebnis R_3

Name	SA
Clive Doris Emily	B, C, D

(d) Ergebnis R_4

Abbildung 9.4: Angriff durch sukzessives Testen

Der Angreifer gelangt aber dadurch an Informationen über den sensiblen Wert von Emily. Wenn dieser A, B oder C wäre, könnte eine seiner Anfragen beantwortet werden. Wäre er zum Beispiel A, würde das Ergebnis zur insgesamt vierten Anfrage aus den sensiblen Werten A, B und C bestehen und könnte ausgegeben werden. Folglich kann der Angreifer drei SA-Werte komplett ausschließen. Dieses Wissen wird mit dem gezeigten Ansatz nicht modelliert und kann dazu benutzt werden, den sensiblen Wert von Emily

herauszufinden (z. B. durch eine weitere Anfrage, die neben dem Tupel von Emily Tupel von Individuen mit den SA-Werten A, B und C selektiert).

Die Ursache für die Verletzung der Privatsphäre liegt darin, dass mit dem in der vorliegenden Arbeit verwendeten Modell das Wissen durch ein nicht herausgegebenes Ergebnis nicht abgebildet wird. Beispielsweise würde das Hinzufügen von künstlichen Werten diesen Angriff verhindern, denn das daraus entstehende Wissen würde gespeichert werden. Andererseits können auch bestimmte Arten von Anfragen im Vorhinein ausgeschlossen werden. In diesem Fall sind es k Anfragen, deren Ergebnisse sich nur durch ein Tupel unterscheiden. Mit dieser Beschränkung würden die Ergebnisse R_2 bis R_4 auch dann nicht ausgegeben werden, wenn sie zu keiner Verletzung des Schutzkriteriums führen würden.

Zusammenfassung der Angriffsszenarien

Anhand dreier Beispiele wurden verschiedene Angriffe auf k-assign Anonymität vorgestellt und diverse Abwehrmechanismen diskutiert. Dabei stellt sich heraus, dass ein Angreifer auch dann bereits Informationen erhält, wenn eine Anfrage nicht beantwortet wird. Das daraus entstehende Wissen kann er zusammen mit anderen Ergebnissen nutzen, um sensible Werte herauszufinden. Auch das Hinzufügen von künstlichen SA-Werten kann zu Verletzungen der Privatsphäre führen. Dabei wird jeweils vorausgesetzt, dass der Angreifer die Anzahl der zusätzlichen Werte exakt bestimmen kann. Eine vollständige Untersuchung aller Angriffe und Gegenmaßnahmen geht jedoch über den Rahmen der vorliegenden Arbeit hinaus. An dieser Stelle werden nur die wichtigsten Ideen kurz erläutert.

Zum einen können Angriffe vermieden werden, wenn der Angreifer nicht weiß, dass er neben realen auch künstliche Daten erhält. Damit modelliert k-assign Anonymität exakt das Wissen, das der Angreifer hat, und schützt die Privatsphäre. Diese Annahme mag in vielen realen Situationen gültig sein, jedoch birgt sie Gefahren in sich. In der Regel ist es nicht ersichtlich, auf welche externen Daten der Anfragende zugreifen kann. Kennt er tatsächlich die quasi-identifizierenden Attributwerte sämtlicher in der Datentabelle enthaltener Individuen, weiß er auch, wie viele Tupel und SA-Werte in den korrekten Ergebnissen vorkommen müssen.

Ein anderer Ansatz verzichtet auf die Minimalität der Anzahl künstlicher Werte. Der vorgestellte Algorithmus würde in diesem Fall nach einem zufälligem Muster weitere künstliche SA-Werte in Ergebnisse einfügen, selbst wenn diese durch das Schutzkriterium nicht benötigt werden. Die vorgestellten Schlussfolgerungen des Angreifers wären damit nicht mehr deterministisch möglich. Dieser Ansatz beruht auf dem Verfahren zur Verhinderung von Minimalitätsattacken von Wong et al. [158].

Eine letzte Idee umfasst das Untersuchen bestimmter Anfragen. Analog zum Ansatz von Kenthapadi et al. [80] werden bestimmte Anfragen abgewiesen, unabhängig davon, ob das Schutzkriterium durch das Ergebnis erfüllt ist oder nicht. Dadurch kann das Wissen, das ein Angreifer durch das Abweisen einer Anfrage erhält, beschränkt werden. Beispiele für solche Anfragen sind die aus dem zweiten und dritten Angriffsszenario.

9.4 Andere Voraussetzungen des Szenarios

In der vorliegenden Arbeit werden einige Annahmen getroffen, die mehrheitlich auch in anderen Ansätzen zum Schutz der Privatsphäre zu finden sind. Während manche Voraussetzungen für das Modell zwingend nötig sind, können die hier vorgestellten Ideen

auch in anderen Szenarien verwendet werden. Auf einige Beispiele wird im Folgenden kurz eingegangen.

Dynamische Daten Bisher wurde stets davon ausgegangen, dass die angefragten Mikrodaten statisch und dementsprechend keine Änderungen zugelassen sind. In der Praxis hingegen werden in Datenbeständen häufig Einträge eingefügt, modifiziert oder gelöscht. Die in dieser Arbeit vorgeschlagenen Algorithmen würden auch mit Aktualisierungen der Daten zurechtkommen, wenn folgende Bedingung umgesetzt wird: Beim Löschen eines Tupels bleibt der entsprechende Eintrag in der Δ-top Matchingtabelle erhalten. Andernfalls könnten zukünftige Anfragen dafür sorgen, dass bestimmte Kanten beziehungsweise Zuordnungen von sensiblen Werten zum gelöschten Tupel entfernt werden. Damit könnte eine Verletzung von k-assign Anonymität bei diesem Tupel nicht mehr gefunden werden. Demgegenüber stellt das Einfügen von Tupeln in die Datentabelle kein Problem dar, denn auch andere Tupel werden eventuell erst durch spätere Anfragen zum ersten Mal selektiert. Das Ändern eines sensiblen Wertes wiederum entspricht dem Löschen und Einfügen des Wertes und kann damit ebenfalls modelliert werden.

Nullwerte Beim Veröffentlichen von Daten werden in der Regel Mikrodaten betrachtet, bei denen keine Nullwerte vorkommen. Aus Datenbanksicht spielen diese aber vor allem in praktischen Anwendungen eine gewisse Rolle. Während nicht vorhandene quasi-identifizierende Attributwerte zu keinem Bruch der Privatsphäre führen, sind Nullwerte bei sensiblen Attributen kritischer zu werten. So stellt sich zum Beispiel die Frage, welchen Grad an Privatsphäre eine QI-Gruppe mit einem sensiblen Wert und drei Nullwerten hat. Werden fehlende Werte wie ein fester anderer Wert angesehen, ist die besagte QI-Gruppe 2-divers, denn dann kommen nur zwei verschiedene Werte vor. Dieses Konzept kann leicht in die Anfragebearbeitung übernommen werden, wobei im Algorithmus Anfragegraphen entstehen, deren SA-Knoten mit dem Wert für Null gelabelt sind. Somit können Nullwerte auch bei k-assign Anonymität beachtet werden. Auf der anderen Seite ergibt sich aber auch die Frage, ob das Nichtvorhandensein eines Wertes die Privatsphäre nicht wesentlich besser schützt als eine mögliche Zuordnung zu anderen Werten. Beispielsweise besteht bei der eben betrachteten QI-Gruppe nur eine 25%ige Wahrscheinlichkeit, dass ein Individuum überhaupt einen sensiblen Wert hat. Damit ist die Privatsphäre stärker geschützt, als wenn in der Gruppe neben dem Wert A noch dreimal der Wert B vorkommen würde. Aus diesem Grund ist es naheliegender, für Nullwerte andere Schutzmodelle zu entwickeln.

Nutzerverwaltung In der vorliegenden Arbeit wurde immer genau eine gegebene Folge von Anfragen betrachtet. In der Praxis wird ein Datenbanksystem jedoch von verschiedenen Nutzern verwendet, die unterschiedliche Anfragen stellen. Wird vorausgesetzt, dass all diese Nutzer untereinander nicht kommunizieren, kann für jeden eine separate Δ-top Matchingtabelle verwaltet werden. Dafür muss eine korrekte Authentifizierung sichergestellt werden, denn jeder Nutzer hat ein anderes Wissen über die sensiblen Werte. Wird allerdings davon ausgegangen, dass die Nutzer untereinander Informationen austauschen können, muss eine globale Matchingtabelle für alle verwendet werden. Darin wird das Wissen einer Person widergespiegelt, die sämtliche bisher zurückgegebenen Ergebnisse kennt.

9.5 Ausblick auf zukünftige Arbeiten

Aus dem in der vorliegenden Arbeit vorgestellten Ansatz, die Privatsphäre bei der Anfragebearbeitung zu schützen, entstehen weitere interessante Fragestellungen, die Grundlage zukünftiger Forschung sein könnten. Ein relevantes Thema sind die bereits angesprochenen möglichen Angriffe, bei denen trotz erfülltem Schutzkriterium durch zusätzliches Wissen eine Verletzung der Privatsphäre entsteht. Weitere Ideen werden im Folgenden kurz besprochen.

Andere Schutzkriterien Wie im Kapitel 9.2.1 bereits diskutiert, gibt es eine Vielzahl von Modellen zum Schutz der Privatsphäre.[8] k-assign Anonymität verfolgt dabei eine ähnliche Idee wie ℓ-Diversität. Es könnte untersucht werden, welche weiteren Modelle des PPDP auf die Anfrageauditierung übertragen werden können.

Definition eines Nutzens Die Evaluation in Kapitel 8 hat gezeigt, dass die Approximationsalgorithmen für beliebige Anfragegraphen aus Kapitel 5 eine bessere Güte haben als der Algorithmus für SA-eindeutige Graphen aus Kapitel 7. Demgegenüber steht, dass die Ergebnismengen des letztgenannten Verfahrens wesentlich kleiner sind als bei den anderen Methoden und damit potentiell einen höheren Informationsgehalt besitzen könnten. Das Entwickeln einer entsprechenden Metrik, welche zurückgegebene Ergebnisse beispielsweise mithilfe eines Nutzenwertes bewertet, stellt eine weitere Herausforderung zukünftiger Arbeiten dar (vgl. Diskussion in Kapitel 9.2.2).

Andere Matchings Der in der vorliegenden Arbeit vorgestellte Approximationsalgorithmus verwendet nur Δ-top Matchings. Insbesondere sorgt die Beschränkung auf Δr-Matchings mit $r = 2$ dafür, dass für die Approximation keine feste Güte angegeben werden kann. Werden Matchings mit $r > 2$ zugelassen, müssen andere Algorithmen und Datenstrukturen entwickelt werden. Der entstehende Vorteil lautet, dass durch größere Werte für r mehr Matchings und damit auch mehr Zuordnungen von SA-Werten zu Tupeln modelliert werden. Der Fehler der Approximation wird dadurch geringer. Das Entwickeln anderer Ideen für größere r stellt dementsprechend ein interessantes Forschungsthema dar.

Mehrere Tupel pro Individuum Veränderte Voraussetzungen liegen vor, wenn mehrere Tupel in der Datentabelle zu einem Individuum gehören können. Beispielsweise könnte eine Person mehrere Krankheiten haben, zu denen jeweils ein eigenständiger Eintrag existiert. Problematisch wird es, wenn ein Ergebnis zu einer Anfrage nur SA-Werte enthält, die zum selben Individuum gehören. Das vorgestellte Modell würde Zuordnungen der dazugehörigen Tupel untereinander zulassen, jedoch widerspricht das dem Schutz der Privatsphäre. Um diese Fälle sinnvoll abzubilden, müssten weitere Bedingungen an die zu berechnenden SA-Werteverteilungen beziehungsweise Matchings gestellt werden. Einerseits dürfen dann Tupel, die zum selben Individuum gehören, nicht miteinander erweitert werden. Andererseits ist es nicht offensichtlich, wie dieser Grundsatz effizient in die beschriebenen Algorithmen integriert werden kann.

[8] In Kapitel 2 wurden die wichtigsten Modelle im Rahmen dieser Arbeit vorgestellt.

Anfragen einschränken In Kapitel 3 wurde gezeigt, dass die Berechnung von k-assign Anonymität mittels SA-Werteverteilungen NP-vollständig ist. Es bietet sich daher an, Anfragen so einzuschränken, dass die Sicherstellung des Schutzkriteriums effizienter gewährleistet werden kann. Wenn sich Ergebnismengen beispielsweise nicht überschneiden, ist die vorgestellte Approximation exakt und produziert keine Ungenauigkeiten. Dahingegen führen überlappende Mengen zum Erweitern von Matchings und erhöhen damit die Wahrscheinlichkeit, dass eine Verletzung von k-assign Anonymität ausgegeben wird, obwohl keine vorliegt. Eine detaillierte Analyse bestimmter Anfragen und Ergebnisse kann eventuell Spezialfälle hervorbringen, in denen effizientere Algorithmen verwendet werden können. Auch eine weitere theoretische Untersuchung der vorgestellten Probleme SA-Werteverteilung und QGM unter bestimmten Einschränkungen kann zu neuen Erkenntnissen führen. Denkbar ist ebenso, den Erwartungswert des Fehlers der Approximation abzuschätzen.

Alle Anfragen zusammen Eine kleine Änderung des Szenarios liegt vor, wenn alle gestellten Anfragen zu Beginn der Berechnungen bekannt sind. Da die vorgestellte Approximation Tupel einer Anfrage in alte und neue unterteilt, ist es ein Unterschied, ob Anfrage Q_i vor oder nach Anfrage Q_j bearbeitet wird. Dementsprechend kann eine andere Reihenfolge der Anfragen zu einer anderen Approximation führen. Ein interessanter Forschungsgegenstand weiterer Überlegungen könnte deshalb das Herleiten eines Kriteriums sein, wie die Anfragen zu ordnen sind, damit möglichst viele beantwortet werden können.

Weitere Operationen Als mögliche Operationen auf relationalen Daten wurden in dieser Arbeit Selektion und Projektion betrachtet. Auch wenn in den Beispielen Projektionen nicht explizit vorkamen, ist es offensichtlich, dass durch die Beschränkung auf bestimmte Attribute das Wissen des Angreifers nur abnehmen kann. Die vorgestellte Abbildung seiner Annahmen kann damit nur restriktiver werden. Weiterführend wäre es deshalb, in anderen Ansätzen zusätzliche Operationen und insbesondere Daten zu betrachten, die in mehreren Relationen gespeichert sind.

Zusammenfassen von sensiblen Werten Bei den aufgeführten Angriffen ist es oft entscheidend, wie häufig ein bestimmter sensibler Wert vorkommt. Wenn beim Veröffentlichen der Ergebnisse aber jeder enthaltene SA-Wert nur einmal aufgeführt wird, können einige Angriffe verhindert werden. Die Privatsphäre wäre stärker geschützt, da bestimmte Zuordnungen von SA-Werten zu Tupeln für den Angreifer zusätzlich möglich erscheinen. Allerdings ist die korrekte Abbildung des Wissens des Angreifers mittels Anfragegraphen mit dem vorgestellten Verfahren nicht mehr möglich und es müsste ein anderes oder erweitertes Modell entwickelt werden. Ebenfalls sinkt der Informationsgehalt des Ergebnisses, denn es kann für manche Nutzer wichtig sein, ob ein SA-Wert, wie eine bestimmte Krankheit, häufiger vorkommt als ein anderer.

Korrelation zwischen sensiblen Werten Eine wichtige Voraussetzung in der vorliegenden Arbeit beruht darauf, dass verschiedene sensible Werte als tatsächlich unterschiedliche Werte angesehen werden. Kommen zum Beispiel in einem Ergebnis drei verschiedene Krebserkrankungen vor, ist 3-assign Anonymität erfüllt. Allerdings kann ein Angreifer daraus schließen, dass die dazugehörigen Individuen ernsthafte Krankheiten haben. Auch

werden Begriffe als verschieden verarbeitet, die Synonyme voneinander darstellen. Abhilfe könnte eine Taxonomie der sensiblen Werte schaffen, bei der zum Beispiel alle Krebsvarianten zum einheitlichen sensiblen Wert „Krebs“ zusammengefasst werden. An Ansätzen dieser Art kann zukünftig geforscht werden.

Integration ins DBMS Bisher wurden alle Ansätze und Algorithmen außerhalb eines Datenbankmanagementsystems getestet. Dabei bestehen die Eingaben im Wesentlichen aus den Ergebnissen, die das darunterliegende DBMS zu den gestellten Anfragen generiert. Um den Schutz der Privatsphäre bei der Anfragebearbeitung in realen Systemen zu integrieren, sollten die vorgestellten Methoden ins DBMS eingebettet werden. Dabei ergeben sich mehrere datenbankspezifische Fragestellungen, wie zum Beispiel nach der effizienten Verwaltung und Speicherung der Δ-top Matchingtabelle. Auch können eventuell bereits bei der Anfrageausführung bestimmte Berechnungen des Schutzkriteriums durchgeführt werden.

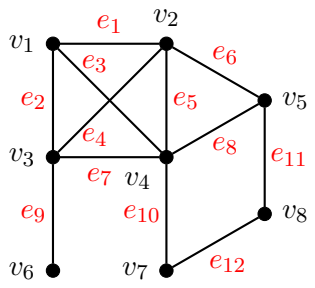
Teil IV

Anhang

A Weiterführende Beispiele

In diesem Teil des Anhangs sind mehrere Beispiele aufgeführt, die keinen Platz im Hauptteil der Arbeit fanden, da sie sehr komplex sind oder nur einen Spezialfall behandeln.

A.1 Graphen und Matchings



(a) Graph G

$$
\begin{aligned}
v_1: &\quad x_{e_1} + x_{e_2} + x_{e_3} \leq 1\\
v_2: &\quad x_{e_1} + x_{e_4} + x_{e_5} + x_{e_6} \leq 1\\
v_3: &\quad x_{e_2} + x_{e_4} + x_{e_7} + x_{e_9} \leq 1\\
v_4: &\quad x_{e_3} + x_{e_5} + x_{e_7} + x_{e_8} + x_{e_{10}} \leq 1\\
v_5: &\quad x_{e_6} + x_{e_8} + x_{e_{11}} \leq 1\\
v_6: &\quad x_{e_9} \leq 1\\
v_7: &\quad x_{e_{10}} + x_{e_{12}} \leq 1\\
v_8: &\quad x_{e_{11}} + x_{e_{12}} \leq 1
\end{aligned}
$$

(b) Nebenbedingungen des ILP

$$
\underbrace{\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array}
\begin{array}{c}
\begin{array}{cccccccccccc} x_{e_1} & x_{e_2} & x_{e_3} & x_{e_4} & x_{e_5} & x_{e_6} & x_{e_7} & x_{e_8} & x_{e_9} & x_{e_{10}} & x_{e_{11}} & x_{e_{12}} \end{array}\\
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{pmatrix}
\end{array}}_{A}
\cdot
\underbrace{\begin{pmatrix}
x_{e_1}\\ x_{e_2}\\ x_{e_3}\\ x_{e_4}\\ x_{e_5}\\ x_{e_6}\\ x_{e_7}\\ x_{e_8}\\ x_{e_9}\\ x_{e_{10}}\\ x_{e_{11}}\\ x_{e_{12}}
\end{pmatrix}}_{x}
\leq
\underbrace{\begin{pmatrix}
1\\1\\1\\1\\1\\1\\1\\1
\end{pmatrix}}_{b}
$$

(c) Nebenbedingungen in Matrixschreibweise

Abbildung A.1: Matchingproblem als ganzzahliges lineares Programm. (a) zeigt den Graphen G, (b) und (c) jeweils die Nebenbedingungen des ILP, wobei zusätzlich $x_{e_i} \in \{0, 1\}$ für alle $1 \leq i \leq 12$ gilt.

Beispiel A.1 In Abbildung A.1 ist in (a) ein Graph G dargestellt, in dem ein größtes Matching berechnet werden soll. Die Nebenbedingungen eines dazugehörigen ganzzahli-

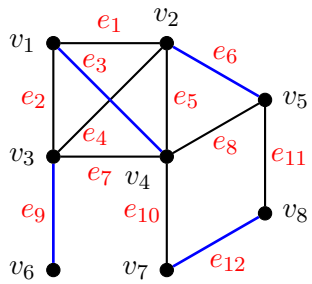


Abbildung A.2: Die Menge der farbigen Kanten bildet ein größtes Matching im Graphen.

gen linearen Programms sind für alle Knoten v_i in (b) und in Matrixschreibweise in (c) angegeben. Die Matrix A entspricht dabei der Indizenzmatrix von G. Die Zielfunktion des ILP ist $\max \sum_{i=1}^{12} x_{e_i}$ und eine mögliche optimale Lösung

$$x_{e_3} = x_{e_6} = x_{e_9} = x_{e_{12}} = 1,$$
$$x_{e_1} = x_{e_2} = x_{e_4} = x_{e_5} = x_{e_7} = x_{e_8} = x_{e_{10}} = x_{e_{11}} = 0.$$

Die Lösung hat den Wert 4 und entspricht dem in Abbildung A.2 dargestellten Matching.

A.2 Anfragegraphen und Δ-top Matchings

Beispiel A.2 Abbildung A.3 zeigt in (a) alle möglichen SA-Werteverteilungen für den ersten Graphen des im Kapitel 3 eingeführten Beispiels. Der entsprechende Anfragegraph $\mathcal{G}^{(1)} = (G_1)$ ist in (b) und die zu den Verteilungen äquivalenten Matchings in (c) bis (n) gegeben. M_1 stellt dabei das Originalmatching in G_1 dar und ist daher farblich anders gekennzeichnet. Zu jedem Matching ist in der Tabelle in (a) der Wert von r für Δr-Matchings bezüglich M_1 angegeben (vgl. Definition 4.6) und es ist markiert, ob das Matching Δ-minimal bezüglich M_1 ist (vgl. Definition 4.4). Zu jedem aufgeführten Matching M_i existiert ein vollständig SA-äquivalentes Matching, das aus M_i dadurch erzeugt werden kann, indem die Matchingparter der beiden SA-Knoten mit Label A vertauscht werden.

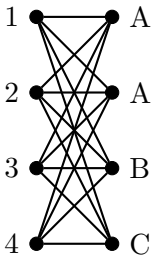
Das folgende Beispiel zeigt, dass die Beschränkung auf Δ2-Matching zu einer Approximation führt, die beliebig schlecht werden kann. Das heißt, der Quotient aus tatsächlichem Wert des Anonymitätsgrades und des durch Δ2-Matching berechneten Wertes wird beliebig groß.

Beispiel A.3 In Abbildung A.4 ist eine Folge von Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3, G_4)$ gegeben. Dabei wurde auf die Angabe von Labels für Tupelknoten verzichtet, die für das Beispiel nicht relevant sind. G_1 enthält unter anderem alle Tupel mit ungerader und G_2 alle mit gerader Nummer. G_3 enthält Tupel 0 und ebenfalls alle ungeraden Tupel, während in G_4 analog 0 und alle geraden Tupel vorkommen. Alle anderen Tupellabel seien jeweils verschieden, das heißt insbesondere, dass alle in G_1 nicht angegebenen Tupellabel nicht in G_3 vorkommen (analog für G_2 und G_4).

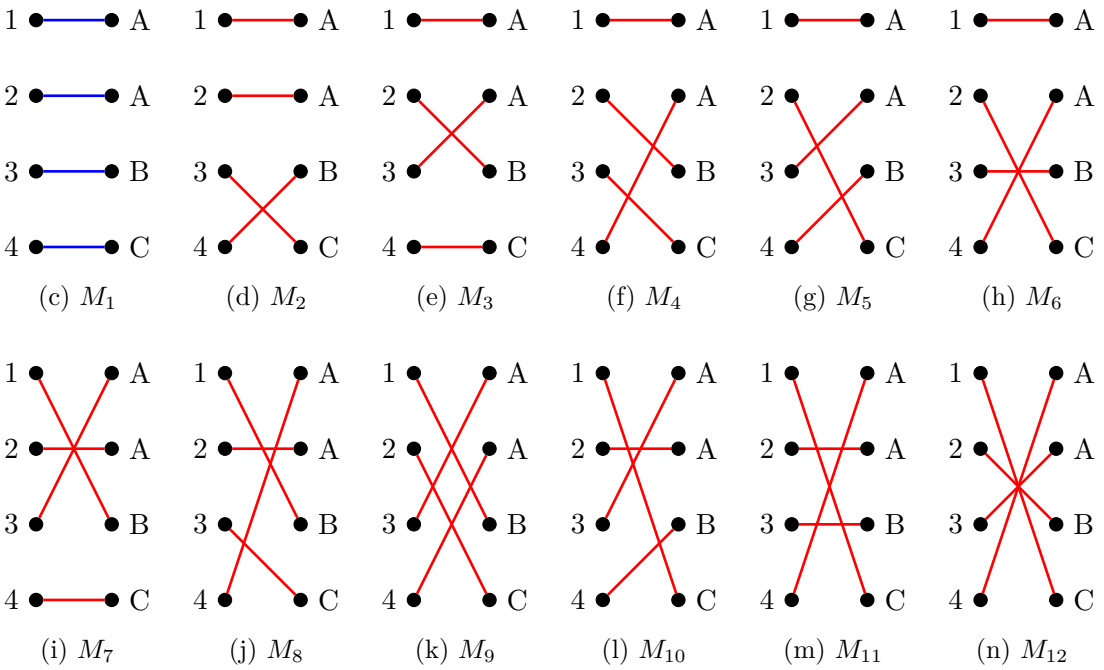
In (f) bis (i) ist das Matching $\mathcal{M}_1$ dargestellt, bei dem Tupel 0 mit SA-Wert A_1 matcht. Die dazugehörige SA-Werteverteilung ist in (a) abzulesen. Aus Symmetriegründen gibt es

ID	Name	SA	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
1	Alison	A	A	A	A	A	A	A	B	B	B	C	C	C
2	Ben	A	A	A	B	B	C	C	A	A	C	A	A	B
3	Clark	B	B	C	A	C	A	B	A	C	A	A	B	A
4	Debra	C	C	B	C	A	B	A	C	A	A	B	A	A
Δr bez. M_1			–	2	2	3	3	2	2	3	2	3	2	2
Δ-min. bez. M_1			–	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	–

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) $\mathcal{G}^{(1)}$



(c) M_1 (d) M_2 (e) M_3 (f) M_4 (g) M_5 (h) M_6

(i) M_7 (j) M_8 (k) M_9 (l) M_{10} (m) M_{11} (n) M_{12}

Abbildung A.3: Perfekte Matchings in $\mathcal{G}^{(1)}$

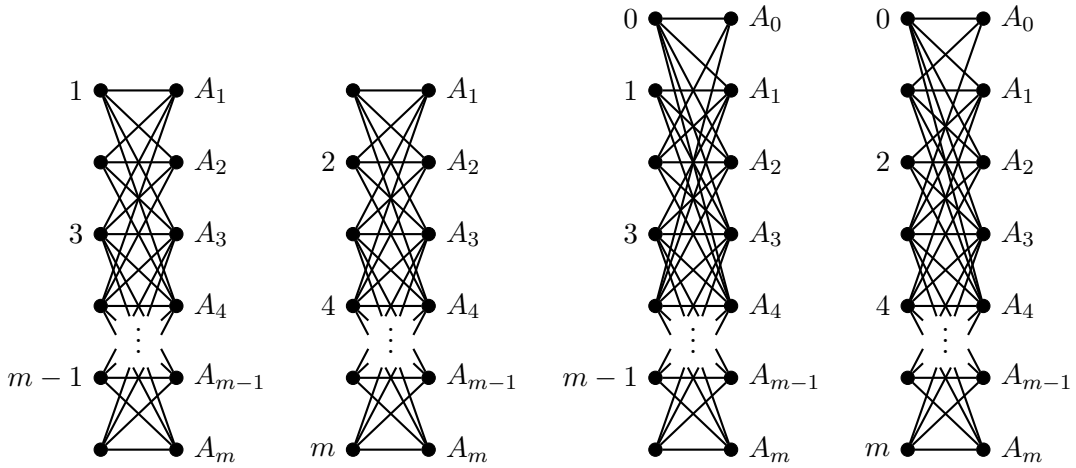
analog zu $\mathcal{M}_1$ für jeden SA-Wert A_i mit $1 \leq i \leq m$ ein Matching $\mathcal{M}_i$, das die Kante $(0, A_i)$ enthält (vgl. Tabelle (a)). Allerdings ist keins dieser Matchings ein $\Delta 2$-Matching bezüglich des Originalmatchings. Dieser Zusammenhang folgt aus der Konstruktion von $\mathcal{G}$. In einem $\Delta 2$-Matching, das die Kante $(0, A_i)$ enthält, müsste in G_3 oder G_4 jeweils auch die Gegenkante (i, A_0) enthalten sein. Da in den beiden Graphen G_1 und G_2 aber jeweils kein SA-Wert A_0 vorkommt, können die Tupel $1, 2, \ldots, m$ nicht mit A_0 matchen und diese Gegenkante kann nicht existieren.

Die Schlussfolgerung aus diesem Beispiel ist, dass die Beschränkung auf $\Delta 2$-Matchings dazu führen kann, dass einem Tupel nur noch sein originaler SA-Wert zugeordnet werden kann, obwohl m Werte möglich wären. Demzufolge kann die Güte $\frac{m}{1}$ der Approximation beliebig schlecht werden.

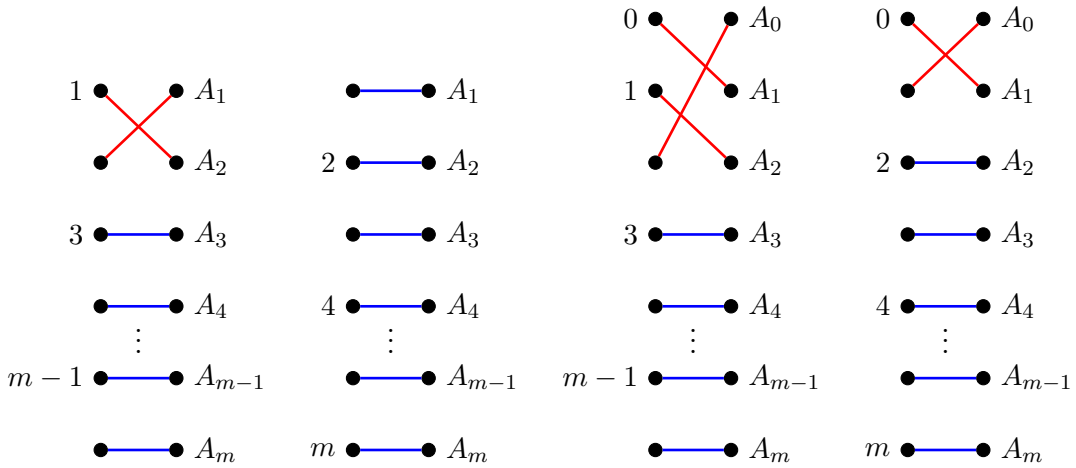
Als Verallgemeinerung der Erkenntnisse aus dem vorigen Beispiel folgt nun, dass die Beschränkung auf Δr-Matchings für jedes beliebige, aber feste r zu einer Approximation führt, für keine feste Güte angegeben werden kann.

ID	SA	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\cdots$	$\mathcal{M}_m$
0	A_0	A_1	A_2	A_3	A_4	$\cdots$	A_m
1	A_1	A_2	A_1	A_1	A_1	$\cdots$	A_1
2	A_2	A_2	A_3	A_2	A_2	$\cdots$	A_2
3	A_3	A_3	A_3	A_4	A_3	$\cdots$	A_3
4	A_4	A_4	A_4	A_4	A_5	$\cdots$	A_4
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$m-1$	A_{m-1}	A_{m-1}	A_{m-1}	A_{m-1}	A_{m-1}	$\cdots$	A_{m-1}
m	A_m	A_m	A_m	A_m	A_m	$\cdots$	A_1

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) G_1 (c) G_2 (d) G_3 (e) G_4

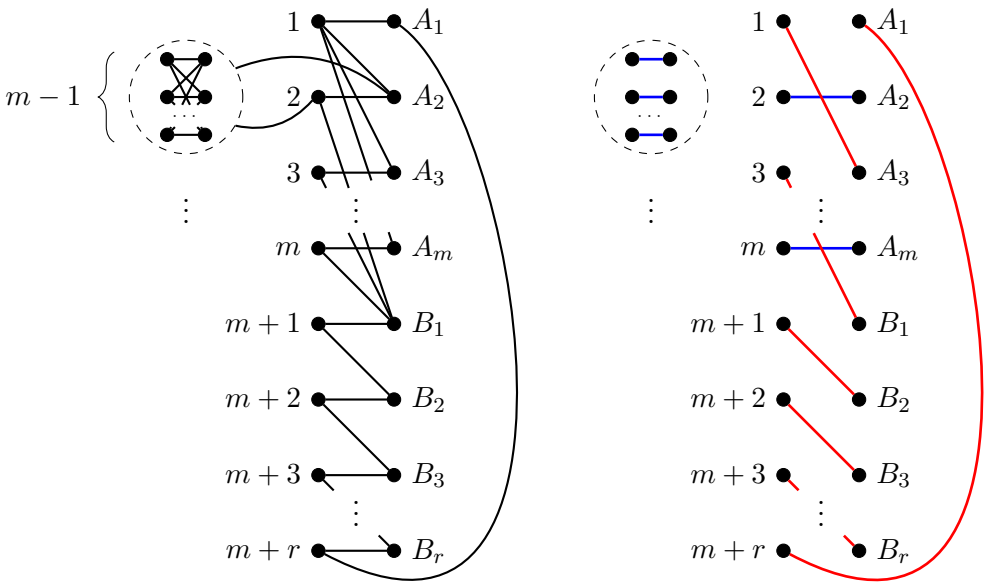


(f) M_1 (g) M_2 (h) M_3 (i) M_4

Abbildung A.4: Für den Anfragegraphen $\mathcal{G} = (G_1, G_2, G_3, G_4)$ in (b) bis (e) ist in (f) bis (i) ein Matching $\mathcal{M}_1 = (M_1, M_2, M_3, M_4)$ dargestellt, das die Kante $(0, A_1)$ enthält, aber kein $\Delta 2$-Matching ist.

ID	SA	M_2	M_3	$\cdots$	M_m	$\cdots$
1	A_1	A_2	A_3	$\cdots$	A_m	$\cdots$
2	A_2	B_1	A_2	$\cdots$	A_2	$\cdots$
3	A_3	A_3	B_1	$\cdots$	A_3	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$
m	A_m	A_m	A_m	$\cdots$	B_1	$\cdots$
$m+1$	B_1	B_2	B_2	$\cdots$	B_2	$\cdots$
$m+2$	B_2	B_3	B_3	$\cdots$	B_3	$\cdots$
$m+3$	B_3	B_4	B_4	$\cdots$	B_4	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$
$m+r$	B_r	A_1	A_1	$\cdots$	A_1	$\cdots$

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) G_n (c) M_3

Abbildung A.5: Anfragegraph ohne Δr-Matching für Tupel 1

Beispiel A.4 Seien m und r zwei beliebige, aber feste natürliche Zahlen. In Abbildung A.5 ist in (b) ein Anfragegraph G_n gegeben. Es sei vereinbart, dass weitere Anfragegraphen $G_1, \ldots, G_{n-1}$ existieren, die hier nicht dargestellt sind. G_n besteht zunächst aus je $m+r$ Tupel- und SA-Knoten, wobei die Tupelknoten mit 1 bis $m+r$ und die SA-Knoten mit A_1 bis A_m sowie B_1 bis B_r gelabelt sind. Alle Tupel $i \in [1, m]$ haben den originalen SA-Wert A_i und alle Tupel $m+j$ mit $j \in [1, r]$ den originalen SA-Wert B_j (vgl. Spalte SA in (a)). Weiterhin existieren Kanten zwischen 1 und allen A_i, allen i und B_1 sowie jeweils $m+j'$ mit $j' \in [1, r-1]$ und $B_{j'+1}$. Den letzten Knoten $m+r$ verbindet eine Kante mit A_1. Damit jeder Knoten mindestens m Nachbarn hat, existiert für jeden Tupelknoten $k \in [2, m+r]$ ein vollständig bipartiter Graph mit jeweils $m-1$ Tupel- und SA-Knoten, die zusammen mit

k einen vollständig bipartiten Graphen der Größe m bilden.[1] Für $k = 2$ ist dieser Graph in der Abbildung skizziert, für die anderen Knoten aus Gründen der Übersichtlichkeit weggelassen. Offensichtlich ist der Anonymitätsgrad jedes Tupels k mindestens m.

In einem validen Matching M_i in G_n, welches die Kante $(1, A_i)$ mit $i \in [2, m]$ enthält, muss i mit B_1 matchen. Denn wenn i mit einem Knoten „seines“ vollständig bipartiten Graphen matchen würde, müsste auch ein Knoten dieses Graphen mit A_i matchen. Aus dem gleichen Grund enthält M_i die Kanten $(m + j', B_{j'+1})$ und die Kante $(m + r, A_1)$. Für jedes i ist das entstehende Matching ein $\Delta(r + 2)$-Matching bezüglich des Originalmatchings und es existiert kein valides Δr-Matching, das die Kante $(1, A_i)$ enthält. (c) zeigt beispielhaft das Matching M_3.

Werden demnach nur Δr-Matchings betrachtet, würde Tupel 1 nur noch sein originaler Wert A_1 zugeordnet werden, obwohl m verschiedene Werte A_1 bis A_m möglich wären. Da für jedes m und jedes r ein solcher Graph G_n angegeben werden kann, zeigt dieses Beispiel, dass die Beschränkung auf Δr-Matchings für jedes beliebige, aber feste r zu einer Approximation führt, deren Güte $\frac{m}{1}$ beliebig schlecht werden kann.

Proposition 4.10 auf Seite 117 gilt nicht für Δr-Matchings mit $r > 2$, was durch das folgende Beispiel illustriert wird.

Beispiel A.5 In Abbildung A.6 ist ein Anfragegraph $\mathcal{G} = (G_1, G_2, G_3, G_4)$, das Originalmatching $\mathcal{M}_{\text{orig}}$ und ein Matching $\mathcal{M}_1$ dargestellt, welches Δ-minimal und ein $\Delta 3$-Matching bezüglich $\mathcal{M}_{\text{orig}}$ ist. In der symmetrischen Differenz der beiden Matchings kommen insgesamt die sechs SA-Labels A, B, C, D, E und F vor. Nach Proposition 4.10 auf Seite 117 sind in der symmetrischen Differenz von Δ-minimalen Δr-Matchings mit $r = 2$ genau zwei verschiedene SA-Labels enthalten. Da jedes Δr-Matching auch ein $\Delta(r + 1)$-Matching ist, zeigt $\mathcal{G}$, dass diese Proposition nicht für den Fall $r > 2$ gilt. Insbesondere sind die SA-Labels aus G_1 komplett verschieden zu denen aus G_4.

Beispiel A.6 Abbildung A.7 zeigt in (a) alle SA-Werteverteilungen, die aus Δ-top Matchings im Anfragegraphen $\mathcal{G}^{(2)} = (G_1, G_2)$ aus (b) erzeugt werden können. $\mathcal{G}^{(2)}$ ist dabei das im Kapitel 3 eingeführte Beispiel. Zum Originalmatching, welches in (c) mit $\mathcal{M}_{1.1}$ angegeben ist, existieren insgesamt sieben Δ-top Matchings, die in (d) bis (j) dargestellt sind. Die Bezeichnung der Matchings entspricht der Nummerierung, die im Kapitel 3 verwendet wurde (vgl. Tabelle 3.8 auf Seite 72).

Beispiel A.7 Abbildung A.8 zeigt in (a) alle SA-Werteverteilungen, die aus Δ-top Matchings im Anfragegraphen $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$ aus (b) entstehen. $\mathcal{G}^{(3)}$ ist dabei das im Kapitel 3 eingeführte Beispiel. Zum Originalmatching, welches in (c) mit $\mathcal{M}_{1.1}$ angegeben ist, existieren insgesamt vier Δ-top Matchings, die in (d) bis (g) dargestellt sind. Die Bezeichnung der Matchings entspricht der Nummerierung, die im Kapitel 3 verwendet wurde (vgl. Tabelle 3.10 auf Seite 73).

A.3 Erweiterung von Matchings

In Kapitel 5.4 wurden zwei Einschränkungen definiert, die die Anzahl möglicher Erweiterungen beschränken. Einerseits sollen neue Tupel nur mit alten erweitert werden und

[1] Das bedeutet, der Tupelknoten k ist mit jedem SA-Knoten „seines“ vollständig bipartiten Graphen verbunden und jeder Tupelknoten des Graphen mit dem originalen SA-Wert von k.

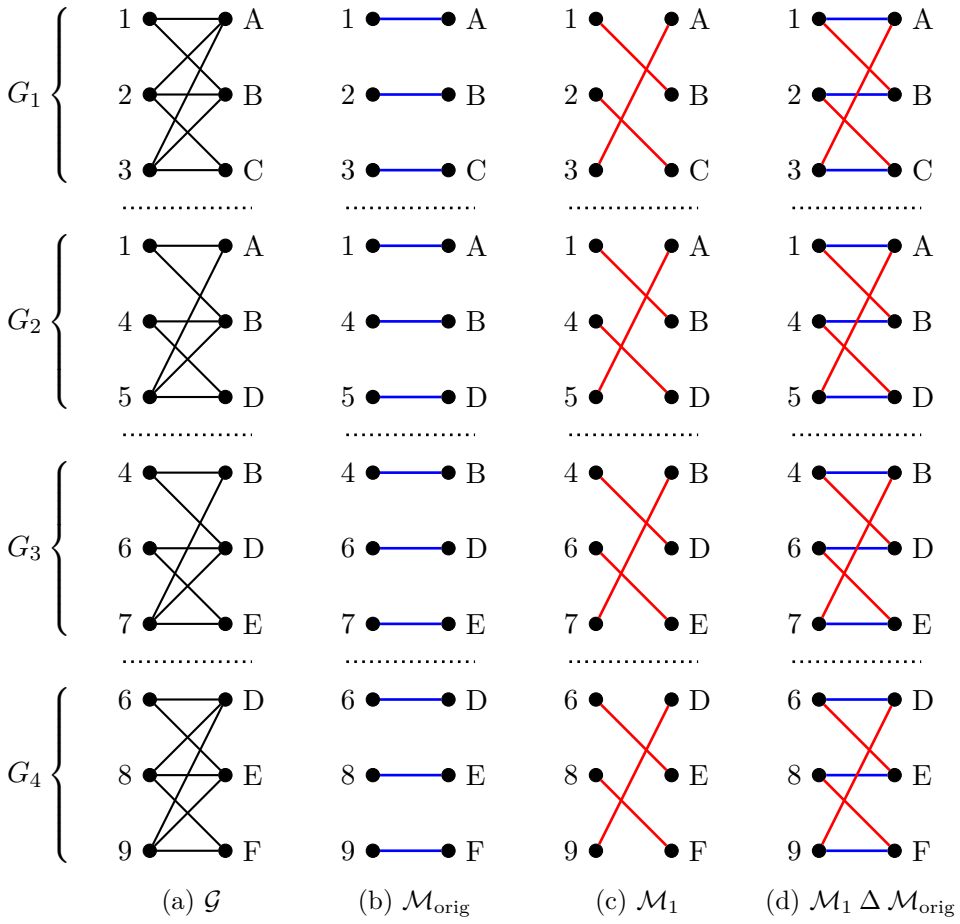


Abbildung A.6: Δ-minimale Δ3-Matchings für $\mathcal{G} = (G_1, G_2, G_3, G_4)$

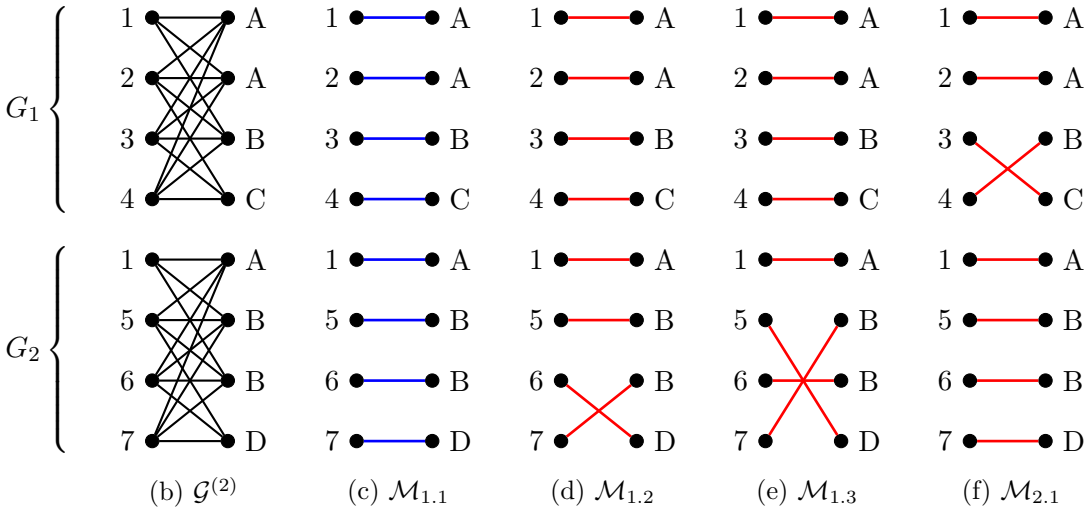
andererseits dürfen alte Tupel, die denselben SA-Wert haben, nicht mit demselben neuen Tupel erweitert werden. An dieser Stelle werden beide Kriterien mithilfe von Beispielen weiter untersucht.

Beispiel A.8 Abbildung A.9 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G}^{(4)}$ und in (e) die entsprechende Δ-top Matchingtabelle. In Spalte $\mathcal{G}^{(3)}$ sind die Δ-top Matchingkanten der Matchings in den ersten drei Graphen aufgelistet. Dabei handelt es sich um die beiden Matchings $\mathcal{M}_1^{(3)}$ und $\mathcal{M}_2^{(3)}$, die auch in (b) beziehungsweise (c) dargestellt sind.

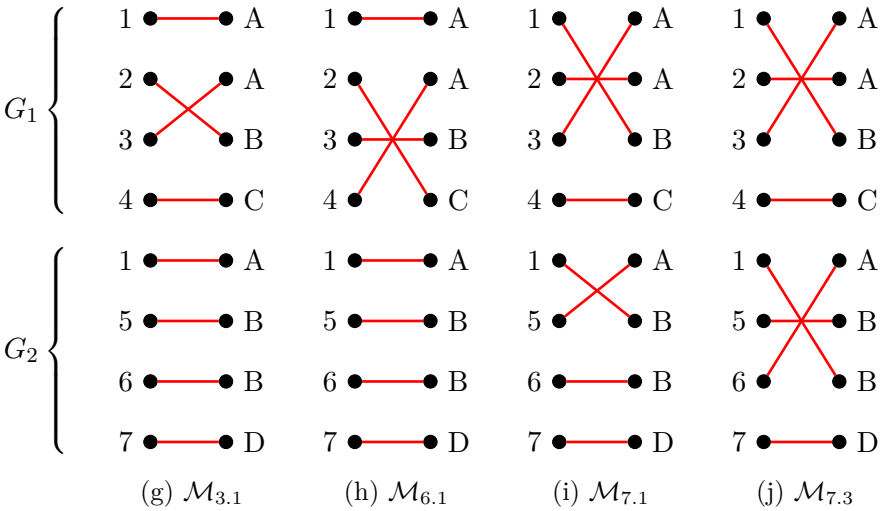
Der Graph G_4 besteht aus den alten Tupeln 1, 3 und 5, dem neuen Tupel 6 sowie den angegebenen SA-Werten. Sind Erweiterungen auch mit alten Tupeln zugelassen, können die Kante $(1, \mathrm{B}(2))$ mit B(4) und die Kante $(4, \mathrm{A}(3))$ mit A(1) erweitert werden. Das dazugehörige Matching $\mathcal{M}_{1+2}^{(4)}$ ist in (d) abgebildet und erfüllt die Δ-top-Eigenschaften. Es entsteht durch ein Zusammenfügen der hier rot dargestellten Kanten. Das sind genau die, die in den jeweiligen Matchings verschieden vom Originalmatching sind. Zwei Matchings können demnach zusammengefügt werden, wenn ihre Kantenmengen disjunkt sind, wobei nur vom Originalmatching verschiedene Kanten betrachtet werden. Daneben muss die Δ-top-Eigenschaft auch im neuen Graphen gelten. Wenn beispielsweise Tupel 6 in G_4 fehlen

ID	Name	SA	$\mathcal{M}_{1.1}$	$\mathcal{M}_{1.2}$	$\mathcal{M}_{1.3}$	$\mathcal{M}_{2.1}$	$\mathcal{M}_{3.1}$	$\mathcal{M}_{6.1}$	$\mathcal{M}_{7.1}$	$\mathcal{M}_{7.3}$
1	Alison	A	A	A	A	A	A	A	B	B
2	Ben	A	A	A	A	A	B	C	A	A
3	Clark	B	B	B	B	C	A	B	A	A
4	Debra	C	C	C	C	B	C	A	C	C
5	Elaine	B	B	B	D	B	B	B	A	B
6	Fiona	B	B	D	B	B	B	B	B	A
7	Gary	D	D	B	B	D	D	D	D	D

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) $\mathcal{G}^{(2)}$ (c) $\mathcal{M}_{1.1}$ (d) $\mathcal{M}_{1.2}$ (e) $\mathcal{M}_{1.3}$ (f) $\mathcal{M}_{2.1}$

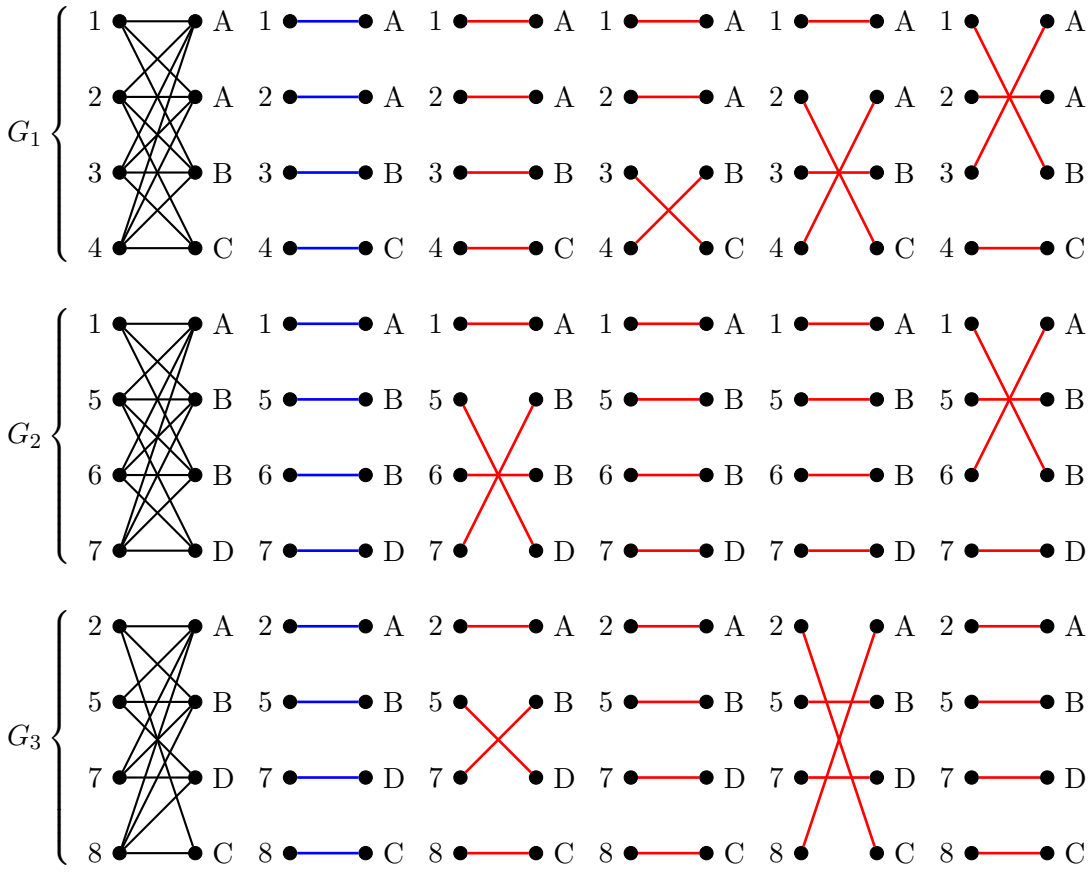


(g) $\mathcal{M}_{3.1}$ (h) $\mathcal{M}_{6.1}$ (i) $\mathcal{M}_{7.1}$ (j) $\mathcal{M}_{7.3}$

Abbildung A.7: Originalmatching und alle Δ-top Matchings in $\mathcal{G}^{(2)}$

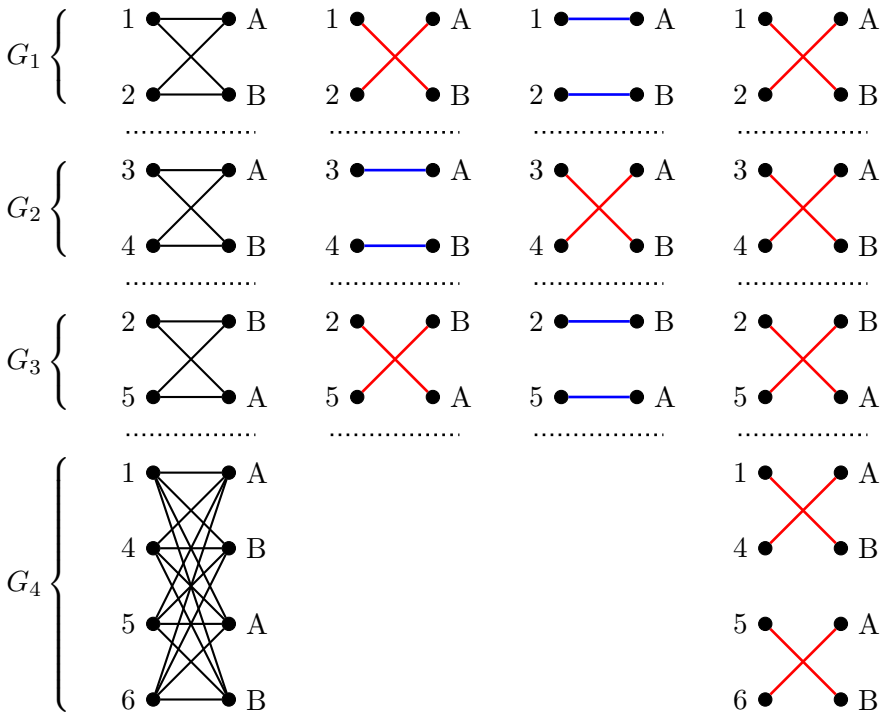
ID	Name	SA	$\mathcal{M}_{1.1}$	$\mathcal{M}_{1.3}$	$\mathcal{M}_{2.1}$	$\mathcal{M}_{6.1}$	$\mathcal{M}_{7.3}$
1	Alison	A	A	A	A	A	B
2	Ben	A	A	A	A	C	A
3	Clark	B	B	B	C	B	A
4	Debra	C	C	C	B	A	C
5	Elaine	B	B	D	B	B	B
6	Fiona	B	B	B	B	B	A
7	Gary	D	D	B	D	D	D
8	Helen	C	C	C	C	A	C

(a) Datentabelle und SA-Werteverteilungen der Matchings



(b) $\mathcal{G}^{(3)}$ (c) $\mathcal{M}_{1.1}$ (d) $\mathcal{M}_{1.3}$ (e) $\mathcal{M}_{2.1}$ (f) $\mathcal{M}_{6.1}$ (g) $\mathcal{M}_{7.3}$

Abbildung A.8: Originalmatching und alle Δ-top Matchings in $\mathcal{G}^{(3)}$



(a) $\mathcal{G}^{(4)}$ (b) $\mathcal{M}_1^{(3)}$ (c) $\mathcal{M}_2^{(3)}$ (d) $\mathcal{M}_{1+2}^{(4)}$

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}^{(4)}/\mathcal{M}_{1+2}^{(4)}$
1	A	B(2)	B(2)	B(2)	B(2, 4)
2	B	A(1)	A(1)	A(1, 5)	A(1, 5)
3	A		B(4)	B(4)	B(4)
4	B		A(3)	A(3)	A(1, 3)
5	A			B(2)	B(2, 6)
6	B				A(5)

(e) Δ-top Matchingtabelle

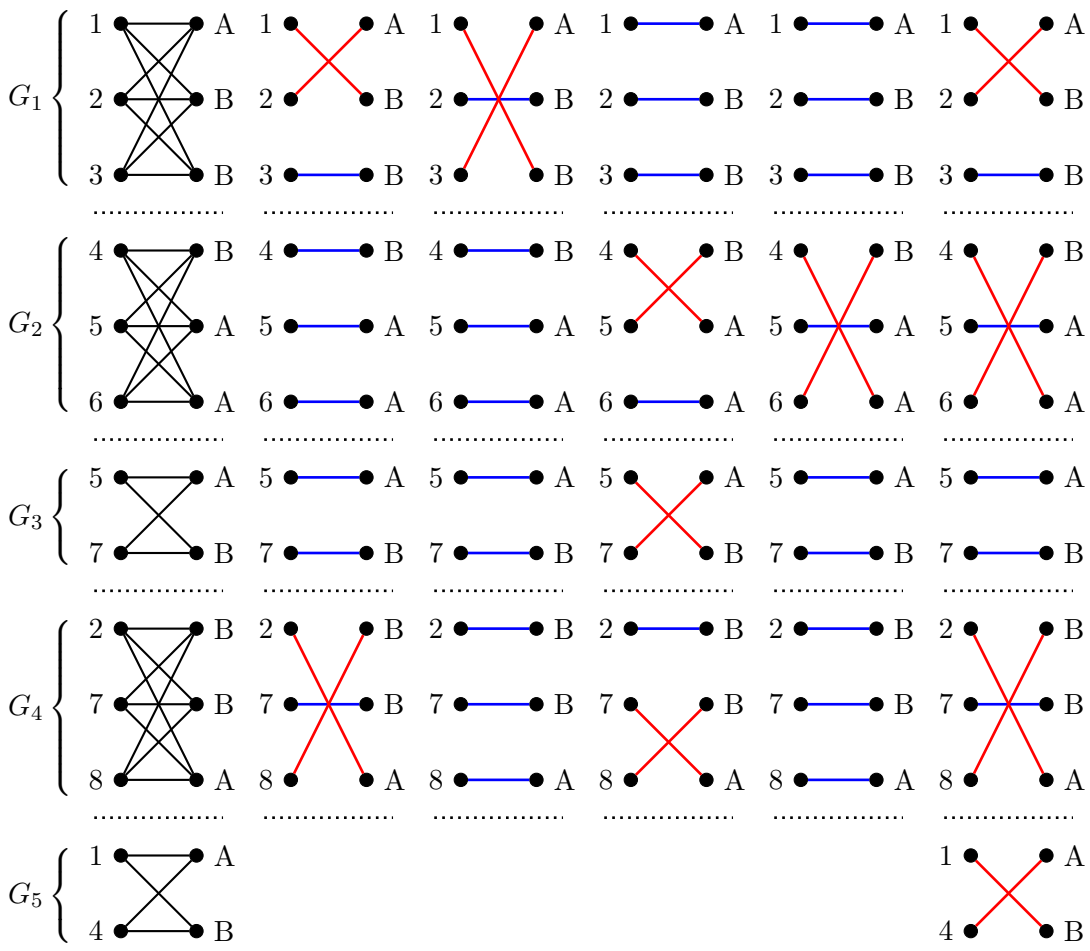
Abbildung A.9: Matchings in $\mathcal{G}^{(3)}$ und $\mathcal{G}^{(4)}$

würde, würde die Erweiterung nicht funktionieren, denn Tupel 5 kann dann nicht mit einem B matchen.

Es wäre auch denkbar, dass die beiden zu vereinigenden Matchings Kanten im selben Graphen besitzen, die keine Originalkanten sind. Zum Beispiel könnte G_2 und G_3 auch genau ein Graph sein, der aus allen vier Tupel- und SA-Knoten besteht. In diesem Fall gäbe es noch mehr mögliche Matchings.

Beispiel A.9 Abbildung A.10 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G}^{(5)}$ und in (g) die dazugehörige Δ-top Matchingtabelle. In Spalte $\mathcal{G}^{(4)}$ sind die Δ-top Matchingkanten der Matchings in den ersten vier Graphen aufgelistet. Dabei handelt es sich um die vier Matchings $\mathcal{M}_1^{(4)}, \mathcal{M}_2^{(4)}, \mathcal{M}_3^{(4)}$ und $\mathcal{M}_4^{(3)}$, die auch in (b) bei (e) dargestellt sind.

Sollen diese Matchings in $\mathcal{G}^{(5)}$ erweitert werden, muss Tupel 1 mit B und Tupel 4 mit A matchen. Es gibt insgesamt je zwei Matchings mit einer Kante $(1, \mathrm{B})$ beziehungsweise



(a) $\mathcal{G}^{(5)}$ (b) $\mathcal{M}_1^{(4)}$ (c) $\mathcal{M}_2^{(4)}$ (d) $\mathcal{M}_3^{(4)}$ (e) $\mathcal{M}_4^{(4)}$ (f) $\mathcal{M}_{1+4}^{(5)}$

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}^{(4)}$	$\mathcal{G}^{(5)}$
1	A	B(2), B(3)	B(2), B(3)	B(2), B(3)	B(2), B(3)	B(2, 4), B(3, 4)
2	B	A(1)	A(1)	A(1)	A(1, 8)	A(1, 8)
3	B	A(1)	A(1)	A(1)	A(1)	A(1)
4	B		A(5), A(6)	A(5), A(6)	A(5), A(6)	A(1, 5), A(1, 6)
5	A		B(4)	B(4, 7)	B(4, 7)	B(4, 7)
6	A		B(4)	B(4)	B(4)	B(4)
7	B			A(5)	A(5, 8)	A(5, 8)
8	A				B(2), B(7)	B(2), B(7)

(g) Δ-top Matchingtabelle

Abbildung A.10: Matchings in $\mathcal{G}^{(4)}$ und $\mathcal{G}^{(5)}$

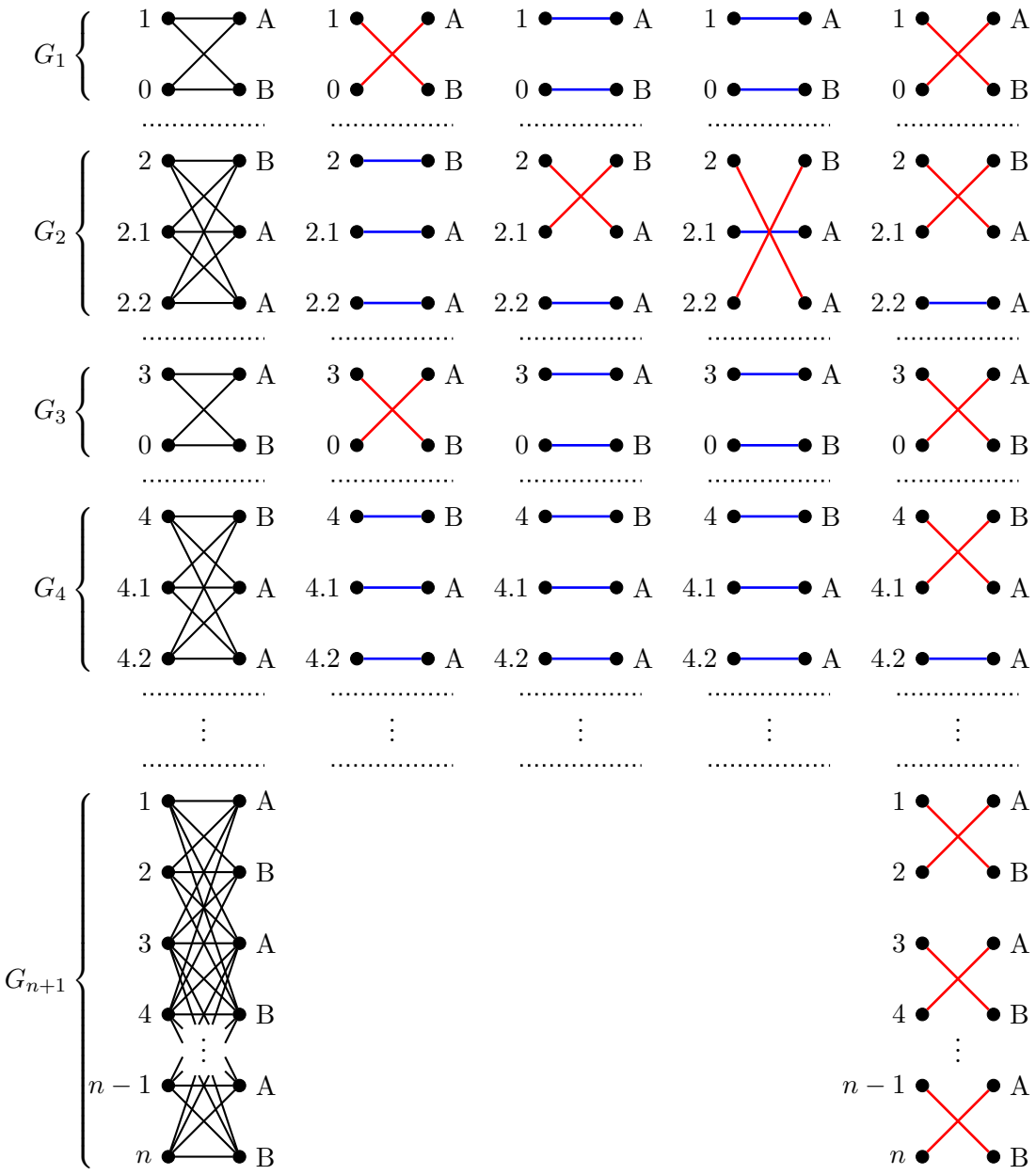
$(4, \mathrm{A})$. Das heißt, es gibt vier mögliche Erweiterungen, die betrachtet werden müssen: das Zusammenfügen von Matching $\mathcal{M}_1^{(4)}$ und $\mathcal{M}_3^{(4)}$, $\mathcal{M}_1^{(4)}$ und $\mathcal{M}_4^{(4)}$, $\mathcal{M}_2^{(4)}$ und $\mathcal{M}_3^{(4)}$ sowie $\mathcal{M}_2^{(4)}$ und $\mathcal{M}_4^{(4)}$. Offensichtlich verhindern die Graphen G_3 und G_4, dass Matchings $\mathcal{M}_1^{(4)}$ und $\mathcal{M}_3^{(4)}$ zusammengelegt werden können. (f) zeigt das Matching $\mathcal{M}_{1+4}^{(5)}$, welches durch das Vereinigen von $\mathcal{M}_1^{(4)}$ und $\mathcal{M}_4^{(4)}$ entsteht. Auch die anderen Kombinationen $\mathcal{M}_2^{(4)}$ und $\mathcal{M}_3^{(4)}$ sowie $\mathcal{M}_2^{(4)}$ und $\mathcal{M}_4^{(4)}$ führen jeweils zu einem Δ-top Matching, welches allerdings hier nicht graphisch dargestellt ist. Somit werden in der Δ-top Matchingtabelle drei Matchings für $\mathcal{G}^{(5)}$ gespeichert.

Dieses Beispiel zeigt, dass alle Anfragegraphen getestet werden müssen, um zu sehen, ob das Zusammenfügen bestimmter Matchings funktioniert. Gibt es nämlich weitere Graphen wie G_3 und G_4, könnten weitere der hier funktionierenden Kombinationen wegfallen. Nur mithilfe der Δ-top Matchingkanten in der Tabelle ist es somit nicht möglich zu entscheiden, ob ein altes Tupel mit einem anderen alten Tupel erweitert werden kann.

Beispiel A.10 In Abbildung A.11 ist ein Teilausschnitt einer komplexeren Folge von Anfragegraphen skizziert. Insgesamt sind $n + 1$ Graphen gegeben. Für alle ungeraden Zahlen $u \leq n$ existiert ein Graph G_u, der analog zu G_1 beziehungsweise G_3 aufgebaut ist. Für alle geraden Zahlen $g \leq n$ existiert ein Graph G_g, der analog zu G_2 beziehungsweise G_4 aufgebaut ist. Damit gibt es in $\mathcal{G}^{(n)}$ ein Matching $\mathcal{M}_1^{(n)}$, bei dem alle ungeraden Labels mit einem B matchen (siehe (b)), und für jede gerade Zahl g jeweils zwei Matchings $\mathcal{M}_{g.1}^{(n)}$ und $\mathcal{M}_{g.2}^{(n)}$, bei denen der Knoten mit Label g mit A matcht (analog zu $\mathcal{M}_{2.1}^{(n)}$ aus (c) und $\mathcal{M}_{2.2}^{(n)}$ aus (d)).

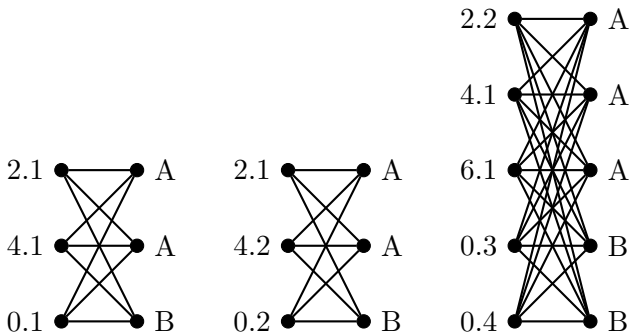
Der Graph G_{n+1} besteht aus Tupelknoten für alle Labels $i \leq n$ und SA-Knoten mit Labels A oder B. Wenn nun das Matching $\mathcal{M}_1^{(n)}$ in $\mathcal{G}^{(n+1)}$ erweitert werden soll, muss jedem Tupelknoten mit ungeradem Label ein B (folgt aus $\mathcal{M}_1^{(n)}$) und demzufolge jedem Tupelknoten mit geradem Label ein A zugeordnet werden. Dafür gibt es insgesamt $\frac{n}{2}!$, also exponentiell viele verschiedene Kombinationen in G_{n+1}, wobei für jede Zuordnung eines geraden Labels g zu A noch einmal zwei verschiedene Matchings existieren ($\mathcal{M}_{g.1}^{(n)}$ und $\mathcal{M}_{g.2}^{(n)}$). Eines dieser Matchings $\mathcal{M}_{1+2.1+4.1}^{(n+1)}$ ist in (e) angegeben.

Anstatt alle Matchings zu speichern, kann man sich überlegen, nur eines auszuwählen, da zumindest die Zuordnung der ungeraden Labels zu B und der geraden Labels zu A bei allen gleich ist. Ignoriert man dazu als Vereinfachung die Zuordnungen in G_{n+1}, so gibt es immer noch für jede gerade Zahl $g \leq n$ zwei verschiedene Matchings, die infrage kommen. Daraus folgt, dass es insgesamt $2^{n/2}$ verschiedene SA-Werteverteilungen gibt. Für den Graphen $\mathcal{G}^{(n+1)}$ aus Abbildung (a) sind alle dieser Matchings Δ-top, daher kann hier einfach ein beliebiges ausgewählt werden. Sind allerdings noch weitere Graphen gegeben, wie zum Beispiel in Abbildung A.12, fallen bestimmte Möglichkeiten weg. Beispielsweise verhindert $G_{0.1}$ aus (a), dass die Matchings $\mathcal{M}_{2.1}^{(n)}$ und $\mathcal{M}_{4.1}^{(n)}$ zusammengefasst werden können, da hierbei die Labels 2.1 und 4.1 jeweils mit B matchen würden. Gleiches gilt für $G_{0.2}$ aus (b) sowie $\mathcal{M}_{2.1}^{(n)}$ und $\mathcal{M}_{4.2}^{(n)}$. Somit kann $\mathcal{M}_{2.1}^{(n)}$ nicht auf $\mathcal{G}^{(n+1)}$ erweitert werden. Weitere Graphen wie $G_{0.3}$ aus (c) verhindern die Kombination von $\mathcal{M}_{2.2}^{(n)}$, $\mathcal{M}_{4.1}^{(n)}$ und $\mathcal{M}_{6.1}^{(n)}$ gleichzeitig. Im schlimmsten Fall funktioniert gar keine Kombination der gegebenen Mat-



(a) $\mathcal{G}^{(n+1)}$ (b) $\mathcal{M}_1^{(n)}$ (c) $\mathcal{M}_{2.1}^{(n)}$ (d) $\mathcal{M}_{2.2}^{(n)}$ (e) $\mathcal{M}_{1+2.1+4.1}^{(n+1)}$

Abbildung A.11: Matchings in $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$



(a) $G_{0.1}$ (b) $G_{0.2}$ (c) $G_{0.3}$

Abbildung A.12: Mögliche zusätzliche Anfragegraphen zu $\mathcal{G}^{(n+1)}$ aus Abbildung A.11

chings für $\mathcal{G}^{(n)}$, was aber erst nach $2^{n/2}$ Schritten festgestellt wird.[2] Um diesen Aufwand zu umgehen, wurden im Kapitel 5.4 Einschränkungen definiert, die verhindern, dass Matchings zusammengefasst werden. Insbesondere werden alte Tupel nicht mit anderen alten Tupeln erweitert.

Beispiel A.11 Abbildung A.13 zeigt in (a) eine Folge von Anfragegraphen $\mathcal{G}^{(4)}$ und in (d) die dazugehörige Δ-top Matchingtabelle. In Spalte $\mathcal{G}^{(3)}$ sind die Δ-top Matchingkanten des einzigen Matchings $\mathcal{M}_1^{(3)}$ in den ersten drei Graphen aufgelistet.

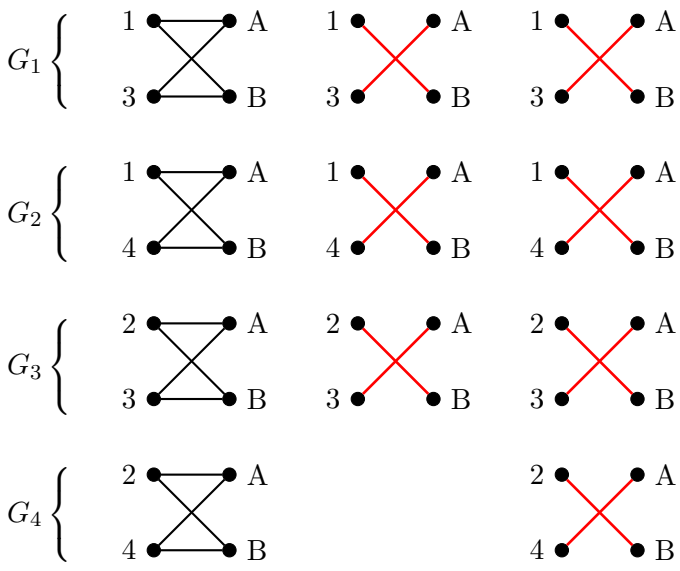
Der Graph G_4 besteht aus den beiden alten Tupeln 2 und 4 sowie den angegebenen SA-Werten. Durch die in Kapitel 5.4 vorgestellten Einschränkungen werden alte Tupel nicht mit anderen alten Tupeln erweitert. Daher würden die Kanten $(2, \mathrm{B}(3))$ und $(4, \mathrm{A}(1))$ und somit das gesamte Matching $\mathcal{M}_1^{(3)}$ gelöscht werden. Keinem Tupel kann nunmehr ein anderer Wert als sein originaler zugeordnet werden (siehe Spalte $\mathcal{G}^{(4)}$) und es würde eine Verletzung der Privatsphäre erkannt werden.

Werden Erweiterungen mit alten Tupeln zugelassen, kann 2 mit 4 und 4 mit 2 erweitert werden. Es entstehen die Kanten $(2, \mathrm{B}(3, 4))$ und $(4, \mathrm{A}(1, 2))$, die zum in (c) dargestellten Δ-top Matching $\mathcal{M}_1^{(4)}$ gehören. Dieses Beispiel zeigt, dass durch die erste Einschränkung nicht mehr alle Δ-top Matchings modelliert werden und dadurch SA-Werteverteilungen fehlen können. Laut der Modellierung wird eine Verletzung der Privatsphäre erkannt, obwohl das nicht der Fall ist (false positive).

Es folgt ein komplexes Beispiel, das Algorithmus 5.3 in Verbindung mit den in Kapitel 5.6 vorgestellten Methoden zur Sicherstellung von k-assign Anonymität durch das Hinzufügen künstlicher SA-Knoten veranschaulicht.

Beispiel A.12 In Abbildung A.14 ist in (f) eine Δ-top Matchingtabelle für eine Folge von $n-1$ Anfragegraphen (Spalte $\mathcal{G}^{(n-1)}$) und in (a) ein weiterer Anfragegraph G_n gegeben. Die Tupel 1 bis 8 seien alte Tupel, die in den hier nicht dargestellten Teilgraphen G_1 bis G_{n-1} vorkommen, und 9 sei ein neues Tupel, für das noch keine Matchings existieren. Alle

[2] Es sei hier vereinfacht angenommen, dass der Test der Matchingeigenschaft einer durch das Zusammenfassen mehrerer Matchings entstehenden Knotenmenge in einem Schritt erfolgen kann. Weiterhin soll keine (oder nur eine vernachlässigbar kleine Anzahl) der $2^{n/2}$ vielen möglichen Kombinationen von vornherein ausgeschlossen werden können.



(a) $\mathcal{G}^{(4)}$ (b) $\mathcal{M}_1^{(3)}$ (c) $\mathcal{M}_1^{(4)}$

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}^{(4)}$
1	A	B(3)	B(3, 5)	B(3, 5)	–
2	A			B(3)	–
3	B	A(1)	A(1)	A(1, 2)	–
4	B		A(1)	A(1)	–

(d) Δ-top Matchingtabelle

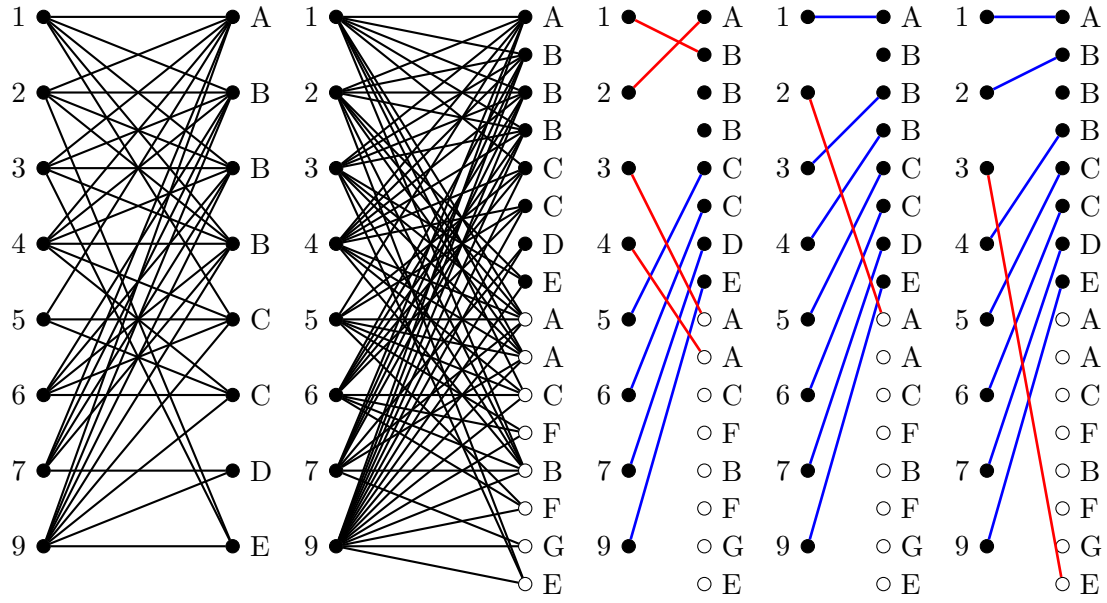
Abbildung A.13: Matchings in $\mathcal{G}^{(3)} = (G_1, G_2, G_3)$

aufgeführten Variablen $x_{i.j}$ stehen für Mengen (bzw. Listen) beliebiger Tupel, die hier nicht angegeben sind. Entsprechend Algorithmus 5.3 sollen die Matchings aus der Tabelle erweitert werden, wobei 3-assign Anonymität gefordert ist, welche durch das Hinzufügen von künstlichen SA-Knoten sichergestellt werden soll.[3] Es sei angenommen, dass die Δ-top Matchingkanten entsprechend der Auflistung in der Tabelle von oben nach unten bearbeitet werden.

Die erste Matchingkante $(1, \mathrm{B}(2, 3, 4))$ führt nach Zeile 12 des Algorithmus zu einem Aufruf der erweiterten *Lösche*-Routine, die bei Bedarf künstliche SA-Knoten in den Graphen einfügt. Wenn die Kante entfernt würde, wäre der Anonymitätsgrad des Tupels 1 nur noch $2 < k = 3$. Da die Tupel 2, 3 und 4 in G_n vorkommen, müssen insgesamt zwei künstliche SA-Knoten mit Label A eingefügt werden, um ein Löschen zu verhindern. Ein mögliches Matching M_1, das die Kante $(1, \mathrm{B}(2, 3, 4))$ im veränderten Graphen G'_n enthält, ist in (c) dargestellt.[4]

Die nächsten beiden Kanten $(1, \mathrm{C}(5))$ und $(2, \mathrm{A}(1))$ bleiben erhalten (Zeile 10). Kante $(2, \mathrm{A}(x_{2.1}))$ könnte zwar gelöscht werden ohne 3-assign Anonymität zu verletzen (Zeile 8), wird aber mithilfe der beiden gerade erzeugten künstlichen SA-Knoten erweitert. Ein mög-

[3] Vgl. Kapitel 5.6.
[4] Originalkanten sind blau und alle anderen Kanten rot gekennzeichnet.



(a) G_n (b) G'_n (c) M_1 (d) M_2 (e) M_3

ID	SA	$\mathcal{G}^{(n-1)}$	$\mathcal{G}'^{(n)}$
1	A	B(2, 3, 4), C(5)	B(2, 3, 4), C(5)
2	B	A(1), A($x_{2.1}$), E($x_{2.2}$)	A(1), A($x_{2.1}$), E($x_{2.2}$, 9)
3	B	A(1), E(x_3)	A(1), E(x_3)
4	B	A(1), C(x_4)	A(1), C(x_4)
5	C	A(1), F(x_5)	A(1), F(x_5)
6	C	B($x_{6.1}$), F($x_{6.2}$)	B($x_{6.1}$), F($x_{6.2}$)
7	D	A($x_{7.1}$), B($x_{7.2}$), G(8)	A($x_{7.1}$), B($x_{7.2}$), G(8)
8	G	A(x_8), D(7)	A(x_8), D(7)
9	E		B(2), A(), B(), C(), F(), G()

(f) Δ-top Matchingtabelle

Abbildung A.14: Matchings in G_n

liches Matching ist in (d) dargestellt. Die Kante $(2, \mathrm{E}(x_{2.2}))$ kann im ersten Schritt weder erweitert noch gelöscht werden und wird für den anschließenden Erweiterungsalgorithmus gespeichert.

Die Behandlung der nächsten Kanten verläuft analog. $(3, \mathrm{A}(1))$, $(4, \mathrm{A}(1))$ und $(5, \mathrm{A}(1))$ bleiben erhalten, $(3, \mathrm{E}(x_3))$ wird für die Erweiterung gespeichert und für die Kanten $(4, \mathrm{C}(x_4))$, $(5, \mathrm{F}(x_5))$, $(6, \mathrm{B}(x_{6.1}))$ sowie $(6, \mathrm{F}(x_{6.2}))$ werden jeweils weitere künstliche SA-Knoten mit entsprechendem SA-Wert eingefügt.[5] Dabei ist zum einen zu beachten, dass $(4, \mathrm{C}(x_4))$ nicht mit 5 oder 6 erweitert werden kann, da beide Tupel alt sind (Zeile 18). Zum anderen müssen insgesamt zwei künstliche Knoten mit Label F eingefügt werden, denn die beiden verursachenden Tupel 5 und 6 haben denselben originalen SA-Wert C. Entspre-

[5] Die künstlichen SA-Knoten für $(4, \mathrm{C}(x_4))$ und $(6, \mathrm{B}(x_{6.1}))$ werden jeweils in Zeile 18 sowie die für $(5, \mathrm{F}(x_5))$ und $(6, \mathrm{F}(x_{6.2}))$ jeweils in Zeile 8 erstellt.

chend den in Kapitel 5.4 eingeführten Einschränkungen dürfen sie nicht mit demselben Tupeln beziehungsweise künstlichen SA-Knoten erweitert werden.

Anders verhält es sich bei Tupel 7 und dessen Kanten $(7, \mathrm{A}(x_{7.1}))$ sowie $(7, \mathrm{B}(x_{7.2}))$. Da der originale SA-Wert von 7 das D ist, können diese Kanten mit den bereits eingefügten künstlichen SA-Knoten mit Label A und B erweitert werden (Zeile 18). Demzufolge ist der Anonymitätsgrad von 7 mindestens 3 und die Kante $(7, \mathrm{G}(8))$ könnte gelöscht werden (Zeile 8). Das hätte zur Folge, dass auch die Gegenkante $(8, \mathrm{D}(7))$ entfernt werden müsste, wodurch der Anonymitätsgrad von Tupel 8 auf $2 < k$ fiele. Somit muss auch $(7, \mathrm{G}(8))$ erweitert werden und es wird ein künstlicher SA-Knoten mit Label G eingefügt.

Als Nächstes wird im Algorithmus 5.3 der Erweiterungsschritt ausgeführt. Da es mit 9 nur ein neues Tupel mit SA-Wert E gibt, kann nur eine der beiden zuvor gespeicherten Kanten erweitert werden. Angenommen, $(2, \mathrm{E}(x_{2.2}))$ wird mit 9 erweitert, so muss nach Zeile 34 auch für die Kante $(3, \mathrm{E}(x_3))$ ein künstlicher SA-Knoten eingefügt werden. Ein Löschen der Kante würde nämlich auch in diesem Fall zu einer Verletzung des Schutzkriteriums führen. M_3 aus (e) ist das entsprechende Matching in G_n.

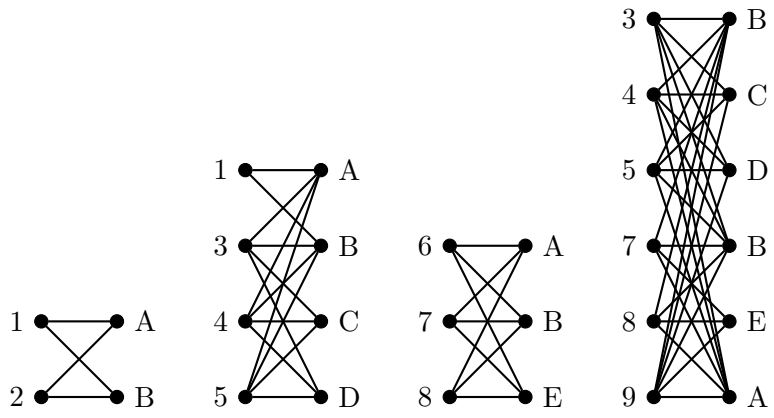
Als Letztes wird das neue Tupel 9 behandelt. Die Kante $(9, \mathrm{B}(2))$ ist bereits durch den Erweiterungsschritt erstellt worden. Da es kein anderes Tupel mit demselben originalen SA-Wert wie 9 gibt, können alle künstliche SA-Werte in die Matchingtabelle übernommen werden. Die Notation $(9, \mathrm{A}())$ gibt beispielsweise an, dass es sich nur um einen künstlichen Matchingpartner handelt.

Der resultierende und um künstliche SA-Knoten erweiterte Anfragegraph ist in (b) dargestellt und mit G'_n bezeichnet.

A.4 Erweiterungsalgorithmen

Beispiel A.13 Abbildung A.15 zeigt eine Folge von vier Anfragegraphen, wobei der vierte Graph G_4 aus den alten Tupeln 3, 4, 5, 7 und 8 sowie dem neuen Tupel 9 besteht. Die Δ-top Matchingtabelle ist in (e) dargestellt. Durch die Einschränkungen beim Erweitern von Matchings kann 9 nur entweder 3 oder 7 zugeordnet werden (da beide Tupel den gleichen SA-Wert B haben). Während 3 bereits 2 Matchingkanten mit C(4) und D(5) hat, hat 7 nur eine mit E(8). In der ersten Variante $v1$ (siehe Spalte $\mathcal{G}^{(4)}_{v1}$) wird daher die Kante $(7, \mathrm{A}(6))$ mit A(9) erweitert, wodurch die Kante $(3, \mathrm{A}(1))$ gelöscht wird. Dadurch fällt aber das Matching mit $(1, \mathrm{B}(2, 3))$ heraus und 1 hat keine weitere Matchingkante mehr. Es entsteht eine Verletzung der Privatsphäre, da 1 nur noch der SA-Wert A zugeordnet werden kann. In der zweiten Variante $v2$ (siehe Spalte $\mathcal{G}^{(4)}_{v2}$) wird $(3, \mathrm{A}(1))$ mit A(9) erweitert. Dadurch wird zwar die Kante $(7, \mathrm{A}(6))$ gelöscht, jedem Tupel können aber weiterhin mindestens zwei verschiedene SA-Werte zugeordnet werden. In diesem Beispiel ist also eine ungleichmäßige Verteilung der neuen Tupel beim Erweitern besser, das heißt, für ein Tupel (hier 3) sollten drei Matchingkanten und für ein anderes (hier 7) nur eine Kante gespeichert werden.

Beispiel A.14 In Abbildung A.16 ist in (a) bis (e) eine Folge von Anfragegraphen $\mathcal{G}^{(5)}$ gegeben. Für den vierten Graphen G_4 gibt es prinzipiell zwei mögliche Erweiterungen. Das neue Tupel 8 kann entweder zu 1 oder zu 4 zugeordnet werden. Wird 1 mit 8 erweitert (siehe Variante $v1$) und ist 2-assign Anonymität gefordert, gibt es keine Verletzung des Schutzkriteriums, da der minimale Anonymitätsgrad 2 ist. Allerdings können im fünften Graphen G_5 die alten Tupel 5 und 7 nicht erweitert werden, da sie nur noch je eine



(a) G_1 (b) G_2 (c) G_3 (d) G_4

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}^{(4)}_{v1}$	$\mathcal{G}^{(4)}_{v2}$
1	A	B(2)	B(2, 3)	B(2, 3)	–	B(2, 3)
2	B	A(1)	A(1)	A(1)	–	A(1)
3	B		A(1), C(4), D(5)	A(1), C(4), D(5)	C(4), D(5)	A(1, 9), C(4), D(5)
4	C		B(3), D(5)	B(3), D(5)	B(3), D(5)	B(3), D(5)
5	D		B(3), C(4)	B(3), C(4)	B(3), C(4)	B(3), C(4)
6	A			B(7), E(8)	B(7), E(8)	E(8)
7	B			A(6), E(8)	A(6, 9), E(8)	E(8)
8	E			A(6), B(7)	A(6, 9), B(7)	A(6, 9), B(7)
9	A				B(7), E(8)	B(3), E(8)

(e) Δ-top Matchingtabelle

Abbildung A.15: Gespeicherte Matchings in $\mathcal{G}^{(4)}$. Dabei werden zwei verschiedene Varianten (Erweiterungsmöglichkeiten) für $\mathcal{G}^{(4)}$ dargestellt.

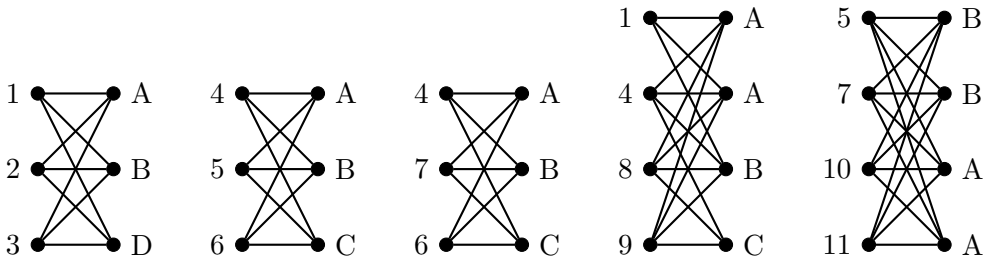
Matchingkante mit einem C haben. Der Anonymitätsgrad der Tupel 5, 7, 10 und 11 beträgt jeweils 1, wodurch es zu 4 Verletzungen des Schutzkriteriums kommt.

Wird in G_4 die andere mögliche Erweiterung gewählt und 4 mit 8 erweitert, entsteht zwar in $\mathcal{G}^{(4)}$ eine Verletzung bei Tupel 1, für $\mathcal{G}^{(5)}$ gibt es aber keine weitere Verletzung. In diesem Fall werden die alten Tupel 5 beziehungsweise 7 mit den neuen Tupeln 10 beziehungsweise 11 erweitert.

Dieses Beispiel zeigt, dass Erweiterungen, die für n Anfragegraphen scheinbar optimal sind, für $n + 1$ Graphen nicht mehr optimal sein müssen.

A.4.1 Erweiterungsalgorithmus: Maximaler Anonmyitätsgrad

Beispiel A.15 Abbildung A.17 zeigt in (a) die Matchingtabelle für eine Folge von n Anfragegraphen (siehe Spalte $\mathcal{G}^{(n)}$). In einem weiteren Graphen G_{n+1} kommen die alten Tupel 1 bis 5 und die neuen Tupel 6 bis 10 vor. Algorithmus 6.3 (*Maximaler Anonymitätsgrad*) wird für die alten Tupel 1, 2 und 3 aufgerufen, da diese denselben SA-Wert A haben. Der Ablauf ist dabei in (b) dargestellt.



(a) G_1 (b) G_2 (c) G_3 (d) G_4 (e) G_5

ID	SA	$\mathcal{G}^{(1)}$	$\mathcal{G}^{(2)}$	$\mathcal{G}^{(3)}$	$\mathcal{G}^{(4)}_{v1}$	$\mathcal{G}^{(5)}_{v1}$	$\mathcal{G}^{(4)}_{v2}$	$\mathcal{G}^{(5)}_{v2}$
1	A	B(2)	B(2)	B(2)	B(2, 8)	B(2, 8)	–	–
		D(3)	D(3)	D(3)	–	–	–	–
2	B	A(1)	A(1)	A(1)	A(1)	A(1)	–	–
		D(3)	D(3)	D(3)	D(3)	D(3)	D(3)	D(3)
3	D	A(1)	A(1)	A(1)	–	–	–	–
		B(2)	B(2)	B(2)	B(2)	B(2)	–	–
4	A		B(5)	B(5, 7)	–	–	B(5, 7, 8)	B(5, 7, 8)
			C(6)	C(6)	C(6, 9)	C(6, 9)	C(6, 9)	C(6, 9)
5	B		A(4)	A(4)	–	–	A(4)	A(4, 10)
			C(6)	C(6)	C(6)	–	C(6)	–
6	C		A(4)	A(4)	A(4)	A(4)	A(4)	A(4)
			B(5)	B(5, 7)	B(5, 7)	–	B(5, 7)	–
7	B			A(4)	–	–	A(4)	A(4, 11)
				C(6)	C(6)	–	C(6)	–
8	B				A(1)	A(1)	A(4)	A(4)
					C(9)	C(9)	C(9)	C(9)
9	B				A(4)	A(4)	A(4)	A(4)
					B(8)	B(8)	B(8)	B(8)
10	A					–		B(5)
11	A					–		B(7)

(f) Δ-top Matchingtabelle

Abbildung A.16: Matchings in $\mathcal{G}^{(5)}$. In (a) bis (e) sind fünf Anfragegraphen gegeben. Für $\mathcal{G}^{(4)}$ sind in der Matchingtabelle in (f) zwei verschiedene Erweiterungen ($v1$ und $v2$) dargestellt. Daraus ergeben sich auch zwei verschiedene Erweiterungen für $\mathcal{G}^{(5)}$.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B($x_{1.1}$)	B($x_{1.1}$, 6)
		C($x_{1.2}$)	–
		D($x_{1.3}$)	D($x_{1.2}$, 9)
		E($x_{1.4}$)	–
2	A	B(4)	B(4)
		C($x_{2.1}$)	C($x_{2.1}$, 8)
		E($x_{2.2}$)	E($x_{2.2}$, 10)
		F($x_{2.3}$)	–
3	A	B(4)	B(4)
		B($x_{3.1}$)	B($x_{3.1}$, 7)
		C(5)	C(5)
		D($x_{3.2}$)	–
		E($x_{3.3}$)	–
4	B	A(2)	A(2)
		A(3)	A(3)
5	C	A(3)	A(3)

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (aus Platzgründen sind nur die alten Tupel angegeben)

ID	SA	E^*_{Erw}	Phase 1.1	Phase 1.2	Phase 1.3	E_{Erw}
1	A	B($x_{1.1}$)	B($x_{1.1}$, 6)	B($x_{1.1}$, 6)	B($x_{1.1}$, 6)	
		C($x_{1.2}$)	C($x_{1.2}$)	–	–	
		D($x_{1.3}$)	D($x_{1.3}$)	D($x_{1.3}$, 9)	D($x_{1.3}$, 9)	
		E($x_{1.4}$)	E($x_{1.4}$)	E($x_{1.4}$)	–	
2	A	C($x_{2.1}$)	C($x_{2.1}$)	C($x_{2.1}$, 8)	C($x_{2.1}$, 8)	
		E($x_{2.2}$)	E($x_{2.2}$)	E($x_{2.2}$)	E($x_{2.2}$, 10)	
3	A	B($x_{3.1}$)	B($x_{3.1}$)	B($x_{3.1}$)	B($x_{3.1}$, 7)	
		D($x_{3.2}$)	D($x_{3.2}$)	–	–	
		E($x_{3.3}$)	E($x_{3.3}$)	E($x_{3.3}$)	–	
6	B		A(1)	A(1)	A(1)	
7	B				A(3)	
8	C			A(2)	A(2)	
9	D			A(1)	A(1)	
10	E				A(2)	

(b) Ablauf des Algorithmus

Abbildung A.17: Erweiterung von Matchings mithilfe des Algorithmus 6.3 (*Maximaler Anonymitätsgrad*)

Für Tupel 2 und 3 gibt es bereits Matchings in $\mathcal{T}_\Delta^{(n+1)}$ mit den Tupeln 4 beziehungsweise 5 (siehe Spalte $\mathcal{G}^{(n+1)}$ in (a)), sodass die aktuellen Anonymitätsgrade $a^*(1) = 1$, $a^*(2) = 2$ und $a^*(3) = 3$ sind. Als Erstes (Phase 1.1) wird daher nur Tupel 1 betrachtet. Die Reihenfolge der neuen Tupel sei 6, 7, 8, 9, 10, woraus folgt, dass 1 mit B(6) erweitert wird (siehe Spalte Phase 1.1).

Danach haben Tupel 1 und 2 jeweils den aktuellen Anonymitätsgrad von 2. Da Tupel 2 mit C und E zwei sowie Tupel 1 mit C, D und E drei verschiedene SA-Werte in Kanten aus E^*_{Erw} hat, liegt in der Reihenfolge der alten Tupel 2 vor 1.[6] Es folgen die Erweiterungen 2 mit C(8) sowie 1 mit D(9) und es müssen Kanten gestrichen werden, die nicht erweitert werden können (siehe Spalte Phase 1.2). Beispielsweise kann $(1, \text{C}(x_{1.2}))$ nicht mehr erweitert werden, denn es gibt kein neues Tupel mehr, das den SA-Wert C hat.

Jetzt haben alle drei alten Tupel den aktuellen Anonymitätsgrad 3 und den Erweiterungsgrad 1. Die Reihenfolge sei zum Beispiel 2, 1, 3. Dann wird 2 mit E(10) erweitert und die Kanten $(1, \text{E}(x_{1.4}))$ und $(3, \text{E}(x_{3.3}))$ werden gelöscht. Dann hat Tupel 1 keine erweiterbare Kante mehr in E^*_{Erw} und es kann nur noch Tupel 3 mit B(7) erweitert werden (siehe Spalte Phase 1.3).

Da alle neuen Tupel zugeordnet wurden, entfällt Phase 2 des Algorithmus. Spalte $\mathcal{G}^{(n+1)}$ aus (a) zeigt aus Platzgründen nur den Teil der Matchingtabelle $\mathcal{T}_\Delta^{(n+1)}$, der aus den alten Tupeln besteht.

Beispiel A.16 Abbildung A.18 zeigt in (a) die Matchingtabelle für eine Folge von n Anfragegraphen (siehe Spalte $\mathcal{G}^{(n)}$). In einem weiteren Graphen G_{n+1} kommen die alten Tupel 1 bis 3 und die neuen Tupel 4 bis 12 vor. Algorithmus 6.3 (*Maximaler Anonymitätsgrad*) wird für alle alten Tupel zusammen aufgerufen, da diese denselben SA-Wert A haben. Der Ablauf ist dabei in (b) dargestellt.

Der aktuelle Anonymitätsgrad ist für jedes alte Tupel 1, da es keine vorhandenen Matchings gibt. Der Erweiterungsgrad der Tupel 1 und 2 beträgt 2, während für Tupel 3 $d(3) = 4$ gilt. Die Reihenfolge der alten Tupel sei somit 1, 2, 3. Insgesamt gibt es vier neue Tupel mit SA-Wert B, drei mit C und je eins mit D beziehungsweise E. Die Reihenfolge der neuen Tupel sei 4, 5, 6, 7, 8, 9, 10, 11, 12, woraus folgt, dass 1 mit B(4) und 2 mit B(5) erweitert wird. Da es jetzt nur noch zwei neue Tupel mit SA-Wert B gibt, ist die neue Reihenfolge 8, 9, 10, 6, 7, 11, 12 und Tupel 3 wird mit C(8) erweitert (siehe Spalte Phase 1.1).

Danach haben alle alten Tupel den aktuellen Anonymitätsgrad 2 und die Reihenfolge sei weiterhin 1, 2, 3.[7] Tupel 1 wird jetzt mit C(9) und Tupel 2 mit D(11) erweitert. Die Reihenfolge der neuen Tupel ist danach 6, 7, 10, 12, denn der SA-Wert B ist jetzt wieder am häufigsten vertreten. Somit wird Tupel 3 mit B(6) erweitert (siehe Spalte Phase 1.2).

Alle Kanten der Tupel 1 und 2 sind damit erweitert und Tupel 3 kann in der Phase 1 nur noch mit E(12) erweitert werden (siehe Spalte Phase 1.2). Es verbleiben die neuen Tupel 7 und 10 und es können in Phase 2 nur noch die Kanten aus der Spalte E_{Erw} erweitert werden. Da Tupel 2 mit B nur einen SA-Wert in Kanten aus E_{Erw} und Tupel 1 sowie 3 jeweils zwei, sei die Reihenfolge der alten Tupel 2, 1, 3.[8] Es folgen die Erweiterungen von 2 mit B(7) und 1 mit C(10) (siehe Spalte Phase 2).

[6] Die Erweiterungsgrade bezüglich E^*_{Erw} sind $d(1) = 3$ und $d(2) = 2$.
[7] Die Erweiterungsgrade bezüglich E^*_{Erw} sind $d(1) = d(2) = 1$ und $d(3) = 3$.
[8] Die Erweiterungsgrade bezüglich E_{Erw} sind $d(1) = d(3) = 2$ und $d(2) = 1$.

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B($x_{1.1}$)	B($x_{1.1}$, 4)
		C($x_{1.2}$)	C($x_{1.1}$, 9)
			C($x_{1.1}$, 10)
2	A	B($x_{2.1}$)	B($x_{2.1}$, 5)
			B($x_{2.1}$, 7)
		D($x_{2.2}$)	D($x_{2.2}$, 11)
3	A	B($x_{3.1}$)	B($x_{3.1}$, 6)
		C($x_{3.2}$)	C($x_{3.2}$, 8)
		D($x_{3.3}$)	–
		E($x_{3.4}$)	E($x_{3.4}$, 12)

(a) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (aus Platzgründen sind nur die alten Tupel angegeben)

ID	SA	E^*_{Erw}	Phase 1.1	Phase 1.2	Phase 1.3	E_{Erw}	Phase 2
1	A	B($x_{1.1}$)	B($x_{1.1}, 4$)	B($x_{1.1}, 4$)	B($x_{1.1}, 4$)	B($x_{1.1}$)	B($x_{1.1}, 4$)
		C($x_{1.2}$)	C($x_{1.2}$)	C($x_{1.2}, 9$)	C($x_{1.2}, 9$)	C($x_{1.2}$)	C($x_{1.2}, 9$)
							C($x_{1.2}, 10$)
2	A	B($x_{2.1}$)	B($x_{2.1}, 5$)	B($x_{2.1}, 5$)	B($x_{2.1}, 5$)	B($x_{2.1}$)	B($x_{2.1}, 5$)
							B($x_{2.1}, 7$)
		D($x_{2.2}$)	D($x_{2.2}$)	D($x_{2.2}, 11$)	D($x_{2.2}, 11$)		D($x_{2.2}, 11$)
3	A	B($x_{3.1}$)	B($x_{3.1}$)	B($x_{3.1}, 6$)	B($x_{3.1}, 6$)	B($x_{3.1}$)	B($x_{3.1}, 6$)
		C($x_{3.2}$)	C($x_{3.2}, 8$)	C($x_{3.2}, 8$)	C($x_{3.2}, 8$)	C($x_{3.2}$)	C($x_{3.2}, 8$)
		D($x_{3.3}$)	D($x_{3.3}$)	–	–		–
		E($x_{3.4}$)	E($x_{3.4}$)	E($x_{3.4}$)	E($x_{3.4}, 12$)		E($x_{3.4}, 12$)
4	B		A(1)	A(1)	A(1)		A(1)
5	B		A(2)	A(2)	A(2)		A(2)
6	B			A(3)	A(3)		A(3)
7	B						A(2)
8	C		A(3)	A(3)	A(3)		A(3)
9	C			A(1)	A(1)		A(1)
10	C						A(1)
11	D			A(2)	A(2)		A(2)
12	E				A(3)		A(3)

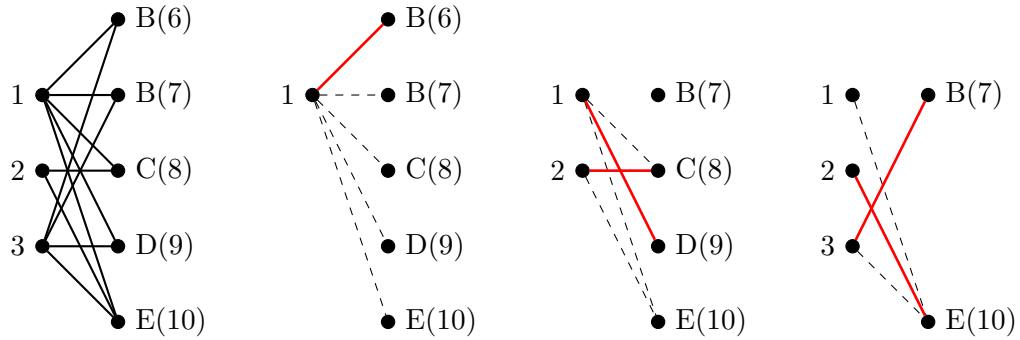
(b) Ablauf des Algorithmus

Abbildung A.18: Erweiterung von Matchings mithilfe des Algorithmus 6.3 (*Maximaler Anonymitätsgrad*)

Spalte $\mathcal{G}^{(n+1)}$ aus (a) zeigt aus Platzgründen nur den Teil der Matchingtabelle $\mathcal{T}_{\Delta}^{(n+1)}$, der aus den alten Tupeln besteht.

A.4.2 Erweiterungsalgorithmus: Einfaches Matching

Beispiel A.17 Gegeben sei dieselbe Δ-top Matchingtabelle wie in Beispiel A.15 (siehe Abbildung A.17a). Es sollen die alten Tupel 1 bis 3 mit den neuen Tupeln 6 bis 10 erweitert werden. Dabei existieren bereits Matchings von 2 beziehungsweise 3 mit den alten Tupeln 4 und 5.



(a) G_{Erw} (b) M_1 (c) M_2 (d) M_3

Abbildung A.19: Ablauf des Algorithmus 6.4 (*Einfaches Matching*). Die farbigen Kanten bilden jeweils ein Matching.

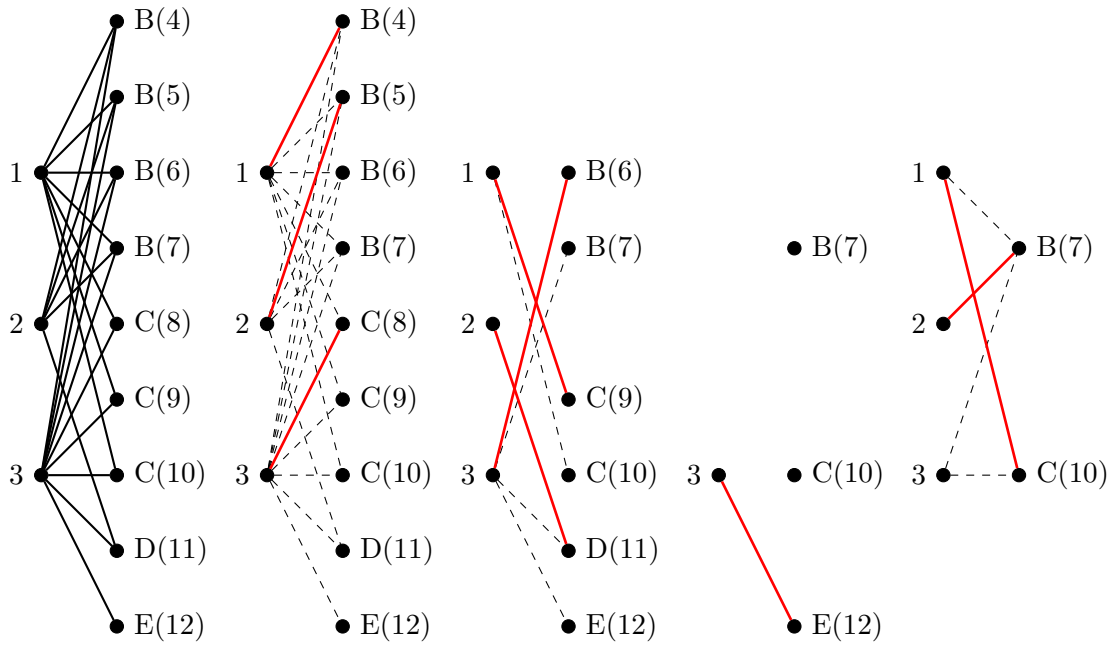
Abbildung A.19 zeigt den Ablauf des Algorithmus 6.4 (*Einfaches Matching*), wobei der Erweiterungsgraph G_{Erw} in (a) dargestellt ist. Anfangs sind die aktuellen Anonymitätsgrade $a^*(1) = 1$, $a^*(2) = 2$ und $a^*(3) = 3$, das heißt, es wird zunächst nur Tupel 1 betrachtet. Ein mögliches Matching ist in (b) berechnet und repräsentiert die Erweiterung von 1 mit B(6). Es sind aber auch andere Matchings denkbar. Nach der entsprechenden Erweiterung wird die Kante $\{1, \text{B}(7)\}$ in G_{Erw} entfernt, da Tupel 7 mit B denselben SA-Wert wie Tupel 6 hat (siehe Zeile 10 im Algorithmus 6.4).

Nach dieser Erweiterung sind die aktuellen Anonymitätsgrade $a^*(1) = a^*(2) = 2$ und $a^*(3) = 3$ und es wird ein Matching mit den alten Tupeln 1 und 2 berechnet. Beispielsweise kann 1 mit D(9) und 2 mit C(8) erweitert werden (siehe (c)). Jetzt haben alle alten Tupel den aktuellen Anonymitätsgrad 3 und (d) zeigt ein mögliches Matching, das alle alten Tupel einbezieht. Da nur noch zwei neue Tupel vorkommen, ist M_3 nicht mehr perfekt (d. h., Tupel 1 hat keinen Matchingpartner). Die entsprechenden Erweiterungen sind 2 mit E(10) und 3 mit B(7).

Phase 2 entfällt, da es kein weiteres neues Tupel gibt. Die resultierende Δ-top Matchingtabelle entspricht der gleichen wie in Beispiel A.15 (siehe Abbildung A.17a), da die gleichen Erweiterungen durchgeführt wurden.

Beispiel A.18 Gegeben sei dieselbe Δ-top Matchingtabelle wie in Beispiel A.16 (siehe Abbildung A.18a). Es sollen die alten Tupel 1 bis 3 mit den neuen Tupeln 4 bis 12 erweitert werden.

Abbildung A.20 zeigt den Ablauf des Algorithmus 6.4 (*Einfaches Matching*), wobei der Erweiterungsgraph G_{Erw} in (a) dargestellt ist. Anfangs haben alle alten Tupel den aktu-



(a) G_{Erw} (b) M_1 (c) M_2 (d) M_3 (e) M_4

Abbildung A.20: Ablauf des Algorithmus 6.4 (*Einfaches Matching*). Die farbigen Kanten bilden jeweils ein Matching.

ellen Anonymitätsgrad 1 und es wird ein Matching mit allen Tupeln berechnet. (b) zeigt eine Möglichkeit, wonach 1 mit B(4), 2 mit B(5) und 3 mit C(8) erweitert wird. Danach werden die Kanten $\{1, \text{B}(6)\}$, $\{1, \text{B}(7)\}$, $\{2, \text{B}(6)\}$, $\{2, \text{B}(7)\}$, $\{3, \text{C}(9)\}$ und $\{3, \text{C}(10)\}$ aus G_{Erw} entfernt (vgl. Zeile 10 im Algorithmus 6.4).

Erneut haben alle alten Tupel denselben aktuellen Anonymitätsgrad und es wird ein Matching berechnet (siehe (c)). Danach haben die Knoten für Tupel 1 und 2 keine ausgehende Kante mehr und werden im dritten Schritt ignoriert.[9] Es folgt nur eine Erweiterung von Tupel 3 mit E(12) (siehe (d)).

In Phase 2 des Algorithmus gibt es nur noch die neuen Tupel 7 und 10. Ein mögliches Matching ist in (e) dargestellt und erweitert 1 mit C(10) und 2 mit B(7). Es entstehen hierbei mehrfach erweiterte Kanten, da 1 bereits in Phase 1 mit einem C und 2 mit einem B erweitert wurde.

Die resultierende Δ-top Matchingtabelle entspricht der gleichen wie in Beispiel A.16 (siehe Abbildung A.18a), da die gleichen Erweiterungen durchgeführt wurden.

A.4.3 Erweiterungsalgorithmus: Klonmatching

Beispiel A.19 Sei eine ähnliche Eingabe wie in Beispiel A.15 gegeben. Dabei sei zur Δ-top Matchingtabelle aus Abbildung A.17a ein Anfragegraph gegeben, der neben den alten Tupeln 1 bis 5 die fünf neuen Tupel 6 bis 10 und zusätzlich ein neues Tupel 11 mit SA-Wert E enthalte. Die entsprechende Matchingtabelle ist in Abbildung A.21c gegeben.

[9] Man beachte, dass die Kante $\{1, \text{C}(10)\}$ aus G_{Erw} entfernt wurde, da Tupel 1 im vorigen Schritt mit einem neuen Tupel mit SA-Wert C erweitert wurde.

ID	$d(t)$	Runde 1		Runde 2	
		$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$
1	4	1	3	2	2
2	2	1	1	2	–
3	3	1	2	2	1
N_T		3		6	
S_T		$\{1,2,3\}$		$\{1,3\}$	
$d_{\max}$		1		0	

Tabelle A.1: Konstruktion des Klongraphen nach Algorithmus 6.5

Zunächst soll ein Klongraph nach Algorithmus 6.5 für das vereinfachte Optimierungsproblem 6.3 erstellt werden. Tabelle A.1 gibt einen Überblick über die sukzessive Konstruktion des Klongraphen. Zu den alten Tupeln sind die Erweiterungsgrade $d(1) = 4$, $d(2) = 2$ und $d(3) = 3$ in der 2. Spalte angegeben. Entscheidend ist die Berechnung von $d_{\max} = \lfloor(|S_{\text{neu}}| - N_T)/|S_T|\rfloor$ in der letzten Zeile.

Im ersten Durchlauf der While-Schleife (siehe Spalte Runde 1) ist $N_T = 3$, $S_T = \{1,2,3\}$ und demnach $d_{\max} = \lfloor(6-3)/3\rfloor = 1 \geq 1$. Für die alten Tupel gilt $d(1)-n(1) = 4-1 = 3 > d_{\max}$, $d(2)-n(2) = 2-1 = 1 = d_{\max}$ beziehungsweise $d(3)-n(3) = 3-1 = 2 > d_{\max}$ und es werden die Klongrade aller Tupel jeweils um $d_{\max} = 1$ erhöht ($n(1) = n(2) = n(3) = 2$). Da Tupel 2 jetzt maximal oft vorhanden ist ($n(2) = d(2)$), wird es aus S_T entfernt.

Danach gilt $d_{\max} = \lfloor(6-6)/2\rfloor = 0 < 1$, denn die Anzahl der alten Knoten N_T entspricht der Anzahl der neuen Tupel $|S_{\text{neu}}|$. Das ist ein Abbruchkriterium der Konstruktion.

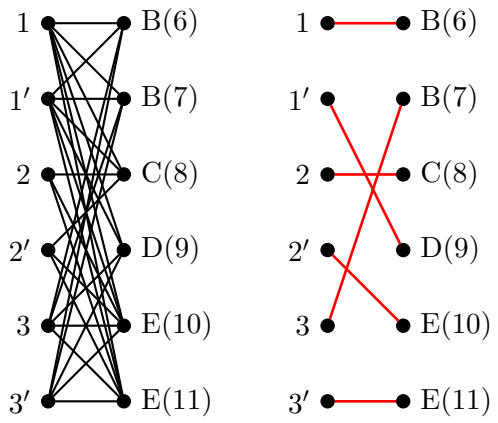
In Abbildung A.21 ist in (a) der konstruierte Klongraph, in (b) ein mögliches Matching und in (c) die daraus resultierende Δ-top Matchingtabelle dargestellt. Jedes alte Tupel ist mit zwei neuen Tupeln erweitert worden, die jeweils einen verschiedenen SA-Wert haben.

Beispiel A.20 Sei die gleiche Eingabe wie in Beispiel A.19 gegeben. Abbildung A.21c zeigt die entsprechende Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$. Diesmal soll mit Algorithmus 6.9 ein Klongraph für das komplexere Optimierungsproblem 6.2 konstruiert werden. Algorithmus 6.10 (*Klonmatching*) berechnet damit eine Erweiterung, bei der der minimale Anonymitätsgrad der alten Tupel maximal ist.

ID	$a^*(t)$	$d(t)$	Runde 1		Runde 2	
			$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$
1	1	4	2	2	3	1
2	2	2	1	1	2	–
3	3	2	0	2	1	1
N_T			3		6	
S_T			$\{1,2,3\}$		$\{1,3\}$	
$d_{\max}$			1		0	

Tabelle A.2: Konstruktion des Klongraphen nach Algorithmus 6.9

Tabelle A.2 zeigt den Ablauf der Konstruktion des Klongraphen. Da bereits die Matchingkanten $(2, \mathrm{B}(4))$, $(3, \mathrm{B}(4))$ und $(3, \mathrm{B}(5))$ in der aktuellen Matchingtabelle existieren, sind die aktuellen Anonymitätsgrade $a^*(1) = 1$, $a^*(2) = 2$ und $a^*(3) = 3$. Die Kan-



(a) G_{Erw} (b) M

ID	SA	$\mathcal{G}^{(n)}$	$\mathcal{G}^{(n+1)}$
1	A	B($x_{1.1}$)	B($x_{1.1}$, 6)
		C($x_{1.2}$)	–
		D($x_{1.3}$)	D($x_{1.2}$, 9)
		E($x_{1.4}$)	–
2	A	B(4)	B(4)
		C($x_{2.1}$)	C($x_{2.1}$, 8)
		E($x_{2.2}$)	E($x_{2.2}$, 10)
		F($x_{2.3}$)	–
3	A	B(4)	B(4)
		B($x_{3.1}$)	B($x_{3.1}$, 7)
		C(5)	C(5)
		D($x_{3.2}$)	–
		E($x_{3.3}$)	E($x_{3.3}$, 11)
4	B	A(2)	A(2)
		A(3)	A(3)
5	C	A(3)	A(3)
6	B		A(1)
7	B		A(3)
8	C		A(2)
9	D		A(1)
10	E		A(2)
11	E		A(3)

(c) Δ-top Matchingtabelle für $\mathcal{G}^{(n)}$ und $\mathcal{G}^{(n+1)}$ (ohne neue Matchings)

Abbildung A.21: Erweiterung von Matchings mithilfe des Algorithmus 6.6 (*Vereinfachtes Klonmatching*)

te $(3, \text{B}(x_{3.1}))$ wird in Phase 1 des *Klonmatching*-Algorithmus nicht beachtet, denn es gibt bereits eine Matchingkante mit Tupel 3 und dem SA-Wert B (vgl. Zeile 3). Demzufolge sind die Erweiterungsgrade $d(1) = 4$ sowie $d(2) = d(3) = 2$ und es werden $n(1) = \min\{a^*_{\max} - a^*(t), d(1)\} = \min\{3-1, 4\} = 2$ Knoten für Tupel 1 sowie $n(2) = 1$ Knoten für Tupel 2 initialisiert. Tupel 3 hat den größten aktuellen Anonymitätsgrad und bekommt daher anfangs keinen Knoten $(n(3) = 0)$.

Im ersten Durchlauf der While-Schleife ist $N_T = 3$, $S_T = \{1, 2, 3\}$ und demnach $d_{\max} = \lfloor (6-3)/3 \rfloor = 1 \geq 1$. Für die alten Tupel gilt $d(1) - n(1) = 4 - 2 = 2 > d_{\max}$, $d(2) - n(2) = 2 - 1 = 1 = d_{\max}$ beziehungsweise $d(3) - n(3) = 2 - 0 = 2 > d_{\max}$ und es werden die Klongrade aller Tupel jeweils um $d_{\max} = 1$ erhöht. Da Tupel 2 jetzt maximal oft vorhanden $(n(2) = d(2))$ ist, wird es aus S_T entfernt.

Danach gilt $d_{\max} = \lfloor (6-6)/2 \rfloor = 0 < 1$, denn die Anzahl der alten Knoten N_T entspricht der Anzahl der neuen Tupel $|S_{\text{neu}}|$. Das ist ein Abbruchkriterium der Konstruktion.

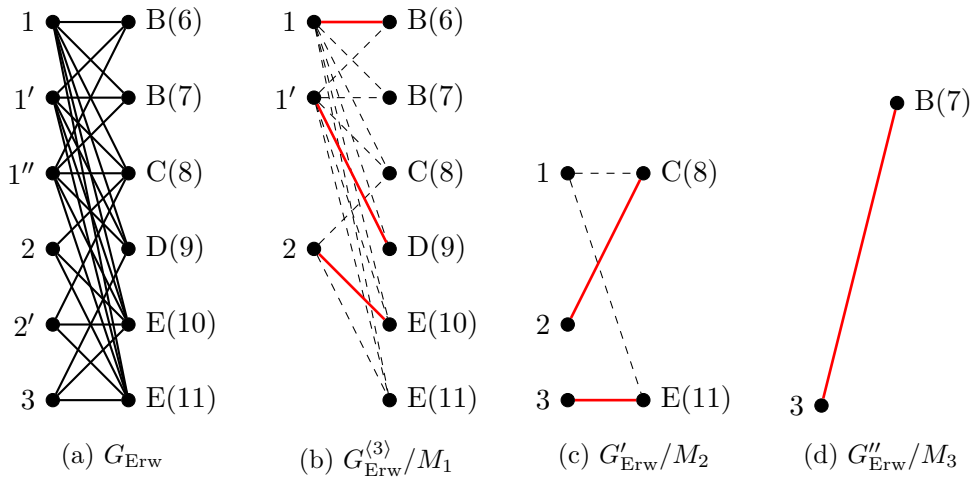
Abbildung A.22a zeigt den resultierenden Klongraphen, in dem aber kein vollständiges Matching existiert. Beispielsweise gibt es zwei neue Knoten mit SA-Wert B, die nur mit den drei Knoten für Tupel 1 verbunden sind. Zwei Knoten für dasselbe alte Tupel dürfen aber in einem vollständigen Matching nicht mit zwei neuen Knoten matchen, die denselben

SA-Wert haben. Da die Anzahl der alten Knoten identisch zur Anzahl der neuen Knoten ist, bleibt in jedem größten Matching mindestens ein alter Knoten unüberdeckt.

An dieser Stelle konstruiert Algorithmus 6.10 (*Klonmatching*) einen Teilgraphen von G_{Erw} und berechnet dazu zwei Werte.

$$n^*_{\max} = \max_{t \in S_{\text{alt}}} (n(t) + a^*(t)) = 4$$
$$a^*_{\min} = \min_{t \in S_{\text{alt}}} a^*(t) \qquad\qquad = 1$$

Es wird das maximale i mit $a^*_{\min} + 1 \leq i \leq n^*_{\max}$ gesucht, für das der Graph $G^{\langle i \rangle}_{\text{Erw}}$ ein vollständiges Matching besitzt. In diesem Beispiel müssen Graphen mit $2 \leq i \leq 4$ getestet werden, wobei der Fall $i = 4$ bereits untersucht wurde ($G^{\langle 4 \rangle}_{\text{Erw}} = G_{\text{Erw}}$ in Abbildung A.22a).



(a) G_{Erw} (b) $G^{\langle 3 \rangle}_{\text{Erw}}/M_1$ (c) G'_{Erw}/M_2 (d) G''_{Erw}/M_3

Abbildung A.22: Erweiterung von Matchings mithilfe des Algorithmus 6.10 (*Klonmatching*). Die farbigen Kanten bilden jeweils ein Matching im Graphen, zu dem zusätzlich noch die gestrichelten Kanten gehören.

Abbildung A.22b zeigt den Graphen $G^{\langle 3 \rangle}_{\text{Erw}}$, der aus G_{Erw} dadurch entsteht, indem für jedes alte Tupel t $\max\{\min\{n(t), 3 - a^*(t)\}, 0\}$ Knoten erzeugt werden.[10] Für Tupel 1 werden damit $\min\{3, 3-a^*(1)\} = 3-1 = 2$ Knoten erstellt, für Tupel 2 $\min\{2, 3-a^*(2)\} = 3 - 2 = 1$ und kein Knoten für Tupel 3, denn der aktuelle Anonymitätsgrad dieses Tupels ist bereits $a^*(3) = 3$. Die farbigen Kanten stellen ein mögliches vollständiges Matching M_1 dar, nach dem an dieser Stelle erweitert werden soll. Somit wird Tupel 1 mit B(6) und D(9) sowie Tupel 2 mit E(10) erweitert.

Damit kommen nur noch die neuen Knoten B(7), C(8) und E(11) für eine Erweiterung in Frage und es wird erneut ein Klongraph erstellt. Tabelle A.3 zeigt den Ablauf der Konstruktion des Klongraphen. Die Erweiterungsgrade sind auf $d(1) = 2$ und $d(2) = d(3) = 1$ gesunken, denn 1 kann mit C(8) und E(11), 2 nur noch mit C(8) sowie 3 nur noch mit E(11) erweitert werden.[11] Das bedeutet, dass das neue Tupel 7 mit SA-Wert B in dieser

[10] Sowohl die gestrichelten als auch die farbig markierten Kanten bilden den Graphen $G^{\langle 3 \rangle}_{\text{Erw}}$.

[11] Da Tupel 1 bereits mit B(6) und Tupel 2 mit E(10) erweitert wurden, können sie in dieser Phase nicht mit einem Tupel erweitert werden, das denselben SA-Wert B beziehungsweise E hat.

ID	$a^*(t)$	$d(t)$	Init	Runde 1	
			$n(t)$	$n(t)$	$d(t) - n(t)$
1	3	2	0	1	1
2	3	1	0	1	–
3	3	1	0	1	–
N_T			0	3	
S_T			–	$\{1\}$	
$d_{\max}$			–	0	

Tabelle A.3: Konstruktion des Klongraphen nach Algorithmus 6.9

Phase nicht mehr beachtet wird und temporär aus S_{neu} entfernt wird. Da der aktuelle Anonymitätsgrad diesmal bei allen alten Tupeln 3 beträgt, werden alle Klongrade nicht mit $\min\{a^*_{\max} - a^*(t), d(t)\} = 0$, sondern mit 1 initialisiert (vgl. Zeile 9 in Algorithmus 6.9). Damit sind Tupel 2 und 3 bereits maximal oft vorhanden und werden nicht in S_T übernommen.

Der erste Durchlauf der While-Schleife liefert bereits ein Abbruchkriterium, denn es gilt $N_T = 3$, $S_T = \{1\}$ und folglich $d_{\max} = \lfloor (2-3)/1 \rfloor = -1 < 1$. Im resultierenden Klongraphen G'_{Erw} in Abbildung A.22c existiert erneut kein vollständiges Matching und somit müssen wieder Teilgraphen $G'^{\langle i \rangle}_{\text{Erw}}$ untersucht werden.

$$n^*_{\max} = \max_{t \in S_{\text{alt}}} (n(t) + a^*(t)) = 4$$
$$a^*_{\min} = \min_{t \in S_{\text{alt}}} a^*(t) \qquad = 3$$

Es gilt $a^*_{\min}+1 = n^*_{\max} = 4$ und damit gilt nur für $i = 4$ die Bedingung $a^*_{\min}+1 \leq i \leq n^*_{\max}$. Der Klongraph $G'^{\langle 4 \rangle}_{\text{Erw}}$ ist aber identisch zum bereits berechneten Graphen G'_{Erw}, in dem es kein vollständiges Matching gibt. Nach Algorithmus *Klonmatching* wird in diesem Fall nur ein größtes Matching in $G'^{\langle 4 \rangle}_{\text{Erw}}$ berechnet, das in Abbildung A.22c dargestellt ist. Tupel 2 wird mit C(8) und Tupel 3 mit E(11) erweitert.

Phase 1 von Algorithmus 6.10 endet an dieser Stelle, da es keine noch nicht erweiterte Kante gibt, die erweitert werden kann ($E^*_{\text{Erw}} = \emptyset$). In Phase 2 wird versucht, alte mit neuen Tupeln zu erweitern, deren SA-Wert bereits in einem Matching vorkommt. In diesem Beispiel wurde die Kante $(3, \text{B}(x_{3.1}))$ am Anfang entfernt und wird nun betrachtet. Der Klongraph für die Phase 2 besteht damit aus dem alten Knoten für Tupel 3 und dem neuen Knoten für Tupel 7 mit einem trivialen Matching (siehe A.22d).

Damit sind alle neuen Tupel zugeordnet und Phase 3 entfällt. Die berechnete Δ-top Matchingtabelle ist in Abbildung A.21c dargestellt und entspricht exakt der, die durch den vereinfachten *Klonmatching*-Algorithmus in Beispiel A.19 berechnet wurde.

Beispiel A.21 Es sei die gleiche Erweiterung wie in Beispiel A.16 gesucht. Die entsprechende Δ-top Matchingtabelle ist in Abbildung A.18a gegeben. Ein weiterer Anfragegraph G_{n+1} enthalte die alten Tupel 1 bis 3 und die neun neuen Tupel 4 bis 12.

Zunächst soll ein Klongraph nach Algorithmus 6.5 für das vereinfachte Optimierungsproblem 6.3 erstellt werden. Tabelle A.4 gibt einen Überblick über die sukzessive Konstruktion des Klongraphen. Zu den alten Tupeln sind die Erweiterungsgrade $d(1) = 2$,

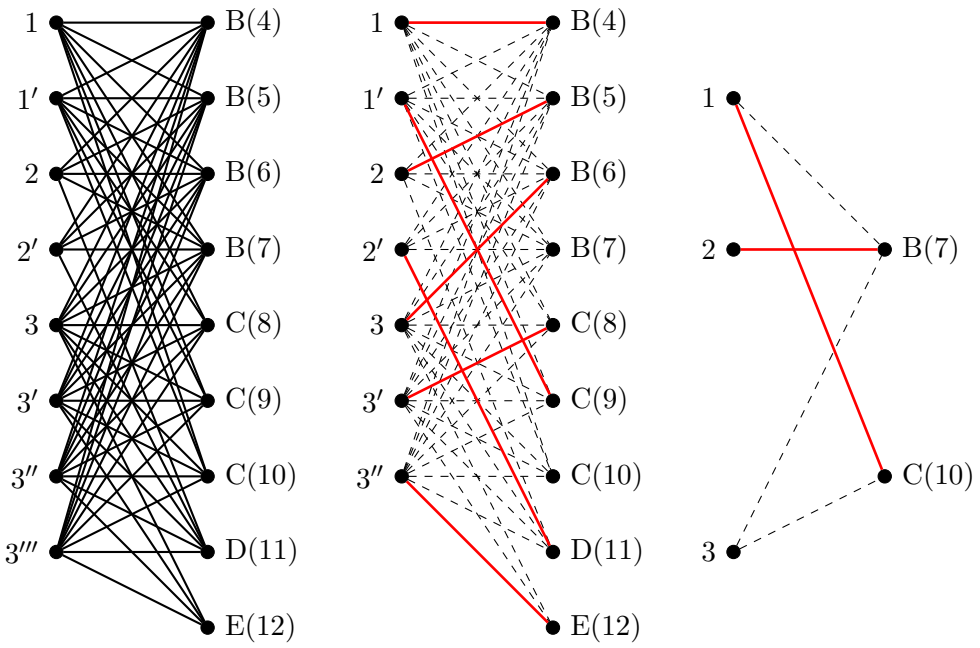
ID	d_t	Runde 1		Runde 2		Runde 3	
		$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$	$n(t)$	$d(t)-n(t)$
1	2	1	1	2	–	2	–
2	2	1	1	2	–	2	–
3	4	1	3	3	1	4	–
N_T		3		7		8	
S_T		$\{1,2,3\}$		$\{3\}$		$\emptyset$	
$d_{\max}$		2		1		–	

Tabelle A.4: Konstruktion des Klongraphen nach Algorithmus 6.5

$d(2) = 2$ und $d(3) = 4$ in der 2. Spalte angegeben. Entscheidend ist die Berechnung von $d_{\max} = \lfloor(|S_{\text{neu}}| - N_T)/|S_T|\rfloor$ in der letzten Zeile.

Im ersten Durchlauf der While-Schleife (siehe Spalte Runde 1) ist $N_T = 3$, $S_T = \{1,2,3\}$ und demnach $d_{\max} = \lfloor(9-3)/3\rfloor = 2 \geq 1$. Für die alten Tupel gilt $d(1) - n(1) = 2 - 1 = 1 < d_{\max}$, $d(2) - n(2) = 2 - 1 = 1 < d_{\max}$ beziehungsweise $d(3) - n(3) = 4 - 1 = 3 > d_{\max}$. Demnach werden die Klongrade für Tupel 1 und 2 jeweils um 1 ($n(1) = n(2) = 2$) und für Tupel 3 um $d_{\max} = 2$ ($n(3) = 3$) erhöht. Da die Tupel 1 und 2 jetzt maximal oft vorhanden sind ($n(i) = d(i)$), werden beide aus S_T entfernt.

Danach gilt $d_{\max} = \lfloor(9-7)/1\rfloor = 2 \geq 1$ und für Tupel 3 $d(3) - n(3) = 4 - 3 = 1 < d_{\max}$. $n(3)$ kann somit um 1 erhöht werden und Tupel 3 wird aus S_T entfernt, da es maximal oft vorhanden ist. Nun gilt aber $S_T = \emptyset$, was ein Abbruchkriterium der Konstruktion ist.



(a) G_{Erw} (b) $G^{[3]}_{\text{Erw}}/M_1$ (c) G'_{Erw}/M_2

Abbildung A.23: Erweiterung von Matchings mithilfe des Algorithmus 6.6 (*Vereinfachtes Klonmatching*). Die farbigen Kanten bilden jeweils ein Matching im Graphen, zu dem zusätzlich noch die gestrichelten Kanten gehören.

Abbildung A.23 zeigt in (a) den resultierenden Klongraphen G_{Erw}, in dem aber kein vollständiges Matching existiert. Da alle alten Knoten, die für dasselbe Tupel stehen, mit neuen Knoten matchen sollen, die jeweils einen anderen SA-Wert haben, müsste sowohl ein Knoten für Tupel 2 als auch einer für Tupel 3 mit D(11) matchen.

An dieser Stelle konstruiert Algorithmus 6.6 (*Vereinfachtes Klonmatching*) einen Teilgraphen von G_{Erw}, bei dem die Anzahl von alten Knoten durch eine Variable i beschränkt ist. Da der maximale Klongrad $n_{\max} = \max_{t \in S_{\text{alt}}} n(t) = 4$ ist, wird das maximale i mit $1 \leq i \leq n_{\max}$ gesucht, für das der Graph $G_{\text{Erw}}^{[i]}$ ein vollständiges Matching besitzt. In diesem Beispiel müssen Graphen mit $1 \leq i \leq 4$ getestet werden, wobei der Fall $i = 4$ bereits untersucht wurde ($G_{\text{Erw}}^{[4]} = G_{\text{Erw}}$ in Abbildung A.23a).

Teil (b) zeigt mit $G_{\text{Erw}}^{[3]}$ einen Klongraphen, bei dem die maximale Anzahl von Knoten pro altem Tupel auf 3 beschränkt ist. M_1 ist darin ein mögliches Matching, nach dem erweitert werden kann. Es verbleiben die neuen Tupel 7 und 10, für die es allerdings keine Δ-top Matchingkanten gibt, die noch nicht erweitert wurden.[12] Demnach endet Phase 1 des Algorithmus mit diesen Erweiterungen.

In Phase 2 werden mit Algorithmus 6.4 (*Einfaches Matching*) bereits einmal erweiterte Kanten erneut erweitert. In (c) ist ein Matching im resultierenden Graphen G'_{Erw} dargestellt. Die aus den Erweiterungen entstehende Δ-top Matchingtabelle ist in Abbildung A.18a angegeben und ist identisch zu der aus Beispiel A.16.

Da alle alten Tupel anfangs den aktuellen Anonymitätsgrad 1 hatten, sind die konstruierten Klongraphen und durchgeführten Berechnungen von Algorithmus 6.10 (*Klonmatching*) analog zu denen von Algorithmus 6.6 (*Vereinfachtes Klonmatching*). In diesem Fall sind offensichtlich die beiden Lösungen der Optimierungsprobleme 6.2 und 6.3 identisch.

A.5 SA-eindeutige Anfragegraphen

Für die Berechnung von Matchings in SA-eindeutigen Anfragegraphen werden im Kapitel 7 spezielle Algorithmen vorgestellt, welche anhand weiterer Beispiele hier vertieft werden.

A.5.1 Clustern der alten Tupel

Beispiel A.22 In Tabelle A.5 ist ein Anfragegraph dargestellt, der aus neun alten Tupeln besteht. Mithilfe von Algorithmus 7.5 soll ein Clustering erstellt werden, wobei in jedem Cluster kein SA-Wert mehrfach vorkommen darf. Insgesamt kommen mit A, B und D drei verschiedene SA-Werte am häufigsten vor, wobei hier für die Initialisierung A gewählt werde. Somit entstehen die beiden Cluster $C_1 = \{1\}$ und $C_2 = \{2\}$. Der weitere Ablauf des Algorithmus ist in Tabelle A.6 skizziert. Seien die restlichen Tupel entsprechend ihren IDs sortiert, dann wird zunächst Tupel 3 betrachtet und das Cluster mit dem größten *sim*-Wert gesucht (vgl. Spalte „3“). Da $sim(3, C_1) = sim(3, \{1\}) = 2$ und $sim(3, C_2) = sim(2, \{2\}) - \infty$, wird 3 in das Cluster mit 1 eingefügt. Analog kommt Tupel 4 danach in Cluster C_2 (vgl. Spalte „4“). Die Tupel 5 und 6 haben jeweils einen maximalen *sim*-Wert von 0, das heißt, sie werden zurückgestellt und noch keinem Cluster zugewiesen. Da Tupel 7 zu beiden Clustern einen *sim*-Wert von $-\infty$ hat, wird ein neues Cluster erstellt, das nur

[12] Die alten Tupel 1, 2 und 3 haben bereits Δ-top Matchingkanten mit neuen Knoten mit den SA-Werten B beziehungsweise C.

aus Tupel 7 besteht. Für Tupel 8 beziehungsweise 9 existiert jeweils ein Cluster, das einen *sim*-Wert von 4 hat und in welches das Tupel einsortiert wird.

ID	SA	*Sig*
1	A	ABE
2	A	ACDE
3	B	ABE
4	C	ACD
5	D	DE
6	F	BF
7	B	BCEF
8	E	ABDE
9	D	ACD

Tabelle A.5: Menge alter Tupel

	Init	3		4		5		6		7	
	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i
C_1	$\{1\}$	2	$\{1,3\}$	$-\infty$	$\{1,3\}$	0	$\{1,3\}$	$-\infty$	$\{1,3\}$	$-\infty$	$\{1,3\}$
C_2	$\{2\}$	$-\infty$	$\{2\}$	2	$\{2,4\}$	$-\infty$	$\{2,4\}$	0	$\{2,4\}$	$-\infty$	$\{2,4\}$
C_3											$\{7\}$

	8		9		5		6	
	sim	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i
C_1	4	$\{1,3,8\}$	$-\infty$	$\{1,3,8\}$	2	$\{1,3,5,8\}$	$-\infty$	$\{1,3,5,8\}$
C_2	2	$\{2,4\}$	4	$\{2,4,9\}$	$-\infty$	$\{2,4,9\}$	0	$\{2,4,9\}$
C_3	2	$\{7\}$	$-\infty$	$\{7\}$	0	$\{7\}$	2	$\{6,7\}$

Tabelle A.6: Clustern von alten Tupeln nach Algorithmus 7.5

Danach erfolgt der zweite Durchlauf der Schleife in Algorithmus 7.5, in dem die beiden Tupel 5 und 6 erneut betrachtet werden. Diesmal gilt $sim(5, C_1) = 2$, wodurch 5 in C_1 eingefügt wird. Tupel 6 hat nur einen positiven *sim*-Wert zu Cluster C_3, das bei der ersten Betrachtung von 6 noch nicht vorhanden war. Folglich wird Tupel 6 Cluster C_3 zugewiesen und die berechneten Cluster lauten $C_1 = \{1, 3, 5, 8\}$, $C_2 = \{2, 4, 9\}$ und $C_3 = \{6, 7\}$.

A.5.2 Clustern der neuen Tupel

Beispiel A.23 In Abbildung A.24 ist ein Anfragegraph dargestellt, der aus sechs alten (siehe (a)) und insgesamt zehn neuen (siehe (b) und (c)) Tupeln besteht. Mithilfe der Algorithmen 7.5 und 7.6 soll ein Clustering erstellt werden, wobei in jedem Cluster kein SA-Wert mehrfach vorkommen darf. Der Ablauf des ersten Algorithmus, der die alten Tupel clustert, ist in Tabelle A.7 dargestellt und soll hier nicht weiter erläutert werden, da er analog zu den vorigen Beispielen verläuft.[13] An dieser Stelle wird nur das Clustern der

[13] Es gibt zwei Stellen im Ablauf, an denen es jeweils zwei Cluster mit maximalem *sim*-Wert gibt. Es sei angenommen, dass sich der Algorithmus für die hier dargestellte Variante entscheidet.

neuen Tupel mittels Algorithmus 7.6 näher untersucht, welches übersichtlich in Tabelle A.8 skizziert ist.

ID	SA	*Sig*
1	A	ABCD
2	A	ABCG
3	B	ABDE
4	B	BCEFH
5	C	ACFG
6	C	BCEFH

(a) Alte Tupel in G_i

ID	SA	*Sig*
7	D	neu
8	E	neu
9	E	neu
10	I	neu
11	J	neu

(b) Neue Tupel in G_i

ID	SA	*Sig*
12	F	neu
13	F	neu
14	G	neu
15	H	neu
16	B	neu

(c) Neue Tupel in G_i

Abbildung A.24: Anfragegraph G_i

	Init	3		4		5		6	
	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i	*sim*	C_i
C_1	$\{1\}$	2	$\{1,3\}$	$-\infty$	$\{1,3\}$	2	$\{1,3\}$	$-\infty$	$\{1,3\}$
C_2	$\{2\}$	2	$\{2\}$	$-\infty$	$\{2\}$	2	$\{2,5\}$	$-\infty$	$\{2,5\}$
C_3				0	$\{4\}$	$-\infty$	$\{4\}$	2	$\{4,6\}$

Tabelle A.7: Clustern von neuen Tupeln nach Algorithmus 7.5

Seien die neuen Tupel entsprechend ihren IDs sortiert, dann wird Tupel 7 zuerst betrachtet. Da sein SA-Wert D nur in der Signatur der alten Tupel 1 und 3 vorkommt, ist $sim_0(7, C_1) = 4 > 0$ und 7 wird in C_1 einsortiert. Der minimale aktuelle Anonymitätsgrad der neuen Tupel in C_1 ist danach $\frac{sim(7,C_1)}{2} + 1 = 3$. Tupel 8 kann C_1 oder C_3 zugefügt werden. Da $sim_0(8, C_1) = 2 < sim_0(8, C_3) = 4$ gilt, wird 8 dem letzteren Cluster zugeordnet. Tupel 9 hat denselben SA-Wert wie Tupel 8 und wird daher in C_1 eingefügt. Der Anonymitätsgrad ändert sich dabei nicht, denn $a^*_{\min}((C_1 \cup \{8\})_{\text{neu}}) = \min\{a^*_{\min}(C_{1\text{neu}}) + 1, \frac{sim(8,C_1)}{2} + 1\} = \min\{3+1, 2+1\} = 3$.

Die Tupel 10 und 11 haben jeweils zu allen Clustern einen sim_0-Wert von 0 und werden im Algorithmus zurückgestellt. Tupel 12 wird C_3 zugeordnet, wodurch dort der Anonymitätsgrad steigt: $a^*_{\min}((C_3 \cup \{12\})_{\text{neu}}) = \min\{4, 4\} = 4$. Die weiteren Tupel 13, 14, 15 und 16 werden wie angegeben in die Cluster C_2 beziehungsweise C_3 eingefügt. Die Anonymitätsgrade vergrößern sich dabei auf $a^*_{\min}(C_{2\text{neu}}) = 4$ und $a^*_{\min}(C_{3\text{neu}}) = 5$. Ist das Schutzkriterium k-assign Anonymität mit $k = 4$, erfüllen alle neuen Tupel der Cluster C_2 und C_3 das Kriterium. Die beiden zurückgestellten Tupel 10 und 11 können daher in Phase 2 des Algorithmus 7.6 nur Cluster C_1 zugeordnet werden. Aus $sim(10, C_1) = 4$ folgt, dass 10 in C_1 einsortiert wird. Der Anonymitätsgrad steigt dadurch nicht, denn $a^*(10) = 3$. Somit wird auch das letzte Tupel 11 C_1 zugeordnet. Das Ergebnis des Algorithmus lautet $C_1 = \{1, 3, 7, 9, 10, 11\}$, $C_2 = \{2, 5, 13, 14, 16\}$ und $C_3 = \{4, 6, 8, 12, 15\}$.

	7			8			9		
	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i
C_1	0	4	$\{1,3,7\}$	3	2	$\{1,3,7\}$	3	2	$\{1,3,7,9\}$
C_2	0	0	$\{2,5\}$	0	0	$\{2,5\}$	0	0	$\{2,5\}$
C_3	0	0	$\{4,6\}$	0	4	$\{4,6,8\}$	3	$-\infty$	$\{4,6,8\}$

	10/11			12			13		
	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i
C_1	3	0	$\{1,3,7,9\}$	3	0	$\{1,3,7,9\}$	3	0	$\{1,3,7,9\}$
C_2	0	0	$\{2,5\}$	0	2	$\{2,5\}$	0	2	$\{2,5,13\}$
C_3	3	0	$\{4,6,8\}$	3	4	$\{4,6,8,12\}$	4	$-\infty$	$\{4,6,8,12\}$

	14			15			16		
	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i	$a^*_{\min}$	sim_0	C_i
C_1	3	0	$\{1,3,7,9\}$	3	0	$\{1,3,7,9\}$	3	$-\infty$	$\{1,3,7,9\}$
C_2	2	4	$\{2,5,13,14\}$	3	0	$\{2,5,13,14\}$	3	2	$\{2,5,13,14,16\}$
C_3	4	0	$\{4,6,8,12\}$	4	4	$\{4,6,8,12,15\}$	5	$-\infty$	$\{4,6,8,12,15\}$

	10			11		
	$a^*_{\min}$	sim	C_i	$a^*_{\min}$	sim	C_i
C_1	3	4	$\{1,3,7,9,10\}$	3	6	$\{1,3,7,9,10,11\}$
C_2	4	–	$\{2,5,13,14,16\}$	4	–	$\{2,5,13,14,16\}$
C_3	5	–	$\{4,6,8,12,15\}$	5	–	$\{4,6,8,12,15\}$

Tabelle A.8: Clustern von neuen Tupeln nach Algorithmus 7.6

B Zusätzliche Beweise

In diesem Teil des Anhangs werden zwei zusätzliche Beweise geführt.

Proposition B.1 *Seien $\mathcal{G} = (G_1, \ldots, G_n)$ eine Folge von Anfragegraphen und $\mathcal{M}, \mathcal{M}'$ zwei valide Matchings in $\mathcal{G}$. Dann ist $\mathcal{M}$ genau dann ein Δ-minimales valides Matching bezüglich $\mathcal{M}'$, wenn $\mathcal{M}'$ ein Δ-minimales valides Matching bezüglich $\mathcal{M}$ ist.*

Beweis: Angenommen, $\mathcal{M}$ ist ein Δ-minimales Matching bezüglich $\mathcal{M}'$, aber nicht umgekehrt. Somit gibt es nach Definition 4.4 ein valides Matching $\mathcal{M}^* \neq \mathcal{M}$ mit $\mathcal{M}^* \,\Delta\, \mathcal{M} \subset \mathcal{M}' \,\Delta\, \mathcal{M}$. Mithilfe des Venn-Diagramms in Abbildung B.1 kann diese Bedingung zu $\mathcal{M}^* \,\Delta\, \mathcal{M} = a \,\dot\cup\, c \,\dot\cup\, e \,\dot\cup\, f \subset a \,\dot\cup\, b \,\dot\cup\, d \,\dot\cup\, f = \mathcal{M}' \,\Delta\, \mathcal{M}$ umgeformt werden. Nun gilt $c = e = \emptyset$ und da $\mathcal{M}^* \neq \mathcal{M}$, folgt $\mathcal{M}^* \,\Delta\, \mathcal{M} \neq \emptyset$. Ferner ist

$$\begin{aligned} \mathcal{M} \,\Delta\, \mathcal{M}' - \mathcal{M} \,\Delta\, \mathcal{M}^* &= (a \,\dot\cup\, b \,\dot\cup\, d \,\dot\cup\, f) - (a \,\dot\cup\, c \,\dot\cup\, e \,\dot\cup\, f) \\ &= (b \,\dot\cup\, d) \\ &= \mathcal{M}^* \,\Delta\, \mathcal{M}' \end{aligned}$$

und folglich $\mathcal{M}^* \,\Delta\, \mathcal{M}' \subset \mathcal{M} \,\Delta\, \mathcal{M}'$. Da aber $\mathcal{M}^* \neq \mathcal{M}'$ ist, widerspricht diese Beziehung der Annahme, dass $\mathcal{M}$ Δ-minimal bezüglich $\mathcal{M}'$ ist. □

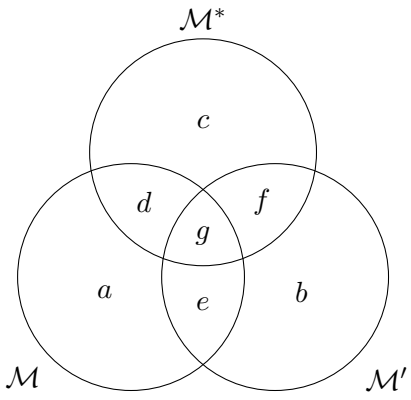


Abbildung B.1: Schnittmengen der drei Matchings $\mathcal{M}, \mathcal{M}'$ und $\mathcal{M}^*$ als Venn-Diagramm

Das folgende Lemma wird für den Beweis von Lemma 6.12 auf Seite 187 benötigt.

Lemma B.2 *Seien M ein eindeutiges Matching in einem Klongraphen, $S_P = \{P_1, \ldots, P_t\}$ eine (inklusions-)maximale streng eindeutig M-augmentierende Menge knotendisjunkter Pfade, Q ein eindeutig augmentierender Pfad in $M' \coloneqq M \,\Delta\, E(P_1) \,\Delta \cdots \Delta\, E(P_t)$. Wenn Q knotendisjunkt zu $P_1, \ldots, P_t$ und eindeutig M-augmentierend ist, dann ist $S_P \cup \{Q\}$ streng eindeutig M-augmentierend.*

Beweis: Nach der Definition der eindeutig augmentierenden Pfade (siehe Definition 6.4 auf Seite 180 und Definition 6.5 auf Seite 185) folgt: es existieren keine Klonkanten in

1. $E_{\bar{M}}(Q) \cup M \setminus E_M(Q)$ (da Q eindeutig M-augmentierend),
2. $E_{\bar{M'}}(Q) \cup M' \setminus E_{M'}(Q)$ (da Q eindeutig M'-augmentierend) und
3. $E_{\bar{M}}(S_P) \cup M \setminus E_M(S_P)$ (da S_P eindeutig M-augmentierend).

Wenn $S_P \cup \{Q\}$ nicht eindeutig M-augmentierend ist, muss es zwei Klonkanten e_1 und e_2 in $E_{\bar{M}}(S_P \cup \{Q\}) \cup M \setminus E_M(S_P \cup \{Q\})$ geben. Da M eindeutig ist, muss eine der beiden Kanten in $E_{\bar{M}}(S_P \cup \{Q\})$ vorkommen. Seien im 1. Fall beide Kanten in $E_{\bar{M}}(S_P \cup \{Q\})$, dann gibt es insgesamt drei verschiedene Möglichkeiten:

- $e_1, e_2 \in E_{\bar{M}}(S_P)$: widerspricht 3,
- $e_1, e_2 \in E_{\bar{M}}(Q)$: widerspricht 1,
- $e_1 \in E_{\bar{M}}(S_P), e_2 \in E_{\bar{M}}(Q)$: daraus folgt $e_1 \in M'$ und, da Q knotendisjunkt zu allen Pfaden aus S_P ist, $e_2 \in E_{\bar{M'}}(Q)$, was 2 widerspricht.

Im 2. Fall gelte $e_1 \in E_{\bar{M}}(S_P \cup \{Q\})$ und $e_2 \in M \setminus E_M(S_P \cup \{Q\})$, dann gibt es hier zwei weitere Möglichkeiten:

- $e_1 \in E_{\bar{M}}(S_P), e_2 \in M \setminus E_M(S_P \cup \{Q\})$: widerspricht 3,
- $e_1 \in E_{\bar{M}}(Q), e_2 \in M \setminus E_M(S_P \cup \{Q\})$: widerspricht 1.

Da Q eindeutig M-augmentierend ist, muss damit $S_P \cup \{Q\}$ streng eindeutig M-augmentierend sein. □

Literaturverzeichnis

[1] Adam, N. R. ; Worthmann, J. C.: Security-control Methods for Statistical Databases: A Comparative Study. In: *ACM Computing Surveys* 21 (1989), Dez., Nr. 4, S. 515–556

[2] Aggarwal, G. ; Feder, T. ; Kenthapadi, K. ; Motwani, R. ; Panigrahy, R. ; Thomas, D. ; Zhu, A. : Approximation Algorithms for k-Anonymity. In: *Journal of Privacy Technology* (2005), Nov.

[3] Alliance for Telecommunication Industry Solutions (ATIS): *ATIS Telecom Glossary 2011.* Version: 2011. `http://www.atis.org/glossary`, Abruf: 24.02.2015

[4] Bacher, J. ; Pöge, A. ; Wenzig, K. : *Clusteranalyse: Anwendungsorientierte Einführung in Klassifikationsverfahren.* 3. München : Oldenbourg Wissenschaftsverlag, 2010. – ISBN 978–3–486–58457–8

[5] Bachmann, P. G. H.: *Zahlentheorie.* Bd. 2: *Die Analytische Zahlentheorie.* Leipzig : B. G. Teubner, 1894

[6] Barták, R. : Constraint Programming: In Pursuit of the Holy Grail. In: *WDS '99: Proceedings of Week of Doctoral Students*, 1999, S. 555–564

[7] Bayardo, R. J. ; Agrawal, R. : Data Privacy through Optimal k-Anonymization. In: Aberer, K. (Hrsg.) ; Franklin, M. J. (Hrsg.) ; Nishio, S. (Hrsg.): *ICDE '05: Proceedings of the 21st International Conference on Data Engineering.* Washington, DC, USA : IEEE Computer Society, Apr. 2005. – ISBN 0–7695–2285–8, S. 217–228

[8] Berge, C. : Two Theorems in Graph Theory. In: *Proceedings of the National Academy of Sciences of the United States of America* 43 (1957), Sept., Nr. 9, S. 842–844

[9] Blum, A. ; Dwork, C. ; McSherry, F. ; Nissim, K. : Practical privacy: the SuLQ framework. In: Li, C. (Hrsg.): *PODS '05: Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA : ACM, Jun. 2005. – ISBN 1–59593–062–0, S. 128–138

[10] Blum, A. ; Ligett, K. ; Roth, A. : A Learning Theory Approach to Non-Interactive Database Privacy. In: Dwork, C. (Hrsg.): *STOC '08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing.* New York, NY, USA : ACM, Mai 2008. – ISBN 978–1–60558–047–0, S. 609–618

[11] Bondy, J.-A. ; Murty, U. S. R.: *Graph Theory.* New York, London : Springer, 2008. – ISBN 978–1–846–28969–9

[12] Bundesrepublik Deutschland: *Bundesdatenschutzgesetz*. Version: 2010. `http://dejure.org/gesetze/bdsg`, Abruf: 24.02.2015

[13] Bundesrepublik Deutschland: *Grundgesetz*. Version: 2012. `https://www.bundestag.de/grundgesetz`, Abruf: 24.02.2015

[14] Byun, J.-W. ; Kamra, A. ; Bertino, E. ; Li, N. : Efficient k-Anonymization Using Clustering Techniques. In: Ramamohanarao, K. (Hrsg.) ; Krishna, P. R. (Hrsg.) ; Mohania, M. K. (Hrsg.) ; Nantajeewarawat, E. (Hrsg.): *DASFAA '07: Advances in Databases: Concepts, Systems and Applications, Proceedings of the 12th International Conference on Database Systems for Advanced Applications* Bd. 4443, Springer, Apr. 2007 (Lecture Notes in Computer Science). – ISBN 978–3–540–71702–7, S. 188–200

[15] Byun, J.-W. ; Sohn, Y. ; Bertino, E. ; Li, N. : Secure Anonymization for Incremental Datasets. In: Jonker, W. (Hrsg.) ; Petkovic, M. (Hrsg.): *SDM '06: Secure Data Management, Proceedings of the 3rd VLDB Workshop on Secure Data Management* Bd. 4165. Berlin, Heidelberg : Springer, Sept. 2006 (Lecture Notes in Computer Science). – ISBN 978–3–540–38984–2, S. 48–63

[16] Campan, A. ; Truta, T. M. ; Cooper, N. : p-Sensitive k-Anonymity with Generalization Constraints. In: *Transactions on Data Privacy* 3 (2010), Nr. 2, S. 65–89

[17] Campan, A. ; Truta, T. M. ; Miller, J. ; Sinca, R. : A Clustering Approach for Achieving Data Privacy. In: Stahlbock, R. (Hrsg.) ; Crone, S. F. (Hrsg.) ; Lessmann, S. (Hrsg.): *DMIN '07: Proceedings of the 2007 International Conference on Data Mining*, CSREA Press, Jun. 2007. – ISBN 1–60132–031–0, S. 321–330

[18] Cao, J. ; Karras, P. : Publishing Microdata with a Robust Privacy Guarantee. In: *Proceedings of the VLDB Endowment* 5 (2012), Jul., Nr. 11, S. 1388–1399. – ISSN 2150–8097

[19] Chawla, S. ; Dwork, C. ; McSherry, F. ; Smith, A. ; Wee, H. : Toward Privacy in Public Databases. In: Kilian, J. (Hrsg.): *TCC '05: Proceedings of the Second Theory of Cryptography Conference* Bd. 3378. Berlin, Heidelberg : Springer, Febr. 2005 (Lecture Notes in Computer Science). – ISBN 3–540–24573–1, S. 363–385

[20] Chen, B.-C. ; LeFevre, K. ; Ramakrishnan, R. : Privacy Skyline: Privacy with Multidimensional Adversarial Knowledge. In: Koch, C. (Hrsg.) ; Gehrke, J. (Hrsg.) ; Garofalakis, M. N. (Hrsg.) ; Srivastava, D. (Hrsg.) ; Aberer, K. (Hrsg.) ; Deshpande, A. (Hrsg.) ; Florescu, D. (Hrsg.) ; Chan, C. Y. (Hrsg.) ; Ganti, V. (Hrsg.) ; Kanne, C.-C. (Hrsg.) ; Klas, W. (Hrsg.) ; Neuhold, E. J. (Hrsg.): *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, ACM, Sept. 2007. – ISBN 978–1–59593–649–3, S. 770–781

[21] Chin, F. Y. L.: Security problems on inference control for SUM, MAX, and MIN queries. In: *Journal of the Association for Computing Machinery (JACM)* 33 (1986), Jul., Nr. 3, S. 451–464. – ISSN 0004–5411

[22] Chin, F. Y. L. ; Ozsoyoglu, G. : Auditing for Secure Statistical Databases. In: Levy, B. (Hrsg.): *ACM '81: Proceedings of the ACM '81 Conference.* New York, NY, USA : ACM, Nov. 1981. – ISBN 0–89791–049–4, S. 53–59

[23] Chvátal, V. : *Linear Programming.* Freeman & Company, 1983 (A Series of books in the mathematical sciences). – ISBN 978–0–7167–1195–7

[24] Ciriani, V. ; De Capitani di Vimercati, S. ; Foresti, S. ; Samarati, P. : Microdata Protection. In: Yu, T. (Hrsg.) ; Jajodia, S. (Hrsg.): *Secure Data Management in Decentralized Systems* Bd. 33. New York, NY, USA : Springer Science+Business Media, LLC, 2007. – ISBN 978–0–387–27694–6, S. 291–321

[25] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. In: *Communications of the ACM (CACM)* 13 (1970), Jun., Nr. 6, S. 377–387. – ISSN 0001–0782

[26] Cook, S. A.: The Complexity of Theorem-Proving Procedures. In: Harrison, M. A. (Hrsg.) ; Banerji, R. B. (Hrsg.) ; Ullman, J. D. (Hrsg.): *STOC '71: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing.* New York, NY, USA : ACM, Mai 1971, S. 151–158

[27] Cormen, T. H. ; Leiserson, C. E. ; Rivest, R. L. ; Stein, C. : *Introduction To Algorithms.* 3. The MIT Press, 2009. – ISBN 978–0–262033848

[28] Dantzig, G. B.: *Linear programming and extensions.* Princeton, NJ : Princeton University Press, 1963 (Rand Corporation Research Study). – ISBN 978–0–691–08000–0

[29] Denning, D. E. ; Denning, P. J. ; Schwartz, M. D.: The Tracker: A Threat to Statistical Database Security. In: *ACM Transactions on Database Systems (TODS)* 4 (1979), März, Nr. 1, S. 76–96. – ISSN 0362–5915

[30] Denning, D. E. ; Schlörer, J. : A Fast Procedure for Finding a Tracker in a Statistical Database. In: *ACM Transactions on Database Systems (TODS)* 5 (1980), März, Nr. 1, S. 88–102. – ISSN 0362–5915

[31] Deutsch, A. ; Papakonstantinou, Y. : Privacy in Database Publishing. In: Eiter, T. (Hrsg.) ; Libkin, L. (Hrsg.): *ICDT '05: Proceedings of the 10th International Conference on Database Theory* Bd. 3363, Springer, Jan. 2005 (Lecture Notes in Computer Science). – ISBN 3–540–24288–0, S. 230–245

[32] Deutsches Institut für Medizinische Dokumentation und Information (DIMDI): *Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme (ICD-10-GM).* Version: 2015. `http://www.dimdi.de/static/de/klassi/icd-10-gm/kodesuche/onlinefassungen/htmlgm2015/index.htm`, Abruf: 24.02.2015

[33] Deutsches Institut für Medizinische Dokumentation und Information (DIMDI): *Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme (ICD-10-GM), Kapitel X: Krankheiten des Atmungssystems (J00–J99).* Version: 2015. `http://www.dimdi.de/static/de/klassi/icd-10-gm/kodesuche/onlinefassungen/htmlgm2015/chapter-x.htm`, Abruf: 24.02.2015

[34] Deutsches Institut für Medizinische Dokumentation und Information (DIMDI): *Internationale statistische Klassifikation der Krankheiten und verwandter Gesundheitsprobleme (ICD-10-GM), Kapitel XI: Krankheiten des Verdauungssystems (K00–K93).* Version: 2015. `http://www.dimdi.de/static/de/klassi/icd-10-gm/kodesuche/onlinefassungen/htmlgm2015/chapter-xi.htm`, Abruf: 24.02.2015

[35] Diestel, R. : *Graphentheorie.* 4. Heidelberg : Springer, 2010. – ISBN 978–3–642–14911–5

[36] Dinur, I. ; Nissim, K. : Revealing Information while Preserving Privacy. In: Neven, F. (Hrsg.) ; Beeri, C. (Hrsg.) ; Milo, T. (Hrsg.): *PODS '03: Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems.* New York, NY, USA : ACM, Jun. 2003. – ISBN 1–58113–670–6, S. 202–210

[37] Dobkin, D. P. ; Jones, A. K. ; Lipton, R. J.: Secure Databases: Protection Against User Influence. In: *ACM Transactions on Database Systems (TODS)* 4 (1979), März, Nr. 1, S. 97–106. – ISSN 0362–5915

[38] Domingo-Ferrer, J. ; Sebé, F. ; Solanas, A. : An Anonymity Model Achievable Via Microaggregation. In: Jonker, W. (Hrsg.) ; Petkovic, M. (Hrsg.): *SDM '08: Secure Data Management, 5th VLDB Workshop* Bd. 5159, Springer, Aug. 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–85258–2, S. 209–218

[39] Du, W. ; Teng, Z. ; Zhu, Z. : Privacy-MaxEnt: Integrating Background Knowledge in Privacy Quantification. In: Wang, J. T.-L. (Hrsg.): *SIGMOD '08: Proceedings of the ACM SIGMOD International Conference on Management of Data.* New York, NY, USA : ACM, Jun. 2008. – ISBN 978–1–60558–102–6, S. 459–472

[40] Duden ; Dudenredaktion (Hrsg.): *Duden – Die deutsche Rechtschreibung.* Bd. 1. 26. Mannheim : Bibliographisches Institut, 2013. – ISBN 978–3–411–04016–2

[41] Dwork, C. : Differential Privacy. In: Bugliesi, M. (Hrsg.) ; Preneel, B. (Hrsg.) ; Sassone, V. (Hrsg.) ; Wegener, I. (Hrsg.): *ICALP '06: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming* Bd. 4052. Berlin, Heidelberg : Springer, Jul. 2006 (Lecture Notes in Computer Science). – ISBN 3–540–35907–9, S. 1–12

[42] Dwork, C. : Differential Privacy: A Survey of Results. In: Agrawal, M. (Hrsg.) ; Du, D.-Z. (Hrsg.) ; Duan, Z. (Hrsg.) ; Li, A. (Hrsg.): *TAMC '08: Proceedings of the 5th International Conference on Theory and Applications of Models of Computation* Bd. 4978, Springer, Apr. 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–79227–7, S. 1–19

[43] Dwork, C. : A Firm Foundation for Private Data Analysis. In: *Communications of the ACM (CACM)* 54 (2011), Jan., Nr. 1, S. 86–95. – ISSN 0001–0782

[44] Elmasri, R. ; Navathe, S. B.: *Grundlagen von Datenbanksystemen.* 3. München : Pearson Studium, 2002. – ISBN 978–3–8273–7021–1

[45] Emden-Weinert, T. ; Hougardy, S. ; Kreuter, B. ; Prömel, H. J. ; Steger, A. : *Einführung in Graphen und Algorithmen*. Version: Sept. 1996. `http://www.or.uni-bonn.de/~hougardy/paper/ga.pdf`, Abruf: 24.02.2015

[46] Europäische Union: *Richtlinie 95/46/EG des Europäischen Parlaments und des Rates*. Version: 1995. `http://eur-lex.europa.eu/legal-content/DE/TXT/?uri=CELEX:31995L0046`, Abruf: 24.02.2015

[47] Evfimievski, A. V. ; Fagin, R. ; Woodruff, D. P.: Epistemic Privacy. In: Lenzerini, M. (Hrsg.) ; Lembo, D. (Hrsg.): *PODS' 08: Proceedings of the 27th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. New York, NY, USA : ACM, Jun. 2008. – ISBN 978–1–60558–108–8, S. 171–180

[48] Federal Register: *The Health Insurance Portability and Accountability Act (HIPAA) – Privacy Rule*. Version: 2002. `http://www.hhs.gov/ocr/privacy/hipaa/administrative/privacyrule/index.html`, Abruf: 24.02.2015

[49] Federal Trade Commission: *Children's Online Privacy Protection Act (COPPA)*. Version: 1998. `http://www.coppa.org`, Abruf: 24.02.2015

[50] Fischer, G. : *Lineare Algebra: Ein Einführung für Studienanfänger*. 18. Wiesbaden : Springer Spektrum, 2013. – ISBN 978–3–658–03944–8

[51] Fouad, M. R. ; Lebanon, G. ; Bertino, E. : ARUBA: A Risk-Utility-Based Algorithm for Data Disclosure. In: Jonker, W. (Hrsg.) ; Petkovic, M. (Hrsg.): *SDM '08: Secure Data Management, 5th VLDB Workshop* Bd. 5159, Springer, Aug. 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–85258–2, S. 32–49

[52] Frank, A. ; Asuncion, A. : *UCI Machine Learning Repository*. Version: 2013. `http://archive.ics.uci.edu/ml`, Abruf: 24.02.2015

[53] Frikken, K. B. ; Zhang, Y. : Yet another privacy metric for publishing micro-data. In: Atluri, V. (Hrsg.) ; Winslett, M. (Hrsg.): *WPES '08: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society*. New York, NY, USA : ACM, Okt. 2008. – ISBN 978–1–60558–289–4, S. 117–122

[54] Fung, B. C. M. ; Wang, K. ; Chen, R. ; Yu, P. S.: Privacy-preserving data publishing: A survey of recent developments. In: *ACM Comput. Surv.* 42 (2010), Jun., Nr. 4, S. 14:1–14:53. – ISSN 0360–0300

[55] Fung, B. C. M. ; Wang, K. ; Fu, A. W.-C. ; Pei, J. : Anonymity for Continuous Data Publishing. In: Kemper, A. (Hrsg.) ; Valduriez, P. (Hrsg.) ; Mouaddib, N. (Hrsg.) ; Teubner, J. (Hrsg.) ; Bouzeghoub, M. (Hrsg.) ; Markl, V. (Hrsg.) ; Amsaleg, L. (Hrsg.) ; Manolescu, I. (Hrsg.): *EDBT '08: Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology* Bd. 261. New York, NY, USA : ACM, März 2008. – ISBN 978–1–59593–926–5, S. 264–275

[56] Fung, B. C. M. ; Wang, K. ; Yu, P. S.: Top-Down Specialization for Information and Privacy Preservation. In: Aberer, K. (Hrsg.) ; Franklin, M. J. (Hrsg.) ; Nishio, S. (Hrsg.): *ICDE '05: Proceedings of the 21st International Conference on*

Data Engineering. Washington, DC, USA : IEEE Computer Society, Apr. 2005. – ISBN 0–7695–2285–8, S. 205–216

[57] Fung, B. C. M. ; Wang, K. ; Yu, P. S.: Anonymizing Classification Data for Privacy Preservation. In: *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 19 (2007), Mai, Nr. 5, S. 711–725. – ISSN 1041–4347

[58] Garcia-Molina, H. ; Ullman, J. D. ; Widom, J. : *Database Systems: The Complete Book*. 2. Upper Saddle River, NJ, USA : Prentice Hall, 2008. – ISBN 978–0–13–187325–4

[59] General Services Administration: *Federal Standard 1037C, Telecommunications: Glossary of Telecommunication Terms*. Version: 1996. `http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm`, Abruf: 24.02.2015

[60] Ghinita, G. ; Karras, P. ; Kalnis, P. ; Mamoulis, N. : Fast Data Anonymization with Low Information Loss. In: Koch, C. (Hrsg.) ; Gehrke, J. (Hrsg.) ; Garofalakis, M. N. (Hrsg.) ; Srivastava, D. (Hrsg.) ; Aberer, K. (Hrsg.) ; Deshpande, A. (Hrsg.) ; Florescu, D. (Hrsg.) ; Chan, C. Y. (Hrsg.) ; Ganti, V. (Hrsg.) ; Kanne, C.-C. (Hrsg.) ; Klas, W. (Hrsg.) ; Neuhold, E. J. (Hrsg.): *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, ACM, Sept. 2007. – ISBN 978–1–59593–649–3, S. 758–769

[61] Goldberger, J. ; Tassa, T. : Efficient Anonymizations with Enhanced Utility. In: Saygin, Y. (Hrsg.) ; Yu, J. X. (Hrsg.) ; Kargupta, H. (Hrsg.) ; Wang, W. (Hrsg.) ; Ranka, S. (Hrsg.) ; Yu, P. S. (Hrsg.) ; Wu, X. (Hrsg.): *ICDM Workshops 2009, IEEE International Conference on Data Mining Workshops*, IEEE Computer Society, Dez. 2009. – ISBN 978–0–7695–3902–7, S. 106–113

[62] Grötschel, M. ; Holland, O. : Solving matching problems with linear programming. In: *Mathematical Programming* 33 (1985), Nr. 3, S. 243–259

[63] Hartigan, J. A.: *Clustering Algorithms*. John Wiley and Sons, 1975 (Wiley series in probability and mathematical statistics: Applied probability and statistics). – ISBN 978–0–471–35645–5

[64] Heller, I. ; Tomkins, C. B.: An Extension of a Theorem of Dantzig. In: Kuhn, H. W. (Hrsg.) ; Tucker, A. W. (Hrsg.): *Linear Inequalities and Related Systems* Bd. 38. Princeton, NJ, USA : Princeton University Press, 1956. – ISBN 978–0–691–07999–8, S. 247–254

[65] Helmberg, C. : *Diskrete Optimierung (Einführung zur Vorlesung)*. Version: 2010. `http://www.tu-chemnitz.de/mathematik/discrete/lehre/ganzz/s10/overview.pdf`, Abruf: 24.02.2015

[66] Hoffman, A. J. ; Kruskal, J. B.: Integral Boundary Points of Convex Polyhedra. In: Kuhn, H. W. (Hrsg.) ; Tucker, A. W. (Hrsg.): *Linear Inequalities and Related Systems* Bd. 38. Princeton, NJ, USA : Princeton University Press, 1956. – ISBN 978–0–691–07999–8, S. 223–246

[67] Hopcroft, J. E. ; Karp, R. M.: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs. In: *SIAM Journal on Computing* 2 (1973), Nr. 4, S. 225–231

[68] Hougardy, S. : *Graphen und Algorithmen 1, Skript zur Vorlesung im Wintersemester 2005/2006.* Version: Febr. 2006. `http://www2.informatik.hu-berlin.de/~hougardy/paper/skripte/ga1.pdf`, Abruf: 12.01.2010

[69] Iwuchukwu, T. ; Naughton, J. F.: K-Anonymization as Spatial Indexing: Toward Scalable and Incremental Anonymization. In: Koch, C. (Hrsg.) ; Gehrke, J. (Hrsg.) ; Garofalakis, M. N. (Hrsg.) ; Srivastava, D. (Hrsg.) ; Aberer, K. (Hrsg.) ; Deshpande, A. (Hrsg.) ; Florescu, D. (Hrsg.) ; Chan, C. Y. (Hrsg.) ; Ganti, V. (Hrsg.) ; Kanne, C.-C. (Hrsg.) ; Klas, W. (Hrsg.) ; Neuhold, E. J. (Hrsg.): *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, ACM, Sept. 2007. – ISBN 978–1–59593–649–3, S. 746–757

[70] Iyengar, V. S.: Transforming data to satisfy privacy constraints. In: *KDD '02: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* New York, NY, USA : ACM, Jul. 2002. – ISBN 1–58113–567–X, S. 279–288

[71] Jansen, K. ; Margraf, M. : *Approximative Algorithmen und Nichtapproximierbarkeit.* Berlin : De Gruyter, 2008. – ISBN 978–3–11–020316–5

[72] Jiang, W. ; Clifton, C. : Privacy-Preserving Distributed k-Anonymity. In: Jajodia, S. (Hrsg.) ; Wijesekera, D. (Hrsg.): *Data and Application Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security* Bd. 3654. Berlin, Heidelberg : Springer, Aug. 2005 (Lecture Notes in Computer Science). – ISBN 978–3–540–28138–2, S. 166–177

[73] Jura Forum: *Schutz der Privatsphäre.* Version: 2013. `http://www.juraforum.de/lexikon/schutz-der-privatsphaere`, Abruf: 24.02.2015

[74] Kann, V. : Maximum Bounded 3-dimensional Matching is MAX SNP-complete. In: *Inf. Process. Lett.* 37 (1991), Jan., Nr. 1, S. 27–35

[75] Karjoth, G. : Sind anonymisierte Daten anonym genug? In: *digma – Zeitschrift für Datenrecht und Informationssicherheit* 8 (2008), März, Nr. 1, S. 18–23. – ISSN 1424–9944

[76] Karmarkar, N. K.: A New Polynomial Time Algorithm for Linear Programming. In: *Combinatorica* 4 (1984), Nr. 4, S. 373–396

[77] Karp, R. M.: Reducibility Among Combinatorial Problems. In: Miller, R. E. (Hrsg.) ; Thatcher, J. W. (Hrsg.): *Proceedings of a symposium on the Complexity of Computer Computations.* New York : Plenum Press, März 1972 (The IBM Research Symposia Series), S. 85–103

[78] Kaufman, L. ; Rousseeuw, P. J.: *Finding Groups in Data: An Introduction to Cluster Analysis.* 1. New York : John Wiley and Sons, 2005. – ISBN 978–0–471–73578–6

[79] Kemper, A. ; Eickler, A. : *Datenbanksysteme: Eine Einführung*. 9. München : Oldenbourg Verlag, 2013. – ISBN 978–3–486–72139–3

[80] Kenthapadi, K. ; Mishra, N. ; Nissim, K. : Simulatable Auditing. In: Li, C. (Hrsg.): *PODS '05: Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. New York, NY, USA : ACM, Jun. 2005. – ISBN 1–59593–062–0, S. 118–127

[81] Khachiyan, L. G.: A Polynomial Algorithm in Linear Programming. In: *Doklady Akademii Nauk SSSR* 244 (1979), Nr. 5, S. 1093–1096. – English translation in [82]

[82] Khachiyan, L. G.: A Polynomial Algorithm in Linear Programming. In: *Soviet Mathematics Doklady* 20 (1979), S. 191–194. – English translation from [81]

[83] Kleinberg, J. M. ; Papadimitriou, C. H. ; Raghavan, P. : Auditing Boolean attributes. In: *Journal of Computer and System Sciences* 66 (2003), Febr., Nr. 1, S. 244–253. – ISSN 0022–0000

[84] Köbler, J. : *Graphalgorithmen*. Version: Jul. 2013. `http://www.informatik.hu-berlin.de/forschung/gebiete/algorithmenII/Lehre/ss13/graphalgo/skript/ga-skript.pdf`, Abruf: 24.02.2015

[85] Korte, B. ; Vygen, J. : *Kombinatorische Optimierung: Theorie und Algorithmen*. 2. Berlin, Heidelberg : Springer, 2012 (Springer-Lehrbuch Masterclass). – ISBN 978–3–642–25400–0

[86] Kuhn, H. W.: The Hungarian Method for the Assignment Problem. In: *Naval Research Logistics Quarterly* 2 (1955), März, S. 83–97

[87] Lambert, D. : Measures of Disclosure Risk and Harm. In: *Journal of Official Statistics* 9 (1993), Mai, Nr. 2, S. 313–331

[88] Landau, E. G. H.: *Handbuch der Lehre von der Verteilung der Primzahlen*. Bd. 2. Leipzig : B. G. Teubner, 1909

[89] LeFevre, K. ; DeWitt, D. J. ; Ramakrishnan, R. : Incognito: Efficient Full-Domain k-Anonymity. In: Özcan, F. (Hrsg.): *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of Data*. New York, NY, USA : ACM, Jun. 2005. – ISBN 1–59593–060–4, S. 49–60

[90] LeFevre, K. ; DeWitt, D. J. ; Ramakrishnan, R. : Mondrian Multidimensional K-Anonymity. In: Liu, L. (Hrsg.) ; Reuter, A. (Hrsg.) ; Whang, K.-Y. (Hrsg.) ; Zhang, J. (Hrsg.): *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*. Los Alamitos, California, USA : IEEE Computer Society, Apr. 2006. – ISBN 0–7695–2570–9, S. 25

[91] LeFevre, K. ; DeWitt, D. J. ; Ramakrishnan, R. : Workload-Aware Anonymization. In: Eliassi-Rad, T. (Hrsg.) ; Ungar, L. H. (Hrsg.) ; Craven, M. (Hrsg.) ; Gunopulos, D. (Hrsg.): *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA : ACM, Aug. 2006. – ISBN 1–59593–339–5, S. 277–286

[92] Li, J. ; Tao, Y. ; Xiao, X. : Preservation of proximity privacy in publishing numerical sensitive data. In: Wang, J. T.-L. (Hrsg.): *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2008. – ISBN 978–1–60558–102–6, S. 473–486

[93] Li, N. ; Li, T. ; Venkatasubramanian, S. : t-Closeness: Privacy Beyond k-Anonymity and ℓ-Diversity. In: Chirkova, R. (Hrsg.) ; Dogac, A. (Hrsg.) ; Özsu, M. T. (Hrsg.) ; Sellis, T. K. (Hrsg.): *ICDE '07: Proceedings of the 23rd International Conference on Data Engineering*. Los Alamitos, California, USA : IEEE Computer Society, Apr. 2007. – ISBN 1–4244–0802–4, S. 106–115

[94] Li, N. ; Li, T. ; Venkatasubramanian, S. : Closeness: A New Privacy Measure for Data Publishing. In: *IEEE Transactions on Knowledge and Data Engineering* 22 (2010), Jul., Nr. 7, S. 943–956. – ISSN 1041–4347

[95] Liu, J. ; Wang, K. : On optimal anonymization for l^+-diversity. In: Li, F. (Hrsg.) ; Moro, M. M. (Hrsg.) ; Ghandeharizadeh, S. (Hrsg.) ; Haritsa, J. R. (Hrsg.) ; Weikum, G. (Hrsg.) ; Carey, M. J. (Hrsg.) ; Casati, F. (Hrsg.) ; Chang, E. Y. (Hrsg.) ; Manolescu, I. (Hrsg.) ; Mehrotra, S. (Hrsg.) ; Dayal, U. (Hrsg.) ; Tsotras, V. J. (Hrsg.): *ICDE '10: Proceedings of the 26th International Conference on Data Engineering*, IEEE, März 2010. – ISBN 978–1–4244–5444–0, S. 213–224

[96] Lloyd, S. P.: Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), März, Nr. 2, S. 129–137

[97] Löser, C. : *Schutz der Privatsphäre und Kernbereich privater Lebensgestaltung*. Version: 2007. `http://www.cloeser.org/ext/Schutz%20der%20Privatsph%E4re.pdf`, Abruf: 24.02.2015

[98] Lovász, L. ; Plummer, M. D.: *Matching Theory*. Providence, Rhode Island : American Mathematical Society Chelsea Publishing, 2009. – ISBN 978–0–8218–4759–6

[99] Lu, H. ; Li, Y. ; Atluri, V. ; Vaidya, J. : An Efficient Online Auditing Approach to Limit Private Data Disclosure. In: Kersten, M. L. (Hrsg.) ; Novikov, B. (Hrsg.) ; Teubner, J. (Hrsg.) ; Polutin, V. (Hrsg.) ; Manegold, S. (Hrsg.): *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology* Bd. 360. New York, NY, USA : ACM, März 2009. – ISBN 978–1–60558–422–5, S. 636–647

[100] Machanavajjhala, A. ; Gehrke, J. : On the Efficiency of Checking Perfect Privacy. In: Vansummeren, S. (Hrsg.): *PODS '06: Proceedings of the 25th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. New York, NY, USA : ACM, Jun. 2006. – ISBN 1–59593–318–2, S. 163–172

[101] Machanavajjhala, A. ; Gehrke, J. ; Kifer, D. ; Venkitasubramaniam, M. : ℓ-Diversity: Privacy Beyond k-Anonymity. In: Liu, L. (Hrsg.) ; Reuter, A. (Hrsg.) ; Whang, K.-Y. (Hrsg.) ; Zhang, J. (Hrsg.): *ICDE '06: Proceedings of the 22nd IEEE International Conference on Data Engineering*, IEEE Computer Society, Apr. 2006, S. 24

[102] Machanavajjhala, A. ; Gehrke, J. ; Kifer, D. ; Venkitasubramaniam, M. : ℓ-Diversity: Privacy Beyond k-Anonymity. In: *ACM Trans. Knowl. Discov. Data* 1 (2007), März, Nr. 1

[103] MacQueen, J. B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Le Cam, L. M. (Hrsg.) ; Neyman, J. (Hrsg.): *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, University of California Press, 1967, S. 281–297

[104] Martin, D. J. ; Kifer, D. ; Machanavajjhala, A. ; Gehrke, J. ; Halpern, J. Y.: Worst-Case Background Knowledge for Privacy-Preserving Data Publishing. In: Chirkova, R. (Hrsg.) ; Dogac, A. (Hrsg.) ; Özsu, M. T. (Hrsg.) ; Sellis, T. K. (Hrsg.): *ICDE '07: Proceedings of the 23rd IEEE International Conference on Data Engineering*. Los Alamitos, California, USA : IEEE Computer Society, Apr. 2007. – ISBN 1–4244–0802–4, S. 126–135

[105] McSherry, F. : Privacy Integrated Queries: An Extensible Platform for Privacy-Preserving Data Analysis. In: Carsten Binnig, B. D. (Hrsg.): *SIGMOD '09: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2009. – ISBN 978–1–60558–551–2, S. 19–30

[106] Meyerson, A. ; Williams, R. : On the Complexity of Optimal K-Anonymity. In: Beeri, C. (Hrsg.) ; Deutsch, A. (Hrsg.): *PODS '04: Proceedings of the 23nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. New York, NY, USA : ACM, Jun. 2004, S. 223–228

[107] Microsoft Developer Network: *Microsoft Solver Foundation*. Version: 2015. `http://msdn.microsoft.com/en-us/library/ff524509%28v=vs.93%29.aspx`, Abruf: 24.02.2015

[108] Miklau, G. ; Suciu, D. : A Formal Analysis of Information Disclosure in Data Exchange. In: Weikum, G. (Hrsg.) ; König, A. C. (Hrsg.) ; Dessloch, S. (Hrsg.): *SIGMOD '04: Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2004. – ISBN 1–58113–859–8, S. 575–586

[109] n-tv.de: *Eingriffe in Privatsphäre – Milliardenklage gegen Facebook*. Version: Mai 2012. `http://www.n-tv.de/wirtschaft/Milliardenklage-gegen-Facebook-article6301881.html`, Abruf: 24.02.2015

[110] Nabar, S. U. ; Kenthapadi, K. ; Mishra, N. ; Motwani, R. : A Survey of Query Auditing Techniques for Data Privacy. In: Aggarwal, C. C. (Hrsg.) ; Yu, P. S. (Hrsg.): *Privacy-Preserving Data Mining – Models and Algorithms* Bd. 34. New York, NY, USA : Springer Science+Business Media, LLC, 2008. – ISBN 978–0–387–70991–8, S. 415–431

[111] Nabar, S. U. ; Marthi, B. ; Kenthapadi, K. ; Mishra, N. ; Motwani, R. : Towards Robustness in Query Auditing. In: Dayal, U. (Hrsg.) ; Whang, K.-Y. (Hrsg.) ; Lomet, D. B. (Hrsg.) ; Alonso, G. (Hrsg.) ; Lohman, G. M. (Hrsg.) ; Kersten, M. L. (Hrsg.) ; Cha, S. K. (Hrsg.) ; Kim, Y.-K. (Hrsg.): *VLDB '06:*

Proceedings of the 32nd International Conference on Very Large Data Bases, ACM, Sept. 2006. – ISBN 1–59593–385–9, S. 151–162

[112] Nergiz, M. E. ; Atzori, M. ; Clifton, C. : Hiding the Presence of Individuals from Shared Databases. In: Chan, C. Y. (Hrsg.) ; Ooi, B. C. (Hrsg.) ; Zhou, A. (Hrsg.): *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2007. – ISBN 978–1–59593–686–8, S. 665–676

[113] Nergiz, M. E. ; Clifton, C. ; Nergiz, A. E.: MultiRelational k-Anonymity. In: Chirkova, R. (Hrsg.) ; Dogac, A. (Hrsg.) ; Özsu, M. T. (Hrsg.) ; Sellis, T. K. (Hrsg.): *ICDE '07: Proceedings of the 23rd International Conference on Data Engineering*. Los Alamitos, California, USA : IEEE Computer Society, Apr. 2007. – ISBN 1–4244–0802–4, S. 1417–1421

[114] Nielsen, J. H.: *Verteilte Anonymisierung von vertikal partitionierten Daten*, Humboldt-Universität zu Berlin, Institut für Informatik, Diplomarbeit, März 2013

[115] Opt-4: *Opt-4 Data Protection Dictionary*. Version: 2015. `http://www.opt-4.co.uk/dictionary/default.asp`, Abruf: 24.02.2015

[116] Park, H. ; Shim, K. : Approximate Algorithms for k-Anonymity. In: Chan, C. Y. (Hrsg.) ; Ooi, B. C. (Hrsg.) ; Zhou, A. (Hrsg.): *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2007. – ISBN 978–1–59593–686–8, S. 67–78

[117] Pei, J. ; Xu, J. ; Wang, Z. ; Wang, W. ; Wang, K. : Maintaining K-Anonymity against Incremental Updates. In: *SSDBM '07: Proceedings of the 19th International Conference on Scientific and Statistical Database Management*. Washington, DC, USA : IEEE Computer Society, Jul. 2007. – ISBN 0–7695–2868–6, S. 5

[118] Peng, S. ; Yang, Y. ; Zhang, Z. ; Winslett, M. ; Yu, Y. : Query Optimization for Differentially Private Data Management Systems. In: Jensen, C. S. (Hrsg.) ; Jermaine, C. M. (Hrsg.) ; Zhou, X. (Hrsg.): *ICDE '13: Proceedings of the 29th IEEE International Conference on Data Engineering*. Los Alamitos, CA, USA : IEEE Computer Society, Apr. 2013. – ISBN 978–1–4673–4909–3, S. 1093–1104

[119] Rastogi, V. ; Hong, S. ; Suciu, D. : The Boundary Between Privacy and Utility in Data Publishing. In: Koch, C. (Hrsg.) ; Gehrke, J. (Hrsg.) ; Garofalakis, M. N. (Hrsg.) ; Srivastava, D. (Hrsg.) ; Aberer, K. (Hrsg.) ; Deshpande, A. (Hrsg.) ; Florescu, D. (Hrsg.) ; Chan, C. Y. (Hrsg.) ; Ganti, V. (Hrsg.) ; Kanne, C.-C. (Hrsg.) ; Klas, W. (Hrsg.) ; Neuhold, E. J. (Hrsg.): *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, ACM, Sept. 2007. – ISBN 978–1–59593–649–3, S. 531–42

[120] Reiss, S. P.: Security in Databases: A Combinatorial Study. In: *Journal of the Association for Computing Machinery (JACM)* 26 (1979), Jan., Nr. 1, S. 45–57. – ISSN 0004–5411

[121] Rubner, Y. ; Tomasi, C. ; Guibas, L. J.: The Earth Mover's Distance as a Metric for Image Retrieval. In: *IJCV: International Journal of Computer Vision* 40 (2000), Nov., Nr. 2, S. 99–121

[122] Russell, S. J. ; Norvig, P. : *Artificial Intelligence: A Modern Approach*. 3. Upper Saddle River, NJ, USA : Prentice Hall, 2009. – ISBN 978–0–13–604259–4

[123] Samarati, P. : Protecting Respondents' Identities in Microdata Release. In: *IEEE Transactions on Knowledge and Data Engineering* 13 (2001), Nov., Nr. 6, S. 1010–1027

[124] Samarati, P. ; Sweeney, L. : Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression. Computer Science Laboratory, SRI International, 1998 (SRI-CSL-98-04). – Forschungsbericht

[125] Schlörer, J. : Identification and Retrieval of Personal Records from a Statistical Data Bank. In: *Methods of Information in Medicine* 14 (1975), Jan., Nr. 1, S. 7–13. – ISSN 0026–1270

[126] Schöning, U. : *Theoretische Informatik – kurzgefasst*. 4. Heidelberg, Berlin : Spektrum Akademischer Verlag, 2001 (Hochschultaschenbuch). – ISBN 3–8274–1099–1

[127] Schönmann, F. : *Lineare Programmierung*. Version: 2003. `http://www.techfak.uni-bielefeld.de/ags/wbski/lehre/digiSA/WS0304/IntAlg/Ausarbeitungen/lp.pdf`, Abruf: 24.02.2015

[128] Schrijver, A. : *Theory of Linear and Integer Programming*. 1. Chichester : John Wiley & Sons, 1998 (Wiley Series in Discrete Mathematics & Optimization). – ISBN 978–0–471–98232–6

[129] Shannon, C. E.: Communication Theory of Secrecy Systems. In: *Bell System Technical Journal* 28 (1949), Okt., Nr. 4, S. 656–715. – ISSN 0005–8580

[130] Silberschatz, A. ; Korth, H. F. ; Sudarshan, S. : *Database System Concepts*. 6. McGraw-Hill, 2010. – ISBN 978–0–07–352332–3

[131] Spiegel Online: *Datenschutz: Europäischer Menschengerichtshof verbietet britische DNA-Konservierung*. Version: Dez. 2008. `http://www.spiegel.de/politik/ausland/datenschutz-europaeischer-menschengerichtshof-verbietet-britische-dna-konservierung-a-594436.html`, Abruf: 24.02.2015

[132] Spiegel Online: *Offengelegte Dokumente: NSA verletzte massiv Privatsphäre von Bürgern*. Version: Sept. 2013. `http://www.spiegel.de/netzwelt/netzpolitik/nsa-legt-dokumente-nach-eff-klage-offen-a-921549.html`, Abruf: 24.02.2015

[133] Sweeney, L. : Uniqueness of Simple Demographics in the U.S. Population. Carnegie Mellon University, Laboratory for International Data Privacy, 2000 (LIDAP-WP4). – Forschungsbericht

[134] Sweeney, L. : Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. In: *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10 (2002), Nr. 5, S. 571–588

[135] Sweeney, L. : k-Anonymity: A Model for Protecting Privacy. In: *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* 10 (2002), Nr. 5, S. 557–570

[136] taz.de: *Sieben Jahre ohne Privatleben.* Version: Okt. 2008. `http://www.taz.de/!23976`, Abruf: 24.02.2015

[137] Truta, T. M. ; Vinay, B. : Privacy Protection: p-Sensitive k-Anonymity Property. In: Barga, R. S. (Hrsg.) ; Zhou, X. (Hrsg.): *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops.* Washington, DC, USA : IEEE Computer Society, Apr. 2006, S. 94

[138] Turau, V. : *Algorithmische Graphentheorie.* 3. München : Oldenbourg Wissenschaftsverlag, 2009. – ISBN 978–3–486–59057–9

[139] Ullman, J. D.: *Principles of Database and Knowledge-Base Systems, Volume I: Classical Database Systems.* Bd. 1. 1. Computer Science Press, 1990. – ISBN 978–0–7167–8158–5

[140] Verein der Richter des Bundesverfassungsgerichts e. V.: *Entscheidungen des Bundesverfassungsgerichts (BVerfGE) 65, 1.* Version: Dez. 1983. `http://www.servat.unibe.ch/dfr/bv065001.html`, Abruf: 24.02.2015

[141] Verein der Richter des Bundesverfassungsgerichts e. V.: *Entscheidungen des Bundesverfassungsgerichts (BVerfGE) 101, 361.* Version: Dez. 1995. `http://www.servat.unibe.ch/dfr/bv101361.html`, Abruf: 24.02.2015

[142] Vimercati, S. De Capitani d. ; Foresti, S. ; Livraga, G. ; Samarati, P. : Protecting Privacy in Data Release. In: Aldini, A. (Hrsg.) ; Gorrieri, R. (Hrsg.): *Foundations of Security Analysis and Design VI – FOSAD Tutorial Lectures* Bd. 6858. Berlin, Heidelberg : Springer, 2011 (Lecture Notes in Computer Science). – ISBN 978–3–642–23081–3, S. 1–34

[143] Wang, K. ; Fung, B. C. M.: Anonymizing Sequential Releases. In: Eliassi-Rad, T. (Hrsg.) ; Ungar, L. H. (Hrsg.) ; Craven, M. (Hrsg.) ; Gunopulos, D. (Hrsg.): *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* New York, NY, USA : ACM, Aug. 2006. – ISBN 1–59593–339–5, S. 414–423

[144] Wang, K. ; Fung, B. C. M. ; Dong, G. : Integrating Private Databases for Data Analysis. In: Kantor, P. B. (Hrsg.) ; Muresan, G. (Hrsg.) ; Roberts, F. S. (Hrsg.) ; Zeng, D. D. (Hrsg.) ; Wang, F. (Hrsg.) ; Chen, H. (Hrsg.) ; Merkle, R. C. (Hrsg.): *Intelligence and Security Informatics, IEEE International Conference on Intelligence and Security Informatics, ISI 2005* Bd. 3495. Berlin, Heidelberg : Springer, Mai 2005 (Lecture Notes in Computer Science). – ISBN 978–3–540–25999–2, S. 171–182

[145] Wang, K. ; Fung, B. C. M. ; Yu, P. S.: Template-Based Privacy Preservation in Classification Problems. In: *ICDM '05: Proceedings of the 5th IEEE International Conference on Data Mining*. Washington, DC, USA : IEEE Computer Society, Nov. 2005. – ISBN 0–7695–2278–5, S. 466–473

[146] Wang, K. ; Fung, B. C. M. ; Yu, P. S.: Handicapping Attacker's Confidence: An Alternative to k-anonymization. In: *Knowledge Information and Systems* 11 (2007), Apr., Nr. 3, S. 345–368

[147] Wang, K. ; Xu, Y. ; Fu, A. W.-C. ; Wong, R. C.-W. : FF-Anonymity: When Quasi-identifiers Are Missing. In: Ioannidis, Y. E. (Hrsg.) ; Lee, D. L. (Hrsg.) ; Ng, R. T. (Hrsg.): *ICDE '09: Proceedings of the 25th International Conference on Data Engineering*. Washington, DC, USA : IEEE Computer Society, März 2009. – ISBN 978–0–7695–3545–6, S. 1136–1139

[148] Weber, H. H.: Khachiyan's Algorithmus. In: *Zeitschrift für Operations Research* 26 (1982), Nr. 1, S. B229–B240. – ISSN 0340–9422

[149] Wei, Q. ; Lu, Y. ; Lou, Q. : (τ, λ)-Uniqueness: Anonymity Management for Data Publication. In: Lee, R. Y. (Hrsg.): *ICIS '08: Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science*. Washington, DC, USA : IEEE Computer Society, Mai 2008. – ISBN 978–0–7695–3131–1, S. 107–112

[150] Westin, A. F.: *Privacy and Freedom*. New York, NY, USA : Atheneum, 1967

[151] Wikipedia: *Datenschutz*. Version: 2015. `http://de.wikipedia.org/wiki/Datenschutz`, Abruf: 24.02.2015

[152] Wikipedia: *Landau-Symbole*. Version: 2015. `http://de.wikipedia.org/wiki/Landau-Symbole`, Abruf: 24.02.2015

[153] Wikipedia: *Lineare Optimierung*. Version: 2015. `http://de.wikipedia.org/wiki/Lineare_Optimierung`, Abruf: 24.02.2015

[154] Wikipedia: *Personenbezogene Daten*. Version: 2015. `http://de.wikipedia.org/wiki/Personenbezogene_Daten`, Abruf: 24.02.2015

[155] Wikipedia: *Privatsphäre*. Version: 2015. `http://de.wikipedia.org/wiki/Privatsphäre`, Abruf: 24.02.2015

[156] Wikipedia: *Relationale Datenbank*. Version: 2015. `http://de.wikipedia.org/wiki/Relationale_Datenbank`, Abruf: 24.02.2015

[157] Wong, R. C.-W. ; Fu, A. W.-C. : *Privacy-Preserving Data Publishing: An Overview*. Morgan & Claypool Publishers, 2010 (Synthesis Lectures on Data Management). – ISBN 978–1–60845–217–0

[158] Wong, R. C.-W. ; Fu, A. W.-C. ; Wang, K. ; Pei, J. : Minimality attack in privacy preserving data publishing. In: Koch, C. (Hrsg.) ; Gehrke, J. (Hrsg.) ; Garofalakis, M. N. (Hrsg.) ; Srivastava, D. (Hrsg.) ; Aberer, K. (Hrsg.) ; Deshpande, A. (Hrsg.) ; Florescu, D. (Hrsg.) ; Chan, C. Y. (Hrsg.) ; Ganti,

V. (Hrsg.) ; Kanne, C.-C. (Hrsg.) ; Klas, W. (Hrsg.) ; Neuhold, E. J. (Hrsg.): *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, ACM, Sept. 2007. – ISBN 978–1–59593–649–3, S. 543–554

[159] Wong, R. C.-W. ; Li, J. ; Fu, A. W.-C. ; Wang, K. : (α, k)-Anonymity: An Enhanced k-Anoymity Model for Privacy-Preserving Data Publishing. In: Eliassi-Rad, T. (Hrsg.) ; Ungar, L. H. (Hrsg.) ; Craven, M. (Hrsg.) ; Gunopulos, D. (Hrsg.): *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA : ACM, Aug. 2006. – ISBN 1–59593–339–5, S. 754–759

[160] Xiao, X. ; Tao, Y. : Anatomy: simple and effective privacy preservation. In: Dayal, U. (Hrsg.) ; Whang, K.-Y. (Hrsg.) ; Lomet, D. B. (Hrsg.) ; Alonso, G. (Hrsg.) ; Lohman, G. M. (Hrsg.) ; Kersten, M. L. (Hrsg.) ; Cha, S. K. (Hrsg.) ; Kim, Y.-K. (Hrsg.): *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, ACM, Sept. 2006. – ISBN 1–59593–385–9, S. 139–150

[161] Xiao, X. ; Tao, Y. : Personalized Privacy Preservation. In: Chaudhuri, S. (Hrsg.) ; Hristidis, V. (Hrsg.) ; Polyzotis, N. (Hrsg.): *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2006. – ISBN 1–59593–256–9, S. 229–240

[162] Xiao, X. ; Tao, Y. : m-Invariance: Towards Privacy Preserving Re-publication of Dynamic Datasets. In: Chan, C. Y. (Hrsg.) ; Ooi, B. C. (Hrsg.) ; Zhou, A. (Hrsg.): *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2007. – ISBN 978–1–59593–686–8, S. 689–700

[163] Xiao, X. ; Tao, Y. : Dynamic Anonymization: Accurate Statistical Analysis with Privacy Preservation. In: Wang, J. T.-L. (Hrsg.): *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA : ACM, Jun. 2008. – ISBN 978–1–60558–102–6, S. 107–120

[164] Xiao, X. ; Tao, Y. ; Koudas, N. : Transparent Anonymization: Thwarting Adversaries Who Know the Algorithm. In: *ACM Transactions on Database Systems (TODS)* 35 (2010), Mai, Nr. 2, S. 8:1–8:48. – ISSN 0362–5915

[165] Zhang, L. ; Jajodia, S. ; Brodsky, A. : Information Disclosure under Realistic Assumptions: Privacy versus Optimality. In: Ning, P. (Hrsg.) ; Vimercati, S. D. C. (Hrsg.) ; Syverson, P. F. (Hrsg.): *CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security*. New York, NY, USA : ACM, Okt. 2007. – ISBN 978–1–59593–703–2, S. 573–583

[166] Zhang, Q. ; Koudas, N. ; Srivastava, D. ; Yu, T. : Aggregate Query Answering on Anonymized Tables. In: Chirkova, R. (Hrsg.) ; Dogac, A. (Hrsg.) ; Özsu, M. T. (Hrsg.) ; Sellis, T. K. (Hrsg.): *ICDE '07: Proceedings of the 23rd International Conference on Data Engineering*. Los Alamitos, CA, USA : IEEE Computer Society, Apr. 2007. – ISBN 1–4244–0802–4, S. 116–125

[167] Zhong, S. ; Yang, Z. ; Wright, R. N.: Privacy-Enhancing k-Anonymization of Customer Data. In: Li, C. (Hrsg.): *PODS '05: Proceedings of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. New York, NY, USA : ACM, Jun. 2005. – ISBN 1–59593–062–0, S. 139–147

Abkürzungsverzeichnis

$\emptyset$	leere Menge
$a_{\mathcal{R}^{(n)}}(t)$	Anonymitätsgrad des Tupels t bezüglich der Folge von Ergebnissen $\mathcal{R}^{(n)}$
$a(t)$	Anonymitätsgrad des Tupels t bezüglich einer Δ-top Matchingtabelle
$a^*(t)$	aktueller Anonymitätsgrad des Tupels t bezüglich einer Δ-top Matchingtabelle
BDSG	Bundesdatenschutzgesetz
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
$d_{E_\Delta}(t)$	Erweiterungsgrad des Tupels t bezüglich der Menge von Δ-top Matchingkanten E_Δ
$diam(G)$	Durchmesser des Graphen G
E	Menge von Kanten
E_Δ	Menge von Δ-top Matchingkanten
$E_\Delta(s)$	Menge von Δ-top Matchingkanten mit SA-Wert s
$E_\Delta(t)$	Menge von Δ-top Matchingkanten des Tupels t
$E_\Delta(t, s)$	Menge von Δ-top Matchingkanten des Tupels t mit SA-Wert s
$E(G)$	Kantenmenge des Graphen G
$E(P)$	Kantenmenge des Pfades P
G	einfacher Graph
G_{Erw}	Erweiterungsgraph, Klongraph
GG	Grundgesetz
$\mathcal{G}^{(n)}$	Folge von Anfragegraphen $(G_1, \ldots, G_n)$
ID	Identifikator, identifizierendes Attribut
K_n	vollständiger Graph mit n Knoten
$K_{r,s}$	vollständiger bipartiter Graph mit Partitionen der Größe $\|A\| = r$ und $\|B\| = s$
$l(M)$	Länge eines kürzesten M-augmentierenden Pfades
$l_e(M)$	Länge eines kürzesten eindeutig M-augmentierenden Pfades
$l(P)$	Länge des Pfades P (= Anzahl der Kanten von P)
m	Kantenzahl eines Graphen
M	Matching
M_{orig}	Originalmatching
$\mathcal{M}^{(n)}$	Matching in einer Folge von Anfragegraphen $\mathcal{G}^{(n)}$
n	Knotenzahl eines Graphen
$\mathbb{N}$	Menge der natürlichen Zahlen: $\mathbb{N} \coloneqq \{0, 1, 2, \ldots\}$
$\mathbb{N}^+$	Menge der positiven ganzen Zahlen: $\mathbb{N}^+ \coloneqq \{1, 2, \ldots\}$
$\nu(G)$	Kardinalität eines größten Matchings im Graphen G
P	Pfad in einem Graphen
Q	Anfrage (engl. *query*)
$\mathbb{Q}$	Menge der rationalen Zahlen

QI	Quasi-Identifikator
PPDP	Privacy-Preserving Data Publishing
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}^+$	Menge der positiven reellen Zahlen: $\mathbb{R}^+ \coloneqq \{x \in \mathbb{R} \mid x > 0\}$
$\mathbb{R}_0^+$	Menge der nicht negativen reellen Zahlen: $\mathbb{R}_0^+ \coloneqq \{x \in \mathbb{R} \mid x \geq 0\}$
$rad(G)$	Radius des Graphen G
SA	sensibles Attribut
$S_{\text{alt}}(G_{n+1})$	Menge der alten Tupel im Graphen G_{n+1}
$Sig_{\mathcal{R}^{(n)}}(t)$	Signatur des Tupels t bezüglich der Folge von Ergebnissen $\mathcal{R}^{(n)}$
$Sig(t)$	Signatur des Tupels t bezüglich einer Δ-top Matchingtabelle
$S_{\text{neu}}(G_{n+1})$	Menge der neuen Tupel im Graphen G_{n+1}
S_P	Menge von Pfaden
SQL	Structured Query Language
S_S	Menge von sensiblen Werten
S_T	Menge von Tupeln
$S_T(\mathcal{G}^{(n)})$	Menge aller Tupel in der Folge von Anfragegraphen $\mathcal{G}^{(n)}$
$S_T(G_n)$	Menge aller Tupel im Graphen G_n
T	Tabelle
$\mathcal{T}_\Delta^{(n)}$	Δ-top Matchingtabelle für Folge von Anfragegraphen $\mathcal{G}^{(n)}$
V	Menge von Knoten
$V(G)$	Knotenmenge des Graphen G
$\mathbb{Z}$	Menge der ganzen Zahlen: $\mathbb{Z} \coloneqq \{\dots, -2, -1, 0, 1, 2, \dots\}$

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Index

Lebenslauf

Der Lebenslauf ist aus Datenschutzgründen in der elektronischen Version dieser Arbeit nicht enthalten.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass

- ich die vorliegende Dissertationsschrift mit dem Titel „Der Schutz der Privatsphäre bei der Anfragebearbeitung in Datenbanksystemen“ selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe;
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin vom 17. Januar 2005, zuletzt geändert am 13. Februar 2006, veröffentlicht im Amtlichen Mitteilungsblatt Nr. 34/2006, bekannt ist.

Berlin, 24.03.2015 Lukas Dölle