Vektoren und Vektorprädikate und ihre Verwendung bei der Entwicklung relationaler Algorithmen

Dissertation

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Dr. rer. nat.)
der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Britta Kehden

Kiel,
2008

1. Gutachter:	Prof. Dr. Rudolf Berghammer
2. Gutachter:	Prof. Dr. Gunther Schmidt
Datum der mündlichen Prüfung:	15.07.2008

Inhaltsverzeichnis

1 Einleitung

Relationen und ihre zahlreichen Anwendungen werden bereits seit dem 19. Jahrhundert untersucht, wobei insbesondere die Arbeiten von De Morgan [18], Peirce [35] und Schröder [43] zu nennen sind. Tarski [45] legte 1941 durch seine Axiomatisierung des Relationenkalküls schließlich die Grundlage für den Einsatz formaler Methoden bei der Spezifikation und Analyse von Problemen, die sich mittels Relationen modellieren lassen. Beispielsweise können Graphen, Hypergraphen, Petrinetze, Gruppen und Verbände durch Relationen repräsentiert werden. Für die relationale Spezifikation werden zunächst meist prädikatenlogische Ausdrücke oder Formeln, die ein Problem beschreiben, in relationale Ausdrücke oder Formeln umgeformt. Dadurch ist es möglich, komplexe Sachverhalte relativ knapp auszudrücken. Unter Verwendung des Relationenkalküls können dann Eigenschaften „komponentenfrei“ bewiesen werden, also ohne dass man auf Elemente der Mengen, zwischen denen die Relation definiert ist, zurückgreift. Diese Art des algebraischen Beweisens hat eine Vielzahl von Vorteilen, zu denen die bessere Verständlichkeit und Lesbarkeit sowie die geringere Fehleranfälligkeit gehören. Zudem sind relationenalgebraische Beweise meist kürzer und besser nachvollziehbar als komponentenbehaftete.

Liegt eine relationale Spezifikation eines Problems vor, so lassen sich daraus oftmals direkt relationale Programme ableiten, die beispielsweise mit dem Computersystem RelView [13] ausgeführt werden können. Dieses Vorgehen hat Programme zur Folge, die „korrekt nach Konstruktion“ sind, deren Herleitung also formalen Ansprüchen genügt, so dass ein späterer Nachweis ihrer Korrektheit nicht notwendig ist. Es gibt zahlreiche Arbeiten, die diese Art der formalen Programmentwicklung mittels relationaler Methoden an Problemen unterschiedlichster Bereiche illustrieren. Neben verschiedenen graphentheoretischen Fragestellungen [6], [7] wurden dabei unter anderem Probleme aus der Ordnungs- und Verbandstheorie [5] sowie der Booleschen Logik [12] behandelt. Dabei wurden bisher meist exakte Verfahren entwickelt, bei denen im Falle der Lösbarkeit des Problems alle Lösungen berechnet wurden.

Ziele der Arbeit

In dieser Arbeit wird der Frage nachgegangen, inwieweit sich auch randomisierte Suchverfahren wie Evolutionäre Algorithmen mittels relationaler Methoden entwickeln, und durch relationale Programme implementieren lassen. Einen besonderen Schwerpunkt bildet dabei die Auswertung von Suchpunkten, der im Selektionsprozess Evolutionärer Algorithmen eine zentrale Bedeutung zukommt. Zur Bewertung der Suchpunkte wird das Konzept der *Vektorabbildungen* und *Vektorprädikate* entwickelt, welches nicht nur bei der relationalen Modellierung Evolutionärer Algorithmen eine wichtige Rolle spielt, sondern auch in anderen Bereichen einsetzbar ist. Wir geben eine Reihe von Beispielen an, bei denen der Einsatz von Vektorabbildungen und Vektorprädikaten zu einer schnellen und unkomplizierten Entwicklung von relationalen Algorithmen beiträgt. Den zweiten Schwerpunkt dieser Arbeit bildet die Repräsentation von Relationen durch Vektoren und die Transformation relationaler in vektorielle Eigenschaften. Dieses Vorgehen ermöglicht es, das Konzept der Vektorabbildungen und Vektorprädikate auf eine größere Klasse von Problemen anzuwenden. Auch für diesen Ansatz werden verschiedene Anwendungsbeispiele angegeben.

Gliederung

Die Arbeit gliedert sich wie folgt. Nachdem in Kapitel 1 die Grundlagen der Relationenalgebra erläutert und eine kurze Einführung in das RelView-System gegeben werden, beschäftigt sich Kapitel 2 mit der relationalen Modellierung Evolutionärer Algorithmen. Es wird zunächst ein Überblick über die wichtigsten Begriffe aus dem Gebiet der Evolutionären Algorithmen gegeben. Anschließend wird geklärt, wie sich Mengen von Suchpunkten (Populationen) durch Relationen modellieren und mittels relationaler Ausdrücke auswerten lassen. Hierbei wird insbesondere eine spezielle Menge relationaler Abbildungen, die sogenannten Vektorabbildungen, definiert, die in dieser Arbeit eine wichtige Rolle spielen. Eine spezielle Teilmenge der Vektorabbildungen stellen die Vektorprädikate dar, die im Selektionsprozess in relationalen Evolutionären Algorithmen einsetzbar sind. Nach der Präsentation einiger Beispiele für die Verwendung von Vektorabbildungen und Vektorprädikaten wird die relationale Modellierung der wichtigsten Module Evolutionärer Algorithmen behandelt. Abschließend wird der Einsatz relationaler Methoden im Zusammenhang mit multikriteriellen Optimierungsproblemen erörtert. Hier wird besonders auf die Reduzierung der Kriterien von Optimierungsproblemen eingegangen, wobei sich die relationale Modellierung als besonders vorteilhaft erweist. Kapitel 3 beschäftigt sich mit der Modellierung von Relationen durch Vektoren.

Die Darstellung einer Relation durch ihre sogenannte *Vektorrepräsentation* ermöglicht es, das Konzept der Vektorabbildungen auf eine größere Klasse von Problemen auszuweiten. Zu diesem Zweck wird in Abschnitt 4.1.10 eine Formel bewiesen, die bei der Transformation relationaler Eigenschaften in Eigenschaften des zugehörigen Vektors von grundlegender Bedeutung ist. Diese Formel wird in späteren Kapiteln an zahlreichen Stellen bei der Entwicklung von Algorithmen verwendet. Nach der Erörterung einiger Beispiele werden abschließend die Ergebnisse verschiedener Experimente präsentiert, um die Möglichkeiten und Grenzen der Programmierung mit Vektorrepräsentation und Vektorabbildungen darzustellen. In Kapitel 4 werden praktische Anwendungen vorgestellt, die sich auf die Verwendung der Vektorrepräsentation von Relationen und die Entwicklung von Vektorprädikaten stützen. Wir erörtern drei verschiedene Stundenplanprobleme und ihre relationale Modellierung und stellen Lösungsansätze vor, die sowohl exakte Verfahren als auch Evolutionäre Algorithmen umfassen. In Kapitel 5 wird eine weitere Anwendung von Vektorrepräsentation und Vektorabbildungen aus dem Bereich der Petrinetze behandelt. Hierbei werden Markierungen eines Petrinetzes durch Vektoren modelliert, so dass zur Analyse auf das Konzept der Vektorabbildungen zurückgegriffen werden kann.

Danksagung

Zunächst möchte ich Herrn Prof. Dr. Rudolf Berghammer für seine gute Betreuung und Unterstützung während meiner Promotionszeit, und Prof. Dr. Gunther Schmidt für die intensive Beschäftigung mit meiner Arbeit danken. Außerdem danke ich Frank Neumann, Alexander Fronk und Florian Diedrich für die angenehme und produktive Zusammenarbeit bei verschiedenen Veröffentlichungen der letzten Jahre. Bei Felix Meyer möchte ich mich für die Kooperation während der Bearbeitung der universitären Stundenplanprobleme bedanken. Meiner Familie und meinen Freunden danke ich für das Korrekturlesen dieser Arbeit und die allgemeine Unterstützung während meines Studiums und meiner Promotion. Christian Buck danke ich für sein Vertrauen in mich, und für seine Geduld.

2 Grundlagen

2.1 Abstrakte und konkrete Relationen

In diesem Kapitel werden die konkrete und die abstrakte Relationenalgebra definiert und wichtige Rechenregeln angegeben. Eine (konkrete) Relation R ist eine Teilmenge des Kartesischen Produktes zweier Mengen M und N, also $R \subseteq M \times N$, wobei wir voraussetzen, dass beide Mengen nicht leer sind. Wir verwenden die Notation $[M \leftrightarrow N]$ zur Beschreibung der Menge $M \times N$ und schreiben $R : M \leftrightarrow N$ statt $R \in [M \leftrightarrow N]$. Wir nennen $[M \leftrightarrow N]$ den *Typ*, M den Vorbereich und N den Nachbereich der Relation R. Des Weiteren wird in dieser Arbeit meist die an Matrizen orientierte Schreibweise R_{xy} verwendet, um auszudrücken, dass $(x, y) \in R$ gilt. In einigen Fällen, beispielsweise wenn es sich bei der Relation um eine Ordnung $\preceq$ handelt, findet sich aus Gründen der Lesbarkeit auch die Notation $x \preceq y$. Da es sich bei Relationen im Wesentlichen um Mengen handelt, stehen die binären mengentheoretischen Operationen $\cap$ und $\cup$ auf Relationen des gleichen Typs zur Verfügung, sowie das unäre Komplement, welches definert ist durch $\overline{R} = (M \times N) \setminus R$. Wir verwenden die Symbole O und L für die *leere Relation* und die *Allrelation* verschiedenen Typs, wobei die leere Relation der leeren Menge entspricht und die Allrelation aus allen Paaren des entsprechenden Kartesischen Produkts besteht. Die Menge $[M \leftrightarrow N]$ bildet mit den Operationen $\cap, \cup$ und $\overline{\ }$ einen vollständigen, Booleschen, atomaren Verband mit kleinstem Element O und größtem Element L. Der *nicht-verbandstheoretische Anteil* der Relationenalgebra ist für drei Mengen M, N und P definiert durch die *identische Relation* I des Typs $[M \leftrightarrow M]$, die für alle $x, y \in M$ definiert ist durch

$$\mathsf{I}_{xy} \Longleftrightarrow x = y$$

und die Operationen *Komposition* und *Transposition*, die im Folgenden eingeführt werden. Die binäre Operation der Komposition $\circ : [M \leftrightarrow N] \times [N \leftrightarrow P] \to [M \leftrightarrow P]$ ist festgelegt durch

$$(R \circ S)_{xy} \Longleftrightarrow \exists z \in \mathbb{N} : R_{xz} \ \wedge \ S_{zy}$$

für alle $x \in M, y \in P$. Die Komposition ist assoziativ. Des Weiteren ist die Transposition $^\top : [M \leftrightarrow N] \to [N \leftrightarrow M]$ definiert durch

$$(R^\top)_{xy} \Longleftrightarrow R_{yx}$$

für alle $x \in M, y \in N$. Bei der Komposition schreiben wir auch RS statt $R \circ S$. Die identische Relation ist das neutrale Element bezüglich der Komposition, d.h es gelten für alle $R : M \leftrightarrow N$ die Gleichungen

$$R\mathsf{I} = R \qquad \text{und} \qquad \mathsf{I}R = R,$$

wobei in der ersten Gleichung die identische Relation vom Typ $[N \leftrightarrow N]$, und in der zweiten die vom Typ $[M \leftrightarrow M]$ gemeint ist. Oftmals werden die Typen von Relationen nicht explizit genannt und bei relationalen Termen eine passende Typisierung vorausgesetzt. Neben den verbandstheoretischen Regeln, die für vollständige Boolesche Verbände gelten (siehe z.B. [17]) sind die folgenden zwei Gesetze grundlegend für das Rechnen mit Relationen.

2.1.1 Satz (Schröder-Äquivalenzen) *Für alle Relationen Q, R und S gelten die Äquivalenzen*

$$QR \subseteq S \iff \overline{S}R^\top \subseteq \overline{Q} \iff Q^\top \overline{S} \subseteq \overline{R}.$$

2.1.2 Satz (Tarski-Regel) *Für alle Relationen R gilt*

$$\mathsf{L}R\mathsf{L} = \mathsf{L} \iff R \neq \mathsf{O}.$$

Mithilfe der bisher erläuterten Eigenschaften lassen sich viele weitere Eigenschaften von Relationen durch ein algebraisches Vorgehen beweisen, ohne dass man dabei auf Komponenten (Elemente einer Relation) zurückgreifen muss. Man kann also von den konkreten Relationen abstrahieren und eine abstrakte Relationenalgebra definieren.

2.1.3 Definition (Abstrakte Relationenalgebra) *Sei T eine Menge, deren Elemente Sorten genannt werden. Eine abstrakte Relationenalgebra ist eine $T \times T$ sortierte Familie [1] $\mathfrak{R} = \{\mathfrak{R}_{mn} \mid m, n \in T\}$ mit den folgenden Eigenschaften:*

1. *Für alle $m, n \in T$ ist $\mathfrak{R}_{mn}$ ein vollständiger Boolescher und atomarer Verband $(\mathfrak{R}_{mn}, \cup, \cap, \overline{\ }, \mathsf{O}, \mathsf{L})$.*

[1] Das bedeutet, aus $\langle m, n\rangle \neq \langle m', n'\rangle$ folgt $\mathfrak{R}_{mn} \cap \mathfrak{R}_{m'n'} = \emptyset$

2. *Für alle $m, n, p \in T$ existiert eine assoziative Abbildung*

$$\circ : \mathfrak{R}_{mn} \times \mathfrak{R}_{np} \to \mathfrak{R}_{mp}$$

mit genau einem linksneutralen Element aus $\mathfrak{R}_{mm}$ und genau einem rechtsneutralen Element aus $\mathfrak{R}_{nn}$, die beide mit I bezeichnet werden.

3. *Für alle $m, n \in T$ gibt es eine Abbildung*

$$^\top : \mathfrak{R}_{mn} \to \mathfrak{R}_{nm}.$$

4. *Es gelten die Tarski-Regel und die Schröder-Äquivalenzen.*

Die konkreten Relationen bilden ein Modell für die abstrakte Relationenalgebra. Daher gelten Gesetze, die durch algebraisches Vorgehen auf Grundlage der Axiome der abstrakten Relationenalgebra bewiesen werden, auch für konkrete Relationen. Solche algebraischen Beweise haben den Vorteil, dass sie meist kürzer, besser nachzuvollziehen und weniger fehleranfällig sind. Wir werden im Folgenden einige wichtige Regeln für das Rechnen mit Relationen angeben.

2.1.4 Satz (Dedekind-Regel) *Für alle Relationen Q, R, S gilt die Inklusion*

$$QR \cap S \subseteq (Q \cap SR^\top)(R \cap Q^\top S).$$

Die Dedekind-Regel ist bei der Axiomatisierung der Relationenalgebra gleichwertig zu den Schröder-Äquivalenzen.

2.1.5 Satz (Rechenregeln)
Es gelten die folgenden Regeln für die Komposition:

$$R\mathsf{O} = \mathsf{O}R = \mathsf{O}$$

$$R \subseteq S \Longrightarrow QR \subseteq QS \;\; \textit{und} \;\; RQ \subseteq SQ$$

$$Q(R \cap S) \subseteq QR \cap QS \;\; \textit{und} \;\; (R \cap S)Q \subseteq RQ \cap SQ$$

$$Q(R \cup S) = QR \cup QS \;\; \textit{und} \;\; (R \cup S)Q = RQ \cup SQ$$

Zusätzlich gelten folgende Regeln für die Transposition:

$$R^{\top\top} = R$$

$$(RS)^\top = S^\top R^\top$$

$\overline{R^\top} = \overline{R}^\top$

$(R \cap S)^\top = R^\top \cap S^\top$ *und* $(R \cup S)^\top = R^\top \cup S^\top$

$R \subseteq S \Longrightarrow R^\top \subseteq S^\top$

Wir nennen eine Relation $R \in \mathfrak{R}_{mm}$ *homogen*, und beliebige Relationen im Unterschied dazu *heterogen*. Jede identische Relation I ist homogen und es gilt

$$\mathsf{I}^\top = \mathsf{I}.$$

Falls O und L homogen sind, gilt ebenfalls

$$\mathsf{L}^\top = \mathsf{L} \qquad \text{und} \qquad \mathsf{O}^\top = \mathsf{O}.$$

Diese Gleichungen gelten auch für heterogene Relationen, wobei hier bei $\mathsf{L}^\top = \mathsf{L}$ auf der linken Seite $\mathsf{L} \in \mathfrak{R}_{mn}$ und auf der rechten Seite $\mathsf{L} \in \mathfrak{R}_{nm}$ gemeint ist (analog für $\mathsf{O}^\top = \mathsf{O}$). Es gilt außerdem die Gleichung $\mathsf{LL} = \mathsf{L}$ bei entsprechender Typisierung.

Für endliche konkrete Relationen, also Relationen R mit endlichem Vor- und Nachbereich definieren wir den Betrag $|R|$ als die Anzahl der Elemente in R. Im folgenden Kapitel werden verschiedene homogene und heterogene Relationen sowie deren spezifische Eigenschaften behandelt.

2.2 Spezielle Relationen

Wir geben zunächst algebraische Definitionen für einige spezielle homogene Relationen an. Eine Relation R heißt

reflexiv $:\Longleftrightarrow \mathsf{I} \subseteq R$,

irreflexiv $:\Longleftrightarrow R \subseteq \overline{\mathsf{I}}$,

symmetrisch $:\Longleftrightarrow R = R^\top$,

antisymmetrisch $:\Longleftrightarrow R \cap R^\top \subseteq \mathsf{I}$,

transitiv $:\Longleftrightarrow RR \subseteq R$.

Wir nennen eine reflexive und transitive Relation R eine *Quasiordnung*. Ist R zudem antisymmetrisch, sprechen wir von einer *Ordnung*. Gilt für eine (Quasi-)ordnung R die Gleichung

$$R \cup R^\top = \mathsf{L}$$

so heißt R *lineare* oder auch *vollständige (Quasi-)ordnung*. Eine *Äquivalenzrelation* ist eine reflexive, transitive und symmetrische Relation.

Im Folgenden werden verschiedene spezielle heterogene Relationen definiert sowie wichtige Eigenschaften für diese angegeben. Eine Relation R heißt

eindeutig $:\Longleftrightarrow R^\top R \subseteq \mathsf{I}$,

total $:\Longleftrightarrow R\mathsf{L} = \mathsf{L}$,

injektiv $:\Longleftrightarrow R^\top$ eindeutig (d.h. $RR^\top \subseteq \mathsf{I}$),

surjektiv $:\Longleftrightarrow R^\top$ total (d.h. $\mathsf{L}R = \mathsf{L}$).

Durch Anwendung der Schröder-Äquivalenzen erhält man sofort auch die folgenden Charakterisierungen für Eindeutigkeit und Injektivität. R ist

eindeutig $\iff R\overline{\mathsf{I}} \subseteq \overline{R}$,

injektiv $\iff \overline{\mathsf{I}}R \subseteq \overline{R}$.

In [42] wird außerdem gezeigt, dass R

total $\iff \mathsf{I} \subseteq RR^\top$,

surjektiv $\iff \mathsf{I} \subseteq R^\top R$.

Eine eindeutige und totale Relation heißt *Funktion*. Ist eine Funktion zudem injektiv und surjektiv, sprechen wir von einer *Bijektion*. Es gelten die folgenden Rechenregeln für alle Relationen R

(1) R ist eindeutig $\iff$ für alle S gilt $R\overline{S} \subseteq \overline{RS}$

(2) R ist total $\iff$ für alle S gilt $\overline{RS} \subseteq R\overline{S}$

(3) R ist eindeutig $\iff$ für alle Q, S gilt $R(Q \cap S) = RQ \cap RS$

Man beachte, dass sich aus (1) und (2) sofort die folgenden Aussagen ergeben:

(4) R ist injektiv $\iff$ für alle S gilt $\overline{S}R \subseteq \overline{SR}$

(5) R surjektiv $\iff$ für alle S gilt $\overline{SR} \subseteq \overline{S}R$

(6) R ist eine Funktion $\iff$ für alle S gilt $R\overline{S} = \overline{RS}$

(7) R ist injektiv und surjektiv $\iff$ für alle S gilt $\overline{S}R = \overline{SR}$

Aus (3) folgt sofort die entsprechende Regel für injektive Relationen.

(8) R ist injektiv $\iff$ für alle Q, S gilt $(Q \cap S)R = QR \cap SR$

2.3 Modellierung von Mengen

Es stehen verschiedene Methoden zur Verfügung, um mittels Relationen Mengen darzustellen. Ein naheliegender Ansatz betrifft die Verwendung von *Vektoren*. Ein Vektor ist eine sogenannte *zeilenkonstante* Relation v, für die $v\mathsf{L} = v$ gilt. Wir verwenden meist Vektoren mit einelementigem Nachbereich, d.h. ein solcher Vektor ist dann eine Relation $v : X \leftrightarrow \mathbf{1}$, wobei die Menge $\mathbf{1}$ nur das Element $\perp$ enthält. Zur Vereinfachung schreiben wir v_x anstelle von $v_{x\perp}$. Mit Vektoren des Typs $[X \leftrightarrow \mathbf{1}]$ kann man in natürlicher Weise Teilmengen von X modellieren, indem jeder Vektor v die Menge $\{i \in X \mid v_i\}$ repräsentiert. Injektive und surjektive Vektoren werden *Punkte* genannt. Punkte $pX \leftrightarrow \mathbf{1}$ entsprechen damit Vektoren mit genau einem Eintrag und modellieren einelementige Teilmengen bzw. einzelne Elemente der Menge X. In einigen Fällen verwenden wir die folgende Notation für Punkte. Für ein Element $x \in X$ sei $\vec{x} : X \leftrightarrow \mathbf{1}$ der durch die folgende Äquivalenz definierte Punkt

$$\vec{x}_i \iff x = i,$$

der Vektor $\vec{x}$ repräsentiert also das Element x. Für jede Relation $R : X \leftrightarrow Y$ sowie alle $x \in X$ und $y \in Y$ gilt offensichtlich

$$R_{xy} \iff (R\vec{y})_x.$$

Bei Relationen mit Vorbereich $\mathbf{1}$ sprechen wir auch von *Zeilenvektoren*, obwohl es sich hierbei nicht um Vektoren im Sinne der obigen Definition handelt.

Mengen können ebenfalls mittels injektiver Einbettungen beschrieben werden. Sei dazu $v : X \leftrightarrow \mathbf{1}$ ein nichtleerer Vektor, der die Teilmenge Y von X repräsentiert. Dann sei $inj(v) : Y \leftrightarrow X$ definiert durch

$$inj(v)_{yx} \iff y = x.$$

Die Relation $inj(v)$ ist eine injektive Funktion, und es gilt

$$inj(v)^\top \mathsf{L} = v.$$

Injektive Einbettungen werden unter anderem zur Einschränkung des Vor- und Nachbereichs von Relationen verwendet. Ist eine Relation $R : X \leftrightarrow Y$ gegeben und repräsentieren die Vektoren v und w die Teilmengen $V \subseteq X$ bzw. $W \subseteq Y$, so ist die Relation

$inj(v)R\,inj(w)^\top$ vom Typ $[V \leftrightarrow W]$ und es gilt

$$(inj(v)R\,inj(w)^\top)_{xy} \Longleftrightarrow R_{xy}.$$

für alle $x \in V$ und $y \in W$.

Eine wichtige Rolle im Zusammenhang mit der relationenalgebraischen Repräsentation von Mengen spielen die *Potenzmengenrelationen* zur Darstellung von Potenzmengen nichtleerer Mengen. Für jede nichtleere Menge X bezeichne M die durch

$$\mathsf{M}_{xY} \Longleftrightarrow x \in Y$$

definierte Relation des Typs $[X \leftrightarrow 2^X]$. Die Relation M wird häufig für exakte relationale Lösungsverfahren verschiedener Probleme verwendet, wie beispielsweise in [12] und [13] erörtert wird.

2.4 Direkte Produkte

Wir nennen ein Paar (π, ρ) von Relationen ein 2-stelliges *direktes Produkt*, wenn es die folgenden 4 Gesetze erfüllt.

$$\pi^\top \pi = \mathsf{I}$$

$$\rho^\top \rho = \mathsf{I}$$

$$\pi\pi^\top \cap \rho\rho^\top = \mathsf{I}$$

$$\pi^\top \rho = \mathsf{L}$$

Es gilt insbesondere, dass π und ρ surjektive Funktionen sind. In der konkreten Relationenalgebra verwendet man häufig sogenannte *natürliche Projektionen* oder Projektionsabbildungen $\pi : X \times Y \leftrightarrow X$ und $\rho : X \times Y \leftrightarrow Y$, welche definiert sind durch

$$\pi_{\langle x,y\rangle x'} \Longleftrightarrow x = x' \qquad \rho_{\langle x,y\rangle y'} \Longleftrightarrow y = y'.$$

Jedes Paar (π, ρ) solcher Projektionen ist ein direktes Produkt, erfüllt also die oben angegebenen Gleichungen.

Direkte Produkte werden unter anderem bei der Transformation zwischen Relationen und Vektoren verwendet. Jede Relation eines beliebigen Typs $[X \leftrightarrow Y]$ kann als Vektor

des Typs $[X \times Y \leftrightarrow \mathbf{1}]$ dargestellt werden. Dazu definieren wir eine Abbildung $vec : [X \leftrightarrow Y] \to [X \times Y \leftrightarrow \mathbf{1}]$ durch

$$vec(R) = (\pi R \cap \rho)\mathsf{L},$$

wobei π und ρ die natürlichen Projektionen von $X \times Y$ sind und L der Allvektor vom Typ $[Y \leftrightarrow \mathbf{1}]$. Wir nennen $vec(R)$ die Vektorrepräsentation der Relation R, denn es gilt

$$R_{xy} \iff vec(R)_{\langle x,y \rangle}.$$

Die Abbildung vec ist ein Boolescher Verbandsisomorphismus und erfüllt damit insbesondere die folgenden Gesetze

$$A \subseteq B \iff vec(A) \subseteq vec(B)$$

$$vec(\overline{A}) = \overline{vec(A)}$$

$$vec(A \cap B) = vec(A) \cap vec(B)$$

$$vec(A \cup B) = vec(A) \cup vec(B)$$

$$vec(\mathsf{O}) = \mathsf{O}$$

$$vec(\mathsf{L}) = \mathsf{L}$$

wie in [42] gezeigt wird. Setzt man die Existenz der entsprechenden direkten Produkte

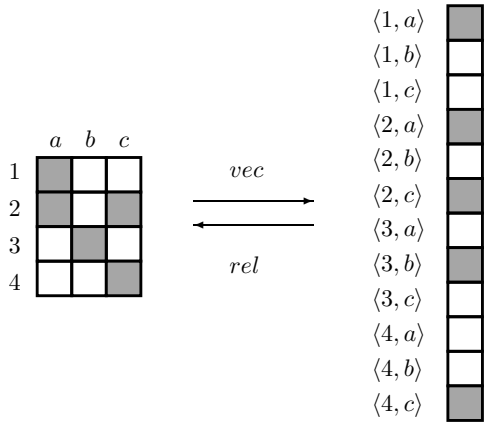


Abbildung 2.1: Transformation zwischen Relationen und Vektoren

voraus, gelten diese Regeln ebenfalls in der abstrakten Relationenalgebra. Die Abbildung $rel : [X \times Y \leftrightarrow \mathbf{1}] \to [X \leftrightarrow Y]$, definiert durch

$$rel(a) = \pi^{\top}(\rho \cap a\mathsf{L}),$$

ist die zu *vec* inverse Abbildung, wie ebenfalls in [42] gezeigt wird.

2.5 Residuen und symmetrische Quotienten

Für Relationen A und B nennt man die Konstruktion

$$A/B := \overline{\overline{A}B^\top}$$

das *Linksresiduum* von A über B, sowie

$$A\backslash B := \overline{A^\top\overline{B}}$$

das *Rechtsresiduum* von B über A. Der *symmetrische Quotient* von A und B ist definiert durch

$$syq(A,B) := (A\backslash B) \cap (A^\top/B^\top)$$

Für alle Relationen A ist $syq(A,A)$ eine Äquivalenzrelation. Im Falle konkreter Relationen $A : X \leftrightarrow Z$ und $B : Y \leftrightarrow Z$ ist (A/B) vom Typ $[X \leftrightarrow Y]$, und es gilt

$$(A/B)_{xy} \Longleftrightarrow \forall z : B_{yz} \rightarrow A_{xz}.$$

Für $A : X \leftrightarrow Y$ und $B : X \leftrightarrow Z$ sind die Relationen $A\backslash B$ und $syq(A,B)$ vom Typ $[Y \leftrightarrow Z]$, und es gelten die Äquivalenzen

$$(A\backslash B)_{yz} \Longleftrightarrow \forall x : A_{xy} \rightarrow B_{xz} \qquad \text{und} \qquad syq(A,B)_{yz} \Longleftrightarrow \forall x : A_{xy} \leftrightarrow B_{xz}.$$

2.6 Das Computersystem RelView

Das RelView-System dient dem prototypischen Manipulieren von diskreten Strukturen, die auf Relationen basieren. Es ist ein interaktives und bildschirmorientiertes Computersystem, welches als Prototyp in den 80er Jahren zunächst an der Technischen Universität München und der Universität der Bundeswehr München entwickelt und später an der Christian-Albrechts-Universität zu Kiel weiterentwickelt und erweitert wurde. Dabei wurde insbesondere die Darstellung von Relationen durch ROBDDs (reduced ordered binary decision diagrams) realisiert, was eine erhebliche Steigerung der Effizienz zur Folge hatte. Für eine detaillierte Beschreibung dieser Repräsentation von Relationen verweisen wir auf [32], [30] und [13].

Die Hauptanwendung des RelView-Systems ist die Auswertung relationenalgebraischer Ausdrücke. Dafür stehen neben den vordefinierten Operationen `-`, `^`, `&`, `|` und `*` für Komplement, Transposition, Schnitt, Vereinigung und Komposition eine Vielzahl an Tests und Basisfunktionen zur Verfügung. Wir werden in dieser Arbeit die binären Tests `eq` und `incl` zum Testen auf Gleichheit bzw. Inklusion sowie den unären Leerheitstest `empty` verwenden. Des Weiteren benötigt man oft verschiedene Funktionen zur Erzeugung wichtiger Konstanten, von denen wir die wichtigsten kurz vorstellen wollen. Mit `L`, `O` und `I` werden Allrelationen, Leere Relationen und Identische Relationen vom Typ der Eingaberelation generiert. Für eine Relation `R` des Typs $[X \leftrightarrow Y]$ erzeugt außerdem `Ln1(R)` den Allvektor vom Typ $[X \leftrightarrow \mathbf{1}]$ und `L1n(R)` die Allrelation vom Typ $[\mathbf{1} \leftrightarrow Y]$, also einen Zeilenvektor. Mithilfe der Funktionen `On1` und `O1n` werden entsprechende leere Relationen konstruiert. Für den Umgang mit Mengen und ihren verschiedenen Darstellungsmöglichkeiten stehen die Funktionen `inj` und `epsi` zur Verfügung, die, angewandt auf einen Vektor `v` mit Vorbereich X die injektive Einbettung $inj(v)$, bzw. die Potenzmengenrelation $\mathsf{M} : X \leftrightarrow 2^X$ erzeugen, wobei im ersten Fall der Vektor nicht leer sein darf. Die Potenzmengenrelationen spielen eine wichtige Rolle bei exakten Verfahren. Da sie sich durch die ROBDD-Implementierung des RelView-Systems sehr effizient darstellen lassen, eignet sich das System besonders gut für Anwendungen, bei denen alle Lösungen eines Problems aufgezählt werden. Ein gutes Beispiel dafür wird in Kapitel 3.4 angegeben. Zur deterministischen Auswahl eines Punktes aus einem nichtleeren Vektor dient die Operation `point`, während durch die Anwendung von `randompoint` auf einen Vektor $v \neq \mathsf{O}$ ein zufälliger, in v enthaltener Punkt zurückgeliefert wird. Für die deterministische Auswahl steht auf den Vor- und Nachbereichen der Relationen eine lineare Ordnung zur Verfügung. Die Operation `point` gibt den Punkt zurück, der dem kleinsten im Eingabevektor enthaltenen Element bezüglich dieser Ordnung entspricht. Repräsentiert ein Punkt `p` mit Vorbereich X ein Element $x \in X$, so wird durch `next(p)` der Punkt berechnet, der dem kleinsten Element in X entspricht, das größer als x ist. Exisiert ein solcher Punkt nicht, weil x das größte Element in X ist, wird der leere Vektor zurückgeliefert.

Da in dieser Arbeit unter anderem relationale Varianten Evolutionärer Algorithmen diskutiert werden, sind die verschiedenen Möglichkeiten des RelView-Systems, randomisiert Relationen zu erzeugen, von besonderem Interesse. Durch die Operationen `random01`, ..., `random99` werden zufällige Relationen generiert, bei denen jedes Paar mit der Wahrscheinlichkeit $\frac{1}{100}, \ldots \frac{99}{100}$ enthalten ist, wobei der Typ der Eingaberelation den Typ der Ausgaberelation bestimmt. Des Weiteren liefert ein Aufruf von `random(R,S)` für $S : X \leftrightarrow Y$ eine Relation vom gleichen Typ wie R, bei der jedes Paar mit Wahr-

scheinlichkeit $\frac{|S|}{|X|\cdot|Y|}$ auftritt. Mit `randomperm` wird eine zufällige Permutation vom Typ der (homogenen) Eingaberelation erzeugt.

Eine relationale Funktion hat die Form $F(X_1, \ldots, X_n) = t$, wobei F der Name der Funktion, die X_i formale Parameter (für Relationen) sind, und t ein relationaler Ausdruck ist, der außer den vordefinierten Basisoperationen und den formalen Parametern auch weitere benutzerdefinierte Funktionen und Programme beinhalten kann. Durch

```
graph(R) = (R | R^) & -I(R)
```

wird beispielsweise zu einer homogenen Relation R die irreflexive, symmetrische Relation $(R^\top \cup R) \cap \bar{\mathsf{I}}$ berechnet. Ein relationales Programm besteht aus einer Kopfzeile mit Programmname und Parameterliste, einem Deklarationsteil und einem Rumpf , der aus einer Anweisung aus der Sprache der While-Programme, und damit aus Zuweisungen, While-Schleifen und Alternativen, besteht. Das folgende Beispiel beschreibt ein einfaches Programm zur Berechnung der transitiven Hülle einer homogenen Relation R.

```
trans(R)
  DECL X, Y
  BEG X = R;
      Y = O(R);
      WHILE -incl(X,Y) DO
        Y = Y | X;
        X = X * R
      OD
      RETURN X
  END.
```

Da in dieser Arbeit den in Abschnitt 2.4 eingeführten Abbildungen *vec* und *rel* besondere Bedeutung zukommt, wollen wir abschließend noch den Umgang mit direkten Produkten und natürlichen Projektionen anhand eines Programms zur Berechnung der Vektordarstellung einer Relation $R : X \leftrightarrow Y$ erläutern.

```
vec(R)
  DECL prod = PROD(R*R^,R^*R);
       pi, rho
  BEG pi = p-1(prod);
      rho = p-2(prod);
      r = (pi*R & rho)*L1n(R)^
  END.
```

Hierbei wird durch `prod = PROD(R*R^,R^*R)` das Kartesische Produkt $X \times Y$ definiert, so dass mit `p-1(prod)` und `p-2(prod)` die natürlichen Projektionen π und ρ auf $X \times Y$ erzeugt werden können.

3 Evolutionäre Algorithmen mit relationalen Methoden

Bereits seit den 60er Jahren werden Algorithmen entwickelt und analysiert, die sich an evolutionären Prozessen in der Natur orientieren. Dazu gehören die von John Holland entwickelten Genetischen Algorithmen (siehe [26]), die von Ingo Rechenberg [37] und Hans-Paul Schwefel [44] entworfenen Evolutionsstrategien sowie das Evolutionäre Programmieren, das auf Larry Fogel [22] zurückgeht. Diese und weitere Ansätze werden heute unter dem Begriff *Evolutionäre Algorithmen* zusammengefasst. Erste Überlegungen für die Verbindung von Evolutionären Algorithmen und relationalen Methoden finden sich in [29] und [28].

In diesem Kapitel werden verschiedene Möglichkeiten für den Einsatz relationaler Methoden im Umfeld Evolutionärer Algorithmen dargestellt. Nach einer kurzen Einführung in das Gebiet der Evolutionären Algorithmen wird erläutert, wie Populationen durch Relationen modelliert und mithilfe relationaler Ausdrücke ausgewertet werden können. Dabei konzentrieren wir uns zunächst auf den Selektionsprozess Evolutionärer Algorithmen und insbesondere auf die Bewertung von Suchpunkten. Für diesen Zweck definieren wir in Abschnitt 3.2.2 eine Menge von sogenannten *Vektorprädikaten*. Dies sind spezielle Abbildungen, die bei der Auswahl von geeigneten Suchpunkten in relationalen Evolutionären Algorithmen eine entscheidende Rolle spielen. Es werden einige Beispiele aus verschiedenen Gebieten angegeben, in denen solche Abbildungen zur Anwendung kommen. Auch im weiteren Verlauf dieser Arbeit kommt den Vektorprädikaten eine elementare Bedeutung zu, beispielsweise bei der Entwicklung exakter Verfahren im Bereich der Stundenplanprobleme in Kapitel 5. In Abschnitt 3.3 wird auf weitere wichtige Bestandteile Evolutionärer Algorithmen und ihre Modellierung mittels relationaler Methoden eingegangen, was wiederum anhand eines Beispiels verdeutlicht wird. Im letzten Abschnitt werden Probleme behandelt, bei denen eine Optimierung bezüglich mehrerer Kriterien angestrebt wird, man spricht hierbei von *multikriterieller* Optimierung. Auch hierbei lassen sich relationale Methoden, beispielsweise zur Reduzierung der Anzahl von Kriterien, einsetzen.

3.1 Evolutionäre Algorithmen

Der Begriff *Evolutionäre Algorithmen* (kurz EA) beschreibt eine Klasse von randomisierten Suchverfahren, welche sich an evolutionären Prozessen in der Natur orientieren. Sie lassen sich für verschiedene Optimierungsprobleme einsetzen und sind besonders dann vorteilhaft, wenn wenig Vorwissen über das Problem vorhanden ist und kein spezieller, auf das Problem zugeschnittener Algorithmus existiert. Es soll an dieser Stelle lediglich eine kurze Einführung gegeben werden, um die später auftretenden Begriffe zu erläutern. Für eine ausführliche Darstellung von Evolutionären Algorithmen verweisen wir auf [20] und [34].

Im Fall eines Maximierungsproblems wird auf der Grundlage eines gegebenen endlichen Suchraumes $\mathcal{S}$ und einer *Fitnessfunktion* f auf $\mathcal{S}$ angestrebt, Suchpunkte, also Elemente von $\mathcal{S}$ zu finden, welche möglichst große f-Werte (*Fitness-Werte*) besitzen. Der Bildbereich der Fitnessfunktion muss also eine geordnete, oder zumindest quasigeordnete Menge sein. Im Fall der *einkriteriellen* Optimierung liegt meist eine Funktion $f : \mathcal{S} \to \mathbb{R}_+$ zugrunde, und es wird ein $x \in \mathcal{S}$ mit $f(x) = max\{f(y) \,|\, y \in \mathcal{S}\}$ gesucht. Bei der *multikriteriellen* Optimierung steht hingegen eine vektorwertige Fitnessfunktion $f : \mathcal{S} \to \mathbb{R}_+^n$ zur Verfügung, wobei durch

$$(x_1, \ldots, x_n) \leq (y_1, \ldots, y_n) \iff \forall i \in [1..n] : x_i \leq y_i$$

eine Quasiordnung auf $\mathbb{R}_+^n$ gegeben ist. Hierbei wird eine Menge von sogenannten *Pareto-optimalen* Elementen aus $\mathcal{S}$ gesucht, wie in Abschnitt 3.4 näher erläutert wird.

Ein EA arbeitet dabei im Allgemeinen mit einer *Population*, also einer Menge von Suchpunkten (auch *Individuen*), die im Lauf des Algorithmus immer wieder verändert und bewertet wird, immer mit der Zielsetzung, möglichst gute Individuen zu erhalten. Hierbei wird in jeder Runde des Algorithmus aus der gegebenen Population (der *Elternpopulation*) unter Verwendung verschiedener *Variationsoperatoren* eine neue Population (die *Offspringpopulation*) erzeugt. Einen einstelligen Variationsoperator stellt die *Mutation* dar. Hierbei wird aus jedem Individuum der Elternpopulation durch zufallsgesteuerte Abwandlung ein neues Individuum erzeugt. Im Folgenden legen wir den Suchraum $\mathcal{S} = \{0, 1\}^n$, die Menge aller Bitstrings der Länge n, zugrunde und bezeichnen das i-te Bit eines Suchpunktes x mit x_i. Durch Mutation mit Wahrscheinlichkeit p wird aus einem Elternindividuum x ein neues Individuum y mit der Eigenschaft

$$Pr(y_i \neq x_i) = p$$

für alle $i \in [1..n]$ erzeugt, d.h., jedes Bit von x wird mit Wahrscheinlichkeit p geflippt. Dieser Prozess kann mehrfach wiederholt werden, so dass man eine Offspringpopulation erhält, die aus mehr Individuen als die Elternpopulation besteht. Bei mehrstelligen Variationsoperatoren spricht man von *Rekombination* oder auch *Crossover*. Hierbei wird aus mehreren Eltern-Individuen ein neues Individuum generiert, wobei von jedem Elternteil gewisse Eigenschaften übernommen werden. Wir beschränken uns auf Rekombinationsoperatoren, bei denen aus zwei Eltern ein neues Individuum entsteht. Hier stehen verschiedene Möglichkeiten zur Verfügung, beispielweise *uniformes Crossover* und *1-Point-crossover*. In beiden Fällen werden aus den Individuen der Elternpopulation zufällig Paare (x, x') gebildet, von denen jedes randomisiert ein Individuum y generiert. Für das uniforme Crossover gilt dann

$$Pr(y_i = x_i) = Pr(y_i = x'_i) = \frac{1}{2},$$

jedes Bit wird also mit gleicher Wahrscheinlichkeit von x oder x' übernommen. Beim 1-Point-crossover wird zufällig eine sogenannte *Trennstelle* $k \in [1..n]$ gewählt und das neue Individuum y mit

$$y_i = x_i \text{ für } 1 \leq i \leq k \qquad \text{und} \qquad y_i = x'_i \text{ für } k < i \leq n$$

erzeugt. Die ersten k Bits werden also von x übernommen, die restlichen von x'.

Die durch Mutation und Rekombination erzeugte Offspringpopulation wird im *Selektionsprozess* bezüglich der Fitnessfunktion f bewertet, und eine Teilmenge der Individuen wird als Elternpopulation für die nächste Runde übernommen. Hierbei kommen verschiedene Selektionsmechanismen zum Einsatz, wir betrachten im Folgenden ausschließlich sogenannte *rangbasierte Selektionsverfahren*. Gehen wir von einer Elternpopulation von μ Individuen und einer Offspringpopulation von λ Individuen aus, so bezeichnet die (μ, λ)-*Selektion* das Vorgehen, dass in die neue Generation ausschließlich Individuen der Offspringpopulation übernommen werden. Selektionsmechanismen, bei denen Individuen sowohl aus der Eltern- als auch aus der Offspringpopulation in Betracht gezogen werden, nennt man $(\mu + \lambda)$-*Selektion*. Wir beschränken uns hier auf die zweite Art der Selektion. Einen Evolutionären Algorithmus mit $(\mu + \lambda)$-Selektion bezeichnet man als $(\mu + \lambda)$-*EA*. Wir konzentrieren uns auf 2 Spezialfälle. Beim $(1 + \lambda)$-EA wird in jeder Runde aus einem Elternindividuum x durch Mutation eine Offspringpopulation $P \subseteq \mathcal{S}$ der Größe λ erzeugt, und aus der Menge $P \cup \{x\}$ wiederum ein neues Elternindividuum x' ausgewählt. Liegt ein Maximierungsproblem vor, muss dabei $f(x') \geq f(x)$

gelten, um möglichst eine Verbesserung zu erreichen. Man kann also bei der Auswahl von x' das Individuum mit dem größten Fitness-Wert in $P \cup \{x\}$ bestimmen oder eine zufällige Wahl aus der Menge $\{y \in P \cup \{x\} \mid f(y) \geq f(x)\}$ treffen. Als zweite Variante des $(\mu + \lambda)$-EA betrachten wir den Fall $\mu = \lambda$ und bezeichnen einen solchen Algorithmus im Folgenden als $(\lambda + \lambda)$-EA. Hierbei wird ausschließlich mit Populationen der Größe λ gearbeitet. Im Selektionsprozess soll dann aus der Vereinigung von Eltern- und Offspringpopulation eine Menge von λ Individuen so gewählt werden, dass sich die Fitness der Population dabei insgesamt verbessert. Eine Möglichkeit besteht darin, die Individuen mit den besten Fitness-Werten zu wählen, eine andere in der sogenannten *Turnierselektion*. Hierbei werden zufällig Paare aus je einem Eltern- und einem Kindindividuum gebildet und von jedem Paar dasjenige Individuum mit dem größeren Fitness-Wert in die neue Generation übernommen.

Durch wiederholte Variation und Selektion wird die durchschnittliche Fitness der Population verbessert, bis ein Abbruchkriterium erfüllt ist. Je nach Problemstellung ist man an einer einzelnen möglichst guten Lösung oder einer Menge von Lösungen interessiert.

Zu den Vorteilen Evolutionärer Algorithmen gehört ihre vielfältige Verwendbarkeit, insbesondere für Probleme, bei denen wenig Einsicht in die Struktur vorhanden ist. Das einfache Grundschema eines Evolutionären Algorithmus lässt sich schnell an unterschiedlichste Probleme anpassen, ohne dass jedesmal ein neuer, problemspezifischer Algorithmus entwickelt werden muss. Von Nachteil ist hingegen, dass sich Aussagen über die Qualität der erzielten Ergebnisse zumeist wesentlich schwieriger treffen lassen als beispielweise bei approximativen Algorithmen. Ebenso fehlen oftmals Garantien bezüglich des Laufzeitverhaltens von Evolutionären Algorithmen.

3.2 Grundlagen des relationalen Modells

Im Selektionsprozess Evolutionärer Algorithmen kommt der Bewertung von Suchpunkten eine fundamentale Bedeutung zu. Dabei wird insbesondere getestet, ob ein Individuum gewisse Eigenschaften erfüllt, um zu entscheiden, ob es in die neue Population übernommen werden soll oder nicht. Wir erläutern die Situation kurz anhand eines Beispiels aus der Graphentheorie. Bei dem Minimalen Knotenüberdeckungs-Problem (siehe auch Abschnitt 3.2.3.3) sucht man eine möglichst kleine Teilmenge T von Knoten, so dass jede Kante des Graphen $g = (V, E)$ mit mindestens einem Knoten aus T inzident ist. Für die Wahl der Fitnessfunktion stehen jetzt verschiedene Möglichkeiten

zur Verfügung. Man könnte die Problemstellung beispielsweise als 2-kriteriell ansehen, also zum einen die Anzahl der Knoten einer Teilmenge bestimmen, und zum anderen die Anzahl der Kanten, die mit dieser Knotenmenge überdeckt werden. Der Evolutionäre Algorithmus müsste dann nach dem ersten Kriterium minimieren und nach dem zweiten maximieren. Die zweite Möglichkeit, auf welche wir uns im Folgenden konzentrieren werden, besteht darin, im gesamten Verlauf des Algorithmus ausschließlich Individuen zu übernehmen, die Knotenüberdeckungen des Graphen darstellen, und innerhalb dieser Menge die Suchpunkte wiederum bezüglich ihrer Größe zu bewerten. Dadurch ergibt sich also ein einkriterielles Optimierungsproblem, wobei die Fitnessfunktion beispielsweise durch

$$f(x) = \begin{cases} |x| & : \quad x \textit{ ist eine Knotenüberdeckung} \\ |V| & : \quad \textit{sonst} \end{cases}$$

gegeben ist. Ein wichtiger Bestandteil des Selektionsprozesses besteht also darin, festzustellen, bei welchen Individuen es sich um Knotenüberdeckungen handelt. Gerade bei graphentheoretischen Fragestellungen bietet sich eine relationale Modellierung an. Individuen, im Falle des Knotenüberdeckungs-Problems also Teilmengen von Knoten, können durch Vektoren, und Populationen durch Relationen dargestellt werden. Auch der Test, ob ein Vektor eine Knotenüberdeckung repräsentiert, läßt sich durch einen relationalen Ausdruck leicht formalisieren. Das folgende Kapitel behandelt eine Klasse von Problemen, bei denen Prädikate, also Abbildungen $\phi : \mathcal{S} \to \mathbb{B}$, im Selektionsprozess eine Rolle spielen, und erläutert insbesondere die Möglichkeiten, solche Prädikate mithilfe relationaler Methoden auszudrücken. Es wird dargestellt, wie Individuen und Populationen durch Relationen modelliert und mithilfe relationaler Terme bewertet werden können. Dabei werden Individuen durch Vektoren und Populationen durch Relationen dargestellt und eine Klasse von Testabbildungen (siehe auch [27]) definiert, mit deren Hilfe Populationen auf einfache Art und Weise auf spezielle Eigenschaften getestet werden können. Die hier eingeführten Testabbildungen spielen in dieser Arbeit eine zentrale Rolle. Sie sind in vielfältiger Weise einsetzbar und kommen nicht nur im Bereich der Evolutionären Algorithmen zur Anwendung, sondern beispielsweise auch bei exakten Verfahren, wie in [29] und [28] erörtert wird. Am Ende des Kapitels werden einige Beispiele für die Verwendung der Testabbildungen angegeben.

3.2.1 Modellierung von Populationen

Im Folgenden sei $n \in \mathbb{N}$ gegeben und wir setzen den Suchraum als $\mathcal{S} = \{0,1\}^n$, der Menge der Bitstrings der Länge n, voraus. Wir modellieren einzelne Suchpunkte aus

$\mathcal{S}$ durch Vektoren des Typs $[[1..n] \leftrightarrow \mathbf{1}]$ und setzen zur Vereinfachung im Folgenden $X := [1..n]$ voraus. Zur Modellierung von Populationen durch Relationen benötigen wir die folgende Notation zur Beschreibung einzelner Spalten einer Relation.

3.2.1.1 Definition *Für eine Relation $R : X \leftrightarrow Y$ und ein Element $y \in Y$ sei $R^{(y)}$ ein Vektor des Typs $[X \leftrightarrow \mathbf{1}]$, der durch die Äquivalenz*

$$R_x^{(y)} \Longleftrightarrow R_{xy}$$

für alle $x \in X$ definiert ist.

Ein Vektor $R^{(y)}$ beschreibt also eine Spalte von R. Eine Population, bestehend aus k Suchpunkten $v^1, \ldots, v^k : X \leftrightarrow \mathbf{1}$ wird durch eine Relation $P : X \leftrightarrow [1..k]$ modelliert, in dem Sinne, dass für alle $j \in [1..k]$ die Gleichung $v^j = P^{(j)}$ gilt. Um mithilfe eines

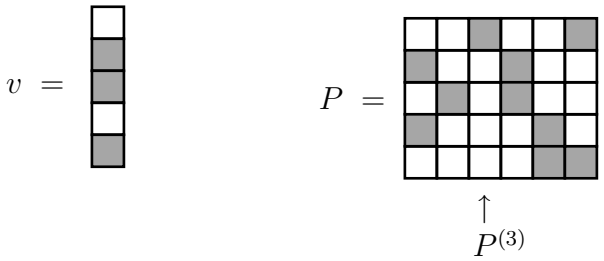


Abbildung 3.1: Individuum und Population

relationalen Ausdrucks ein einzelnes Individuum der Population P zu erhalten, benötigt man einen Punkt $p : [1..k] \leftrightarrow \mathbf{1}$. Der Punkt p repräsentiert dann ein Element $j \in [1..k]$, d.h. es gilt $p = \vec{j}$. Man erhält das Individuum $P^{(j)}$ durch Multiplikation von P mit dem Punkt p, also

$$P^{(j)} = P\vec{j}.$$

Im Folgenden geht es um die Auswertung solchermaßen repräsentierter Populationen, also um die relationale Formalisierung gewisser Prädikate auf dem gegebenen Suchraum. Hierbei sollen nicht einzelne Individuen aus der Population herausgelöst und bewertet werden, sondern die als Relation repräsentierte Population soll als Eingabe einer relationalen Abbildung dienen, die als Ergebnis die Menge der Individuen liefert, welche das zugrundeliegende Prädikat erfüllen.

3.2.2 Auswertung von Populationen

Zur relationalen Auswertung von Suchpunkten aus $\mathcal{S} = [X \leftrightarrow \mathbf{1}]$ wird in diesem Kapitel eine Menge $\mathcal{VP}$ von *Vektorprädikaten* definiert, die zunächst nur der Auswertung

einzelner Individuen dient. Bei den Vektorprädikaten handelt es sich um Abbildungen, die Vektoren eines bestimmten Typs in die Menge $[\mathbf{1} \leftrightarrow \mathbf{1}]$ abbilden. Da die Menge $[\mathbf{1} \leftrightarrow \mathbf{1}]$ ausschließlich aus den Relationen O und L besteht, können diese als boolesche Werte interpretiert werden. Mit einer Abbildung $\varphi : [X \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ kann also eine Eigenschaft von Vektoren des Typs $[X \leftrightarrow \mathbf{1}]$ modelliert werden, in dem Sinne, dass $\varphi(v) = \mathsf{L}$ genau dann gilt, wenn v die durch φ modellierte Eigenschaft erfüllt. Im Selektionsprozess Evolutionärer Algorithmen können Vektorprädikate eingesetzt werden, um einzelne Individuen auf bestimmte Eigenschaften zu testen. Da aber meist eine ganze Menge von Individuen, also eine Population ausgewertet werden muss, erweitern wir die Vektorprädikate zu allgemeineren Testabbildungen, welche auf durch Relationen repräsentierte Populationen angewandt werden. Die Anwendung einer Testabbildung auf eine Population liefert dann die Menge der Individuen innerhalb der Population, die das zugrundeliegende Vektorprädikat erfüllen. Im Folgenden gehen wir von Populationen der Größe λ aus, die als Relationen des Typs $[X \leftrightarrow [1..\lambda]]$ dargestellt werden. Ein Vektorprädikat $\varphi : [X \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ soll dann in der Weise zu einer Testabbildung

$$\varphi^{[1..\lambda]} : [X \leftrightarrow [1..\lambda]] \to [\mathbf{1} \leftrightarrow [1..\lambda]]$$

modifiziert werden, dass die Anwendung von $\varphi^{[1..\lambda]}$ auf eine Population P einen Zeilenvektor liefert, der die Individuen in P repräsentiert, welche die durch φ modellierte

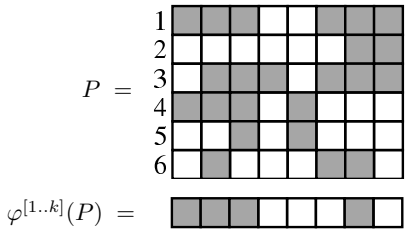


Abbildung 3.2: Auswertung einer Population

Eigenschaft erfüllen. Der in Abbildung 3.2 dargestellte Zeilenvektor gibt die vier Spalten der Relation P an, die die durch φ modellierte Eigenschaft erfüllen. Es gilt hier $\varphi(P^{(j)}) = \mathsf{L}$ für $j \in \{1, 2, 3, 7\}$. Für jedes $\varphi \in \mathcal{VP}$, jede Population P der Größe λ und jedes $j \in [1..\lambda]$ soll also die Äquivalenz

$$\varphi(P^{(j)}) \Longleftrightarrow \varphi^{[1..\lambda]}(P)_{\perp j}$$

gelten. Diese Eigenschaft kann komponentenfrei folgendermaßen formuliert werden: Für alle Relationen $P : X \leftrightarrow [1..\lambda]$ und alle Punkte $p : [1..\lambda] \leftrightarrow \mathbf{1}$ gilt:

$$\varphi(Pp) = \varphi^{[1..\lambda]}(P)p$$

Durch $\varphi^{[1..\lambda]}$ erhält man somit eine Testabbildung zur Auswertung von Populationen, die geeignete Individuen mit bestimmten Eigenschaften aus einer Population herausfiltert. Zur Konstruktion der Menge $\mathcal{VP}$ definieren wir zunächst eine Obermenge $\mathcal{VA}$ (*Vektorabbildungen*) von $\mathcal{VP}$. Sei dazu im Folgenden X eine Menge, sowie $\mathcal{U}$ eine Menge von Mengen mit $X \in \mathcal{U}$ und $\mathbf{1} \in \mathcal{U}$. Wir definieren eine Menge von Abbildungen

$$\mathcal{VA} \subseteq \bigcup_{Y \in \mathcal{U}} [[X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]],$$

so dass für jede Abbildung $\varphi \in \mathcal{VA}$ des Typs $[X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]$ und jede Menge Z eine Abbildung $\varphi^Z : [X \leftrightarrow Z] \to [Y \leftrightarrow Z]$ mit der folgenden Eigenschaft existiert: Für

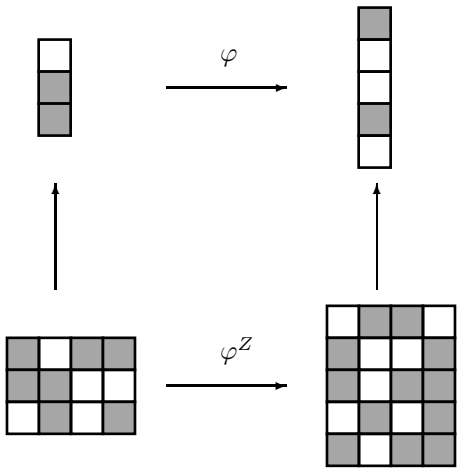


Abbildung 3.3:

jeden Punkt $p : Z \leftrightarrow \mathbf{1}$ und jede Relation $P : X \leftrightarrow Z$ gilt

$$\varphi(Pp) = \varphi^Z(P)p.$$

Diese Gleichung bedeutet anschaulich, dass die Spalten des Bildes von P unter φ^Z genau die Bilder der Spalten von P unter φ sind, also $\varphi^Z(P)^{(z)} = \varphi(P^{(z)})$ für alle $z \in Z$. Für die Konstruktion der Menge $\mathcal{VA}$ benötigen wir zunächst einige Notationen.

3.2.2.1 Definition *Seien A, B, C, D, E Mengen.*

- *Mit $id_{[A \leftrightarrow B]}$ sei die identische Abbildung auf $[A \leftrightarrow B]$ bezeichnet, es sei also*

$id_{[A \leftrightarrow B]} : [A \leftrightarrow B] \to [A \leftrightarrow B]$ *definiert durch*

$$id_{[A \leftrightarrow B]}(R) = R.$$

- *Für jede Relation* $S : C \leftrightarrow D$ *sei* $\chi_S : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *die konstante Abbildung mit*

$$\chi_S(R) = S.$$

- *Für* $\varphi : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *sei* $\overline{\varphi} : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *definiert durch*

$$\overline{\varphi}(R) = \overline{\varphi(R)}.$$

- *Für zwei Abbildungen* $\varphi_1, \varphi_2 : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *seien die Abbildungen* $\varphi_1 \cap \varphi_2, \varphi_1 \cup \varphi_2 : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *definiert durch*

$$(\varphi_1 \cap \varphi_2)(R) = \varphi_1(R) \cap \varphi_2(R) \text{ und } (\varphi_1 \cup \varphi_2)(R) = \varphi_1(R) \cup \varphi_2(R).$$

- *Für jede Abbildung* $\varphi : [A \leftrightarrow B] \to [C \leftrightarrow D]$ *und jede Relation* $S : E \leftrightarrow C$ *sei* $S\varphi : [A \leftrightarrow B] \to [E \leftrightarrow D]$ *definiert durch*

$$(S\varphi)(R) = S\varphi(R).$$

Unter Verwendung dieser Notationen kann nun die Menge $\mathcal{VA}$ induktiv definiert werden.

3.2.2.2 Definition *Sei* $\mathcal{U}$ *eine Menge von Mengen mit* $X \in \mathcal{U}$ *und* $\mathbf{1} \in \mathcal{U}$*. Dann ist die Menge* $\mathcal{VA}$ *von Abbildungen induktiv durch die folgenden fünf Regeln definiert.*

1. *Identität:*

$$id_{[X \leftrightarrow \mathbf{1}]} \in \mathcal{VA}.$$

2. *Konstante Abbildungen: Für jede Menge* $Y \in \mathcal{U}$ *und jeden Vektor* c *des Typs* $[Y \leftrightarrow \mathbf{1}]$ *sei*

$$\chi_c \in \mathcal{VA}.$$

3. *Komplement: Für jedes* $\varphi \in \mathcal{VA}$ *sei auch*

$$\overline{\varphi} \in \mathcal{VA}.$$

4. *Schnitt und Vereinigung: Für je zwei Abbildungen $\varphi_1, \varphi_2 \in \mathcal{VA}$ vom gleichen Typ $[X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]$ sei*

$$\varphi_1 \cap \varphi_2 \in \mathcal{VA} \text{ und } \varphi_1 \cup \varphi_2 \in \mathcal{VA}.$$

5. *Linksmultiplikation: Für alle Mengen $W, Y \in \mathcal{U}$, alle $\varphi \in \mathcal{VA}$ des Typs $[X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]$ und alle $R : W \leftrightarrow Y$ sei*

$$R\varphi \in \mathcal{VA}.$$

Die Abbildungen der eben definierten Menge $\mathcal{VA}$ können auf Vektoren des Typs $[X \leftrightarrow \mathbf{1}]$ angewandt werden und liefern Vektoren des Typs $[Y \leftrightarrow \mathbf{1}]$, wobei $Y \in \mathcal{U}$ ist. Im Folgenden wird jeder Abbildung $\varphi \in \mathcal{VA}$ eine Abbildung φ^Z zugeordnet, welche auf Relationen des Typs $[X \leftrightarrow Z]$ angewandt werden kann und Relationen des Typs $[Y \leftrightarrow Z]$ liefert.

3.2.2.3 Definition *Sei Z eine Menge. Für jedes $\varphi \in \mathcal{VA}$ definieren wir induktiv ein $\varphi^Z \in \bigcup_{Y \in \mathcal{U}} [[X \leftrightarrow Z] \to [Y \leftrightarrow Z]]$*

1. *Identität:*

$$id^Z_{[X \leftrightarrow \mathbf{1}]} := id_{[X \leftrightarrow Z]}$$

2. *Konstante Abbildungen: Für jedes $Y \in \mathcal{U}$ und jeden Vektor $c : Y \leftrightarrow \mathbf{1}$ sei*

$$\chi_c^Z := \chi_{c\mathsf{L}},$$

   wobei L ein Allvektor des Typs $[\mathbf{1} \leftrightarrow Z]$ sei.

3. *Komplement: Für $\varphi \in \mathcal{VA}$ sei*

$$\overline{\varphi}^Z := \overline{\varphi^Z}$$

4. *Schnitt und Vereinigung: Für typgleiche $\varphi_1, \varphi_2 \in \mathcal{VA}$ sei*

$$(\varphi_1 \cap \varphi_2)^Z := \varphi_1^Z \cap \varphi_2^Z$$

   und

$$(\varphi_1 \cup \varphi_2)^Z := \varphi_1^Z \cup \varphi_2^Z.$$

5. *Linksmultiplikation: Für alle Mengen W, $Y \in \mathcal{U}$, Relationen $R : W \leftrightarrow Y$ und Abbildungen $\varphi \in \mathcal{VA}$ des Typs $[X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]$ sei*

$$(R\varphi)^Z := R\varphi^Z.$$

Für jedes $\varphi \in \mathcal{VA}$ mit $\varphi : [X \leftrightarrow \mathbf{1}] \to [Y \leftrightarrow \mathbf{1}]$ hat die zugeordnete Abbildung φ^Z den Typ $[X \leftrightarrow Z] \to [Y \leftrightarrow Z]$. Das folgende Theorem beschreibt den engen Zusammenhang zwischen φ und φ^Z.

3.2.2.4 Theorem *Für jede Menge Z, jedes $\varphi \in \mathcal{VA}$, jeden Punkt $p : Z \leftrightarrow \mathbf{1}$ und jede Relation $P : X \leftrightarrow Z$ gilt die Gleichung*

$$\varphi(Pp) = \varphi^Z(P)p.$$

Beweis: *Im Folgenden sei Z eine Menge, $P : X \leftrightarrow Z$ eine Relation und $p : Z \leftrightarrow \mathbf{1}$ ein Punkt. Der Beweis erfolgt durch strukturelle Induktion.*

I.A.: Die Basis der induktiv definierten Menge $\mathcal{VA}$ besteht aus der identischen Abbildung und den konstanten Abbildungen. Für die identische Abbildung gilt

$$id_{[X \leftrightarrow \mathbf{1}]}(Pp) = Pp = id_{[X \leftrightarrow Z]}(P)p = id^Z_{[X \leftrightarrow \mathbf{1}]}(P)p,$$

was unmittelbar aus Definition 3.2.2.3 folgt.

Sei nun $c : Y \leftrightarrow \mathbf{1}$ ein Vektor mit $Y \in \mathcal{U}$. Seien weiter $\mathsf{L} : \mathbf{1} \leftrightarrow Z$ sowie $\mathsf{L}' : \mathbf{1} \leftrightarrow \mathbf{1}$ Allreationen. Da p ein Punkt und damit surjektiv, und c ein Vektor ist, gilt $c\mathsf{L}p = c\mathsf{L}' = c$ und es folgt für die konstante Abbildung χ_c:

$$\chi_c(Pp) = c = c\mathsf{L}p = \chi_{c\mathsf{L}}(P)p = \chi_c^Z(P)p$$

I.S.: Seien φ_1 und φ_2 typgleiche Abbildungen aus $\mathcal{VA}$ und es gelte $\varphi_i(Pp) = \varphi_i^Z(P)p$ für $i \in \{1, 2\}$. Für $\varphi = \overline{\varphi_1}$ gilt dann:

$$\begin{aligned}
\varphi(Pp) &= \overline{\varphi_1}(Pp) \\
&= \overline{\varphi_1(Pp)} && (\textit{Definition } \overline{\varphi_1}) \\
&= \overline{\varphi_1^Z(P)p} && (\textit{Induktionshypothese}) \\
&= \overline{\varphi_1^Z(P)}p && (p \textit{ injektiv und surjektiv}) \\
&= \overline{\varphi_1^Z}(P)p && (\textit{Definition } \overline{\varphi_1^Z}) \\
&= \overline{\varphi_1}^Z(P)p. && (\textit{Definition 3.2.2.3}) \\
&= \varphi^Z(P)p
\end{aligned}$$

Für $\varphi = \varphi_1 \cap \varphi_2$ gilt

$$\varphi(Pp) = (\varphi_1 \cap \varphi_2)(Pp)$$

$$
\begin{aligned}
&= \varphi_1(Pp) \cap \varphi_2(Pp) && (\textit{Def. } \varphi_1 \cap \varphi_2)\\
&= \varphi_1^Z(P)p \cap \varphi_2^Z(P)p && (\textit{Induktionshypothese})\\
&= (\varphi_1^Z(P) \cap \varphi_2^Z(P))p && (p \textit{ injektiv})\\
&= (\varphi_1^Z \cap \varphi_2^Z)(P)p && (\textit{Def. } \varphi_1^Z \cap \varphi_2^Z)\\
&= (\varphi_1 \cap \varphi_2)^Z(P)p && (\textit{Definition 3.2.2.3})\\
&= \varphi^Z(P)p
\end{aligned}
$$

Die Gleichung

$$(\varphi_1 \cup \varphi_2)(Pp) = (\varphi_1 \cup \varphi_2)^Z(P)p$$

wird analog gezeigt. Sei nun $\varphi = R\varphi$*, wobei* R *eine Relation des Typs* $[W \leftrightarrow Y]$ *mit* $W \in \mathcal{U}$ *ist. Dann gilt:*

$$
\begin{aligned}
\varphi(Pp) &= (R\varphi_1)(Pp)\\
&= R(\varphi_1(Pp)) && (\textit{Def. } R\varphi_1)\\
&= R(\varphi_1^Z(P)p) && (\textit{Induktionshypothese})\\
&= (R(\varphi_1^Z)(P))p\\
&= ((R\varphi_1^Z)(P))p && (\textit{Def. } R\varphi_1^Z\ )\\
&= (R\varphi)_1^Z(P)p && (\textit{Definition 3.2.2.3}).\\
&= \varphi^Z(P)p
\end{aligned}
$$

$$
\begin{array}{ccc}
[X \leftrightarrow Z] & \xrightarrow{\varphi^Z} & [Y \leftrightarrow Z]\\
\Big\downarrow{\scriptstyle \mu_p} & & \Big\downarrow{\scriptstyle \mu_p}\\
[X \leftrightarrow \mathbf{1}] & \xrightarrow{\varphi} & [Y \leftrightarrow \mathbf{1}]
\end{array}
$$

Abbildung 3.4: Kommutatives Diagramm zu Theorem 3.2.2.4

Abbildung 3.4 veranschaulicht die Aussage von Theorem 3.2.2.4. Dabei sei $p : Z \leftrightarrow \mathbf{1}$ ein Punkt und μ_p die Rechtsmultiplikation mit p, also die Abbildung $R \mapsto Rp$. Das Diagramm veranschaulicht die Gleichung

$$\varphi \circ \mu_p = \mu_p \circ \varphi^Z,$$

wobei $\circ$ die Komposition von Abbildungen bezeichnet. Beispiele für die Entwicklung und Verwendung von Abbildungen der Menge $\mathcal{VA}$ finden sich in den Abschnitten 3.2.3.1

und 3.2.3.2 sowie in Kapitel 6. Im Zusammenhang mit Evolutionären Algorithmen ist die Teilmenge $\mathcal{VP} \subseteq \mathcal{VA}$ von besonderem Interesse, die aus den Abbildungen mit dem Resultatbereich $[\mathbf{1} \leftrightarrow \mathbf{1}]$ besteht. Sei also im Folgenden

$$\mathcal{VP} := \mathcal{VA} \cap [[X \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]]$$

die Menge der Vektorprädikate auf X. Wegen Theorem 3.2.2.4 kann jedes Vektorprädi-

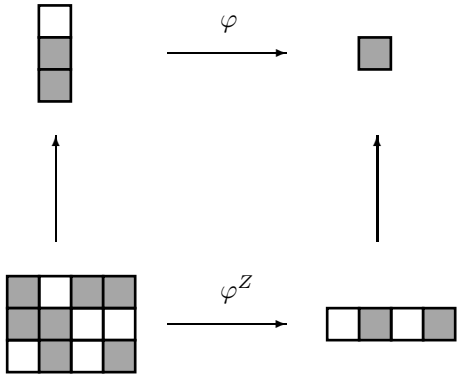


Abbildung 3.5: Ein Vektorprädikat

kat φ zu einer Testabbildung $\varphi^{[1..\lambda]}$ zur Auswertung von Populationen P der Größe λ erweitert werden. $\varphi(P)i^{[1..\lambda]}$ liefert einen Zeilenvektor aus der Menge $[\mathbf{1} \leftrightarrow [1..\lambda]]$, der die Spalten $P^{(j)}$ mit $\varphi(P^{(j)})$ repräsentiert. Die hier definierten Testabbildungen können also im Selektionsprozess von Evolutionären Algorithmen eingesetzt werden, um zu bestimmen, welche Individuen einer Population bestimmte erwünschte Eigenschaften erfüllen und daher für die Übernahme in die nächste Generation in Frage kommen. Kapitel 3.2.3.3 und 3.3.3 liefern Beispiele für dieses Vorgehen.

Vektorprädikate und Testabbildungen können auch in exakten Lösungsverfahren verwendet werden. Hierzu wird die Potenzmengenrelation $\mathsf{M} : X \leftrightarrow 2^X$ verwendet, die alle Vektoren des Typs $[X \leftrightarrow \mathbf{1}]$ als Spalten enthält, und somit den gesamten Suchraum darstellt. Ein Vektorprädikat $\varphi : [X \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$ wird dann, wie in Definition 3.2.2.3 erklärt, zu einer Testabbildung $\varphi^{2^X} : [X \leftrightarrow 2^X] \rightarrow [\mathbf{1} \leftrightarrow 2^X]$ erweitert und auf M angewandt. Der Zeilenvektor $\varphi^{2^X}(\mathsf{M})$ ist dann vom Typ $[\mathbf{1} \leftrightarrow 2^X]$ und spezifiziert alle Spalten von M, die die von φ modellierte Eigenschaft erfüllen. Dieses Vorgehen verwenden wir beispielsweise in Kapitel 5 zur exakten Lösung von Stundenplanproblemen.

3.2.3 Beispiele

Im Folgenden werden Beispiele aus verschiedenen Bereichen der Informatik vorgestellt, in denen sich Vektorabbildungen und Vektorprädikate als praktisch erweisen. Zunächst werden zwei Beispiele behandelt, die nicht aus dem Umfeld der Evolutionären Algorithmen stammen. Das erste Beispiel, aus dem Bereich der Petrinetze, beschäftigt sich mit Bedingungs-Ereignis-Netzen (siehe auch [2]) und zeigt, wie man mithilfe von Vektorabbildungen die durch eine Markierung aktivierten Transitionen sowie Nachfolgemarkierungen in Bedingungs-Ereignis-Netzen berechnen kann. Das zweite Beispiel behandelt Mengensysteme und die Berechnung von Schnittmengen mittels einer Vektorabbildung. Das dritte Beispiel stammt aus dem Bereich der Graphentheorie und wurde bereits in [28] veröffentlicht. Hier werden Vektorprädikate entwickelt, die in Evolutionären Algorithmen eingesetzt werden können, um innerhalb einer Population die Individuen zu bestimmen, die Knotenüberdeckungen, Cliquen oder Unabhängige Mengen eines Graphen repräsentieren.

Eigenschaften von Vektoren werden häufig als Inklusionen ausgedrückt. Für die Umwandlung solcher Inklusionen in Vektorprädikate kann das folgende Lemma verwendet werden.

3.2.3.1 Lemma *Für Vektoren v und w des Typs $[X \leftrightarrow \mathbf{1}]$ gilt die folgende Äquivalenz:*

$$v \subseteq w \iff \overline{\mathsf{L}(v \cap \overline{w})} = \mathsf{L}$$

Beweis: *Da v und w Vektoren sind, gilt $(v \cap \overline{w})\mathsf{L} = v \cap \overline{w}$. Mit der Tarski-Regel folgt*

$$\begin{aligned} v \subseteq w &\iff v \cap \overline{w} = \mathsf{O} \\ &\iff \mathsf{L}(v \cap \overline{w}) = \mathsf{O} \\ &\iff \overline{\mathsf{L}(v \cap \overline{w})} = \mathsf{L}. \end{aligned}$$

Wählt man also L vom Typ $[\mathbf{1} \leftrightarrow X]$, so hat der Ausdruck $\overline{\mathsf{L}(v \cap \overline{w})}$ den Typ $\mathbf{1} \leftrightarrow \mathbf{1}$ und man erhält für jeden Vektor w das Vektorprädikat

$$\varphi : [X \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}], \quad \varphi(v) = \overline{\mathsf{L}(v \cap \overline{w})},$$

so dass $\varphi(v) = \mathsf{L}$ genau dann gilt wenn $v \subseteq w$.

3.2.3.1 Bedingungs-Ereignis-Netze

Bedingungs-Ereignis-Netze bilden eine spezielle Klasse der Petrinetze, auf die in Kapitel 6 noch ausführlich eingegangen wird. In [11] und [2] wird bereits die relationale Modellierung und Analyse von Bedingungs-Ereignis-Netzen diskutiert. Im Folgenden wollen wir dieses Thema erneut, diesmal unter dem Gesichtspunkt der im vorangegangenen Abschnitt definierten Abbildungen, erörtern. Ein Bedingungs-Ereignis-Netz (auch B/E-Netz) kann relational durch ein Tupel $\mathcal{N} = (B, E, R, S)$ dargestellt werden, wobei B und E endliche, zueinander disjunkte Mengen und R und S Relationen des Typs $[B \leftrightarrow E]$ bzw. $[E \leftrightarrow B]$ sind. Die Elemente von B werden Bedingungen, die von E Ereignisse genannt. Zu einem Ereignis e sei $\mathit{pred}(e) = \{b \,|\, R_{be}\}$ die Menge der Vorgänger und $\mathit{succ}(e) = \{b \,|\, S_{eb}\}$ die Menge der Nachfolger von e. Teilmengen von B werden Markierungen des Netzes $\mathcal{N}$ genannt. Eine Markierung M aktiviert ein Ereignis e, falls alle Vorgänger von e und kein Nachfolger von e markiert sind, d.h. wenn

$$\mathit{pred}(e) \subseteq M \;\text{ und }\; \mathit{succ}(e) \subseteq \overline{M}$$

gilt. Im Folgenden werden Markierungen durch Vektoren des Typs $[B \leftrightarrow \mathbf{1}]$ und Mengen von Ereignissen durch Vektoren des $[E \leftrightarrow \mathbf{1}]$ dargestellt und eine Abbildung

$$\varphi_{act} : [B \leftrightarrow \mathbf{1}] \to [E \leftrightarrow \mathbf{1}]$$

entwickelt, die zu einer Markierung die Menge der von ihr aktivierten Ereignisse berechnet. Für eine Markierung M, modelliert durch $m : B \leftrightarrow \mathbf{1}$, gilt:

$$\begin{aligned}
M \text{ aktiviert } e &\iff \mathit{pred}(e) \subseteq M \;\wedge\; \mathit{succ}(e) \subseteq \overline{M} \\
&\iff (\forall b : R_{be} \to m_b) \;\wedge\; (\forall b : S_{eb} \to \neg m_b) \\
&\iff (\neg\exists b : R^{\top}_{eb} \;\wedge\; \overline{m_b}) \;\wedge\; (\neg\exists b : s_{eb} \;\wedge\; m_b) \\
&\iff (\overline{R^{\top}\overline{m}} \cap \overline{Sm})_e \\
&\iff (\overline{R^{\top}\overline{m} \cup Sm})_e
\end{aligned}$$

Man erhält also die Abbildung $\varphi_{act} : [B \leftrightarrow \mathbf{1}] \to [E \leftrightarrow \mathbf{1}]$ mit $\varphi_{act}(m) = (\overline{R^{\top}\overline{m} \cup Sm})$, die zu einer Markierung m die Menge der von m aktivierten Ereignisse in Form eines Vektors des Typs $[E \leftrightarrow \mathbf{1}]$ berechnet. Die Abbildung φ_{act} ist offensichtlich ein Element der in Kapitel 3.2.2 definierten Menge $\mathcal{VA}$. Damit kann φ_{act} für jede Menge Z zu einer Abbildung $\varphi^Z_{act} : [B \leftrightarrow Z] \to [E \leftrightarrow Z]$ mit $\varphi^Z_{act}(P) = (\overline{R^{\top}\overline{P} \cup SP})$ fortgesetzt werden. Dann entspricht für jedes $j \in Z$ der Vektor $\varphi^Z_{act}(P)^{(j)}$ der Menge der Ereignisse,

die von der Markierung $P^{(j)}$ aktiviert werden, es gilt also

$$\varphi^Z_{act}(P)_{ej} \Longleftrightarrow P^{(j)}\text{aktiviert } e.$$

Auch das Schalten von aktivierten Ereignissen kann durch Verwendung einer Vektorabbildung aus $\mathcal{VA}$ realisiert werden. Sei dazu ein Ereignis e gegeben, das von einer Markierung M aktiviert ist. Dann erzeugt das Schalten von e unter M die Nachfolgemarkierung

$$N = (M \cap \overline{pred(e)}) \cup succ(e),$$

d.h. die Markierung M wird dahingehend verändert, dass die Vorgänger des zu schaltenden Ereignisses ihre Markierungen verlieren, während die Nachfolger markiert werden. Modelliert der Vektor $n : B \leftrightarrow \mathbf{1}$ die Menge N, folgt

$$\begin{aligned}
n_b &\Longleftrightarrow b \in N \\
&\Longleftrightarrow b \in (M \cap \overline{pred(e)}) \cup succ(e) \\
&\Longleftrightarrow (m_b \ \wedge \ \overline{R}_{be}) \ \vee \ S_{eb} \\
&\Longleftrightarrow (m_b \ \wedge \ \overline{(R\vec{e})}_b) \ \vee \ (S^\top \vec{e})_b) \qquad \text{(Abschnitt 2.3)} \\
&\Longleftrightarrow ((m \cap \overline{R\vec{e}}) \cup S^\top \vec{e})_b,
\end{aligned}$$

wobei $\vec{e}$, wie in Kapitel 2.3 erläutert, der Punkt vom Typ $[E \leftrightarrow \mathbf{1}]$ ist, der das Ereignis e repräsentiert. Wir erhalten damit eine Abbildung $\varphi_{\mathit{fire}} : [B \leftrightarrow \mathbf{1}] \times [E \leftrightarrow \mathbf{1}] \to [B \leftrightarrow \mathbf{1}]$ durch $\varphi_{\mathit{fire}}(m, p) = (m \cap \overline{Rp}) \cup S^\top p$, so dass $\varphi_{\mathit{fire}}(m, p)$ für jeden Punkt p mit $p \subseteq \varphi_{act}(m)$ die Nachfolgemarkierung modelliert, die aus m durch das Schalten von p entsteht. Die Abbildung φ_{fire} kann dazu verwendet werden, zu einer Markierung m eine Relation zu berechnen, die alle Nachfolgemarkierungen von m repräsentiert. Dafür wird die Abbildung zunächst auf ein festes m eingeschränkt. Offensichtlich ist für jedes m die Abbildung

$$\varphi_{\mathit{fire}_m} : [E \leftrightarrow \mathbf{1}] \to [B \leftrightarrow \mathbf{1}], \quad \varphi_{\mathit{fire}_m}(v) = (m \cap \overline{Rv}) \cup S^\top v$$

ein Element von $\mathcal{VA}$. Sei nun $A \subseteq E$ die durch $\varphi_{act}(m)$ dargestellte Menge, d.h. die Markierung m aktiviert die Menge A von Ereignissen. Wir erhalten gemäß Kapitel 3.2.2 die Abbildung $\varphi^A_{\mathit{fire}_m} : [E \leftrightarrow A] \to [B \leftrightarrow A]$ durch

$$\varphi^A_{\mathit{fire}_m}(P) = (m\mathsf{L} \cap \overline{RP}) \cup S^\top P,$$

wobei L vom Typ $[\mathbf{1} \leftrightarrow A]$ ist. Die Abbildung kann insbesondere auf die Relation $P := inj(\varphi_{act}(m))^\top$ des Typs $[E \leftrightarrow A]$ angewandt werden. Da nach Kapitel 2.3 die Relation $inj(\varphi_{act}(m))$ eindeutig ist, ist P injektiv, somit ist jede Spalte von P ein Punkt. Wegen $inj(\varphi_{act}(m))^\top \mathsf{L} = \varphi_{act}(m)$ ist außerdem jede Spalte von P in $\varphi_{act}(m)$ enthalten. Die Spalten von P repräsentieren also genau die von m aktivierten Ereignisse. Für alle $j \in A$ ist $P^{(j)}$ ein Punkt mit $P^{(j)} \subseteq \varphi_{act}(m)$, für alle $e \in E$ mit $\varphi_{act}(m)_e$ gibt es ein $j \in A$ mit $\vec{e} = P^{(j)}$ und es gilt

$$\varphi^A_{\mathit{fire}_m}(P)^{(j)} = \varphi_{\mathit{fire}_m}(P^{(j)}) = \varphi_{\mathit{fire}}(m, P^{(j)}) = \varphi_{\mathit{fire}}(m, \vec{e}).$$

Die Relation $\varphi^A_{\mathit{fire}_m}(P)$ gibt also alle Markierungen an, die aus m durch das Schalten eines von m aktivierten Ereignisses entstehen, genauer gesagt, jede Spalte $\varphi^A_{\mathit{fire}_m}(P)^{(j)}$ repräsentiert die Markierung, die aus m durch das Schalten des durch den Punkt $P^{(j)}$ dargestellten Ereignisses entsteht.

3.2.3.2 Mengensysteme

Im Folgenden wird eine Vektorabbildung entwickelt, mit der Schnitte von Mengen berechnet werden, und daraus ein Vektorprädikat zur Ermittlung spezieller Teilmengen eines Mengensystems abgeleitet. Die hier entwickelten Abbildungen kommen später im Zusammenhang mit multikriterieller Optimierung zur Anwendung. Sei im Folgenden X eine Menge und $\mathcal{M} \subseteq 2^X$ eine Menge von Teilmengen von X. Wir betrachten im Folgenden Schnitte über diese Mengen, also Ausdrücke der Form

$$\bigcap_{T \in \mathcal{N}} T,$$

wobei $\mathcal{N}$ eine Teilmenge von $\mathcal{M}$ sei. Zur Vereinfachung sei $\mathcal{M}$ endlich mit $|\mathcal{M}| = k$ und $\mathcal{M} = \{T_1, \ldots, T_k\}$. Dann kann $\mathcal{M}$ durch die Relation $M : X \leftrightarrow [1..k]$ mit M_{xj} genau dann wenn $x \in T_j$ modelliert werden. Jede Spalte $M^{(j)}$ enspricht also einer Menge $T_j \in \mathcal{M}$. Für jede Teilmenge $N \subseteq [1..k]$, dargestellt durch einen Vektor $v : [1..k] \leftrightarrow \mathbf{1}$, wird dann der Schnitt $\bigcap_{j \in N} T_j$ von Mengen aus $\mathcal{M}$ durch den Vektor $\overline{\overline{M}v}$ repräsentiert, denn es gilt:

$$\begin{aligned} x \in \bigcap_{j \in N} T_j &\iff \forall j \in N : x \in T_j \\ &\iff \forall j \in N : M^{(j)}_x \\ &\iff \forall j : v_j \rightarrow M_{xj} \end{aligned}$$

$$\begin{aligned}
&\iff \neg\exists j : v_j \ \wedge\ \overline{M}_{xj} \\
&\iff \overline{\overline{M}v}_x
\end{aligned}$$

Man erhält also eine Abbildung $\psi_{cut} : [[1..k] \leftrightarrow \mathbf{1}] \to [X \leftrightarrow \mathbf{1}]$ durch

$$\psi_{cut}(v) = \overline{\overline{M}v},$$

die zu jedem Vektor v einen Vektor $c : X \leftrightarrow \mathbf{1}$ liefert, der die Menge $\bigcap\{T_j \mid v_j\}$ modelliert. Offenbar ist $\psi_{cut} \in \mathcal{VA}$, und kann damit auf beliebige Mengen fortgesetzt werden, d.h. für jede Menge Z gibt es eine Abbildung

$$\psi^Z_{cut} : [[1..k] \leftrightarrow Z] \to [X \leftrightarrow Z],$$

definiert durch $\psi^Z_{cut}(P) = \overline{\overline{M}P}$, und für jedes $i \in Z$ entspricht der Vektor $\psi^Z_{cut}(P)^{(i)}$ dem Schnitt über alle Mengen T_j, für die $P^{(i)}_j$ gilt. Es gilt also

$$\psi^Z_{cut}(P)^{(i)} = \bigcap\{M^{(j)} \mid P^{(i)}_j\}.$$

Interessant sind diese Abbildungen beispielsweise bei der Berechnung von Teilmengen $\mathcal{N} \subseteq \mathcal{M}$, die den gesamten Schnitt erzeugen, für die also

$$\bigcap \mathcal{N} = \bigcap \mathcal{M}$$

gilt. Die Menge $\bigcap \mathcal{M}$, also der Schnitt über alle Mengen in $\mathcal{M}$, entspricht offenbar dem Vektor

$$c := \psi_{cut}(\mathsf{L}).$$

Gesucht werden also Vektoren v, für die $\psi_{cut}(v) = c$ gilt. Für alle $v : [1..k] \leftrightarrow \mathbf{1}$ ist offensichtlich $c \subseteq \psi_{cut}(v)$, es folgt also

$$\begin{aligned}
\psi_{cut}(v) = c &\iff \psi_{cut}(v) \subseteq c \\
&\iff \overline{\overline{M}v} \subseteq c \\
&\iff \overline{\mathsf{L}(\overline{\overline{M}v} \cap \overline{c})} = \mathsf{L} && \text{(Lemma 3.2.3.1)} \\
&\iff \overline{\mathsf{L}(\overline{\overline{M}v} \cup c)} = \mathsf{L}.
\end{aligned}$$

Man erhält das Vektorpädikat $\varphi_{cut} : [[1..k] \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch

$$\varphi_{cut}(v) = \overline{\mathsf{L}(\overline{\overline{M}v} \cup c)},$$

so dass $\varphi_{cut}(v) = \mathsf{L}$ genau dann gilt, wenn $\psi_{cut}(v) = c$. Die Abbildung $\varphi^Z_{cut} : [[1..k] \leftrightarrow Z] \rightarrow [\mathbf{1} \leftrightarrow Z]$, definiert durch $\varphi^Z_{cut}(P) = \mathsf{L}(\overline{\overline{M}P \cup c\mathsf{L}})$, filtert dann aus einer Relation P die Spalten $P^{(i)}$ heraus, für die $\psi_{cut}(P^{(i)}) = c$ gilt, d.h.

$$\varphi^Z_{cut}(P)_{\perp i} \iff \psi_{cut}(P^{(i)}) = c \iff c = \bigcap\{M^{(j)} \mid P^{(i)}_j\}.$$

Die hier entwickelten Abbildungen finden beispielsweise Verwendung bei der Reduzierung von Kriterien in der multikriteriellen Optimierung, wie in Kapitel 3.4 beschrieben.

3.2.3.3 Probleme aus der Graphentheorie

Im Folgenden werden drei bekannte Probleme der Kombinatorischen Optimierung auf Graphen vorgestellt und relational modelliert. Es wird gezeigt, wie man in einfacher Weise Vektorprädikate entwickeln kann, mit deren Hilfe in einer Population diejenigen Individuen bestimmt werden können, die gewisse erwünschte Eigenschaften aufweisen. Sei $G = (V, E)$ ein ungerichteter Graph, also V eine Menge von Knoten und E eine

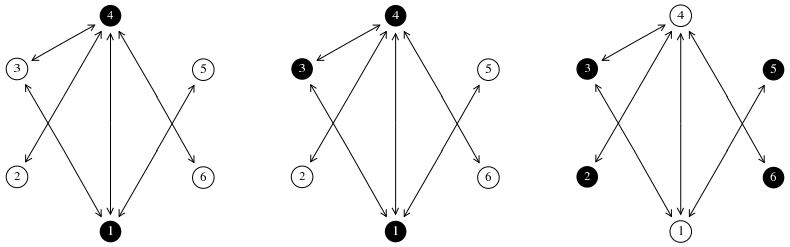


Abbildung 3.6: Knotenüberdeckung, Clique und Unabhängige Menge

Menge von Kanten, d.h. $E \subseteq \{T \subseteq V \mid |T| = 2\}$ eine Menge von zweielementigen Teilmengen von V. Dann heißt eine Teilmenge $T \subseteq V$ eine *Knotenüberdeckung* von G, falls $e \cap T \neq \emptyset$ für jede Kante $e \in E$ gilt. T heißt *Clique*, falls $\{x, y\} \in E$ für alle verschiedenen $x, y \in T$ gilt. Die Menge T heißt *unabhängig*, falls $\{x, y\} \notin E$ für alle $x, y \in T$. Ist das Ziel eines Evolutionären Algorithmus beispielsweise, eine minimale Knotenüberdeckung zu finden, so ist es notwendig, in jedem Selektionsschritt alle Individuen der Population auf die Knotenüberdeckungs-Eigenschaft zu testen, und die Anzahl ihrer Knoten zu bestimmen. Indem man den Ansatz aus Kapitel 3.2.2 verwendet, kann man für ersteres relationale Testabbildungen verwenden, die sich direkt aus Vektorprädikaten ableiten lassen. Zunächst werden Graphen und Teilmengen von Knoten relational modelliert, um dann Vektorprädikate zu entwickeln, mit deren Hilfe man testen kann, ob ein Vektor eine Knotenüberdeckung, eine Clique oder eine unabhängige Menge repräsentiert. Jeder ungerichtete Graph kann durch eine homogene, symmetrische und irreflexive Relation $R : V \leftrightarrow V$, seine Adjazenzrelation, modelliert werden, indem R

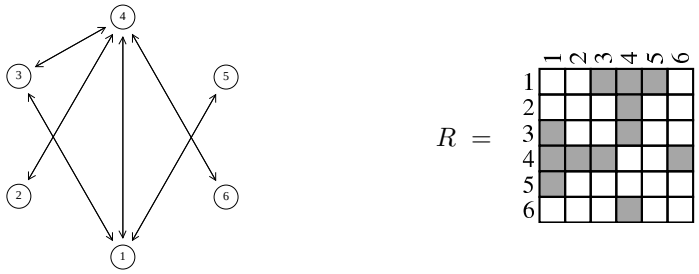


Abbildung 3.7: Ein Graph und seine Adjazenzrelation

definiert wird durch

$$R_{xy} \Longleftrightarrow \{x, y\} \in E.$$

Wie üblich werden Teilmengen von Knoten durch Vektoren $v : V \leftrightarrow \mathbf{1}$ repräsentiert. Im Folgenden werden Vektorprädikate entwickelt, die für solche Vektoren entscheiden, ob es sich um Knotenüberdeckungen, Cliquen oder unabhängige Mengen des Graphen handelt. Ein Vektor v repräsentiert eine Knotenüberdeckung von R genau dann, wenn die prädikatenlogische Formel

$$\forall x, y : R_{xy} \rightarrow (v_x \ \vee \ v_y)$$

erfüllt ist. Wie man leicht sieht, entspricht dies der relationalen Inklusion $R\overline{v} \subseteq v$. Mithilfe von Lemma 3.2.3.1 folgt:

$$v \text{ ist eine Knotenüberdeckung} \iff \overline{\mathsf{L}(R\overline{v} \cap \overline{v})} = \mathsf{L}$$

Wir erhalten also die Abbildung $\varphi_{vc} : [V \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch $\varphi_{vc}(v) = \overline{\mathsf{L}(R\overline{v} \cap \overline{v})}$ und es gilt $\varphi_{vc}(v) = \mathsf{L}$ genau dann, wenn v eine Knotenüberdeckung von R ist. Die

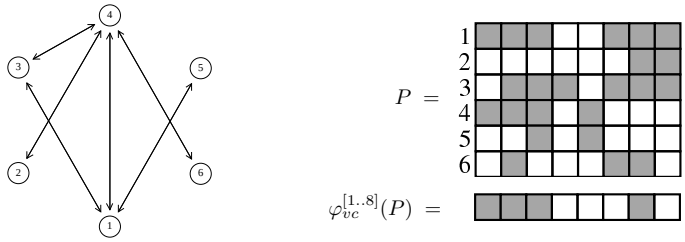


Abbildung 3.8: Knotenüberdeckungen innerhalb der Population P

Abbildung φ_{vc} ist offensichtlich ein Vektorprädikat und kann damit für jede beliebige Menge Z zu einer Testabbildung $\varphi_{vc}^{Z} : [V \leftrightarrow Z] \rightarrow [\mathbf{1} \leftrightarrow Z], \varphi_{vc}^{Z}(P) = \overline{\mathsf{L}(R\overline{P} \cap \overline{P})}$ erweitert werden, mit deren Hilfe aus einer Population $P : V \leftrightarrow Z$, bestehend aus $|Z|$

Individuen, diejenigen heraus gefiltert werden, die Knotenüberdeckungen von R sind. Das heißt, es gilt $\varphi^Z_{vc}(P)_{\perp i}$ genau dann, wenn das Individuum $P^{(i)}$ eine Knotenüberdeckung ist. Da später die relationale Modellierung von Evolutionären Algorithmen am Beispiel des Minimalen Knotenüberdeckungs-Problems erläutert wird, wollen wir an dieser Stelle die entsprechende RelView-Funktion explizit angeben.

```
isVertexCover(R,P) = -(L1n(R)*(R*(-P) & -P))
```

Sie ergibt sich direkt aus der oben entwickelten Testabbildung und berechnet für eine symmetrische, irreflexive Relation $R : X \leftrightarrow X$ und eine Relation P mit Urbildbereich X einen Zeilenvektor, der die Spalten von P angibt, welche Knotenüberdeckungen von R repräsentieren. In den Abschnitten 3.3.3 und 3.3.4 wird erklärt, wie die RelView-Funktion im Selektionsprozess Evolutionärer Algorithmen einsetzbar ist.

Für die Entwicklung eines Vektorprädikats für unabhängige Mengen benutzen wir die Tatsache, dass ein Vektor v genau dann eine unabhängige Menge repräsentiert, wenn $\overline{v}$ eine Knotenüberdeckung ist. Es gilt:

$$\begin{aligned} v \text{ ist eine unabhängige Menge} &\iff \forall x, y : R_{xy} \rightarrow (\overline{v}_x \ \vee \ \overline{v}_y) \\ &\iff \overline{v} \text{ ist eine Knotenüberdeckung} \\ &\iff \overline{\mathsf{L}(Rv \cap v)} = \mathsf{L} \end{aligned}$$

Wir erhalten also $\varphi_{um} : [V \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch $\varphi_{um}(v) = \overline{\mathsf{L}(Rv \cap v)}$ und es gilt $\varphi_{um}(v) = \mathsf{L}$ genau dann, wenn v eine unabhängige Menge in G ist. Weiter ist eine Teilmenge von Knoten genau dann eine Clique in G, wenn sie eine unabhänginge Menge im Komplementgraphen von G ist. Da die Adjazenzrelation des Komplementgraphen die Relation $\overline{R} \cap \overline{\mathsf{I}}$ ist, folgt

$$\begin{aligned} v \text{ ist eine Clique von } R &\iff v \text{ ist eine unabhängige Menge von } \overline{R} \cap \overline{\mathsf{I}} \\ &\iff \overline{\mathsf{L}((\overline{R} \cap \overline{\mathsf{I}})v \cap v)} = \mathsf{L} \\ &\iff \overline{\mathsf{L}((\overline{R \cup \mathsf{I}})v \cap v)} = \mathsf{L}. \end{aligned}$$

Damit erhalten wir das Vektorprädikat $\varphi_{cl} : [V \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch $\varphi_{cl}(v) = \overline{\mathsf{L}((\overline{R \cup \mathsf{I}})v \cap v)}$ und es gilt $\varphi_{cl}(v) = \mathsf{L}$ genau dann, wenn v eine Clique in G ist.

3.3 Variation und Selektion

In diesem Kapitel wird dargestellt, wie Variations- und Selektionsverfahren in Evolutionären Algorithmen durch relationale Terme realisiert werden können. Dabei beschränken wir uns auf zwei Arten von Algorithmen, den $(1+\lambda)$-EA und den $(\lambda+\lambda)$-EA. Eine gekürzte Version der hier vorgestellten Ergebnisse findet sich in [28]. Das Beispiel am Ende des Kapitels verdeutlicht das Vorgehen anhand des Minimalen Knotenüberdeckungs-Problems.

3.3.1 Mutation

Beim $(1+\lambda)$-EA geht man von einem Individuum aus, das durch Mutation λ Nachkommen erzeugt, aus denen dann wiederum genau eines ausgewählt wird. Relational betrachtet, besteht die Aufgabe also darin, aus einem Vektor $v : X \leftrightarrow \mathbf{1}$ durch Mutation eine Population in Form einer Relation $P : X \leftrightarrow [1..\lambda]$ zu erzeugen, wobei sich jeder Eintrag jedes Individuums, also jeder Spalte von P, mit einer gewissen Wahrscheinlichkeit p von dem entsprechenden Eintrag von v unterscheidet. Das heißt, für alle $j \in [1..\lambda]$ soll $Pr(\overline{P}_x^{(j)}) = p$ gelten, falls v_x gilt, und $Pr(P_x^{(j)}) = p$, falls v_x nicht gilt. Zur Berechnung der Nachfolge-Population P verwenden wir eine zufällig erzeugte Relation $M : X \leftrightarrow [1..\lambda]$, für die $Pr(M_{xj}) = p$ für alle $x \in X$ und alle $j \in [1..\lambda]$ gilt. Die Kinder von v werden dann modelliert durch die Relation

$$P = (v\mathsf{L} \cap \overline{M}) \cup (\overline{v\mathsf{L}} \cap M),$$

wobei L vom Typ $[\mathbf{1} \leftrightarrow [1..\lambda]]$ ist. Für $x \in X$ mit $\overline{v}_x$ folgt $(\overline{v\mathsf{L}})_{xj}$ für alle $j \in [1..\lambda]$, und damit gilt

$$\begin{aligned}
P_x^{(j)} &\iff P_{xj} \\
&\iff ((v\mathsf{L} \cap \overline{M}) \cup (\overline{v\mathsf{L}} \cap M))_{xj} \\
&\iff (\overline{v\mathsf{L}} \cap M)_{xj} && (\mathsf{da}\ (\overline{v\mathsf{L}})_{xj}\ \mathsf{gilt}) \\
&\iff M_{xj}
\end{aligned}$$

für alle $j \in [1..\lambda]$. Es folgt sofort $Pr(P_x^{(j)}) = Pr(M_{xj}) = p$ für alle x mit $\overline{v}_x$. Um zu zeigen, dass $Pr(\overline{P}_x^{(j)}) = p$ für alle x mit v_x gilt, verwenden wir die folgende Gleichung:

$$\begin{aligned}
\overline{P} &= \overline{((v\mathsf{L} \cap \overline{M}) \cup (\overline{v\mathsf{L}} \cap M))} \\
&= ((\overline{v\mathsf{L}} \cup M) \cap (v\mathsf{L} \cup \overline{M})) \\
&= (\overline{v\mathsf{L}} \cap v\mathsf{L}) \cup (\overline{v\mathsf{L}} \cap \overline{M}) \cup (M \cap v\mathsf{L}) \cup (M \cap \overline{M})
\end{aligned}$$

$$= (\overline{v\mathsf{L}} \cap \overline{M}) \cup (M \cap v\mathsf{L}).$$

Es gelte nun v_x und damit $(v\mathsf{L})_{xj}$ für alle j. Dann folgt

$$\begin{aligned}
\overline{P}_x^{(j)} &\Longleftrightarrow \overline{P}_{xj} \\
&\Longleftrightarrow ((\overline{v\mathsf{L}} \cap \overline{M}) \cup (M \cap v\mathsf{L}))_{xj} \\
&\Longleftrightarrow (M \cap v\mathsf{L})_{xj} \\
&\Longleftrightarrow M_{xj},
\end{aligned}$$

und damit gilt $Pr(\overline{P}_x^{(j)}) = Pr(M_{xj}) = p$. Man erhält das folgende RelView-Programm, dass zu einem Vektor v eine Population von $|l|$ Nachkommen erzeugt, wobei l eine Allrelation eines beliebigen Zeilenvektor-Typs ist. Der Parameter $p : Y \leftrightarrow \mathbf{1}$ bestimmt den Füllgrad der Relation M in der Weise, dass $pr(M_{ij}) = |p|/|Y|$ gilt.

```
mutation1(v,p,l)
  DECL V, M, P
  BEG V = v*l;
      M = random(V,p);
      P = (V & -M) | (-V & M)
      RETURN P
  END.
```

Der $(\lambda+\lambda)$-EA arbeitet ausschließlich mit Populationen der Größe λ. Jedes Individuum der Elternpopulation erzeugt durch Mutation einen Nachkommen, der sich an jeder Stelle mit Wahrscheinlichkeit p von ihm unterscheidet. Das bedeutet, aus einer Relation $P : X \leftrightarrow [1..\lambda]$, welche die Elternpopulation modelliert, soll eine Relation C des gleichen Typs erzeugt werden, so dass für alle $j \in [1..\lambda]$ und alle $x \in X$ gilt: Aus $P_x^{(j)}$ folgt $Pr(\overline{C}_x^{(j)}) = p$ und aus $\overline{P}_x^{(j)}$ folgt $Pr(C_x^{(j)}) = p$. Auch hier wird eine zufällig erzeugte Relation M des Typs $[X \leftrightarrow [1..\lambda]]$ verwendet, um die Nachfolge-Population durch

$$C = (P \cap \overline{M}) \cup (\overline{P} \cap M)$$

zu erzeugen. Für $x \in X$ und $j \in [1..\lambda]$ mit $\overline{P}_x^{(j)}$ folgt

$$\begin{aligned}
C_x^{(j)} &\Longleftrightarrow C_{xj} \\
&\Longleftrightarrow (P \cap \overline{M}) \cup (\overline{P} \cap M)_{xj} \\
&\Longleftrightarrow (\overline{P} \cap M)_{xj} && (\text{da } \overline{P}_{xj} \text{ gilt }) \\
&\Longleftrightarrow M_{xj}.
\end{aligned}$$

Setzt man also auch hier für alle $x \in X, j \in [1..\lambda]$ die Wahrscheinlichkeit $Pr(M_{xj}) = p$ voraus, so erhält man $Pr(C_x^{(j)}) = Pr(M_{xj}) = p$ für alle $x \in X$ und $j \in [1..\lambda]$ mit $\overline{P}_x^{(j)}$. Analog zu oben wird gezeigt, dass $\overline{C} = (\overline{P} \cap \overline{M}) \cup (P \cap M)$ gilt. Aus $P_x^{(j)}$ folgt damit

$$\begin{aligned}\overline{C}_x^{(j)} &\Longleftrightarrow \overline{C}_{xj} \\ &\Longleftrightarrow ((\overline{P} \cap \overline{M}) \cup (P \cap M))_{xj} \\ &\Longleftrightarrow (P \cap M)_{xj} \\ &\Longleftrightarrow M_{xj}.\end{aligned}$$

Es gilt also $Pr(\overline{C}_x^{(j)}) = Pr(M_{xj}) = p$ für x und j mit $P_x^{(j)}$. Das folgende REL-VIEW-Programm erzeugt aus der Population P durch Mutation jedes Individuums eine Nachfolge-Population.

```
mutation2(P,p)
  DECL M, C
  BEG M = random(P,p);
      C = (P & -M) | (-P & M)
      RETURN C
  END.
```

Auch hier bestimmt der Parameter p die Mutations-Wahrscheinlichkeit.

3.3.2 Rekombination

Im Fall des $(\lambda + \lambda)$-EA hat man als weiteren Variationsoperator neben der unären Mutation noch die binäre Rekombination (auch: Crossover) zur Verfügung. Hierbei werden zunächst randomisiert λ Paare aus den Individuen der Elternpopulation P gebildet. Jedes Paar generiert dann ein Individuum der Offspring-Generation, wobei aus beiden Elternteilen Informationen übernommen werden. Bei der relationalen Modellierung der Rekombination erfolgt die Paarbildung durch die Erzeugung einer Population P', die aus P durch Permutation der Spalten entsteht. Dann bilden $(P^{(j)}, P'^{(j)})$ für $j \in [1..\lambda]$ die Elternpaare, die im Folgenden jeweils ein Kind erzeugen. Um festzulegen, welche Einträge welches Elternteils auf das Kind übergehen, verwenden wir, wie bei der Mutation, eine Relation $M : X \leftrightarrow [1..\lambda]$ und erzeugen die Offspring-Generation C durch

$$C = (P \cap M) \cup (P' \cap \overline{M}).$$

Die verschiedenen Arten von Crossover können durch die Verwendung unterschiedlicher Relationen M realisiert werden. Für das uniforme Crossover wird eine zufällig erzeugte

Relation M mit $Pr(M_{xj}) = \frac{1}{2}$ verwendet. Falls M_{xj} gilt, folgt $C_x^{(j)}$ genau dann wenn $P_x^{(j)}$, d.h., der Eintrag an der Stelle x wird vom Individuum $P^{(j)}$ übernommen. Im Fall von $\overline{M}_{xj}$ gilt $C_x^{(j)}$ genau dann wenn $P_x'^{(j)}$, und damit wird der Eintrag an der Stelle x vom Individuum $P'^{(j)}$ übernommen. Da beide Situationen mit Wahrscheinlichkeit $\frac{1}{2}$ eintreten, handelt es sich bei diesem Vorgehen um uniformes Crossover.

Zur Realisierung des 1-Point-Crossover wählt man für jede Spalte j eine Trennstelle k_j und definiert M durch

$$M_{xj} \Longleftrightarrow 1 \leq x \leq k_j$$

für alle $j \in [1..\lambda]$. Dann gilt $C_x^{(j)}$ genau dann wenn $P_x^{(j)}$ für alle $1 \leq x \leq k_j$, und $C_x^{(j)}$ genau dann wenn $P_x'^{(j)}$ für $k_j \leq x \leq n$. Damit ergibt sich das Individuum $C^{(j)}$ aus $P^{(j)}$

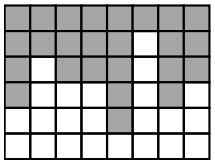


Abbildung 3.9: Die Relation M für das 1-Point-Crossover

und $P'^{(j)}$, indem alle Einträge zwischen 1 und k_j von $P^{(j)}$, und die restlichen von $P'^{(j)}$ übernommen werden.

In vielen Fällen soll der Crossover-Operator nicht auf alle Individuen in jedem Durchlauf des Algorithmus angewandt werden, sondern jedes Paar $(P^{(j)}, P'^{(j)})$ wird mit einer gewissen Wahrscheinlichkeit p_c für die Rekombination ausgewählt. Um dieses Vorgehen relational zu modellieren, benötigen wir einen Zeilenvektor $m : \mathbf{1} \leftrightarrow [1..\lambda]$ mit $Pr(m_{\perp j}) = p_c$ für alle $j \in [1..\lambda]$. Dann wird aus der ursprünglichen Relation M eine neue Relation M' desselben Tys durch

$$M' = M \cap \mathsf{L}m$$

erzeugt, wobei die Allrelation L ein Vektor vom Typ $[X \leftrightarrow \mathbf{1}]$ ist. Der Crossover-Operator wird dann unter Verwendung von M' angewandt, es gilt also $C = (P \cap M') \cup (P' \cap \overline{M'})$. Dadurch folgt aus $m_{\perp j}$, dass $M'^{(j)} = M^{(j)}$ gilt, und damit ist

$$C^{(j)} = (P^{(j)} \cap M^{(j)}) \cup (P'^{(j)} \cap \overline{M}^{(j)}).$$

Das Individuum $C^{(j)}$ ensteht also durch Crossover aus $P^{(j)}$ und $P'^{(j)}$, indem die Relation M verwendet wird. Gilt hingegen $\overline{m}_{\perp j}$, so ist $M'^{(j)} = \mathsf{O}$ und damit $C^{(j)} = P'^{(j)}$. Das

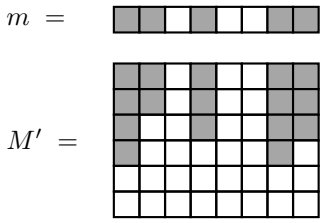


Abbildung 3.10: Die Relationen m und M'

Individuum wird also direkt aus der Elternpopulation übernommen. Durch das folgende RelView-Programm wird auf der Population P uniformes Crossover durchgeführt.

```
crossover(P,p)
  DECL Q,m, M, C
  BEG Q = P*randomperm(L1n(P)^);
      m = random(L1n(P),p);
      M =  random50(P) & Ln1(P)*m;
      C = (P & M) | (Q & -M)
      RETURN P
  END.
```

Der Parameter p bestimmt, mit welcher Wahrscheinlichkeit jedes Paar für die Rekombination ausgewählt wird.

3.3.3 Selektion

Im Gegensatz zu den eben beschriebenen Variationsoperatoren, die problemunabhänging sind, hängt der Selektionsschritt eines EAs wesentlich von dem gewählten Problem ab. In diesem Abschnitt beschäftigen wir uns ausschließlich mit einkriterieller Optimierung und gehen von einem Minimierungsproblem aus, d.h., es soll im Suchraum $\mathcal{S} = \{0,1\}^n$ ein Individuum v mit möglichst geringem Fitness-Wert $f(v)$ gefunden werden, wobei f eine Funktion des Typs $\mathcal{S} \to \mathbb{R}_+$ ist. Zur Verdeutlichung nehmen wir außerdem Bezug auf Abschnitt 3.2.3.3 und erläutern die vorgestellten Verfahren anhand des Problems, eine Minimale Knotenüberdeckung eines Graphen G zu finden. Unserem Beispiel, bezogen auf einen Graphen mit n Knoten, liegt die Fitnessfunktion

$$f(v) = \begin{cases} |v| & : \quad \varphi_{vc}(v) = \mathsf{L} \\ n & : \quad sonst \end{cases}$$

zugrunde. Für zwei Suchpunkte v und v' gilt offenbar $f(v) \leq f(v')$, wenn v eine Knotenüberdeckung ist und dies für v' nicht zutrifft. Die Verwendung von f sichert also zu,

dass im Laufe des Algorithmus nie Individuen in die neue Generation übernommen werden, die keine Knotenüberdeckungen des Graphen sind, sofern man als Initialisierung ebenfalls eine (oder mehrere) Knotenüberdeckungen wählt. Im Relationalen Modell wird dieses Vorgehen realisiert, indem aus einer Population $P : X \leftrightarrow [1..\lambda]$ zunächst alle Knotenüberdeckungen herausgefiltert werden, die dann im Folgenden auf die Anzahl ihrer Knoten untersucht werden. Wir berechnen also zunächst den Zeilenvektor

$$\varphi_{vc}^{[1..\lambda]}(P) : \mathbf{1} \leftrightarrow [1..\lambda],$$

der die Menge der Knotenüberdeckungen in der Population P repräsentiert. Im Folgenden konzentriert sich der Selektionsprozess nur noch auf die Individuuen $P^{(j)}$ mit $\varphi_{vc}^{[1..\lambda]}(P)_{\perp j}$.

Im Selektionsschritt stehen für die relationale Modellierung eines einkriteriellen $(1+\lambda)$-EA mit Fitnessfunktion f im wesentlichen zwei Vorgehensweisen zur Verfügung. Ist v das Elternindividuum und P die aus v durch Mutation hervorgegangene Population, wählt man im Selektionsschritt sinnvollerweise ein Individuum $P^{(j)}$ aus, dessen Fitness-Wert höchstens so groß ist wie der von v, d.h. es gilt $f(P^{(j)}) \leq f(v)$. Hierbei wählt man entweder zufällig aus der Menge $M = \{P^{(j)} \mid f(P^{(j)}) \leq f(v)\}$, oder man entscheidet sich zielgerichtet für ein Individuum mit dem kleinsten Fitness-Wert in M. In der relationalen Modellierung vergleicht man daher die Werte $|P^{(j)}|$ für j mit $\varphi_{vc}^{[1..\lambda]}(P)_{\perp j}$ mit $|v|$ und wählt eine geeignete Spalte aus. Um überflüssige Vergleiche zu vermeiden, kann außerdem eine Abbildung verwendet werden, mit der getestet wird, welche Spalten der Population Obermengen von v sind. Es gilt nach Lemma 3.2.3.1 für jeden Vektor w, dass $v \subseteq w$ genau dann gilt, wenn $\overline{\mathsf{L}(v \cap \overline{w})} = \mathsf{L}$, also erhalten wir durch $\varphi_{sup_v}(w) = \overline{\mathsf{L}(v \cap \overline{w})}$ das Vektorprädikat φ_{sup_v} zum Test auf die Obermengen-Beziehung bezüglich v. Für die Testabbildung $\varphi_{sup_v}^{[1..k]} : [X \leftrightarrow [1..k]] \to [\mathbf{1} \leftrightarrow [1..k]]$ mit

$$\varphi_{sup_v}^{[1..k]}(P) = \overline{\mathsf{L}(v\mathsf{L} \cap \overline{P})}$$

gilt damit $\varphi_{sup_v}^{[1..k]}(P)_{\perp i}$ genau dann wenn $v \subseteq P^{(i)}$. Zunächst wird also

$$d = (\varphi_{vc}^{[1..k]} \cap \overline{\varphi_{sup_v}^{[1..k]}})(P)$$

berechnet, um anschließend für j mit $d_j^\top$ die Kardinalität von $P^{(j)}$ und v zu vergleichen. Mit

```
isSuperSet(P,v) = -(Ln1(v)^*(v*L1n(P) & -P))
```

und der RelView-Funkion `isVertexCover` aus Abschnitt 3.2.3.3 erhalten wir das folgende Programm zur Selektion eines geeigneten Individuums aus einer Population P.

```
selection(R,v,P)
  DECL w, p
  BEG  w = v;
       d = (isVertexCover(R,P) & -isSuperSet(P,v))^;
       WHILE -empty(d) DO
          p = point(d);
          IF cardgt(P*p,v) THEN d = d & -p
                           ELSE w = P*p; d = 0(d)
          FI
       OD
       RETURN w
  END.
```

Hierbei wird $w = P^{(j)}$ zurückgeliefert mit $j = min\{i \mid d_i^\top \wedge |P^{(i)}| \leq |v|\}$ falls ein solches j existiert. Anderenfalls wird $w = v$ zurückgegeben. Wir verwenden an dieser Stelle die Basisfunktion `cardgt`, die zwei Relationen Q und S bezüglich der Anzahl ihrer Einträge vergleicht und `L` zurückliefert, sofern $|Q| > |S|$ gilt. Da die Individuen aus P zufällig erzeugt sind und somit auch ihre Positionen in P zufällig sind, entspricht dieses Vorgehen einer zufälligen Wahl eines Individuums aus der Menge $\{P^{(j)} \mid f(P^{(j)}) \leq f(v)\}$. Durch die Abwandlung der While-Schleife zu

```
WHILE -empty(d) DO
  p = point(d);
  IF cardgt(P*p,w) THEN w = P*p FI
  d = d & -p
OD
```

ermittelt man ein Individuum mit der geringsten Anzahl von Knoten innerhalb der Menge $\{P^{(i)} \mid d_i^\top\} \cup \{v\}$.

Beim $(\lambda + \lambda)$-EA wird häufig mit Turnierselektion gearbeitet. Dabei werden zufällig λ Turniere aus je einem Individuum der Elternpopulation und einem Individuum der Offspringpopulation gebildet und dasjenige mit dem besseren (hier: geringerem) Fitness-Wert in die neue Generation übernommen. Seien also Relationen $P, C : X \leftrightarrow [1..\lambda]$ gegeben, die die Elternpopulation, bzw. Offspringpopulation modellieren. Es ensteht

dann $\tilde{C}$ aus C durch zufällige Permutation der Spalten. Die Turniere sind dann alle Paare $(P^{(j)}, \tilde{C}^{(j)})$ mit $j \in [1..\lambda]$. Schließlich wird entsprechend der Fitnessfunktion f ein Zeilenvektor $d : \mathbf{1} \leftrightarrow [1..\lambda]$ berechnet, der angibt, welches Individuum jeden Paares den besseren Fitness-Wert besitzt, und damit in die neue Generation N übernommen wird. Der *Entscheidungsvektor*[1] d soll die Individuen spezifizieren, die aus der Elternpopulation übernommen werden, es gilt also

$$d_{\perp j} \Longleftrightarrow f(P^{(j)}) \leq f(\tilde{C}^{(j)}).$$

Da wir in unserem Beispiel davon ausgehen, dass alle Individuen der Elternpopulation Knotenüberdeckungen repräsentieren, und da Individuen, die keine Knotenüberdeckungen sind, grundsätzlich verworfen werden, können wir die Äquivalenz schreiben als

$$d_{\perp j} \Longleftrightarrow \overline{\varphi_{vc}(\tilde{C}^{(j)})} \ \vee \ |P^{(j)}| \leq |\tilde{C}^{(j)}|.$$

Um auch hier die Anzahl der Kardinalitätsvergleiche zu minimieren, verwenden wir wie eben eine (diesmal zweistellige) Testabbildung

$$\varphi_{sup} : [X \leftrightarrow [1..k]] \times [X \leftrightarrow [1..k]] \rightarrow [\mathbf{1} \leftrightarrow [1..k]]$$

mit

$$\varphi_{sup}(A, B) = \overline{\mathsf{L}(B \cap \overline{A})}$$

für die $\varphi_{sup}(A, B)_{\perp i}$ genau dann wenn $B^{(i)} \subseteq A^{(i)}$ gilt. Mit

$$\tilde{d} = \overline{\varphi_{vc}^{[1..k]}}(\tilde{C}) \cup \varphi_{sup}(\tilde{C}, P)$$

erhalten wir also einen Teilvektor von d. Unter Verwendung der Funktion

```
isSuperSet(C,P) = -(Ln1(P)^*(P & -C)).
```

erhalten wir also das folgende Programm zur Berechnung des Entscheidungsvektors d, wobei wir als Eingaberelation C bereits die spaltenpermutierte Offspringpopulation voraussetzen.

[1] Der Entscheidungsvektor ist kein Vektor im eigentlichen Sinne, sondern ein Zeilenvektor.

```
tournDecision(R,P,C)
   DECL d,z,p
   BEG d = -isVertexCover(R,C) | isSuperSet(C,P);
       z = -d^;
       WHILE -empty(z) DO
         p = point(z);
         IF cardgt(C*p, P*p) THEN d = d | p^ FI;
         z = z & -p
       OD
       RETURN d
   END.
```

Die neue Generation N wird dann durch

$$N = (P \cap \mathsf{L}d) \cup (\tilde{C} \cap \overline{\mathsf{L}d})$$

berechnet, wobei L vom Typ $[X \leftrightarrow \mathbf{1}]$ ist. Es folgt dann $N^{(j)} = P^{(j)}$ aus $d_{\perp j}$, denn für alle $x \in X$ gilt

$$\begin{aligned}
N_x^{(j)} &\iff N_{xj} \\
&\iff ((P \cap \mathsf{L}d) \cup (\tilde{C} \cap \overline{\mathsf{L}d}))_{xj} \\
&\iff (P \cap \mathsf{L}d)_{xj} && (\overline{(\tilde{C} \cap \overline{\mathsf{L}d})}_{xj}) \\
&\iff P_{xj} && ((\mathsf{L}d)_{xj}) \\
&\iff P_x^{(j)}.
\end{aligned}$$

Analog wird gezeigt, dass $N^{(j)} = \tilde{C}^{(j)}$ im Fall von $\overline{d}_{\perp j}$ gilt. Im ersten Fall wird also das

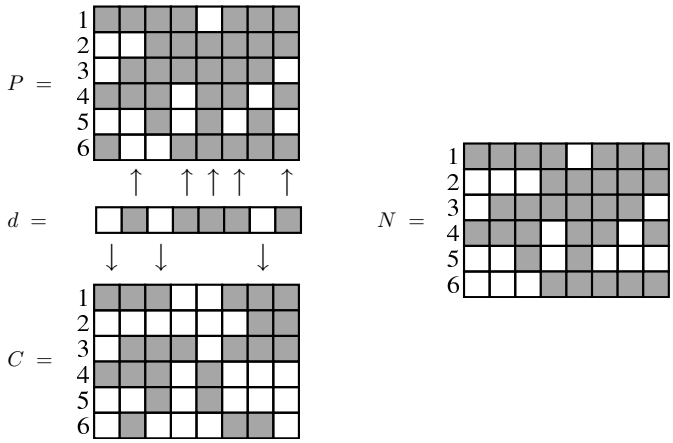


Abbildung 3.11: Selektion mithilfe des Entscheidungsvektors d

Individuum aus der Elternpopulation P in die neue Generation N übernommen, im zweiten Fall das Individuum aus der Offspring-Population, also immer dasjenige mit dem geringen Fitness-Wert. Man erhält

```
tournSelection(P,C,d)= (P & Ln1(P)*d) | (C & -(Ln1(P)*d))
```

wobei C wieder als die spaltenpermutierte Relation $\tilde{C}$ vorausgesetzt wird und der Parameter d dem Resultat von `tournDecision(R,P,C)` entspricht.

3.3.4 Beispiel

Im Folgenden werden unter Verwendung der bisher eingeführten Funktionen und Programme zwei Evolutionäre Algorithmen für das Minimale Knotenüberdeckungs-Problem vorgestellt. Teile der hier vorgestellten Ergebnisse wurden bereits in [29] und [28] veröffentlicht. Die relationale Variante des $(1+\lambda)$-EA benötigt fünf Eingabeparameter R, v, p, l, t, wobei R die Adjazenzrelation des betrachteten Graphen ist, v ein Vektor, der eine Knotenmengen repräsentiert, welche eine Knotenüberdeckung des Graphen ist, p ein Vektor zur Festlegung der Mutations-Wahrscheinlichkeit, l ein Zeilenvektor der Länge λ zur Festlegung der Populationsgröße und t ein Allvektor beliebiger Größe, der die Anzahl der erzeugten Generationen bestimmt. Das folgende RelView-Programm stellt eine einfache Möglichkeit dar, den $(1+\lambda)$-EA zu implementieren.

```
EA1(R,v,p,l,t)
   DECL  z, w, P, d
   BEG z = t;
       w = v;
       WHILE -empty(z) DO
          P = mutation1(w,p,l);
          w = selection(w,P);
          z = z & -point(z)
        OD
        RETURN w
    END.
```

In jedem Schleifendurchlauf wird durch Mutation aus dem Elternindividuum w eine Population P generiert und aus dieser ein neues Elternindividuum ausgewählt. Nach $|t|$ Durchläufen gibt der Algorithmus das aktuelle Individuum zurück. Eine alternative Terminerungbedingung besteht darin, durch t eine maximale Anzahl von Durchläufen

anzugeben, bei denen sich der Suchpunkt w nicht verändert. Dies kann innerhalb der While-Schleife folgendermaßen realisiert werden

```
WHILE -empty(z) DO
  P = mutation1(w1,p,l);
  w2 = selection(w1,P);
  IF eq(w1,w2) THEN z = z & -point(z) ELSE z = t FI;
  w1 = w2
OD
```

Der Algorithmus bricht also ab, wenn in $|t|$ hintereinander erfolgenden Durchläufen kein neues Elternindividuum gefunden wird. Die im Folgenden beschriebenen Experimente wurden mit dieser zweiten Variante des $(1 + \lambda)$-EA durchgeführt, wobei als Parameter p ein Punkt des Typs $[X \leftrightarrow \mathbf{1}]$ gewählt wurde. Wegen $|p| = 1$ und $|X| = n$ gilt damit $Pr(M_{ij}) = \frac{|p|}{|X|} = \frac{1}{n}$ für die durch `mutation1(w1,p,l)` erzeugten Relationen M und alle $i \in X$, $j \in [1..\lambda]$. Somit unterscheidet sich jeder Eintrag jedes Individuums in der Population P mit Wahrscheinlichkeit $\frac{1}{n}$ vom entsprechenden Eintrag des Elternindividuums. Als initiale Knotenüberdeckung v wurde der Allvektor des Typs $[X \leftrightarrow \mathbf{1}]$ verwendet.

Das Programm `EA1` wurde zum einen mit zwei anderen relationalen Algorithmen zur Approximierung einer minimalen Knotenüberdeckung verglichen, zum anderen wurden die Laufzeiten des `EA1` in Abhängigkeit der Parameter λ und t ermittelt.

Zum Vergleich wurde die folgende relationale Imlementierung des Algorithmus von Gavril und Yannakakis (siehe [16]) verwendet.

```
Gavril(R)
  DECL E, c, e
  BEG  c = On1(R);
       E = R;
       WHILE -empty(E) DO
         e = edge(E);
         c = c | dom(e);
         E = E & -(e*L(e) & R) & -(L(e)*e & R)
         OD
       RETURN c
  END.
```

Dabei wird, ausgehend von der leeren Menge, eine Knotenüberdeckung erzeugt, indem sukzessive eine Kante $e = \{x, y\}$ aus dem Graphen gewählt wird, beide Knoten x und y in die Menge übernommen und alle mit x und y inzidenten Kanten aus dem Graphen entfernt werden, bis man den leeren Graphen erhält. Auf diese Weise wird eine Knotenüberdeckung berechnet, das höchstens doppelt so viele Knoten enthält wie eine optimale Lösung.

In [3] wird ein generischer Algorithmus zur Berechnung inklusionsminimaler Teilmengen mit bestimmten Eigenschaften entwickelt, der sich in Kombination mit dem Algorithmus von Gavril und Yannakakis zur Berechnung inklusionsminimaler Knotenüberdeckungen eignet.

```
GavrilMinCover(R)
 DECL c, a, b, p
 BEG  c = Gavril(R);
      a = c;
      b = c;
      WHILE -empty(b) DO
        p = point(b);
        IF incl(R*p,a) THEN a = a & -p FI;
        b = b & -p
        OD
      RETURN a
 END.
```

Hierbei wird, ausgehend von einer durch das Programm `Gavril` ermittelten Knotenüberdeckung c eine inklusionsminimale, in c enthaltende Knotenüberdeckung berechnet, indem Knoten aus aus c entfernt werden, falls dadurch die Knotenüberdeckungs-Eigenschaft nicht verletzt wird. In Abbildung 3.12 werden die Kardinalitäten der durch die drei Algorithmen `Gavril`, `GavrilMinCover` und `EA1` ermittelten Knotenüberdeckungen von 20 zufälligen Graphen mit 100 Knoten und ungefähr 500 Kanten dargestellt. Dabei wurde $\lambda = 100$ und $t = 272$ gewählt. Um mit großer Wahrscheinlichkeit zu gewährleisten, dass der $(1 + \lambda)$-EA zumindest inklusionsminimale, und damit mindestens so gute Resultate wie der Algorithmus aus [3] erzielt, sollte die Gleichung

$$\lambda t = ek^2$$

gelten, wie im Folgenden näher erläutert wird. Wir setzen zunächst einen Suchpunkt w

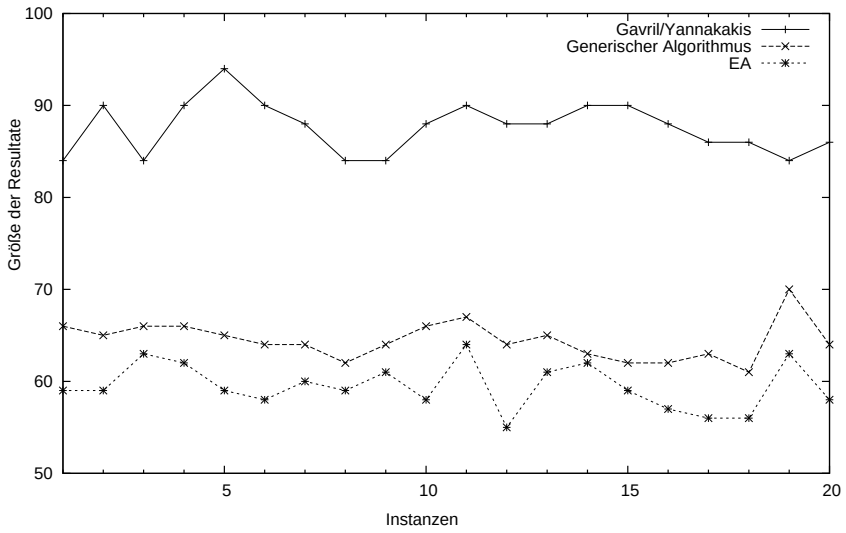


Abbildung 3.12: Vergleich der Algorithmen

voraus, der eine nicht inklusionsminimale Knotenüberdeckung darstellt, woraus folgt, dass ein Punkt $x : X \leftrightarrow \mathbf{1}$ existiert, so dass $w \cap \overline{x}$ ebenfalls eine Knotenüberdeckung ist. Die Wahrscheinlichkeit, dass durch Mutation von w das Individuum $w \cap \overline{x}$ erzeugt wird, entspricht der Wahrscheinlichkeit, dass es ein $j \in [1..\lambda]$ mit $M^{(j)} = x$ gibt. Es repräsentiere der Punkt x das Element $i \in X$, dann ergibt sich für jedes $j \in [1..\lambda]$ die folgende Wahrscheinlichkeit:

$$\begin{aligned} Pr(M^{(j)} = x) &= Pr(M_i^{(j)}) \prod_{\substack{\ell \in X \\ \ell \neq i}} Pr(\overline{M}_\ell^{(j)}) \\ &= \frac{1}{n} \prod_{\substack{\ell \in X \\ \ell \neq i}} (1 - \frac{1}{n}) \\ &= \frac{1}{n}(1 - \frac{1}{n})^{n-1} \end{aligned}$$

Die Wahrscheinlichkeit für jeden Nachkommen von w, eine in w enthaltene Knotenüberdeckung zu sein, beträgt also mindestens $\frac{1}{n}(1-\frac{1}{n})^{n-1}$, da dies die Wahrscheinlichkeit ist, dass genau das Individuum $w \cap \overline{x}$ erzeugt wird. Aus $(1-\frac{1}{n})^{n-1} \geq \frac{1}{e}$ folgt $\frac{1}{n}(1-\frac{1}{n})^{n-1} \geq \frac{1}{en}$ und damit ist die erwartete Anzahl von Individuen, die erzeugt werden müssen, um eine Verbesserung zu erzielen, höchstens en. Zu jedem einmal selektierten Individuum werden maximal λt Nachkommen erzeugt, bevor der Algorithmus abgebrochen wird. Bei Berücksichtigung der obigen Gleichung werden also inbesondere aus w bis zu en^2 Nachkommen generiert. Da die erwartete Anzahl, die zu einer Verbesserung führt, höchstens

en beträgt, folgt mit der Markov-Ungleichung (siehe [33]) sofort, dass die Wahrscheinlichkeit, dass eine solche Verbesserung in Form einer in w enthaltenen Knotenüberdeckung nicht generiert wird, durch $\frac{1}{n}$ beschränkt wird. Damit ist also bei dieser Wahl der Parameter λ und t die Wahrscheinlich hoch, dass das Resultat des $(1+\lambda)$-EA eine inklusionsminimale Knotenüberdeckung ist. Die Experimente bestätigen die Überlegenheit des $(1+\lambda)$-EA im Vergleich mit den beiden anderen Algorithmen, wie Abbildung 3.12 verdeutlicht. Der relationale $(1+\lambda)$-EA erzielt in jedem der getesteten Fälle das beste Ergebnis der drei Algorithmen.

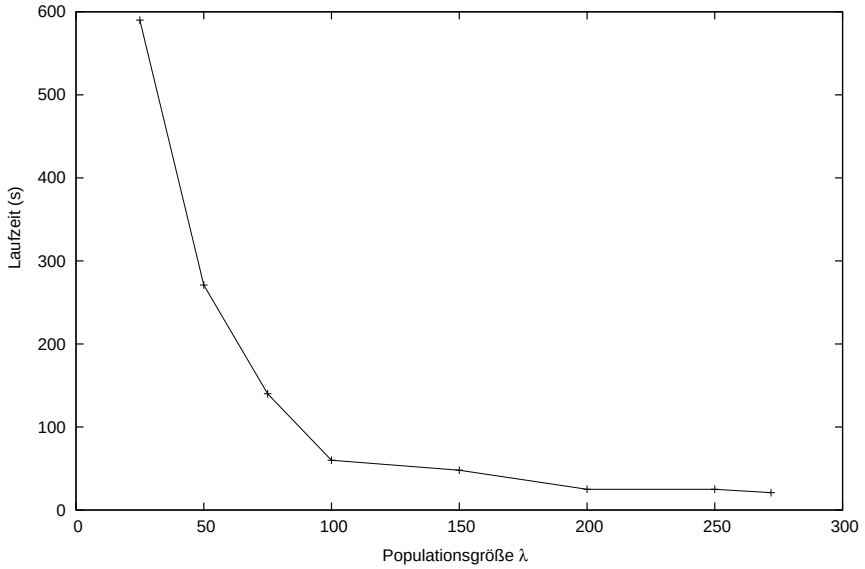


Abbildung 3.13: Laufzeit des $(1+\lambda)$-EA in Abhängigkeit von λ und t

In weiteren Experimenten (siehe Abbildung 3.13) wurde untersucht, wie sich die Wahl der Parameter λ und t auf die Laufzeit des Algorithmus auswirkt, wenn die Gleichung $\lambda t = en^2$ berücksichigt wird. Auf der x-Achse sind die verschiedenen Werte für λ aufgelistet, die y-Achse beschreibt die Laufzeit des $(1+\lambda)$-EA in Sekunden. Man sieht deutlich, dass man mit steigendem λ, und somit fallendem t gravierend kürzere Laufzeiten erzielt. Es erscheint also tatsächlich sinnvoll, bei dieser Art von Problemen mit größeren Populationen zu arbeiten.

Der $(\lambda+\lambda)$-EA kann beispielsweise durch das folgende RelView-Programm implementiert werden. Als Eingaben benötigt man hier R und t wie oben, außerdem eine aus Knotenüberdeckungen bestehende Startpopulation, modelliert als Relation $S : X \leftrightarrow [1..\lambda]$, sowie Relationen p_M und p_C, die die Wahrscheinlichkeiten bei Mutation und Crossover festlegen, wie in den Abschnitten 3.3.1 und 3.3.2 beschrieben.

```
EA2(R,S,pM, pC,t)
  DECL P, z, C, d
  BEG P = S;
      z = t;
      WHILE -empty(z) DO
        C = crossover(P,pC);
        C = mutation2(C, pM);
        C = C*randomperm(L1n(C)^);
        d = tournDecision(R,P,C);
        P = tournSelection(P,C,d);
        z = z & -point(z)
      OD
      RETURN P
  END.
```

Ausgehend von einer Startpopulation S, die aus λ Knotenüberdeckungen besteht, wird hier durch uniformes Crossover und anschließende Mutation in jedem Schleifendurchlauf aus der Elternpopulation P eine Nachfolgepopulation C generiert. Dann werden randomisiert Paare gebildet, die je aus einem Individuum der Eltern- und einem der Nachfolgepopulation bestehen, aus denen mit Turnierselektion jeweils das bessere für die nächste Generation bestimmt wird. Um die Qualität des Algorithmus zu beurteilen, wurde er auf 40 zufällig erzeugten Graphen mit 50 Knoten und verschiedenen Kantenzahlen getestet, wobei die Startpopulationen, bestehend aus 50 Individuen, durch eine randomisierte Variante des Algorithmus von Gavril und Yannakakis erzeugt wurde (siehe Anhang B). Als Parameter pM wurde, wie beim $(1+\lambda)$-EA, ein beliebiger Punkt des Typs $[X \leftrightarrow \mathbf{1}]$ gewählt, was zu $p_M = \frac{1}{50}$ führt, als pC wurde ein Vektor des Typs $[[1..5] \leftrightarrow \mathbf{1}]$ mit 4 Einträgen verwendet, so dass für jedes Paar mit der Wahrscheinlichkeit $p_C = 0.8$ der Crossover-Operator angewandt wurde. Nach $t = 100$ Schleifendurchläufen bricht der Algorithmus ab und liefert die aktuelle Population zurück. Zum Vergleich wurden mithilfe eines exakten Verfahrens unter Verwendung der Testabbildung $\varphi_{vc}^{2^X}$ die optimalen Ergebnisse berechnet. In Abbildung 3.14 wird zu jeder Instanz die Größe einer kleinsten Knotenüberdeckung der Startpopulation und der durch den $(\lambda+\lambda)$-EA erzeugten Population dargestellt, sowie die Knotenanzahl einer optimalen Lösung. Die ersten 10 Eingaben sind Graphen mit einer Kantenwahrscheinlichkeit von 0.05, die nächsten mit 0.1, u.s.w. Man sieht, dass bereits 100 Iterationen ausreichen um nahezu optimale Ergebnissse zu erzielen. In vielen Fällen wird sogar das Optimum erreicht.

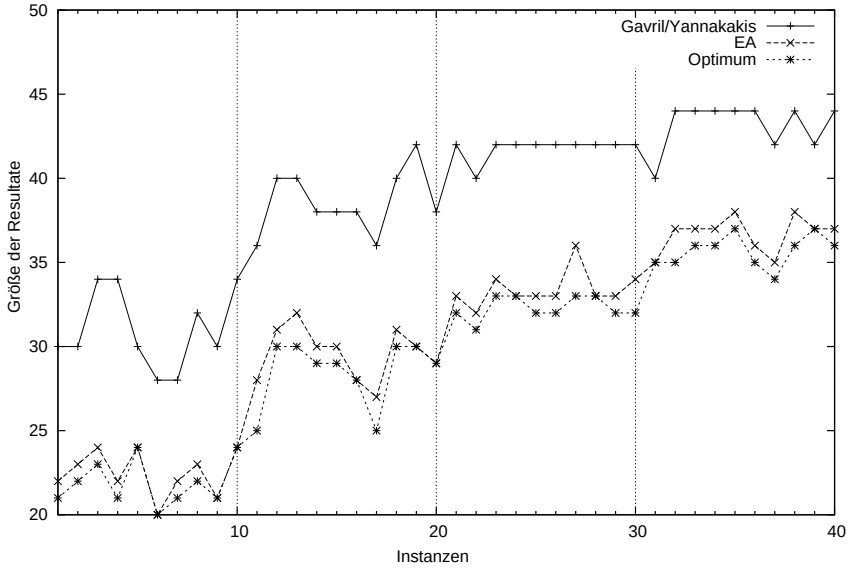


Abbildung 3.14: Qualität des $(\lambda + \lambda)$-EA

3.4 Multikriterielle Optimierung

In diesem Abschnitt wird der Frage nachgegangen, inwieweit sich relationale Methoden im Bereich der multikriteriellen Optimierung einsetzen lassen, wobei die Reduzierung der Optimierungskriterien einen besonderen Schwerpunkt darstellt. Teile der hier diskutierten Ergebnisse wurden bereits in [19] veröffentlicht.

3.4.1 Grundlagen

In den vorangehenden Kapiteln wurde ausschließlich einkriterielle Optimierung behandelt, also die Optimierung bezüglich einer Fitnessfunktion $f : X \to \mathbb{R}_+$. Durch die Fitnessfunktion wird mit

$$x \preceq x' \iff f(x) \leq f(x')$$

eine vollständige Quasiordnung $\preceq$ auf X induziert, und es wird (im Falle eines Minimierungsproblems) versucht, im Suchraum X ein kleinstes Element bezüglich $\preceq$ zu finden. Bei multikriteriellen Problemen steht eine Menge $\mathcal{F} = \{f_1, \ldots, f_k\}$ von Funktionen $f_i : X \to \mathbb{R}_+$ zur Verfügung, nach denen gleichzeitig optimiert wird. Man erhält also eine vektorwertige Fitnessfunktion $f : X \to \mathbb{R}_+^k$ durch $f = (f_1, \ldots f_k)$. Die durch f induzierte *schwache Dominanzrelation* $\preceq$ auf X ist definiert durch

$$x \preceq x' \iff \forall i \in [1..k] : f_i(x) \leq f_i(x').$$

Damit ist $\preceq$ ebenfalls eine Quasiordnung, die jedoch im allgemeinen nicht vollständig ist. Jedes der *Kriterien* f_i induziert eine vollständige Quasiordnung $\preceq_i$ und der Schnitt aller $\preceq_i$ ist genau die durch f induzierte schwache Dominanzrelation $\preceq$. Es gilt also

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i .$$

Die *starke Dominanzrelation* $\prec$ ist definiert durch

$$x \prec x' \iff f(x) \leq f(x') \ \wedge \ \exists i \in [1..k] : f_i(x) < f_i(x').$$

Es ergibt sich die starke direkt aus der schwachen Dominanzrelation durch die Gleichung

$$\prec = \preceq \cap \overline{\preceq}^{\top},$$

denn für zwei Suchpunkte x und x' gilt

$$\begin{aligned}
x' \prec x &\iff \forall i \in [1..k] : f_i(x') \leq f_i(x) \ \wedge \ \exists i \in [1..k] : f_i(x') < f_i(x) \\
&\iff x' \preceq x \ \wedge \ \neg \forall i \in [1..k] : f_i(x) \leq f_i(x') \\
&\iff x' \preceq x \ \wedge \ \neg (x \preceq x') \\
&\iff x' \preceq x \ \wedge \ x' \overline{\preceq}^{\top} x \\
&\iff x' (\preceq \cap \overline{\preceq}^{\top}) x
\end{aligned}$$

Ein Suchpunkt x heißt *Pareto-optimal*, wenn er von keinem Suchpunkt *dominiert* wird, das heißt, es gibt kein $x' \in X$ mit $x' \prec x$. Die Menge aller Pareto-optimalen Elemente wird *Pareto-Menge* genannt. Ziel der multikriteriellen Optimierung ist es, eine gute Approximierung der Pareto-Menge zu erreichen.

Für kleine Instanzen kann dieses Problem mit einem einfachen relationalen Programm gelöst werden, wenn man die Relationen $\preceq_1, \ldots, \preceq_k$ als gegeben annimmt. Aus praktischen Gründen modellieren wir zunächst die Menge $\mathcal{R} = \{\preceq_1, \ldots, \preceq_k\}$ als Relation $R : X \times X \leftrightarrow [1..k]$, deren Spalten genau den Vektordarstellungen (siehe Abschnitt 2.4) der Relationen $\preceq_i$ entsprechen. Es gilt also

$$R^{(i)} = vec(\preceq_i)$$

für alle $i \in [1..k]$, und damit $R_{\langle x,x' \rangle i}$ genau dann wenn $x \prec x'$ für alle Suchpunkte x und x'. Mithilfe der Abbildung $\psi_{cut} : [X \times X \leftrightarrow [1..k]] \to [X \times X \leftrightarrow \mathbf{1}]$ aus Beispiel

3.2.3.2, die den Schnitt aller Spalten einer Relation berechnet, erhält man die folgende Darstellung der schwachen Dominanzrelation als Vektor des Typs $[X \times X \leftrightarrow \mathbf{1}]$. Es gilt

$$w := vec(\preceq) = \psi_{cut}(\mathsf{L}) = \overline{\overline{R}\mathsf{L}},$$

man erhält also sofort die RelView-Funktion

```
weakDom(R) = -(-R * L1n(R)^).
```

und, darauf aufbauend, das folgende Programm zur Berechnung der starken Dominanzrelation.

```
strongDom(R,Q)
  DECL w, W, S
  BEG w = weakDom(R);
      W = rel(w,Q);
      S = W & -W^
      RETURN S
  END.
```

Dabei ist der zweite Parameter eine beliebige Relation des Typs $[X \leftrightarrow X]$, die dazu dient, den Typ der Ausgaberelation anzugeben. Mit der starken Dominanzrelation ist es ist es nun möglich, die Menge aller Pareto-optimalen Suchpunkte zu ermitteln. Es gilt:

$$\begin{aligned} x \text{ ist Pareto-optimal} &\iff \neg\exists x' : x' \prec x \\ &\iff \neg\exists x' : x \prec^{\top} x' \\ &\iff \neg(\prec^{\top} \mathsf{L})_x \\ &\iff \overline{(\prec^{\top} \mathsf{L})}_x. \end{aligned}$$

Die Pareto-Menge wird also durch den Vektor

$$o = \overline{\prec^{\top} \mathsf{L}}$$

des Typs $[X \leftrightarrow \mathbf{1}]$ repräsentiert und wir erhalten die RelView-Funktion

```
ParetoOpt(R,Q) = -(strongDom(R,Q)^ * Ln1(Q)).
```

Innerhalb der Pareto-optimalen Suchpunkte können Elemente mit gleichen Fitness-

Werten existieren, also Suchpunkte x, x' mit $f(x) = f(x')$. In vielen Fällen ist man jedoch daran interessiert, pro Fitness-Vektor genau einen Pareto-optimalen Suchpunkt zu erhalten. Dazu definieren wir zunächst eine Äquivalenzrelation $\approx$ vom Typ $[X \leftrightarrow X]$, die sich aus der schwachen Dominanzrelation durch

$$\approx := \preceq \cap \preceq^{\top}$$

ergibt. Die Relation beschreibt also die Fitness-Wert-Gleichheit zweier Suchpunkte, es gilt also $x \approx x'$ genau dann wenn $f(x) = f(x')$. Aus der Pareto-Optimalität eines Suchpunktes x folgt offensichtlich, dass die gesamte Äquivalenzklasse $[x]_{\approx}$ Pareto-optimal ist. Um einen Vektor $r \subseteq o$ von Repräsentanten aller Pareto-optimaler Äquivalenzklassen zu berechnen, verwenden wir eine beliebige vollständige Ordnung O und übernehmen aus jeder Pareto-optimalen Äquivalenzklasse das kleinste Element bezüglich O. Es gilt

$$\begin{aligned}
r_x &\iff o_x \ \wedge\ \forall x' : x \approx x' \rightarrow O_{xx'} \\
&\iff o_x \ \wedge\ \neg\exists x' : x \approx x' \ \wedge\ \overline{O}_{xx'} \\
&\iff o_x \ \wedge\ \neg\exists x' : (\approx \cap\, \overline{O})_{xx'} \\
&\iff o_x \ \wedge\ \overline{(\approx \cap\, \overline{O})\mathsf{L}}_x \\
&\iff (o \cap \overline{(\approx \cap\, \overline{O})\mathsf{L}})_x
\end{aligned}$$

Wir erhalten also den Vektor

$$r = o \cap \overline{(\approx \cap\, \overline{O})\mathsf{L}},$$

der genau einen Repräsentanten jeder Pareto-optimalen Äquivalenzklasse enthält, mit dem folgenden RelView-Program.

```
ParetoOptRep(R,O)
  DECL W, o, r
  BEG W = rel(weakdom(R),O);
      o = ParetoOpt(R,O);
      r = o & -((W & W^ & -O)*L(o))
      RETURN r
  END.
```

Dabei ist der Parameter `O` eine vollständige Ordnung auf X. Wie man mithilfe des RelView-Systems vollständige Ordnungen erzeugt, wird im folgenden Kapitel erläutert.

3.4.2 Reduzierung der Kriterien

Bei multikriteriellen Problemen geht es meist darum, eine gute Approximation der Pareto-Menge bezüglich einer Anzahl von Kriterien zu berechnen. Problematisch ist dabei oft, aus der ermittelten Menge geeignete Suchpunkte auszuwählen, da man sich dabei mit einer großen Anzahl von Kriterien auseinandersetzen muss. Dabei stellt sich also die Frage, ob innerhalb der berechneten Menge $Y \subseteq X$ von Suchpunkten tatsächlich alle Kriterien aus $\mathcal{F}$ eine Rolle spielen, oder ob echte Teilmengen von $\mathcal{F}$ existieren, die innerhalb von Y die gleiche schwache Dominanzrelation induzieren, d.h. also Teilmengen $T \in [1..k]$ mit

$$x \preceq x' \iff x(\bigcap_{i \in T} \preceq_i)x'$$

für alle $x, x' \in Y$. Allgemein kann man das Problem folgendermaßen formulieren. Für eine beliebige Teilmenge Y von X und $\preceq_1, \ldots, \preceq_k$ sowie $\preceq$ die durch die Menge $\mathcal{F}$ auf Y induzierten Quasiordnungen, also Relationen des Typs $[Y \leftrightarrow Y]$ mit

$$\forall i \in [1..k], x, x' \in Y : x \preceq_i x' \Longleftrightarrow f_i(x) \leq f_i(x')$$

und

$$\preceq = \bigcap_{i \in [1..k]} \preceq_i,$$

berechne eine Teilmenge $T \subseteq [1..k]$ minimaler Größe mit

$$\preceq = \bigcap_{i \in T} \preceq_i .$$

In [15] wird dieses Problem diskutiert und sowohl ein Greedy-Algorithmus als auch ein exaktes Verfahren angegeben. Im Folgenden wird ein relationaler Algorithmus entwickelt, der das Problem ebenfalls exakt löst und der dabei wesentlich effizienter ist als der in [15] angegebene. Hierzu kann das in Beispiel 3.2.3.2 entwickelte Vektorprädikat φ_{cut} verwendet werden. Sei also

$$\varphi_{cut} : [[1..k] \leftrightarrow \mathbf{1}] \rightarrow [\mathbf{1} \leftrightarrow \mathbf{1}]$$

definiert durch

$$\varphi_{cut}(t) = \overline{\mathsf{L}(\overline{\overline{Rt} \cup w})},$$

wobei $R : Y \times Y \leftrightarrow [1..k]$ die durch die Quasiordnungen $\preceq_i$ auf Y induzierte Relation ist, und $w = vec(\preceq)$ die Vektordarstellung der schwachen Dominanzrelation auf Y. Nach Beispiel 3.2.3.2 gilt dann für die durch den Vektor t repräsentierte Teilmenge $T \subseteq [1..k]$ die Äquivalenz

$$\preceq = \bigcap_{i \in T} \preceq_i \iff \varphi_{cut}(t) = \mathsf{L}.$$

Die Abbildung φ_{cut} testet also Vektoren auf ihre Eignung zur Reduzierung der Optimierungs-Kriterien. Anders ausgedrückt gilt

$$\preceq = \bigcap\{\preceq_i \mid t_i\} \iff \varphi_{cut}(t) = \mathsf{L}.$$

Da es sich bei φ_{cut} um ein Vektorprädikat im Sinne von Abschnitt 3.2.2 handelt, erhält man sofort für jede Menge Z eine Testabbildung $\varphi^Z_{cut} : [[1..k] \leftrightarrow Z] \to [\mathbf{1} \leftrightarrow Z]$ durch

$$\varphi^Z_{cut}(M) = \overline{\mathsf{L}(\overline{\overline{R}M \cup w\mathsf{L}})},$$

wobei L vom Typ $[\mathbf{1} \leftrightarrow Z]$ ist. Für jede Relation $M : [1..k] \leftrightarrow Z$ repräsentiert der Zeilenvektor $\varphi^Z_{cut}(M)$ die Spalten von M, welche die Anzahl der Kriterien reduzieren, d.h. es gilt

$$\varphi^Z_{cut}(M)_{\perp j} \iff \varphi_{cut}(M^{(j)}) = \mathsf{L} \iff \preceq = \bigcap\{\preceq_i \mid M^{(j)}_i\}.$$

Unter Benutzung der Potenzmengenrelation $\mathsf{M} : [1..k] \leftrightarrow 2^{[1..k]}$ berechnet man mit $c := \varphi^{2^{[1..k]}}_{cut}(\mathsf{M})$ alle geeigneten Vektoren zur Reduzierung der Kriterien. Man erhält sofort das folgende RelView-Programm.

```
cut(R)
  DECL w, M, c
  BEG w  = weakDom(R);
      M  = epsi(L1n(R^));
      c  = -(Ln1(R)^ * -(-R * M | w * L1n(M)))
      RETURN c
  END.
```

Um die kleinsten Vektoren mit der erwünschten Eigenschaft zu finden, wird die Größenvergleichsrelation $\mathsf{C} : 2^{[1..k]} \leftrightarrow 2^{[1..k]}$ mit C_{XY} genau dann wenn $|X| \leq |Y|$ verwendet. Wir definieren eine Abbildung *se*, die für eine gegebene Quasiordnung Q und einen Vektor v die bzgl. Q kleinsten Elemente in v berechnet. Mit $se(Q, v) = v \cap \overline{\overline{Q}v}$ erhalten

wir einen Vektor für den

$$se(Q,v)_x \iff v_x \wedge \forall y : v_y \to Q_{xy}$$

gilt. Der Vektor $s = se(\mathsf{C}, c^\top)$ repräsentiert also die anzahlminimalen Teilmengen $T \subseteq [1..k]$, die die Eigenschaft $\preceq = \bigcap_{i\in T} \preceq_i$ erfüllen, d.h. es gilt

$$s_{\perp j} \iff \preceq = \bigcap\{\preceq_i \mid \mathsf{M}_i^{(j)}\} \quad \wedge \quad \forall t : |t| < |\mathsf{M}^{(j)}| \to \preceq \neq \bigcap\{\preceq_i \mid t_i\}.$$

Unter Benutzung des Vektorprädikats φ_{cut} kann man die Äquivalenz wie folgt ausdrücken.

$$s_{\perp j} \iff \varphi_{cut}(\mathsf{M}^{(j)}) = \mathsf{L} \quad \wedge \quad \forall t : |t| < |\mathsf{M}^{(j)}| \to \varphi_{cut}(t) = \mathsf{O}$$

Das folgende RelView-Programm berechnet den Vektor s, wobei die Größenvergleichsrelation auf $2^{[1..k]}$ durch `cardrel(L1n(R)^)` generiert wird.

```
smallCuts(R)
  DECL c, C, s
  BEG c = cut(R);
      C = cardrel(L1n(R)^);
      s = se(C,c^)
      RETURN s
  END.
```

3.4.3 Experimente

Im Folgenden wird zunächst der im vorangegangenen Kapitel entwickelte relationale Algorithmus mit dem exakten Verfahren aus [15] verglichen. Die verwendeten Instanzen bestehen aus Approximationen der Pareto-Menge multikriterieller Knapsack-Probleme, die mit dem Evolutionären Algorithmus SPEA2 (siehe [49]) berechnet wurden. Es wurden Instanzen mit 50 Suchpunkten und 5 bis 35 Kriterien getestet, d.h., jede Eingabe bestand aus 5 bis 35 Relationen der Größe 50×50, was für das RelView-Programm (im Fall von 35 Kriterien) zu einer Eingaberelationen R der Größe 2500×35 führte. Der exakte Algorithmus aus [15] konnte mit Rechenzeiten bis zu 53 Minuten für alle Instanzen bis zu 30 Kriterien ein Ergebnis zu errechen. Das RelView-Progamm erzielte für alle Instanzen eine optimale Lösung innnerhalb weniger Sekunden und erreichte wesentlich kürzere Rechenzeiten, wie Tabelle 3.1 zeigt.
Zusätzlich wurde das Laufzeitverhalten des relationalen Algorithmus auf zufällig generierten Instanzen untersucht. Dazu wurden mithilfe des RelView-Systems zufällige Quasiordnungen generiert, wie im Folgenden erläutert wird. Basierend auf einer belie-

Tabelle 3.1: Vergleich des RelView-Progamms mit dem exakten Verfahren aus [15] (Zeit in Millisekunden)

Kriterien	Laufzeit RelView	Laufzeit exaktes Verfahren [15]
5	40	178
10	70	4369
15	590	166343
20	170	197690
25	430	5135040
30	1360	3203227
35	5990	-

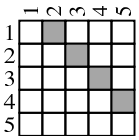
$S =$ 

Abbildung 3.15: Die 5×5-Nachfolger-Relation

bigen Hasse-Relation S und einer zufällig erzeugten Permutation P des Typs $[X \leftrightarrow X]$ berechnet man $O = (PSP^\top)^*$, die reflexiv-transitive Hülle von $PSP^\top$, und erhält damit eine zufällige vollständige Ordnung. Das folgende RelView-Programm generiert eine Ordnung auf diese Weise, wobei T eine beliebige Relation des Typs $[X \leftrightarrow X]$ ist, die lediglich den Typ der erzeugten Relationen bestimmt.

```
randomOrder(T)
  DECL S,P,O
  BEG S = succ(Ln1(T));
      P = randomperm(Ln1(T));
      O = refl(trans(P*S*P^))
      RETURN O
  END.
```

Dabei generiert `succ(Ln1(T))` die Nachfolger-Relation S auf X (siehe Abbildung 3.4.3), die eine Hasse-Relation auf X darstellt. Durch Anwendung von `refl` und `trans` wird die reflexiv-transitive Hülle berechnet. Eine vollständige Ordnung kann durch Hinzufügen von Einträgen zu einer vollständigen Quasiordnung modifiziert werden. Dazu wird eine zufällige Relation A erzeugt und $A \cup A^\top$ mit $PSP^\top$ vereinigt, bevor die reflexiv-transitive Hülle berechnet wird. Die Quasiordnung ist also durch $(PSP^\top \cup A \cup A^\top)^*$ gegeben. Im folgenden RelView-Programm ist die Eingabe T eine nichtleere Relation,

die den Typ der erzeugten Relationen vorgibt und die Anzahl der Einträge der erzeugten Quasiordnung beeinflusst. Mit `random(T,T)` wird eine zufällige Relation $A : X \leftrightarrow X$ mit der Eigenschaft erzeugt, dass für alle $i, j \in X$ die Wahrscheinlichkeit, dass A_{ij} gilt, $|T|/|X|^2$ beträgt.

```
randomQuasiOrder(T)
  DECL S, P, A, Q
  BEG S = succ(Ln1(T));
      P = randomperm(Ln1(T));
      A = random(T,T);
      Q = refl(trans(P*S*P^ | A | A^))
      RETURN Q
  END.
```

Mit diesem Programm können jetzt Eingaben, bestehend aus Relationen des Typs $[X \times X \leftrightarrow [1..k]]$, erzeugt werden, indem nacheinander k Quasiordnungen Q des Typs $[X \leftrightarrow X]$ und ihre Vektordarstellung q generiert werden.

```
randomInput(T,k)
  DECL R, z, Q, q, p
  BEG R = 0(vec(T)*k^);
      z = k
      WHILE -empty(z) DO
        Q = randomQuasiOrder(T);
        q = vec(Q);
        p = point(z);
        R = R | q*p^;
        z = z & -p
      OD
      RETURN R
  END.
```

Ist p ein Punkt, der ein $i \in [1..k]$ repräsentiert, so wird durch `R = R |q*p^` der Vektor q als i-te Spalte in die Relation R eingefügt, wobei R vom Typ $[X \times X \leftrightarrow [1..k]]$ ist. Mithilfe dieses RelView-Programms wurden verschiedene Eingaben erzeugt, die Mengen von 5 bis 145 Quasiordnungen auf 50 Elementen entsprechen. Die für diese Instanzen benötigten Laufzeiten des relationalen Algorithmus sind in Tabelle 3.2 dargestellt. Dabei bezeichnet p die Wahrscheinlichkeit für jeden Eintrag der Relation A, d.h. p bestimmt den Füllgrad der einzelnen Quasiordnungen.

Tabelle 3.2: Resultate für zufällige Eingaben (Zeit in Sekunden)

p	1/2500		1/500		1/250		3/500		1/125		1/50	
Krit.	Zeit	Kr.	Zeit	Kr.	Zeit	Kr.	Zeit	Kr.	Zeit	Kr.	Zeit	Kr.
5	0.04	5	0.01	5	0.01	5	0.01	4	0.01	4	0.16	2
15	0.63	8	0.51	15	0.48	14	0.46	12	0.47	9	0.38	3
25	5.71	7	0.25	11	0.07	14	0.07	14	0.06	14	0.05	8
35			1.22	13	0.20	22	0.16	23	0.16	14	0.15	5
45					0.49	26	0.41	28	0.41	25	0.38	19
55					39.06	17	0.87	30	0.80	30	0.79	18
65					5.29	23	2.44	29	3.19	32	1.54	17
75							2.91	34	2.72	38	3.52	23
85							7.86	27	5.61	33	4.43	20
95									9.24	40	7.87	27
105									11.60	40	23.86	21
115									16.80	42	24.43	26
125									27.37	44	23.19	29
135											32.52	32
145											77.77	30

Es zeigt sich eine starke Abhängigkeit der Laufzeiten vom Füllgrad der Eingaben. Bei den Experimenten wurde das Programm jeweils abgebrochen, wenn nach 10 Minuten kein Ergebnis geliefert wurde. Die Tabelle zeigt, dass die in dieser Zeitspanne zu bewältigende Problemgröße mit steigendem Füllgrad der Eingaberelationen stark zunimmt. Der Algorithmus ist in der Lage, für 145 Kriterien, die alle stark gefüllte Quasirelationen induzieren, eine optimale Reduzierung der Kriterien in weniger als 2 Minuten zu berechnen.

Neben den Laufzeiten werden auch die Mächtigkeiten der optimalen Lösungen angegeben, also die minimale Anzahl von Kriterien, die ausreichen, um die schwache Dominanzrelation zu erzeugen. Anhand der Tabelle wird deutlich, dass in den meisten Fällen die Anzahl der Kriterien erheblich reduziert werden kann.

4 Relationen und Vektoren

In Abschnitt 3.2.2 wurde erläutert, wie Mengen von Suchpunkten als Relationen dargestellt und mithilfe spezieller Testabbildungen ausgewertet werden können. Dieser Ansatz soll im folgenden Kapitel auf eine größere Klasse von Problemen erweitert werden. Bisher wurden ausschließlich Probleme behandelt, bei denen die Suchpunkte in natürlicher Weise durch Vektoren modelliert werden können, da es sich um Teilmengen, beispielsweise von Knoten eines Graphs, handelt. Dieses Kapitel zielt nun darauf ab, das Verfahren auch auf Problemstellungen anzuwenden, bei denen der Suchraum aus Relationen besteht, wie es bei zahlreichen Problemen der Kombinatorischen Optimierung der Fall ist. Typische Beispiele dafür sind alle Arten von Graphfärbungsproblemen, bei denen unter Einhaltung gewisser Einschränkungen jeder Knoten (oder jede Kante) eines Graphs einer Farbe zugeordet werden soll, sowie Stundenplanprobleme, bei denen beispielweise Veranstaltungen auf Zeitschienen oder Räume verteilt werden. Bei dieser Art von Problemen geht es also darum, Relationen mit speziellen Eigenschaften zu finden. Um hierbei den Ansatz aus Kapitel 3.2.2 verwenden zu können, ist es sinnvoll, Relationen als Vektoren darzustellen, wie in Abschnitt 2.4 erläutert. Mittels der Vektorrepräsentation lässt sich eine Menge von Relationen eines Typs $[X \leftrightarrow Y]$ als eine Relation mit Urbildbereich $X \times Y$ modellieren, wie bereits in Abschnitt 3.4 erörtert wurde. So dargestellte Populationen können dann unter Verwendung der in Kapitel 3.2.2 definierten Vektorprädikate ausgewertet werden, d.h., es werden genau die Spalten aus der Population herausgefiltert, die bestimmte Eigenschaften aufweisen. Um eine solche Auswertung zu ermöglichen, ist es notwendig, Eigenschaften einer Relation als Eigenschaften des zugehörigen Vektors auszudrücken. Ziel ist es, nur anhand der Vektordarstellung einer Relation zu entscheiden, ob die zugrundeliegende Relation eine spezielle Eigenschaft besitzt. Im folgenden wird die Transformation von Relations- in Vektoreigenschaften diskutiert. Zunächst wird mit Theorem 4.1.10 eine Formel bewiesen, die sich in diesem Zusammenhang als sehr nützlich erweist. Dabei werden alle Beweise in der abstrakten Relationenalgebra geführt, da sie in dieser Form besser lesbar und nachvollziehbar sind. Anschließend werden einige Beispiele für die Verwendung der Formel angegeben sowie verschiedene Experimente dargestellt, die verdeutlichen, in welchen Fällen der in diesem Kapitel dargestellte Ansatz sinnvoll eingesetzt wer-

den kann. Eine gekürzte Version der hier vorgestellten Resultate wurde bereits in [27] veröffentlicht.

4.1 Transformation relationaler Eigenschaften

Eigenschaften von Relationen werden häufig durch relationale Gleichungen oder Inklusionen dargestellt. Eine Relation S des Typs $[X \leftrightarrow Y]$ ist beispielweise genau dann eindeutig, wenn die Inklusion $S\overline{\mathsf{I}} \subseteq \overline{S}$ gilt. Modelliert man nun mithilfe der in Kapitel 2.4 vorgestellten Abbildung *vec* Relationen als Vektoren, stellt sich die Frage, wie man anhand eines Vektors s des Typs $[X \times Y \leftrightarrow \mathbf{1}]$ die Eindeutigkeit der zugrundeliegenden Relation feststellen kann, ohne die Relationendarstellung $rel(s)$ zu berechnen. In diesem Kapitel wird eine Formel bewiesen, die es ermöglicht, eine große Klasse von Eigenschaften einer Relation in Eigenschaften des zugehörigen Vektors zu transformieren. Man kann also relationale Ausdrücke und Inklusionen entwickeln, mit denen direkt am Vektor getestet wird, ob die zugrundeliegende Relation eine gewisse Eigenschaft erfüllt, ohne dass die Relationendarstellung des Vektors benötigt wird. Hierbei spielt das Konzept der *parallelen Komposition* von Relationen eine wichtige Rolle.

4.1.1 Definition *Seien A, B Relationen, und (π, ρ) und (τ, σ) direkte Produkte, so dass $\pi A \tau^\top \cap \rho B \sigma^\top$ definiert ist. Im folgenden sei*

$$(A \| B) := \pi A \tau^\top \cap \rho B \sigma^\top$$

die parallele Komposition von A und B.

In der konkreten Relationenalgebra hat die Konstruktion der parallelen Komposition die folgende Bedeutung. Seien Relationen $A : X \leftrightarrow Y$ und $B : Z \leftrightarrow W$ gegeben. Unter Verwendung der natürlichen Projektionen von $X \times Z$ bzw. $Y \times W$ als relationale direkte Produkte erhält man durch Anwendung der parallelen Komposition die Relation $A \| B$ des Typs $[X \times Z \leftrightarrow Y \times W]$ mit der Eigenschaft

$$(A \| B)_{\langle x,z \rangle \langle y,w \rangle} \iff A_{xy} \ \wedge \ B_{zw}.$$

In engem Zusammenhang mit der parallelen Komposition steht das Konzept des *Tuplings* von Relationen, das im Folgenden erklärt wird.

4.1.2 Definition *Seien A und B Relationen, so dass $AB^\top$ definiert ist. Sei weiter ein direktes Produkt (π, ρ) gegeben für das $\pi A \cap \rho B$ definiert ist. Im folgenden sei*

$$[A, B] := \pi A \cap \rho B$$

das Tupling von A und B.

Offensichtlich gilt $A\|B = [A\pi_2^\top, B\rho_2^\top]$. In Kapitel 2.4 wurde bereits die Vektordarstellung einer Relation als $vec(A) = (\pi A \cap \rho)\mathsf{L}$ definiert, die sich mithilfe des Tuplings auch als

$$vec(A) = [A, \mathsf{I}]\mathsf{L}$$

darstellen lässt. Betrachtet man konkrete Relationen, so kann Tupling ausschließlich auf Relationen mit gleichem Nachbereich angewandt werden. Für $A : X \leftrightarrow Y$ und $B : Z \leftrightarrow Y$ erhält man dann die Relation $[A, B] : X \times Z \leftrightarrow Y$ mit der Eigenschaft

$$[A, B]_{\langle x,z\rangle y} \Longleftrightarrow A_{xy} \ \wedge \ B_{zy},$$

wobei als relationales direktes Produkt die natürlichen Projektionen von $X \times Z$ angenommen werden. Es sollte erwähnt werden, dass die hier verwendete Definition des Tuplings von der im Allgemeinen verwendeten Konstruktion abweicht. Meist wird das Tupling zweier Relationen A und B als $A\pi^\top \cap B\rho^\top$ definiert, wobei die passenden Typisierungen vorausgesetzt seien (siehe z.B. [48] und [14]). Wenn also im Folgenden von bereits in [14] bewiesenen Resultaten die Rede ist, die das Tupling betreffen, sind damit immer Aussagen über die eben erwähnte Konstruktion gemeint, die sich jedoch durch Transposition ($(\pi A \cap \rho B)^\top = A^\top\pi^\top \cap B^\top\rho^\top$) einfach auf das hier verwendete Tupling übertragen lassen. Im folgenden werden einige Aussagen über das Tupling und die parallele Komposition bewiesen. Einige der Resultate sind ebenfalls in [14] zu finden, aber in weniger allgemeiner Form. Zunächst werden einige Gleichungen und Inklusionen bezüglich der parallelen Komposition gezeigt.

4.1.3 Lemma *Seien Q und R Relationen, (π, ρ) und (τ, σ) direkte Produkte, so dass $\pi Q\tau^\top \cap \rho R\sigma^\top$ definiert ist. Dann gelten die folgenden Aussagen.*

1. $(Q\|R)^\top = Q^\top\|R^\top$
2. $(Q\|R)\tau \subseteq \pi Q$
3. *Ist R total, gilt sogar* $(Q\|R)\tau = \pi Q$.
4. $(Q\|R)\sigma \subseteq \rho R$
5. *Ist Q total, gilt sogar* $(Q\|R)\sigma = \rho R$.

Beweis: *1. Die Gleichung folgt direkt aus der Definition der parallelen Komposition:*

$$(Q\|R)^\top = (\pi Q\tau^\top \cap \rho R\sigma^\top)^\top = \tau Q^\top\pi^\top \cap \sigma R^\top\rho^\top = (R^\top\|Q^\top)$$

2. *Zum Beweis dieser Inklusion wird eine Eigenschaft des direkten Produkts verwendet. Mit* $\tau^\top\tau = \mathsf{I}$ *erhalten wir*

$$(Q\|R)\tau = (\pi Q\tau^\top \cap \rho R\sigma^\top)\tau \subseteq \pi Q\tau^\top\tau = \pi Q.$$

3. *Sei nun* R *total. Es bleibt zu zeigen, dass* $\pi Q \subseteq (Q\|R)\tau$ *gilt. Da* ρ *ebenfalls total ist und laut Definition des direkten Produkts* $\sigma^\top\tau = \mathsf{L}$ *gilt, folgt zunächst* $\rho R\sigma^\top\tau = \rho R\mathsf{L} = \rho\mathsf{L} = \mathsf{L}$. *Ausserdem gilt nach [42] für eindeutige Relationen* B *und alle Relationen* A *und* C *die Gleichung*

$$(AB^\top \cap C)B = A \cap CB.$$

Da τ *eindeutig ist, folgt also*

$$(Q\|R)\tau = (\pi Q\tau^\top \cap \rho R\sigma^\top)\tau = \pi Q \cap R\sigma^\top\tau = \pi Q.$$

Die verbleibenden Aussagen 4 *und* 5 *beweist man analog zu* 2 *und* 3.

Aus der Totalität der Identitäten ergibt sich als Spezialfall von Lemma 4.1.3 sofort das folgende Korollar.

4.1.4 Korollar *Für alle* Q *und* R *gilt*

1. $(Q\|\mathsf{I})\tau = \pi Q$
2. $(\mathsf{I}\|R)\sigma = \rho R$

wobei (π, ρ) *und* (τ, σ) *wie in Lemma 4.1.3 vorausgesetzt seien.*

Um eine Multiplikationsformel für das Tupling und die parallele Komposition beweisen zu können, benötigen wir ein weiteres Lemma, das eine direkte Folgerung der Dedekind-Regel ist.

4.1.5 Lemma *Seien* A, B, C *und* D *Relationen, so dass* $AB \cap C$ *definiert ist. Dann gilt*

$$A^\top C \subseteq D \;\Rightarrow\; AB \cap C \subseteq A(B \cap D)$$

Beweis: *Sei* $A^\top C \subseteq D$. *Es folgt sofort aus der Dedekind-Regel*

$$AB \cap C \subseteq (A \cap CB^\top)(B \cap A^\top C) \subseteq A(B \cap D).$$

Mit von Korollar 4.1.4 und Lemma 4.1.5 können nun Multiplikationsformeln für Tupling und parallele Komposition gezeigt werden, für den Spezialfall, dass eine der Relationen in der parallelen Komposition die Identität ist. Mithilfe dieser Spezialfälle kann später eine allgemeine Formel bewiesen werden.

4.1.6 Lemma *Seien Q, R, (π, ρ) and (τ, σ) wie in Lemma 4.1.3 und S, T weitere Relationen, so dass $\tau S \cap \sigma T$ definiert ist.*

1. *Wenn $\rho\sigma^\top$ existiert, gilt $(Q\|\mathsf{I})[S,T] = [QS, T]$.*
2. *Wenn $\pi\tau^\top$ existiert, gilt $(\mathsf{I}\|R)[S,T] = [S, RT]$.*

Beweis: *Wir beweisen nur die erste Gleichung, da der Beweis für 2 analog geführt werden kann.*

"$\subseteq$" Der Beweis der ersten Inklusion basiert hauptsächlich auf Lemma 4.1.3.

$$\begin{aligned}
(Q\|\mathsf{I})[S,T] &= (Q\|\mathsf{I})(\tau S \cap \sigma T) \\
&\subseteq (Q\|\mathsf{I})\tau S \cap (Q\|\mathsf{I})\sigma T && (\textit{Kapitel 2.1}) \\
&\subseteq \pi QS \cap \rho\mathsf{I}T && (\textit{Lemma 4.1.3}) \\
&= [QS, T].
\end{aligned}$$

"$\supseteq$" Aus Lemma 4.1.3 folgt sofort, dass die Inklusion $(Q\|\mathsf{I})^\top \rho T \subseteq \sigma T$ gilt. Damit erhalten wir

$$\begin{aligned}
[QS,T] &= \pi QS \cap \rho T \\
&= (Q\|\mathsf{I})\tau S \cap \rho T && (\textit{Korollar 4.1.4}) \\
&\subseteq (Q\|\mathsf{I})(\tau S \cap \sigma T) && (\textit{Inklusion und Lemma 4.1.5}) \\
&= (Q\|\mathsf{I})[S,T].
\end{aligned}$$

Eine direkte Folgerung aus Lemma 4.1.6 ist eine Multiplikationsformel für einen Spezialfall der parallelen Komposition.

4.1.7 Korollar *Für alle Relationen Q und R gilt die folgende Gleichung*

$$(Q\|\mathsf{I})(\mathsf{I}\|R) = (\mathsf{I}\|R)(Q\|\mathsf{I}) = (Q\|R)$$

Beweis: *Sei das direkte Produkt (τ, σ) wie in Lemma 4.1.3 vorausgesetzt. Wegen $(\mathsf{I}\|R) = [\tau^\top, R\sigma^\top]$ kann Lemma 4.1.6 angewandt werden, und es gilt:*

$$(Q\|\mathsf{I})(\mathsf{I}\|R) = (Q\|\mathsf{I})[\tau^\top, R\sigma^\top] = [Q\tau^\top, R\sigma^\top] = (Q\|R)$$

Die zweite Gleichung wird analog gezeigt.

Im folgenden wird mithilfe von Lemma 4.1.6 und Korollar 4.1.7 eine Multiplikationsformel für parallele Komposition und Tupling bewiesen. Das Theorem ist eine Generalisierung einer Aussage in [14], wo diese Formel für injektive Relationen S and R bewiesen wird.

4.1.8 Theorem *Seien Q, R, S und T Relationen, so dass $[S,T]$, QS und RT definiert sind. Dann gilt*

$$(Q\|R)[S,T] = [QS, RT].$$

Beweis: *Der Beweis ist eine direkte Folgerung von Lemma 4.1.6 und Korollar 4.1.7:*

$$(Q\|R)[S,T] = (Q\|\mathsf{I})(\mathsf{I}\|R)[S,T] = (Q\|\mathsf{I})[S,RT] = [QS,RT]$$

Das Theorem wird im Folgenden dazu verwendet, ein elementares Problem bezüglich der Vektorrepräsentation von Relationen zu lösen, indem eine Formel für die Vektordarstellung von Produkten von Relationen bewiesen wird. Wir nehmen an, die Vektordarstellung s einer Relation S sei bekannt, nicht aber die Relation S selber. Dann soll die Vektordarstellung der Komposition von S mit einer weiteren Relation berechnet werden, also beispielsweise $vec(QS)$ oder $vec(SR)$, ohne dabei aus dem Vektor s die Relation $S = rel(s)$ zu berechnen. Man benötigt zunächst eine weitere Eigenschaft des Tuplings, die in [42] gezeigt wird.

4.1.9 Bemerkung *Für alle Relationen A, B, für die das Produkt AB existiert, gilt*

$$[AB, \mathsf{I}]\mathsf{L} = [A, B^\top]\mathsf{L}.$$

Mithilfe dieser Bemerkung und Theorem 4.1.8 kann nun das Hauptresultat dieses Kapitels bewiesen werden.

4.1.10 Theorem *Seien Q, S, R Relationen, für die das Produkt QSR existiert. Dann gilt*

$$vec(QSR) = (Q\|R^\top)vec(S).$$

Beweis: *Die Gleichung wird unter Verwendung von Theorem 4.1.8 gezeigt*

$$\begin{aligned}
vec(QSR) &= [QSR, \mathsf{I}]\mathsf{L} && (\textit{Formel nach Def. 4.1.2}) \\
&= [QS, R^\top]\mathsf{L} && (\textit{Bemerkung 4.1.9}) \\
&= (Q\|R^\top)[S, \mathsf{I}]\mathsf{L} && (\textit{Theorem 4.1.8}) \\
&= (Q\|R^\top)vec(S). && (\textit{Formel nach Def. 4.1.2})
\end{aligned}$$

4.1.11 Korollar *Als unmittelbare Folge von Theorem 4.1.10 erhalten wir die folgenden Spezialfälle.*

1. $vec(QS) = (Q\|\mathsf{I})vec(S)$
2. $vec(SR) = (\mathsf{I}\|R^\top)vec(S)$

Abbildung 4.1 visualisiert die Aussage von Theorem 4.1.10 für den Fall konkreter Relationen. Dabei seien Q und R Relationen vom Typ $[Z \leftrightarrow X]$ bzw. $[Y \leftrightarrow W]$ und es

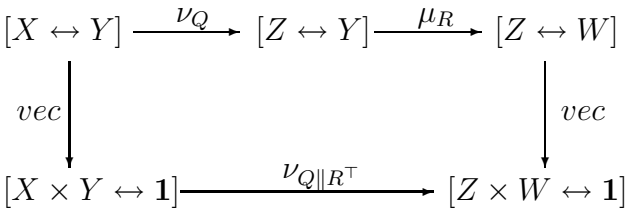


Abbildung 4.1: Kommutatives Diagramm

bezeichne ν_Q die Linksmultiplikation mit Q, also die Abbildung $S \mapsto QS$, sowie μ_R die Rechtsmultiplikation mit R, also die Abbildung $S \mapsto SR$. Die Aussage von Theorem 4.1.10 kann durch die Gleichung

$$vec \circ \mu_R \circ \nu_Q = \nu_{Q\|R^\top} \circ vec.$$

ausgedrückt werden. Beispiele für die vielfältige Verwendbarkeit des hier bewiesenen Theorems werden im folgenden Abschnitt sowie in den Kapiteln 5 und 6 angegeben.

4.2 Beispiele

Im Folgenden werden einige Anwendungen von Theorem 4.1.10 behandelt. Die Formel ermöglicht es, verschiedene Eigenschaften von Relationen als Eigenschaften der zugehörigen Vektoren auszudrücken, was zunächst an den Beispielen Eindeutigkeit, Totalität, Injektivität und Surjektivität verdeutlicht wird. Mit Hilfe von Theorem 4.1.10 werden Abbildungen entwickelt, die auf Vektoren angewandt werden und mit denen entschieden werden kann, ob die einem Vektor zugrundeliegende Relation eine dieser Eigenschaften besitzt. Da es sich bei den Abbildungen um Vektorprädikate im Sinne von Kapitel 3.2.2 handelt, können sie zu Testabbildungen erweitert werden, die aus Relationen die Spalten herausfiltern, welche die entsprechenden Eigenschaften aufweisen. Solchermaßen entwickelte Testabbildungen können in exakten Verfahren, aber

beispielsweise auch in Evolutionären Algorithmen eingesetzt werden. Sei im Folgenden S eine Relation und $s = vec(S)$ ihre Vektor-Darstellung. Des Weiteren sei (π, ρ) ein direktes Produkt mit der Eigenschaft, dass $\pi S \rho^\top$ definiert ist. Es gelten die folgenden Äquivalenzen.

1. S ist eindeutig $\iff (\mathsf{I}|\overline{\mathsf{I}})s \subseteq \overline{s}$
2. S ist injektiv$\iff (\overline{\mathsf{I}}\|\mathsf{I})s \subseteq \overline{s}$
3. S ist total $\iff \pi^\top s = \mathsf{L}$
4. S ist surjektiv $\iff \rho^\top s = \mathsf{L}$

Den Beweis beginnen wir mit den folgenden relationalen Charakterisierungen aus Abschnitt 2.2.

1. S ist eindeutig $\iff S\overline{\mathsf{I}} \subseteq \overline{S}$
2. S ist injektiv $\iff \overline{\mathsf{I}}S \subseteq \overline{S}$
3. S ist total $\iff S\mathsf{L} = \mathsf{L}$
4. S ist surjektiv$\iff \mathsf{L}S = \mathsf{L}$

Wir verwenden die in Abschnitt 2.4 aufgeführten Eigenschaften der Abbildung *vec* und Theorem 4.1.10.

1. Es gilt die folgende Äquivalenz:

$$
\begin{aligned}
S \text{ ist eindeutig} &\iff S\overline{\mathsf{I}} \subseteq \overline{S} \\
&\iff vec(S\overline{\mathsf{I}}) \subseteq vec(\overline{S}) && (vec \text{ ist monoton}) \\
&\iff (\mathsf{I}\|\overline{\mathsf{I}}^\top)vec(S) \subseteq \overline{vec(S)} && (\text{Theorem 4.1.10, Abschnitt 2.4}) \\
&\iff (\mathsf{I}\|\overline{\mathsf{I}})s \subseteq \overline{s} && (\ \overline{\mathsf{I}} \text{ ist symmetrisch})
\end{aligned}
$$

2. Für die Injektivität erhalten wir:

$$
\begin{aligned}
S \text{ ist injektiv} &\iff \overline{\mathsf{I}}S \subseteq \overline{S} \\
&\iff vec(\overline{\mathsf{I}}S) \subseteq vec(\overline{S}) && (vec \text{ ist monoton}) \\
&\iff (\overline{\mathsf{I}}\|\mathsf{I}^\top)vec(S) \subseteq \overline{vec(S)} && (\text{Theorem 4.1.10, Abschnitt 2.4}) \\
&\iff (\overline{\mathsf{I}}\|\mathsf{I})s \subseteq \overline{s} && (\ \mathsf{I} \text{ ist symmetrisch})
\end{aligned}
$$

3. Man benötigt zunächst eine Hilfsaussage: $(*)$ Für jede surjektive Funktion A und jede Relation B gilt $AB = \mathsf{L} \iff B = \mathsf{L}$.

   "$\Rightarrow$" Sei $AB = \mathsf{L}$. Da A eine Funktion, und damit eindeutig ist, gilt $A^\top A \subseteq \mathsf{I}$. Da außerdem A surjektiv ist, ist $A^\top$ total, es gilt also $A^\top \mathsf{L} = \mathsf{L}$ und damit folgt

$$\begin{aligned} \mathsf{L} &= A^\top \mathsf{L} && (A \text{ ist surjektiv}) \\ &= A^\top AB && (\ AB = \mathsf{L}) \\ &\subseteq \mathsf{I}B && (\ A \text{ ist eindeutig}) \\ &= B, \end{aligned}$$

   also gilt $B = \mathsf{L}$.

   "$\Leftarrow$" Aus $B = \mathsf{L}$ und der Totalität von A folgt sofort $AB = A\mathsf{L} = \mathsf{L}$.

   Des Weiteren gilt:

$$\begin{aligned} vec(S\mathsf{L}) &= (\mathsf{I} \| \mathsf{L}) vec(S) && (\text{Theorem 4.1.10}) \\ &= (\pi \mathsf{I} \pi^\top \cap \rho \mathsf{L} \rho^\top) s \\ &= (\pi\pi^\top \cap \mathsf{L}) s && (\rho \text{ ist total und surjektiv }) \\ &= \pi\pi^\top s \end{aligned}$$

   Schließlich folgt mit $(*)$:

$$\begin{aligned} S \text{ is total} &\iff S\mathsf{L} = \mathsf{L} \\ &\iff vec(S\mathsf{L}) = vec(\mathsf{L}) \\ &\iff \pi\pi^\top s = \mathsf{L} && (\text{siehe oben}) \\ &\iff \pi^\top s = \mathsf{L} \end{aligned}$$

   Da π eine surjektive Funktion ist, kann im letzten Schritt $(*)$ angewendet werden.

4. Die letzte Äquivalenz wird ähnlich bewiesen. Aus Theorem 4.1.10 und der Totalität und Surjektivität von π folgt, analog zu 3., $vec(\mathsf{L}S) = \rho\rho^\top s$ und damit

$$\begin{aligned} S \text{ ist surjektiv} &\iff S\mathsf{L} = \mathsf{L} \\ &\iff vec(\mathsf{L}S) = vec(\mathsf{L}) \\ &\iff \rho\rho^\top s = \mathsf{L} && (\text{siehe oben}) \\ &\iff \rho^\top s = \mathsf{L}, \end{aligned}$$

wobei im letzten Schritt wiederum $(*)$ benutzt wird.

Mit Hilfe der eben bewiesenen Äquivalenzen und Lemma 3.2.3.1 können nun Vektorprädikate entwickelt werden. Für eine konkrete Relation des Typs $[X \leftrightarrow Y]$ gilt beispielsweise

$$\begin{aligned} S \text{ ist eindeutig} &\Longleftrightarrow (\mathsf{I}\|\overline{\mathsf{I}})s \subseteq \overline{s} \\ &\Longleftrightarrow \overline{\mathsf{L}((\mathsf{I}\|\overline{\mathsf{I}})s \cap s)} = \mathsf{L}. \qquad \text{(Lemma 3.2.3.1)} \end{aligned}$$

Man erhält also die Abbildung $\varphi_{eind} : [X \times Y \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ mit

$$\varphi_{eind}(v) = \overline{\mathsf{L}((\mathsf{I}\|\overline{\mathsf{I}})v \cap v)}$$

und es gilt $\varphi_{eind}(v) = \mathsf{L}$ genau dann, wenn die Relation $rel(v)$ eindeutig ist. Offensichtlich ist φ_{eind} ein Vektorprädikat und kann damit für jede beliebige Menge Z zu einer Testabbildung $\varphi^Z_{eind} : [X \times Y \leftrightarrow Z] \to [\mathbf{1} \leftrightarrow Z]$ mit $\varphi^Z_{eind}(P) = \overline{\mathsf{L}((\mathsf{I}\|\overline{\mathsf{I}})P \cap P)}$ erweitert werden. Es gilt dann

$$\varphi^Z_{eind}(P)_{\perp i} \Longleftrightarrow rel(P^{(i)}) \text{ ist eine eindeutige Relation.}$$

Der Zeilenvektor $\varphi^Z_{eind}(P)$ gibt also die Spalten von P an, die Vektordarstellungen von eindeutigen Relationen sind.

Analog erhält man das Vektorprädikat φ_{inj} durch $\varphi_{inj}(v) = \overline{\mathsf{L}((\overline{\mathsf{I}}\|\mathsf{I})v \cap v)}$ für den Test auf Injektivität. Das Vektorprädikat φ_{to} für den Test auf Totalität der zugrundeliegenden Relation ergibt sich wie folgt:

$$\begin{aligned} S \text{ ist total} &\Longleftrightarrow \pi^\top s = \mathsf{L} \\ &\Longleftrightarrow \overline{\pi^\top s} = \mathsf{O} \\ &\Longleftrightarrow \mathsf{L}\overline{\pi^\top s} = \mathsf{O} \qquad (\mathsf{L} : \mathbf{1} \leftrightarrow X \times Y) \\ &\Longleftrightarrow \overline{\mathsf{L}\overline{\pi^\top s}} = \mathsf{L} \qquad (\mathsf{L} : \mathbf{1} \leftrightarrow \mathbf{1}) \end{aligned}$$

Man erhält also $\varphi_{to}(v) = \overline{\mathsf{L}\overline{\pi^\top v}}$, und durch eine analoge Herleitung das Vektorprädikat φ_{sur} mit $\varphi_{sur}(v) = \overline{\mathsf{L}\overline{\rho^\top v}}$. Die Vektorprädikate können miteinander geschnitten und vereinigt werden, so erhält man z.B. durch $\varphi_{fun} = \varphi_{eind} \cap \varphi_{to}$ eine Abbildung, die testet, ob es sich bei der einem Vektor zugrundeliegede Relation um eine Funktion handelt.

4.3 Experimente

Um eine Vorstellung davon zu bekommen, inwieweit sich die Formel aus Theorem 4.1.10 in praktischen Anwendungen einsetzen lässt, und in welchen Fällen ihr Einsatz Vorteile in der Effizienz von Berechnungen bewirkt, haben wir verschiedene Experimente durchgeführt. Zunächst betrachten wir die Abbildung $\psi_{A,B} : [X \leftrightarrow Y] \to [Z \leftrightarrow W]$, definiert

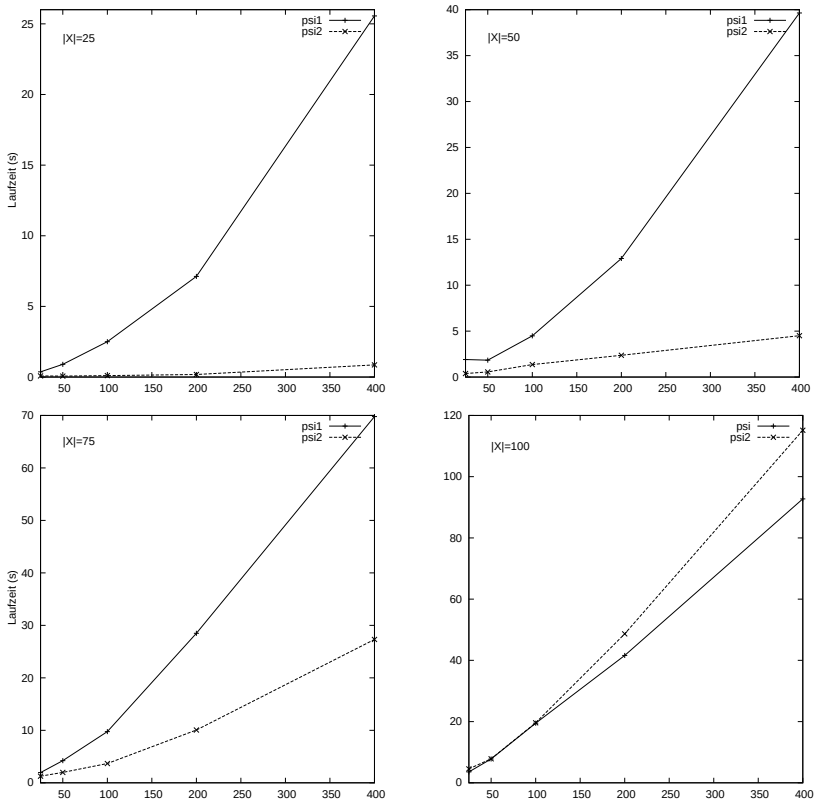


Abbildung 4.2: Vergleich der beiden Algorithmen

durch

$$\psi_{A,B}(S) = ASB,$$

wobei A und B Relationen des Typs $[Z \leftrightarrow X]$ bzw. $[Y \leftrightarrow W]$ sind. Es soll nun durch ein RelView-Programm zu einer Menge $\mathcal{R} \subseteq [X \leftrightarrow Y]$ die Menge $\psi_{A,B}(\mathcal{R})$, also die Menge aller Bilder von Relationen aus $\mathcal{R}$ unter der Abbildung $\psi_{A,B}$ berechnet werden. Da das Programm beliebig große Mengen von Relationen als Eingabe akzeptieren soll, wird eine Menge $\mathcal{R}$ von n Relationen als eine Relation R des Typs $[X \times Y \leftrightarrow [1..n]]$ modelliert. Die Ausgabe des Programms ist dann eine Relation T des Typs $[Z \times W \leftrightarrow [1..n]]$, so dass

$$rel(T^{(i)}) = \psi_{A,B}(R^{(i)}) = A\, rel(R^{(i)})\, B$$

für alle $i \in [1..n]$ gilt. Im naiven Ansatz ohne Verwendung von Theorem 4.1.10 muss also für jede Spalte $R^{(i)}$ der Eingaberelation die Relationendarstellung $rel(R^{(i)})$ berechnet werden, diese mit den Relationen A und B multipliziert und schließlich $vec(A\, rel(R^{(i)})\, B)$ als Spalte in die Resultatrelation T eingefügt werden. Man erhält das folgende RelView-Programm.

```
psi1(R,A,B,S)
  DECL T, z, p, r
  BEG T = 0(vec(Ln1(A)*L1n(B)) * L1n(R));
      z = L1n(List)^;
      WHILE -empty(z) DO
        p = point(z);
        r = vec(A * rel(R*p,S) * B);
        T = T | r*p^;
        z = z & -p
      OD
      RETURN T
  END.
```

Hierbei ist der Parameter S eine beliebige Relation des Typs $[X \leftrightarrow Y]$, die benötigt wird, um die Relationendarstellung der Spalten von R zu berechnen. Im zweiten Ansatz verwendet man die Tatsache, dass

$$vec(\psi_{A,B}(S)) = vec(ASB) = (A \| B^\top) vec(S)$$

nach Theorem 4.1.10 gilt, und dass insbesondere die Abbildung $v \mapsto (A \| B^\top)v$ ein Element der Menge $\mathcal{VA}$ (vgl. Kapitel 3.2.2) ist. Daher gilt, mit $T := (A \| B^\top)R$, die Gleichung

$$\begin{aligned} rel(T^{(i)}) &= rel(((A \| B^\top)R)^{(i)}) \\ &= rel((A \| B^\top)R^{(i)}) \\ &= rel(vec(A\, rel(R^{(i)})\, B)) \\ &= A\, rel(R^{(i)})\, B. \end{aligned}$$

Man erhält also die folgende RelView-Funktion

```
psi2(R,A,B) = pk(A,B^)*R,
```

wobei `pk(A,B^)` die parallele Komposition der Relationen A und B berechnet (siehe Anhang C). Zur Vereinfachung haben wir die Tests auf homogenen Relationen des gleichen Typs durchgeführt, d.h. die Programme berechnen Bilder der Abbildung

$$\psi_{A,B} : [X \leftrightarrow X] \to [X \leftrightarrow X],$$

wobei A und B zufällig erzeugte Relationen sind. Es wurden verschiedene Instanzen der Größenordnung zwischen $|X| = 25$ und $|X| = 100$ getestet, wobei die ebenfalls zufällig erzeugte Eingaberelation R aus 25 bis 400 Spalten bestand. Abbildung 4.2 zeigt das Laufzeitverhalten beider Programme für die verschiedenen Instanzen. Dabei gibt die x-Achse jeweils die Anzahl der Spalten der Eingaberelation R, und die y-Achse die Laufzeit in Sekunden an. Man sieht deutlich, dass das Programm `psi2`, welches sich auf Theorem 4.1.10 stützt, im Größenbereich von $|X| = 25$ bis $|X| = 75$ erheblich effizienter ist als der naive Ansatz, was sich insbesondere bei größeren Eingaberelationen R zeigt. Ab einer Größe von $|X| = 100$ erzielt das Programm `psi1` die besseren Ergebnisse. Insgesamt kann man also davon ausgehen, dass der in diesem Kapitel vorgestellte Ansatz erhebliche Vorteile bringt, wenn es darum geht Bilder von größeren Mengen von Relationen unter gewissen Abbildungen zu berechnen, sofern Urbild- und Bildmenge der Relationen nicht zu groß werden. Besonders interessant ist dies unter dem Gesichtspunkt von exakten Verfahren, da hierbei die gesamte Menge $[X \leftrightarrow Y]$ aller Relationen eines Typs als Eingaberelation des Typs $[X \times Y \leftrightarrow 2^{X \times Y}]$ modelliert wird. Dabei kann man mit dem neuen Verfahren Größenordungen erreichen, die mit dem naiven Ansatz, wenn überhaupt, nur mit sehr langen Laufzeiten erreichbar sind.

Im folgenden Kapitel werden anhand von Stundenplanproblemen Beispiele für exakte Lösungsverfahren angegeben, die sich auf Theorem 4.1.10 stützen. Da hierbei insbesondere eindeutige Relationen eine Rolle spielen, gehen wir im Folgenden auf den Vergleich zweier Verfahren ein, mit denen eine Menge von Relationen auf Eindeutigkeit getestet werden kann. Dabei sei wieder eine Menge $\mathcal{R} \subseteq [X \leftrightarrow Y]$ durch eine Relation $R : X \times Y \leftrightarrow [1..n]$ repräsentiert. Es soll nun ein Vektor u berechnet werden, für den u_i genau dann gilt, wenn $rel(R^{(i)})$ eine eindeutige Relation ist. Für einen fairen Vergleich der Verfahren benötigen wir zunächst ein einfaches Testkriterium für den naiven Ansatz. Es gilt für jede Relation Q

$$Q \text{ ist eindeutig} \iff Q\overline{\mathsf{I}} \subseteq \overline{Q} \iff Q\overline{\mathsf{I}} \cap Q = \mathsf{O},$$

und wir erhalten sofort das folgende RelView-Programm.

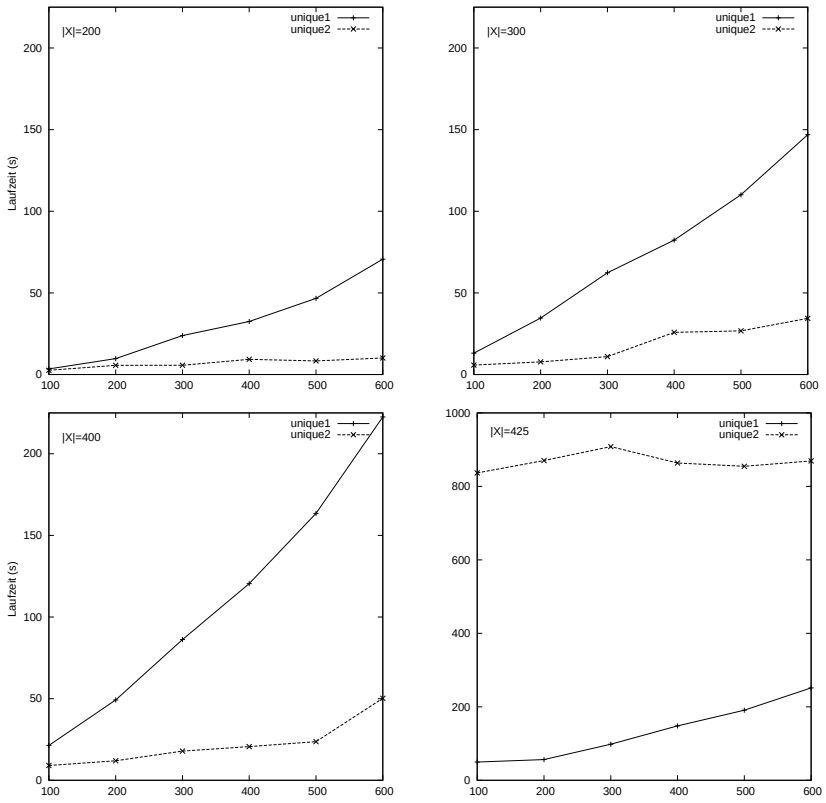


Abbildung 4.3: Vergleich der beiden Algorithmen zum Test auf Eindeutigkeit

```
unique1(R,S)
  DECL z, v, N, L1, L2, p, Q
  BEG z = L1n(R)^;
      v = O(z);
      N = -I(S^*S);
      WHILE -empty(z) DO
        p = point(z);
        Q = rel(R*p,S);
        IF empty(Q*N & Q) THEN v = v | p FI;
        z = z & -p
      OD
      RETURN v
  END.
```

Hierbei wird zu jeder Spalte von R ihre Relationendarstellung berechnet, und, falls diese eindeutig ist, ein entsprechender Eintrag im Ergebnisvektor gesetzt. In Abschnitt 4.2 wurde mithilfe von Theorem 4.1.10 bereits ein Vektorprädikat φ_{eind} für den Test auf

Eindeutigkeit der einem Vektor zugrundeliegenden Relation entwickelt. Daraus leitet sich sofort die folgende RelView-Funktion ab, die einen Zeilenvektor zurückliefert, welcher die Spalten der Eingaberelation R spezifiziert, die eindeutigen Relationen entsprechen.

```
unique2(R,S) = -(Ln1(R)^*(pk(I(S*S^),-I(S^*S))*R & R))
```

Zum Vergleich der Laufzeiten beider Programme wurden Experimente auf Instanzen durchgeführt, die sich auf homogene Relationen $S : X \leftrightarrow X$ stützen, wobei die Mächtigkeit der Menge X zwischen 200 und 425 lag. Als Eingaberelationen wurden zufällig erzeugte Relationen des Typs $[X \times X \leftrightarrow [1..k]]$ mit k zwischen 100 und 600 verwendet. Abbildung 4.3 zeigt für $|X| \leq 400$ die erheblich geringeren Laufzeiten von `unique2` gegenüber `unique1` insbesondere bei steigender Größe von k. Bei den Instanzen mit $|X| = 425$ fällt bei `unique2` ein erheblicher Anstieg der Laufzeit auf. Waren bei den kleineren Instanzen Laufzeiten von unter 50 Sekunden ausreichend, so werden hier Laufzeiten zwischen 800 und 1000 Sekunden benötigt, um den Test auf Eindeutigkeit durchzuführen. Hier erweist sich das Programm `unique1` als wesentlich effizienter.

5 Anwendung 1: Stundenplanprobleme

In vielen verschiedenen Bereichen des täglichen Lebens spielt die Erstellung von Zeitplänen eine wichtige Rolle. Typische Beispiele sind Stundenplanungen und Prüfungsplanungen an Schulen und Universitäten, Personaleinsatzplanung in Unternehmen sowie die Planung von Sportereignissen. In der Informatik werden Stundenplanprobleme als Teilgebiet des Scheduling bereits seit den 60er Jahren diskutiert (siehe z.B. [25] und [1]) In [41] werden ein Überblick über die bis 1979 erschienenen Publikationen zum Thema Stundenplanprobleme gegeben und die wichigsten Modelle und Lösungstechniken vorgestellt. Allgemein bestehen Stundenplanprobleme darin, eine Anzahl von Recourcen unter Einhaltung gewisser Restriktionen auf eine begrenzte Anzahl von Zeitschienen oder Räumlichkeiten so zu verteilen, dass eine Menge von Zielsetzungen zu einem möglichst hohen Grad erfüllt sind (siehe [47]). Dazu können verschiedene Methoden wie beispielsweise lineare Programmierung, Ansätze, die sich auf Graph-Färbung stützen, Cluster-Methoden, Constraint-basierte Verfahren, Metaheuristiken oder Evolutionäre Algorithmen verwendet werden. Für einen detaillierten Überblick über Stundenplanprobleme siehe [31]. Stundenplanprobleme sind im allgemeinen NP-vollständig, wie beispielsweise anhand eines einfachen Modells von Gotlieb (siehe [25]) in [21] gezeigt wird. Erste Ansätze zur Lösung von Stundenplanproblemen mithilfe relationaler Methoden finden sich in [39] und [40]. In diesem Kapitel werden Anwendungen aus dem Bereich der Stundenplanprobleme dargestellt, die mit Theorem 4.1.10 aus dem vorangehenden Kapitel und dem Konzept der Vektorprädikate aus Kapitel 3 bearbeitet werden. Es werden drei verschiedene Modelle von Stundenplanproblemen vorgestellt. Abschnitt 5.1 behandelt ein eher allgemeines Modell, welches bereits in [39] mithilfe relationaler Methoden diskutiert wurde. In den beiden folgenden Abschnitten werden konkrete Stundenplanprobleme aus der Praxis dargestellt, die im Zusammenhang mit der Einführung des Bachelor/Master-Systems an der Universität Kiel auftraten und durch die Verwendung relationaler Methoden und den bisher entwickelten Konzepten modelliert und gelöst werden konnten. Das Vorgehen ist in allen drei Fällen das gleiche. Eine informelle Problembeschreibung wird formalisiert und in eine relationale Pro-

blembeschreibung transformiert, wobei Eingabe-Relationen festgelegt und notwendige Eigenschaften von Lösungen des Problems angegeben werden. Daraus werden mithilfe von Theorem 4.1.10 Vektorprädikate entwickelt, die sowohl in exakten Verfahren, als auch in Suchheuristiken eingesetzt werden können. Am Ende des Kapitels gehen wir noch spezieller auf die Möglichkeit ein, Stundenplanprobleme mit Evolutionären Algorithmen zu lösen. Hier werden spezielle Variationsoperatoren entwickelt, und durch Experimente ermittelt, inwieweit sich die Qualität der erzielten Ergebnisse mithilfe dieser Operatoren gegenüber herkömmlichen Variationsoperatoren (siehe Abschnitt 3.3) verbessern lässt.

5.1 Klassisches Stundenplanproblem

In diesem Abschnitt wird Theorem 4.1.10 zur Behandlung eines einfachen Stundenplanproblems verwendet, welches sich auf viele Situationen anwenden lässt. In dem Modell, das auch in [39] und [40] diskutiert wird, geht man von einer Menge von Meetings aus, an denen jeweils eine bestimmte Gruppe von Personen teilnimmt. Diese Meetings sollen nun unter Einhaltung bestimmter Bedingungen auf eine Menge von Zeitschienen verteilt werden, dabei dürfen beispielsweise zwei Meetings nicht zum gleichen Zeitpunkt stattfinden, wenn sie einen gemeinsamen Teilnehmer haben. Es wird also nach einer Relation zwischen Meetings und Zeitschienen gesucht, die bestimmte Eigenschaften erfüllt. Diese erwünschten Eigenschaften lassen sich durch einfache relationale Ausdrücke formalisieren. Damit man nun eine Menge von möglichen Lösungen des Stundenplanproblems (Suchpunkte) hinsichtlich der erwünschten Eigenschaften auswerten kann, werden Relationen durch Vektoren repräsentiert, wie in Abschnitt 2.4 erklärt wurde. Die relationalen Eigenschaften werden unter Verwendung von Theorem 4.1.10 in Eigenschaften der zugehörigen Vektoren und schließlich in Vektorprädikate im Sinne von Abschnitt 3.2.2 transformiert.

Es sollte noch erwähnt werden, dass das in [21] diskutierte Stundenplanproblem von Gotlieb sich in unser Modell überführen lässt. Insbesondere handelt es sich daher um ein NP-vollständiges Problem.

5.1.1 Relationale Modellierung

Im Folgenden sei ein Stundenplanproblem durch ein Tupel

$$\mathcal{T} = (\mathcal{M}, \mathcal{P}, \mathcal{H}, A, P)$$

gegeben, wobei $\mathcal{M}$ eine endliche Menge von Meetings, $\mathcal{P}$ eine endliche Menge von Teilnehmern, $\mathcal{H}$ eine endliche Menge von Zeitschienen sowie $A : \mathcal{M} \leftrightarrow \mathcal{H}$ und $P : \mathcal{M} \leftrightarrow \mathcal{P}$ Relationen sind. Die Relation A beschreibt, welche Meetings in welche Zeitschienen eingeordnet werden können, es gilt also A_{mh}, wenn die Zeitschiene h für das Meeting m geeignet ist. Beispielsweise könnten sich die Zeitschienen bezüglich ihrer Größe unterscheiden, und es könnten Meetings existieren, die besonders viel Zeit und somit eine große Zeitschiene beanspruchen. Die Relation P gibt zu jedem Meeting die Menge der Teilnehmer an, d.h. es gilt P_{mp}, falls p am Meeting m teilnimmt. Zur Vereinfachung definieren wir eine weitere Relation C, die im Weiteren *Konfliktrelation* genannt wird, und die sich direkt aus der Relation P ergibt. Zwei verschiedene Meetings m und m' stehen in Konflikt miteinander, falls sie einen gemeinsamen Teilnehmer haben. Formal bedeutet das:

$$\begin{aligned} m \text{ und } m' \text{ sind in Konflikt} &\iff m \neq m' \ \wedge \ \exists p : P_{mp} \ \wedge \ P_{m'p} \\ &\iff \overline{\mathsf{I}}_{mm'} \ \wedge \ (PP^\top)_{mm'} \\ &\iff (\overline{\mathsf{I}} \cap PP^\top)_{mm'} \end{aligned}$$

Definieren wir also die Relation $C : \mathcal{M} \leftrightarrow \mathcal{M}$ durch

$$C := \overline{\mathsf{I}} \cap PP^\top,$$

so sind m und m' genau dann in Konflikt, wenn $C_{mm'}$ gilt. Eine Lösung für $\mathcal{T}$ ordnet jedem Meeting eine geeignete Zeitschiene zu, so dass in Konflikt stehende Meetings nicht zur gleichen Zeit stattfinden. Eine Lösung für $\mathcal{T}$ ist also eine Relation $S : \mathcal{M} \leftrightarrow \mathcal{H}$ mit den folgenden Eigenschaften

(L1) $\forall\, m, h : S_{mh} \rightarrow A_{mh}$

(L2) $\forall\, m, m', h : (C_{mm'} \ \wedge \ S_{m'h}) \rightarrow \neg S_{mh}$

(L3) S ist eindeutig

(L4) S ist total

Erfüllt S nur die Eigenschaften (L1) - (L3), so nennen wir S eine Teillösung des Stundenplanproblems $\mathcal{T}$. Die Eigenschaft (L1) drückt aus, dass ein Meeting m nur einer Zeitschiene h zugeordnet werden kann, die gemäß der Relation A als für m geeignet festgelegt ist. Durch (L2) wird zugesichert, dass in Konflikt stehende Meetings nicht zur gleichen Zeit stattfinden, und durch Eindeutigkeit und Totalität wird sichergestellt,

das jedes Meeting genau zu einer Zeit stattfindet. Zunächst werden (L1) - (L4) durch relationale Inklusionen dargestellt. Es gilt

(L1) $S \subseteq A$

(L2) $CS \subseteq \overline{S}$

(L3) $S\bar{\mathsf{I}} \subseteq \overline{S}$

(L4) $S\mathsf{L} = \mathsf{L}$

Denn offensichtlich gilt für (L1):

$$\forall\, m, h : S_{mh} \rightarrow A_{mh} \Longleftrightarrow S \subseteq A,$$

und in Fall von (L2):

$$\begin{aligned}
&\forall\, m, m', h : (C_{mm'} \ \wedge \ S_{m'h}) \rightarrow \neg S_{mh} \\
&\Longleftrightarrow \forall\, m, h : (\exists m' : C_{mm'} \ \wedge \ S_{m'h}) \rightarrow \overline{S}_{mh} \\
&\Longleftrightarrow \forall\, m, h : (CS)_{mh} \rightarrow \overline{S}_{mh} \\
&\Longleftrightarrow CS \subseteq \overline{S}
\end{aligned}$$

Damit liegt ein relationales Modell des Stundenplanproblems vor, das im Folgenden schrittweise zu einem Lösungsverfahren weiterentwickelt wird.

5.1.2 Entwicklung eines Vektorprädikats

In diesem Abschnitt wird aus der relationalen Beschreibung des Problems ein Vektorprädikat entwickelt, das, angewandt auf einen Vektor des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$, bestimmt, ob die dem Vektor zugrundeliegende Relation eine Lösung des Stundenplanproblems $\mathcal{T}$ ist. Zunächst werden die Eigenschaften (L1) - (L4) einer Relation als Eigenschaften ihrer Vektorrepräsentation ausgedrückt. Seien also im Folgenden $s = vec(S)$ und $a = vec(A)$ die Vektorrepräsentationen der Relationen S und A. Dann ist S eine Lösung von $\mathcal{T}$ genau dann, wenn s die folgenden 4 Eigenschaften erfüllt.

(L1) $s \subseteq a$

(L2) $(C\|\mathsf{I})s \subseteq \overline{s}$

(L3) $(\mathsf{I}\|\bar{\mathsf{I}})s \subseteq \overline{s}$

(L4) $\pi^{\top} s = \mathsf{L}$

Aus der Monotonie von *vec* folgt sofort die Äquivalenz von $S \subseteq A$ und $s \subseteq a$. Für (L2) gilt

$$\begin{aligned} CS \subseteq \overline{S} &\iff vec(CS) \subseteq vec(\overline{S}) && \text{(Monotonie von } vec\text{)} \\ &\iff (C\|\mathsf{I})vec(S) \subseteq \overline{vec(S)} && \text{(Theorem 4.1.10, Abschnitt 2.4)} \\ &\iff (C\|\mathsf{I})s \subseteq \overline{s} \end{aligned}$$

Eindeutigkeit und Totalität wurden bereits in Abschnitt 4.2 behandelt. Durch Anwendung von Lemma 3.2.3.1 erhält man sofort die folgenden Vektorprädikate, mit deren Hilfe Vektoren auf die Eigenschaften (L1), ..., (L4) getestet werden können. Mit

$$\varphi_{(L1)}(s) = \overline{\mathsf{L}(s \cap \overline{a})},$$

$$\varphi_{(L2)}(s) = \overline{\mathsf{L}(C\|\mathsf{I})s \cap s)},$$

$$\varphi_{(L3)}(s) = \overline{\mathsf{L}((\mathsf{I}\|\bar{\mathsf{I}})s \cap s)},$$

$$\varphi_{(L4)}(s) = \overline{\mathsf{L}\overline{\pi^\top s}}$$

folgt nun insgesamt

$$S \text{ ist eine Lösung für } \mathcal{T} \iff \varphi_{(L1)}(s) \cap \varphi_{(L2)}(s) \cap \varphi_{(L3)}(s) \cap \varphi_{(L4)}(s) = \mathsf{L},$$

und es gilt

$$\begin{aligned} &\varphi_{(L1)}(s) \cap \varphi_{(L2)}(s) \cap \varphi_{(L3)}(s) \cap \varphi_{(L4)}(s) \\ &= \overline{\mathsf{L}(s \cap \overline{a})} \cap \overline{\mathsf{L}(C\|\mathsf{I})s \cap s)} \cap \overline{\mathsf{L}((\mathsf{I}\|\bar{\mathsf{I}})s \cap s)} \cap \overline{\mathsf{L}\overline{\pi^\top s}} \\ &= \overline{\mathsf{L}((s \cap \overline{a}) \cup (C\|\mathsf{I})s \cap s) \cup ((\mathsf{I}\|\bar{\mathsf{I}})s \cup s) \cup \mathsf{L}\overline{\pi^\top s})} \\ &= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup \ (C\|\mathsf{I})s \cup (\mathsf{I}\|\bar{\mathsf{I}})s) \cap s))} \\ &= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup \ ((C\|\mathsf{I}) \cup (\mathsf{I}\|\bar{\mathsf{I}}))s) \cap s))}. \end{aligned}$$

Man erhält also das Vektorprädikat $\varphi_{st} : [\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch

$$\varphi_{st}(s) = \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup \ ((C\|\mathsf{I}) \cup (\mathsf{I}\|\bar{\mathsf{I}}))s) \cap s))},$$

mit dem für einen Vektor $s : \mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}$ getestet wird, ob *rel*(s) einer Lösung für $\mathcal{T}$ entspricht. Das Vektorprädikat kann für jede beliebige Menge Z, wie in Abschnitt 3.2.2 erklärt, zu einer Testabbildung $\varphi_{st}^Z : [\mathcal{M} \times \mathcal{H} \leftrightarrow Z] \to [\mathbf{1} \leftrightarrow Z]$ fortgesetzt werden, um aus einer Relation $P : \mathcal{M} \times \mathcal{H} \leftrightarrow Z$ die Spalten herauszufiltern, die gültigen Lösungen

für $\mathcal{T}$ des Stundenplanproblems entsprechen. Je nach Größe des Problems kann die Testabbildung in einem exakten Verfahren oder in einem Evolutionären Algorithmus eingesetzt werden, wie im nächsten Abschnitt genauer erläutert wird.

5.1.3 Einsatz der Vektorprädikate

Um das hier diskutierte Stundenplanproblem mithilfe des Vektorprädikats φ_{st} exakt zu lösen, und damit alle Lösungen zu berechnen, verwendet man die Potenzmengenrelation $\mathsf{M} : \mathcal{M} \times \mathcal{H} \leftrightarrow 2^{\mathcal{M}\times\mathcal{H}}$, und modelliert damit die Menge aller Relationen des Typs $[\mathcal{M} \leftrightarrow \mathcal{H}]$ in Vektordarstellung. Der Zeilenvektor $\varphi_{st}^{2^{\mathcal{M}\times\mathcal{H}}}(\mathsf{M})$ des Typs $[\mathbf{1} \leftrightarrow 2^{\mathcal{M}\times\mathcal{H}}]$ gibt dann die Spalten von M an, die Lösungen von $\mathcal{T}$ darstellen, man erhält also die Menge aller Lösungen von $\mathcal{T}$. Falls also das Stundenplanproblem nicht lösbar ist, gilt $\varphi_{st}^{2^{\mathcal{M}\times\mathcal{H}}}(\mathsf{M}) = \mathsf{O}$. Gilt hingegen $\varphi_{st}^{2^{\mathcal{M}\times\mathcal{H}}}(\mathsf{M}) \neq \mathsf{O}$, erhält man mit $\mathsf{M}p$ eine Lösung des Stundenplanproblems in Vektordarstellung, wobei p ein Punkt mit $p \subseteq \varphi_{st}^{2^{\mathcal{M}\times\mathcal{H}}}(\mathsf{M})^\top$ ist. Die Relation $rel(\mathsf{M}p)$ ist dann eine Lösung in Relationendarstellung.

Um das Stundenplanproblem durch Verwendung eines Evolutionären Algorithmus zu lösen, kann beispielsweise das Vektorprädikat $\varphi_{tl} = \varphi_{(L1)} \cap \varphi_{(L2)} \cap \varphi_{(L3)}$ verwendet werden, mit dem getestet wird, ob ein Vektor eine Teillösung von $\mathcal{T}$ repräsentiert. Ziel des Algorithmus ist es dann, möglichst große Teillösungen zu finden, also Teillösungen, in denen möglichst viele Meetings einer Zeitschiene zugeordnet sind. Dazu werden im Selektionsschritt ausschließlich zulässige Teillösungen übernommen, die mithilfe des Vektorprädikats φ_{tl} aus der Population herausgefiltert werden. Die so bestimmten Individuen werden dann zusätzlich hinsichtlich der Anzahl ihrer Einträge bewertet, und dementsprechend in die nächste Generation übernommen.

Um die Suche nach geeigneten Teillösungen zielgerichteter zu gestalten, können statt der in den Abschnitten 3.3.1 und 3.3.2 vorgestellten Variationsoperatoren speziellere Varianten verwendet werden, so dass ausschließlich Nachkommen mit gewissen Eigenschaften erzeugt werden. Die einfachste Möglickeit besteht darin, jede Offspring-Generation C vor dem Selektionsschritt zu

$$C' = C \cap vec(A)\mathsf{L}$$

zu modifizieren, und damit sicherzustellen, dass für jedes der Individuen in C' bereits die Bedingung (L1) gilt. Im Selektionsschritt wird dann das Vektorprädikat $\varphi_{(L2)} \cap \varphi_{(L3)}$ verwendet, um zulässige Individuen zu bestimmen. Eine weitere, schwieriger zu realisierende Möglichkeit ist die Beschränkung auf eindeutige Nachkommen, also die Er-

zeugung von Populationen, in denen jede Spalte eine eindeutige Relation repräsentiert. Hierauf wird später in Abschnitt 5.4 ausführlicher eingegangen. Stellt man sicher, dass im Verlauf des Algorithmus ausschließlich eindeutige Individuen auftreten, kann im Selektionsschritt das Vektorprädikat $\varphi_{(L1)} \cap \varphi_{(L2)}$ verwendet werden, um zulässige Teillösungen zu bestimmen.

Im Folgenden wird eine Möglichkeit erläutert, mit der die Effizienz sowohl des exakten als auch des randomisierten Verfahrens gesteigert werden kann, indem ausgenutzt wird, dass zulässige Individuen immer Teilvektoren eines festen Vektors (der Vektordarstellung der Verfügbarkeitsrelation A) sind. Hierdurch kann der Suchraum eingeschränkt werden, was je nach der Beschaffenheit von A einen erheblichen Einfluss auf die Effizienz des Verfahrens haben kann. Der Übersichtlichkeit halber betrachten wir zunächst eine verallgemeinerte Situation. Sei im Folgenden $a : X \leftrightarrow \mathbf{1}$ mit $a \neq \mathsf{O}$ gegeben. Es repräsentiere der Vektor a die Teilmenge Y von X und es sei $J := \mathit{inj}(a)$, also insbesondere J eine injektive Funktion des Typs $[Y \leftrightarrow X]$. Das folgende Lemma bildet die Grundlage für den Ansatz.

5.1.3.1 Lemma *Seien $R : X \leftrightarrow X$ und $v : X \leftrightarrow \mathbf{1}$, sowie $R' = JRJ^\top$ und $v' = Jv$ gegeben. Dann ist R' vom Typ $[Y \leftrightarrow Y]$ und v' vom Typ $[Y \leftrightarrow \mathbf{1}]$ und es gilt die folgende Äquivalenz*

$$(Rv \subseteq \overline{v} \ \wedge \ v \subseteq a) \Longleftrightarrow R'v' \subseteq \overline{v'}$$

Beweis: *„⇒“ Es gelte $Rv \subseteq v$ und $v \subseteq a$. Dann folgt*

$$\begin{aligned} R'v' &= JRJ^\top Jv \\ &\subseteq JRv && (J \text{ ist eindeutig}) \\ &\subseteq J\overline{v} \\ &\subseteq \overline{Jv} && (J \text{ist eindeutig}) \\ &= \overline{v'} \end{aligned}$$

„⇐“ Es gilt $R = J^\top R'J$ und $v = J^\top v'$. Aus $R'v' \subseteq \overline{v'}$ folgt zunächst

$$\begin{aligned} Rv &= J^\top R'JJ^\top v' \\ &\subseteq J^\top R'v' && (J \text{ injektiv}) \\ &\subseteq J^\top \overline{v'} && (R'v' \subseteq \overline{v'}) \\ &\subseteq \overline{J^\top v}. && (J^\top \text{ eindeutig}) \\ &= \overline{v} \end{aligned}$$

Außerdem gilt $v = J^\top v' \subseteq J^\top \mathsf{L} = inj(a)^\top \mathsf{L} = a$ nach Abschnitt 2.3.

Das Lemma kann wie folgt bei der Bestimmung von Teillösungen des Stundenplanproblems verwendet werden. Definiert man $R := (C\|\mathsf{I}) \cup (\mathsf{I}\|\bar{\mathsf{I}})$, so ergibt sich die folgende Äquivalenz:

$$
\begin{aligned}
& s \text{ ist eine Teillösung} \\
\iff & s \subseteq a \ \wedge \ (C\|\mathsf{I})s \subseteq \overline{s} \ \wedge \ (\mathsf{I}\|\bar{\mathsf{I}})s \subseteq \overline{s} && \text{(Abschnitt 5.1.2)} \\
\iff & s \subseteq a \ \wedge \ (C\|\mathsf{I})s \cup (\mathsf{I}\|\bar{\mathsf{I}})s \subseteq \overline{s} \\
\iff & s \subseteq a \ \wedge \ ((C\|\mathsf{I}) \cup (\mathsf{I}\|\bar{\mathsf{I}}))s \subseteq \overline{s} \\
\iff & s \subseteq a \ \wedge \ Rs \subseteq \overline{s}.
\end{aligned}
$$

Mit $s' = inj(a)s$ und $R' = inj(a)R\, inj(a)^\top$ folgt also

$$s \text{ ist eine Teillösung} \iff R's' \subseteq \overline{s'}.$$

Es ergibt sich das Vektorprädikat $\varphi_{R'} : [Y \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ durch $\varphi_{R'}(v) = \overline{\mathsf{L}(R'v \cap v)}$, wobei Y die Teilmenge von $\mathcal{M} \times \mathcal{H}$ ist, die durch den Vektor $a = vec(A)$ repräsentiert wird. Für jeden Vektor $v : Y \leftrightarrow \mathbf{1}$ ist $inj(a)^\top v$ vom Typ $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ und es gilt

$$\varphi_{R'}(v) = \mathsf{L} \iff inj(a)^\top v \text{ ist eine Teillösung.}$$

Wie üblich, kann das Vektorprädikat $\varphi_{R'}$ für jede beliebige Menge Z zu einer Testabbildung $\varphi^Z_{R'}(M) : [Y \leftrightarrow Z] \to [\mathbf{1} \leftrightarrow Z]$ erweitert werden, so dass für jede Relation $M : Y \leftrightarrow Z$ die Äquivalenz

$$\varphi^Z_{R'}(M)_{\perp i} \iff inj(a)^\top M^{(i)} \text{ ist eine Teillösung}$$

gilt. Insbesondere erhält man durch

$$inj(a)^\top \mathsf{M}\, inj(\varphi^{2^Y}_{R'}(\mathsf{M})^\top)^\top$$

die Menge aller Teillösungen, modelliert als Liste von Vektoren des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$, wobei hierbei die Potenzmengenrelation $\mathsf{M} : Y \leftrightarrow 2^Y$ verwendet wird. Im Vergleich zum herkömmlichen exakten Verfahren, welches die Potenzmengenrelation vom Typ $[\mathcal{M} \times \mathcal{H} \leftrightarrow 2^{\mathcal{M}\times\mathcal{H}}]$ verwendet, ist durch diesen Ansatz je nach Füllgrad der Relation A eine erhebliche Effizienzsteigerung zu erwarten.

Für einen Vergleich der beiden exakten Verfahren entwickeln wir zunächst ein weiteres

Vektorprädikat φ_{R^a} auf der Grundlage der vorangehenden Abschnitte. Dazu benötigen wir das folgende Lemma.

5.1.3.2 Lemma *Für Vektoren s und a gilt die Äquivalenz*

$$s \subseteq a \iff \overline{aa^\top} s \subseteq \overline{s}.$$

Beweis: *Wir zeigen zunächst die Äquivalenz der Inklusionen $s \subseteq a$ und $ss^\top \subseteq aa^\top$, wobei die Richtung von links nach rechts trivial ist. Da s ein Vektor ist, ist s entweder der leere Vektor oder surjektiv. Ist $s = \mathsf{O}$, folgt sofort $s \subseteq a$. Ist $s \neq \mathsf{O}$ und damit surjektiv, gilt $\mathsf{O} \neq s\mathsf{L} = ss^\top\mathsf{L} \subseteq aa^\top\mathsf{L}$ und damit ist auch $a \neq \mathsf{O}$ und es folgt $s\mathsf{L} \subseteq aa^\top\mathsf{L} = a\mathsf{L}$ und damit $s \subseteq a$. Mit den Schröder-Äquivalenzen folgt*

$$s \subseteq a \iff ss^\top \subseteq aa^\top \iff \overline{aa^\top} s \subseteq \overline{s}.$$

Wir haben bereits gezeigt, dass ein Vektor s genau dann eine Teillösung des Stundenplanproblems ist, wenn $s \subseteq a$ und $Rs \subseteq \overline{s}$ gilt. Mit dem vorangegangenen Lemma folgt nun für $R^a := R \cup \overline{aa^\top}$:

$$\begin{aligned} s \text{ ist eine Teillösung} &\iff s \subseteq a \ \wedge \ Rs \subseteq \overline{s} \\ &\iff \overline{aa^\top} s \subseteq \overline{s} \ \wedge \ Rs \subseteq \overline{s} \\ &\iff (R \cup \overline{aa^\top}) s \subseteq \overline{s} \\ &\iff R^a s \subseteq \overline{s} \end{aligned}$$

Wir erhalten also analog zum Vektorprädikat $\varphi_{R'}$ das Vektorprädikat

$$\varphi_{R^a} : [\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$$

durch $\varphi_{R^a}(v) = \overline{\mathsf{L}(R^a v \cap v)}$, so dass man unter Verwendung der Potenzmengenrelation M auf $\mathcal{M} \times \mathcal{H}$ mit

$$\varphi_{R^a}^{2^{\mathcal{M}\times\mathcal{H}}}(\mathsf{M})$$

den Zeilenvektor erhält, der alle Teillösungen des Stundenplanproblems spezifiziert. Daraus ergibt sich sofort die RelView-Funktion

```
phi(X,P) = -(L1n(X)*(X*P & P))
```

zur Implementierung der aus den Vektorprädikaten φ_{R^a} und $\varphi_{R'}$ abgeleiteten Testabbildungen, sowie das folgende Programm zur Berechnung der Relation R aus der Kon-

fliktrelation C, wobei die Verfügbarkeitsrelation A nur zur Erzeugung einer Identitätsrelation auf $\mathcal{H}$ dient .

```
R(A,C)
  DECL I1, I2, R
  BEG I1 = I(C);
      I2 = I(L1n(A)^*L1n(A));
      R = pk(C,I2) | pk(I1,-I2)
      RETURN R
   END.
```

Durch die so erzeugte Relation R wird im Folgenden zur Berechnung der Relationen $R^a = R \cup \overline{aa^\top}$ und $R' = inj(a) R\, inj(a)^\top$ verwendet. Darauf aufbauend kann man nun zwei exakte Algorithmen zur Bestimmung aller größten Teillösungen implementieren. Wir betrachten zunächst die herkömmliche Variante.

```
grePartSol1(A,C)
  DECL a, Ra, M, vec, Cr, g, list
  BEG a = vec(A);
      Ra = R(A,C) | -(a*a^);
      M = epsi(a);
      vec = phi(Ra,M);
      Cr = cardrel(a);
      g = ge(Cr,vec^);
      list = M*inj(g)^
      RETURN list
   END
```

Hierbei wird die Potenzmengenrelation vom Typ $[\mathcal{M} \times \mathcal{H} \leftrightarrow 2^{\mathcal{M}\times\mathcal{H}}]$ erzeugt und mit φ_{R^a} der Zeilenvektor aller Teillösungen ermittelt. Unter Verwendung der Kardinalitätsvergleichsrelation werden dann die größten Teillösungen bestimmt und diese als Liste von Vektoren ausgegeben. Die verbesserte Variante arbeitet mit einem verkleinerten Suchraum. Hier wird die Potenzmengenrelation M vom Typ $[Y \leftrightarrow 2^Y]$ verwendet, wobei $Y = \{\langle m, h\rangle \mid A_{mh}\}$ ist [1]. Mithilfe des Vektorprädikats $\varphi_{R'}$ wird der Zeilenvektor aller Teillösungen ermittelt, der hierbei vom Typ $[\mathbf{1} \leftrightarrow 2^Y]$ ist, und unter Verwendung der Kardinalitätsvergleichsrelation vom Typ $[2^Y \leftrightarrow 2^Y]$ der Vektor g bestimmt, welcher die größten Teillösungen repräsentiert. Mit $inj(a)^\top \mathsf{M}\, inj(g)^\top$ wird schließlich die Liste aller größten Teillösungen zurückgegeben.

[1] Die Menge Y entspricht also der Relation A

```
grePartSol2(A,C)
  DECL a, J, L, Rs, M, vec, Cr, gre, list
  BEG a = vec(A);
      J = inj(a);
      L = Ln1(J);
      Rs = J*R(A,C)*J^;
      M = epsi(L);
      vec = phi(Rs,M );
      Cr = cardrel(L);
      g = ge(Cr,vec^);
      list = J^*M*inj(g)^
      RETURN list
  END.
```

Wir haben die Laufzeiten beider Algorithmen bezüglich verschiedener randomisiert erzeugter Instanzen mit 30 Meetings und 4 Zeitschienen verglichen. Dabei wurde deutlich, dass die Laufzeiten beider Algorithmen stark vom Füllgrad der verwendeten Eingaberelationen abhängen. Je mehr Einträge die größten Teillösungen haben, desto mehr Teillösungen existieren insgesamt, so dass beide Algorithmen mehr Zeit benötigen, um diese zu berechnen. Abbildung 5.1 stellt die Laufzeiten der beiden Algorithmen

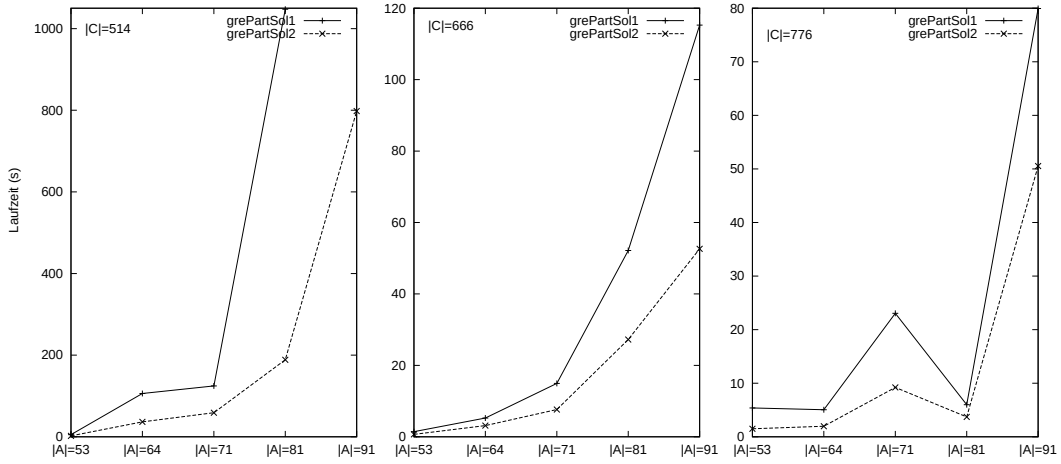


Abbildung 5.1: Vergleich der Algorithmen

in Abhängigkeit der Eingaberelationen C und A dar. Es wurden 3 verschiedene Konfliktrelationen verwendet, bei denen die Anzahl der Einträge zwischen 541 und 776 lag, sowie 5 Verfügbarkeitsrelationen mit $|A|$ zwischen 53 und 91. Bei allen getesteten Instanzen weist das hier entwickelte, optimierte Verfahren deutlich geringere Laufzei-

ten auf. Überraschend ist hierbei, dass die Unterschiede zwischen den Algorithmen bei geringer gefüllten Verfügbarkeitsrelationen abnehmen, obwohl man erwarten könnte, dass gerade hier der Vorteil des optimierten Verfahrens deutlicher zutage tritt, da hier deutlich kleinere Potenzmengen- und Kardinalitätsvergleichsrelationen verwendet werden. Das Phänomen ist jedoch damit zu erklären, dass durch die geringen Anzahl der Einträge der Verfügbarkeitsrelationen nur wenige Teillösungen existieren, so dass beide Verfahren sehr schnell sind, ohne dass sich ein deutlicher Unterschied abzeichnet. In den Fällen mit eher gering besetzter Konfliktrelation und eher stark gefüllter Verfügbarkeitsrelationen ist das optimierte Verfahren dem herkömmlichen Verfahren deutlich überlegen.

5.2 Universitäres Stundenplanproblem

In diesem Abschnitt wird ein Problem aus der Praxis behandelt, das bei der Einführung der Bachelor- und Masterstudiengänge an der Universität Kiel auftrat. Es sollte unter anderem Bachelor-Studiengänge für das Lehramt der Sekundarstufe 2, sowie als Ersatz für die frühren Magister-Studiengänge eingeführt werden. Um einen Bachelor-Studiengang für diese Studiengänge anbieten zu können, war es notwendig, einen Studienplan zu entwickeln, der es den Studierenden ermöglicht, ihr Studium in 3 Jahren, bzw für bestimmte Ausnahmefälle in höchstens 4 Jahren abzuschließen. Da sowohl von den Lehramts- als auch von den Magisterstudenten traditionell zwei Fächer studiert werden, ergeben sich zwangsläufig Überschneidungen zwischen den Veranstaltungen der einzelnen Fächer, die eine genaue Voraussage der Studiendauer erschweren. Aus dieser Situation ergibt sich ein Stundenplanproblem, welches im Folgenden erläutert und mit relationalen Methoden gelöst wird. Auch hierbei kommt dem im vorangehenden Kapitel bewiesenen Theorem 4.1.10 eine zentrale Bedeutung zu. Eine gekürzte Fassung der hier vorgestellten Ergebnisse wurde bereits in [9] veröffentlicht.

5.2.1 Informelle Problembeschreibung

Bei diesem Stundenplanproblem aus dem universitären Bereich soll ein Stundenplan für Lehramts- und Magisterstudenten entwickelt werden. Dabei sollen die Veranstaltungen für verschiendene Fächer zeitlich so verteilt werden, dass gewisse Fächerkombinationen ohne Überschneidungen und Verzögerungen studiert werden können. Es wird eine Menge von Schulfächern angeboten, aus der jeder Student zwei auswählen muss. Dabei gibt es Kombinationen, die häufig gewählt werden, oder denen aus anderen Gründen eine größere Bedeutung zukommt, wie z.B. den Kombinationen Mathematik und Phy-

sik oder Deutsch und Englisch. Diese werden wir im Folgenden als Kombinationen der Kategorie 2 bezeichnen. Seltener gewünschte Kombinationen (wie Kunst und Englisch) ordnen wir der Kategorie 1 zu. Schließlich bezeichnet Kategorie 0 die Fächerkombinationen, die fast nie studiert werden. Die Studierenden müssen für jedes ihrer beiden Fächer verschiedene Veranstaltungen belegen. Ziel ist es nun, einen Stundenplan zu erstellen, in dem je zwei Fächer, deren Kombination zur Kategorie 2 gehört, sich zeitlich nicht überschneiden. Fächerkombinationen der Kategorie 1 sollen immerhin eingeschränkt d.h., mit einer verlängerten Studiendauer studierbar sein, was im Folgenden genauer erklärt wird.

Das Studium erstreckt sich im Idealfall über 3 Jahre, wobei für jedes Jahr ein gesonderter Stundenplan erstellt wird. Dafür wird eine Woche in drei (sich nicht überschneidende) Zeitschienen eingeteilt, die über die Zeit konstant bleiben. Für jeweils ein Jahr werden dann die Veranstaltungen auf die drei Zeitschienen verteilt, so dass alle Veranstaltungen eines Faches in derselben Zeitschiene stattfinden. Fächerkombinationen der Kategorie 2 sollen immer in verschiedenen Zeitschienen stattfinden, damit sich keine Überschneidungen ergeben. Die Fächerkombinationen der Kategorie 1 sollen jeweils 2 Jahre lang auf verschiedene Zeitschienen verteilt werden, um dann im dritten Jahr parallel, also innerhalb derselben Zeitschiene stattzufinden. Daher kann im dritten Jahr nur eines der Fächer (oder beide Fächer nur mit einem Teil der Veranstaltungen) studiert werden, was ein viertes Studienjahr zur Folge hat. Bei dem konkreten Problem der Universität Kiel hat man eine Menge $\mathcal{F}$ von 34 Schulfächern und eine Menge $\mathcal{G} = \{A, B, C, U, V, W, X, Y, Z\}$ von 9 Gruppen zur Verfügung. Jedes Fach soll in eine Gruppe eingeordnet werden. Die Gruppen sind wiederum in 3 Blöcke eingeteilt, d.h., es sei die Menge $\mathcal{B} = \{1, 2, 3\}$ gegeben und die Abbildung $b : \mathcal{G} \to \mathcal{B}$ durch $b(A) = b(B) = b(C) = 1, b(U) = b(V) = b(W) = 2$ und $b(X) = b(Y) = b(Z) = 3$ definiert.

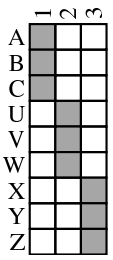


Abbildung 5.2: Die Abbildung b

Weiterhin ist der folgende Zeitplan gegeben, der für einen Zyklus von 3 Jahren die Veranstaltungen jeder Fächergruppe einer Zeitschiene zuordnet.

Block	Gruppe \ Jahr	1	2	3
1	A	1	1	1
	B	2	2	2
	C	3	3	3
2	U	1	2	3
	V	2	3	1
	W	3	1	2
3	X	1	3	2
	Y	2	1	3
	Z	3	2	1

Abbildung 5.3: Zeitplan mit rotierenden Zeitschienen

Die Fächer, die in die Gruppen A, B und C des Blocks 1 eingeordnet sind, haben also feste Zeitschienen, während die Zeitschienen für die restlichen Gruppen jährlich wechseln. Wie man anhand von Abbildung 5.3 sieht, kann man 2 Fächer genau dann überschneidungsfrei studieren, wenn sie verschiedenen Gruppen desselben Blocks zugeordnet sind. Wählt man Fächer aus verschiedenen Blöcken, z.B Fach a aus A und Fach v aus V, so kann man im ersten und zweiten Jahr an den Veranstaltungen beider Fächer teilnehmen, während im dritten Jahr beide in der Zeitschiene 1 angesiedelt sind, so dass es zu Überschneidungen kommt, die ein viertes Studienjahr zur Folge haben. Eine Funktion $K : \mathcal{F} \times \mathcal{F} \to \{0, 1, 2\}$ ordnet den Fächerkombinationen ihre Kategorien zu. Dabei gilt aus praktischen Gründen $K(f, f) = 0$ für alle Fächer f. Im Folgenden wird eine formale Beschreibung des oben erklärten Stundenplanproblems angegeben, und definiert, welche Eigenschaften eine Lösung besitzen muss. Anschließend wird ein relationales Lösungsverfahren entwickelt, mit dem alle Lösungen des Problems berechnet werden können. Zur Verdeutlichung der Problemstellung und des Lösungsverfahrens verwenden wir ein fortlaufendes Beispiel, das aus Gründen der Übersichtlichkeit gegenüber dem eigentlichen Problem verkleinert ist. Wir gehen hierbei von einer Menge $\mathcal{F}$ aus, die nur aus den 10 Schulfächern Mathematik, Deutsch, Englisch, Geschichte, Physik, Chemie, Biologie, Erdkunde Kunst und Sport besteht. Die Mengen $\mathcal{B}$ und $\mathcal{G}$ ist ebenfalls verkleinert, $\mathcal{G}$ besteht in diesem Beispiel aus den Gruppen A, B, C, die zum Block 1 gehören, sowie U, V, W, die dem Block 2 zugeordnet sind, wie in Abbildung 5.4 dargestellt ist.

5.2.2 Formale Problembeschreibung

Ein Stundenplanproblem ist ein Tupel $(\mathcal{F}, \mathcal{G}, \mathcal{B}, b, K)$, wobei $\mathcal{F}, \mathcal{G}, \mathcal{B}$ Mengen, und $b : \mathcal{G} \to \mathcal{B}$ und $K : \mathcal{F} \times \mathcal{F} \to \{0, 1, 2\}$ Abbildungen sind. Die Abbildung K gibt die

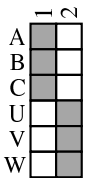


Abbildung 5.4: Die Abbildung b

Wichtigkeit der einzelnen Kombinationen an. In dem durch Abbildung 5.5 gegebenen Beispiel ist z.B. Mathematik/Physik eine wichtige Kombination, und gehört damit in die Kategorie 2. Die Kombination aus Deutsch und Biologie wird als weniger wichtig eingestuft und daher der Kategorie 1 zugeordnet, und Chemie/Englisch bekommt die Kategorie 0, d.h., es handelt sich eine Kombination, die als nicht wichtig angesehen wird. Eine Lösung des Problems $(\mathcal{F}, \mathcal{G}, \mathcal{B}, b, K)$, also ein zulässiger Stundenplan, ist

	Ma	De	En	Ge	Ph	Che	Bio	Er	Ku	Sp
Ma	0	0	1	1	2	2	0	0	0	0
De	0	0	2	1	0	0	0	1	0	1
En	1	2	0	0	1	0	1	0	0	1
Ge	1	1	0	0	0	0	1	2	2	0
Ph	2	0	1	0	0	2	1	0	0	1
Che	2	0	0	0	2	0	0	1	0	0
Bio	0	0	1	1	1	0	0	2	0	2
Er	0	1	0	2	0	1	2	0	1	0
Ku	0	0	0	2	0	0	0	1	0	0
Sp	0	1	2	0	1	0	2	0	0	0

Abbildung 5.5: Die Abbildung K

eine Abbildung $S : \mathcal{F} \to \mathcal{G}$ mit den folgenden Eigenschaften:

(L1) $\forall f, f' \in \mathcal{F} : S(f) = S(f') \to K(f, f') = 0$

(L2) $\forall f, f' \in \mathcal{F} : K(f, f') = 2 \to b(S(f)) = b(S(f'))$

Die erste Eigenschaft (L1) sagt aus, dass innerhalb einer Gruppe nur Fächerkombinationen der Kategorie 0 vorkommen. Damit sind die Kombinationen der Kategorie 1 immer in verschiedenen Gruppen, und damit (zumindest eingeschränkt) studierbar. Die zweite Eigenschaft sagt aus, dass Fächerkombinationen der Kategorie 2 immer in Gruppen desselben Bocks eingeteilt werden. Da sie wegen Eigenschaft (L1) nicht in derselben Gruppe liegen können, hat dies zur Folge, dass diese wichtigen Fächerkombinationen optimal studierbar sind.

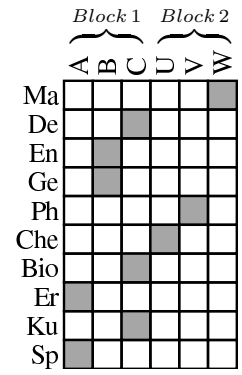


Abbildung 5.6: Eine Lösung S des Stundenplanproblems

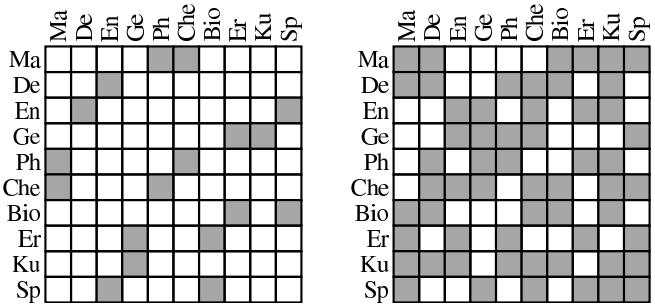


Abbildung 5.7: Die Relationen J und N

5.2.3 Relationale Modellierung

Seien die Mengen $\mathcal{F}, \mathcal{G}$ und $\mathcal{B}$, sowie die Relation $b : \mathcal{G} \leftrightarrow \mathcal{B}$ als relationale Version der Abbildung b gegeben. Die Abbildung K modellieren wir durch zwei Relationen N und J zur Spezifikation der Fächerkombinationen der Kategorien 0 und 2. Seien also $J, N : \mathcal{F} \leftrightarrow \mathcal{F}$ definiert durch die folgenden Äquivalenzen:

$$J_{ff'} \iff K(f, f') = 2 \qquad \text{und} \qquad N_{ff'} \iff K(f, f') = 0$$

Insbesondere sind J und N symmetrisch, N ist reflexiv und J irreflexiv. Außerdem gilt $J \cap N = \mathsf{O}$. Offensichtlich sind die Fächerkombinationen der Kategorie 1 durch die Relation $\overline{J \cup N}$ spezifiziert. Abbildung 5.7 zeigt die Relationen J und N, die sich aus der in Abbildung 5.5 dargestellten Abbildung K ergeben. Wir definieren außerdem eine Relation B, die für zwei Gruppen angibt, ob sie dem selben Block zugeordnet sind. Sei $B : \mathcal{G} \leftrightarrow \mathcal{G}$ durch $B = bb^\top$ gegeben, so gilt für alle $g, g' \in \mathcal{G}$:

$$b(g) = b(g') \iff \exists i : b_{gi} \wedge b_{g'i} \iff (bb^\top)_{gg'} \iff B_{gg'}.$$

Insbesondere ist B reflexiv und symmetrisch. Durch die Relationen J, N und B ist das

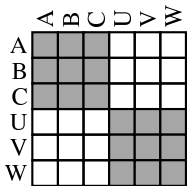


Abbildung 5.8: Die Relation B

Stundenplanproblem $(\mathcal{F}, \mathcal{G}, \mathcal{B}, b, K)$ vollständig definiert und wir können nun mithilfe relationaler Ausdrücke die Eigenschaften formulieren, die eine Lösung des Problems erfüllen muss.

Eine Relation $S : \mathcal{F} \leftrightarrow \mathcal{G}$ ist genau dann eine Lösung des durch die Relationen J, N und B gegebenen Stundenplanproblems, wenn sie die folgenden Eigenschaften (L1) - (L4) erfüllt.

(L1) $\overline{N}S \subseteq \overline{S}$

(L2) $JS \subseteq \overline{S\overline{B}}$

(L3) S ist eindeutig

(L4) S ist total

Da als Lösung des Stundenplanproblems eine Abbildung gefordert ist, ist klar, dass die Relation S eindeutig und total sein muss. Die beiden ersten Inklusionen entsprechen genau den in Abschnitt 5.2.2 angegebenen prädikatenlogischen Formeln (L1) und (L2), wie im Folgenden gezeigt wird. Zu (L1): Nach der Definition von N gilt

$$\begin{aligned}
&\forall f, f' \in \mathcal{F} : S(f) = S(f') \rightarrow K(f, f') = 0 \\
&\iff \forall f, f' \in \mathcal{F}, g \in \mathcal{G} : S_{fg} \ \wedge \ S_{f'g} \rightarrow N_{ff'} \\
&\iff \neg\exists f, f' \in \mathcal{F}, g \in \mathcal{G} : S_{fg} \ \wedge \ S_{f'g} \ \wedge \ \overline{N}_{ff'} \\
&\iff \neg\exists f \in \mathcal{F}, g \in \mathcal{G} : (\overline{N}S)_{fg} \ \wedge \ S_{fg} \\
&\iff \forall f \in \mathcal{F}, g \in \mathcal{G} : (\overline{N}S)_{fg} \rightarrow \overline{S}_{fg} \\
&\iff \overline{N}S \subseteq \overline{S}.
\end{aligned}$$

Zu (L2): Nach den Definitionen von J und B gilt

$$\begin{aligned}
&\forall f, f' \in \mathcal{F} : K(f, f') = 2 \rightarrow b(S(f)) = b(S(f')) \\
&\iff \forall f, f \in \mathcal{F}, g, g' \in \mathcal{G} : J_{ff'} \ \wedge \ S_{fg} \ \wedge \ S_{f'g'} \rightarrow b(g) = b(g')
\end{aligned}$$

$$
\begin{aligned}
&\iff \forall f, f \in \mathcal{F}', g, g' \in \mathcal{G} : J_{ff'} \ \wedge \ S_{fg} \ \wedge \ S_{f'g'} \to B_{gg'} \\
&\iff \neg\exists f, f' \in \mathcal{F}, g, g' \in \mathcal{G} : J_{ff'} \ \wedge \ S_{fg} \ \wedge \ S_{f'g'} \ \wedge \ \overline{B}_{gg'} \\
&\iff \neg\exists f \in \mathcal{F}, g' \in \mathcal{G} : (JS)_{fg'} \ \wedge \ (S\overline{B})_{fg'} \\
&\iff \forall f \in \mathcal{F}, g' \in \mathcal{G} : (JS)_{fg'} \to \overline{(S\overline{B})}_{fg'} \\
&\iff JS \subseteq \overline{S\overline{B}}.
\end{aligned}
$$

Damit ist das Stundenplanproblem $(\mathcal{F}, \mathcal{G}, \mathcal{B}, b, K)$ in ein äquivalentes Relationales Stundenplanproblem (J, N, B) überführt. Im Folgenden wird ein relationales Lösungsverfahren für das Problem angegeben.

5.2.4 Entwicklung eines Vektorprädikats

In diesem Abschnitt wird ein Vektorprädikat entwickelt, das, auf einen Vektor des Typs $[\mathcal{F} \times \mathcal{G} \leftrightarrow \mathbf{1}]$ angewandt, bestimmt, ob die dem Vektor zugrundeliegende Relation eine Lösung des Stundenplanproblems ist. Zunächst werden die Eigenschaften (L1) und (L2) einer Relation als Eigenschaften ihrer Vektorrepräsentation angegeben. Seien also eine Relation $S : \mathcal{F} \leftrightarrow \mathcal{G}$ und ihre Vektor-Darstellung $s = vec(S)$ gegeben. Dann sind die Forderungen (L1) und (L2) äquivalent zu den folgenden Forderungen:

(L1) $(\overline{N} \| \mathsf{I}) s \subseteq \overline{s}$

(L2) $(J \| \mathsf{I}) s \subseteq \overline{(\mathsf{I} \| \overline{B}) s}$

Für den Beweis verwenden wir Eigenschaften der Abbildung *vec*, und insbesondere Theorem 4.1.10. Zu (L1):

$$
\begin{aligned}
\overline{N}S \subseteq \overline{S} &\iff vec(\overline{N}S) \subseteq vec(\overline{S}) && \text{(Monotonie von } vec\text{)} \\
&\iff (\overline{N} \| \mathsf{I}) s \subseteq \overline{s} && \text{(Theorem 4.1.10)}
\end{aligned}
$$

Zu (L2):

$$
\begin{aligned}
JS \subseteq \overline{S\overline{B}} &\iff vec(JS) \subseteq vec(\overline{S\overline{B}}) && \text{(Monotonie von } vec\text{)} \\
&\iff vec(JS) \subseteq \overline{vec(S\overline{B})} && \text{(Abschnitt 2.4)} \\
&\iff (J \| \mathsf{I}) s \subseteq \overline{(\mathsf{I} \| \overline{B}^{\top}) s} && \text{(Theorem 4.1.10)} \\
&\iff (J \| \mathsf{I}) s \subseteq \overline{(\mathsf{I} \| \overline{B}) s} && (B \text{ ist symmetrisch})
\end{aligned}
$$

Aus den Inklusionen (L1) und (L2) ergeben sich mithilfe von Lemma 3.2.3.1 sofort die Vektorprädikate $\varphi_{(L1)}$ und $\varphi_{(L2)} : [\mathcal{F} \times \mathcal{G} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ mit

$$\varphi_{(L1)}(s) = \overline{\mathsf{L}((\overline{N}\|\mathsf{I})s \cap s)}$$

$$\varphi_{(L2)}(s) = \overline{\mathsf{L}((J\|\mathsf{I})s \cap (\mathsf{I}\|\overline{B})s)}.$$

Zusammen mit den Vektorprädikaten $\varphi_{(L3)} = \varphi_{eind}$ und $\varphi_{(L4)} = \varphi_{to}$ für den Test auf Eindeutigkeit bzw. Totalität aus Abschnitt 5.1.2 ergibt sich das Vektorprädikat $\varphi_{st} : [\mathcal{F} \times \mathcal{G} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$, das definiert ist durch

$$\varphi_{st}(s) = \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup (((\mathsf{I}\|\bar{\mathsf{I}}) \cup (\overline{N}\|\mathsf{I}))s \cap s) \cup ((J\|\mathsf{I})s \cap (\mathsf{I}\|\overline{B})s))}.$$

Wie im Folgenden bewiesen wird, ist $rel(s)$ genau dann eine Lösung des durch J, N und B gegebenen Stundenplanproblems, wenn $\varphi_{st}(s) = \mathsf{L}$ gilt. Nach den vorangegangenen Überlegungen erhalten wir

$$rel(s) \text{ ist eine Lösung } \iff \varphi_{(L1)} \cap \varphi_{(L2)} \cap \varphi_{(L3)} \cap \varphi_{(L4)} = \mathsf{L},$$

und es ist

$$\begin{aligned}
&\varphi_{(L1)} \cap \varphi_{(L2)} \cap \varphi_{(L3)} \cap \varphi_{(L4)} \\
&= \overline{\mathsf{L}((\overline{N}\|\mathsf{I})s \cap s)} \cap \overline{\mathsf{L}((J\|\mathsf{I})s \cap (\mathsf{I}\|\overline{B})s)} \cap \overline{\mathsf{L}((\mathsf{I}\|\bar{\mathsf{I}})s \cap s)} \cap \overline{\mathsf{L}\overline{\pi^\top s}} \\
&= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\mathsf{I}\|\bar{\mathsf{I}})s \cap s) \cup ((\overline{N}\|\mathsf{I})s \cap s) \cup ((J\|\mathsf{I})s \cap (\mathsf{I}\|\overline{B})s))} \\
&= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup (((\mathsf{I}\|\bar{\mathsf{I}}) \cup (\overline{N}\|\mathsf{I}))s \cap s) \cup ((J\|\mathsf{I})s \cap (\mathsf{I}\|\overline{B})s))}.
\end{aligned}$$

Da offensichtlich $\varphi_{st} \in \mathcal{VP}$ gilt, kann in der in Abschnitt 3.2.2 angegebenen Weise unmittelbar eine Testabbildung φ_{st}^Z für jede Menge Z abgeleitet werden. Da das konkrete Stundenplanproblem mit 34 Fächern und 9 Gruppen noch relativ klein ist, kann es mithilfe der Potenzmengenrelation $\mathsf{M} : \mathcal{F} \times \mathcal{G} \leftrightarrow 2^{\mathcal{F}\times\mathcal{G}}$ exakt gelöst werden. Das heißt, wir berechnen $\varphi_{st}^{2^{\mathcal{F}\times\mathcal{G}}}(\mathsf{M})$ und erhalten einen Zeilenvektor des Typs $[\mathbf{1} \leftrightarrow 2^{\mathcal{F}\times\mathcal{G}}]$, der alle Lösungen des Stundenplanproblems repräsentiert. Es gilt also

$$\varphi_{st}^{2^{\mathcal{F}\times\mathcal{G}}}(\mathsf{M})_{\perp i} \iff rel(\mathsf{M}^{(i)}) \text{ ist eine Lösung des Stundenplanproblems.}$$

Dieses Vorgehen hat insbesondere den Vorteil, dass im Fall von $\varphi_{st}^{2^{\mathcal{F}\times\mathcal{G}}}(\mathsf{M}) = \mathsf{O}$ zweifelsfrei feststeht, dass für das Stundenplanproblem keine Lösung existiert. Gibt es jedoch Lösungen, so erhält man eine einzelne davon, indem man einen Punkt $p \subseteq \varphi_{st}^{2^{\mathcal{F}\times\mathcal{G}}}(\mathsf{M})^\top$ wählt und $rel(\mathsf{M}p)$ berechnet. Selbstverständlich kann man statt M auch beliebige Listen von Vektoren des Typs $[\mathcal{F} \times \mathcal{G} \leftrightarrow \mathbf{1}]$ verwenden und innerhalb dieser mit einer entsprechenden Testabbildung φ_{st}^Z nach Lösungen suchen. Zu unserem durch die Abbildungen 5.4 und 5.5 gegebenenen Beispiel erhält man durch das exakte Verfahren einen

Vektor der Länge 2^{60} mit genau 144 Einträgen. Es gibt für dieses Problem also 144 Lösungen. Der folgende Abschnitt beschäftigt sich mit der weiteren Bewertung solcher Mengen von Lösungen.

5.2.5 Isomorphe Lösungen

Erhält man aus einer Liste $M : \mathcal{F} \times \mathcal{G} \leftrightarrow [1..n]$ von n Vektoren durch die Anwendung der Testabbildung $\varphi_{st}^{[1..n]}$ eine Anzahl von Lösungen für das durch J, N und B gegebene Stundenplanproblem, können diese von unterschiedlicher Qualität sein. Beispielsweise ist wünschenswert, dass möglichst viele der Fächerkombinationen der Kategorie 1 überschneidungsfrei studierbar sind. Man muss also die einzelnen Lösungen einer zusätzlichen Bewertung unterziehen, um den bestmöglichen Stundenplan unter ihnen zu bestimmen. Dabei ergibt sich das Problem, dass zu jeder Lösung eine große Menge von sogenannten *isomorphen* und damit gleichwertigen Lösungen auftritt, so dass es ausreichend ist, eine von diesen zu betrachten und den Rest zu verwerfen. Unter der Isomorphie zweier Lösungen verstehen wir in diesem Zusammenhang, dass die Mengen der studierbaren und der eingeschränkt studierbaren Fächerkombinationen der beiden Lösungen übereinstimmen. Um die berechneten Lösungen effizienter bewerten und vergleichen zu können, wird im olgenden ein Verfahren vorgestellt, mit dem zu einer Lösung S die Liste aller zu S isomorphen Lösungen berechnet wird. Mithlife dieser Liste werden aus dem Zeilenvektor $\varphi_{st}^{[1..n]}(M)$, der alle Lösungen in M repräsentiert, diejenigen gelöscht, die zu S isomorph sind. Durch sukzessive Anwendung dieses Verfahrens wird die Anzahl der Lösungen im allgemeinen stark verringert, was eine effizientere Bewertung zur Folge hat. Für die formale Definition der Isomorpie zweier Lösungen entwickeln wir zunächst zwei relationale Ausdrücke, die zu einem Stundenplan die Menge der studierbaren, bzw. eingeschränkt studierbaren Fächerkombinationen angeben. Studierbar ist eine Kombination zweier Fächer genau dann, wenn sich die Veranstaltungen der beiden Fächern nie überschneiden, was gleichbedeutend damit ist, dass die Fächer verschiedenen Gruppen desselben Blocks zugeodnet sind. Eingeschränkte Studierbarkeit einer Kombination bedeutet, dass die Fächer in Gruppen verschiedener Blöcke liegen, und es damit zum Teil zu Überschneidungen zwischen den Veranstaltung kommt.

5.2.5.1 Definition *Zu einer Lösung S des Stundenplanproblems (J, N, B) seien die Relationen $sb(S)$ und $esb(S)$ des Typs $[\mathcal{F} \leftrightarrow \mathcal{F}]$ definiert durch*

$$sb(S) = S(B \cap \overline{\mathsf{I}})S^\top \quad \textit{und} \quad esb(S) = S\overline{B}S^\top.$$

Das folgende Lemma erklärt die Bedeutung der Relationen $sb(S)$ und $esb(S)$.

5.2.5.2 Lemma *Für alle Fächer $f, f' \in \mathcal{F}$ gelten die folgenden Äquivalenzen.*

1. $sb(S)_{ff'} \Longleftrightarrow$ *f und f' sind studierbar*

2. $esb(S)_{ff'} \Longleftrightarrow$ *f und f' sind eingeschränkt studierbar*

Beweis: *Für zwei Fächer $f, f' \in \mathcal{F}$ gilt*

$$\begin{aligned} f \text{ und } f' \text{ sind studierbar} &\Longleftrightarrow \exists g, g' : S_{fg} \ \wedge \ S_{f'g'} \ \wedge \ b(g) = b(g') \ \wedge \ g \neq g' \\ &\Longleftrightarrow \exists g, g' : S_{fg} \ \wedge \ S_{f'g'} \ \wedge \ (B \cap \bar{\mathsf{I}})_{gg'} \\ &\Longleftrightarrow (S(B \cap \bar{\mathsf{I}})S^{\top})_{ff'} \\ &\Longleftrightarrow sb(S)_{ff'}. \end{aligned}$$

Für die zweite Ausage folgt

$$\begin{aligned} f \text{ und } f' \text{ sind eingeschränkt studierbar} & \\ &\Longleftrightarrow \exists g, g' : S_{fg} \ \wedge \ S_{f'g'} \ \wedge \ b(g) \neq b(g') \\ &\Longleftrightarrow \exists g, g' : S_{fg} \ \wedge \ S_{f'g'} \ \wedge \ \overline{B}_{gg'} \\ &\Longleftrightarrow (S\overline{B}S^{\top})_{ff'} \\ &\Longleftrightarrow esb(S)_{ff'}. \end{aligned}$$

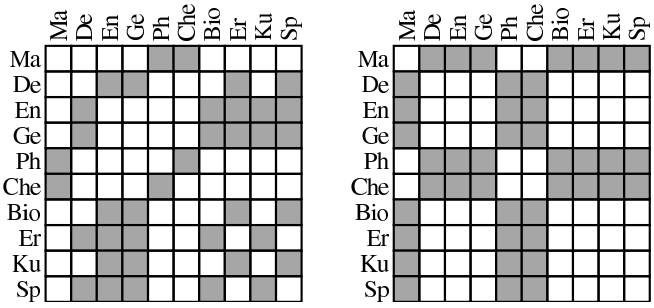


Abbildung 5.9: $sb(S)$ und $esb(S)$

Abbildung 5.9 zeigt die studierbaren bzw. eingeschränkt studierbaren Fächerkombinationen der in Abbildung 5.6 dargestellten Lösung. Offensichtlich gilt $J \subseteq sb(S)$ und $\overline{J \cup N} \subseteq esb(S) \cup sb(S)$ für jede Lösung S, da alle Kombinationen der Kategorie 2 studierbar, und alle Kombinationen der Kategorie 1 mindestens eingeschränkt studierbar sind. Für die weitergehende Bewertung der Qualität einer Lösung S spielen die Relationen $sb(S)$ und $esb(S)$ eine zentrale Rolle. Hierbei sind insbesondere die Relationen $sb(S) \cap \overline{J \cup N}$, $sb(S) \cap N$ und $esb(S) \cap N$ interessant. Die Relation $sb(S) \cap \overline{J \cup N}$ gibt

die Fächerkombinationen der Kategorie 1 an, die mit dem Stundenplan S überschneidungsfrei studierbar sind, $sb(S) \cap N$ spezifiziert die überschneidungsfrei studierbaren Fächerkombinationen der Kategorie 0, und $esb(S) \cap N$ gibt die Fächerkombinationen an, die ebenfalls zur Kategorie 0 gehören, und mit S eingeschränkt studierbar sind. Des Weiteren ist die Relation $\overline{sb(S) \cup esb(S)}$ von Bedeutung, denn es gilt:

$$\begin{aligned}\overline{(sb(S) \cup esb(S)} &= \overline{S(B \cap \overline{\mathsf{I}})S^\top \cup S\overline{B}S^\top} \\ &= \overline{S\overline{\mathsf{I}}S^\top} \\ &= SS^\top \qquad (S \text{ Funktion})\end{aligned}$$

Damit folgt aus $\overline{(sb(S) \cup esbS)}_{ff'}$, dass die Fächer f und f' im Stundenplan S in die gleiche Gruppe gehören. Mithilfe der Relationen $sb(S)$ und $esb(S)$ kann nun auch die Isomorphie zweier Lösungen formal definiert werden.

5.2.5.3 Definition *Zwei Lösungen S und S' des Stundenplanproblems (J, N, B) heißen isomorph ($S \cong S'$) genau dann wenn $sb(S) = sb(S')$ und $esb(S) = esb(S')$ gilt.*

Offensichtlich können die Relationen $sb(S)$ und $esb(S)$ sowie der Begriff der Isomorphie analog für beliebige Relationen S des Typs $[\mathcal{F} \leftrightarrow \mathcal{G}]$ definiert werden. Für das weitere Vorgehen muss geklärt werden, ob jede zu einer Lösung isomorphe Relation ebenfalls eine Lösung ist.

5.2.5.4 Lemma *Für eine Lösung S und eine beliebige Relation $S' : \mathcal{F} \leftrightarrow \mathcal{G}$ folgt aus der Isomorphie $S \cong S'$, dass S' eindeutig ist und die Bedingungen (L1) und (L2) erfüllt.*

Beweis: *Sei S eine Lösung und S' eine beliebige Relation des Typs $[\mathcal{F} \leftrightarrow \mathcal{G}]$. Zum Beweis der Eindeutigkeit der Relation S' zeigt man zunächst die Äquivalenz zwischen der Eindeutigkeit einer beliebigen Relation R des Typs $[\mathcal{F} \leftrightarrow \mathcal{G}]$ und der Irreflexivität der Relationen $sb(R)$ und $esb(R)$. Sei zunächst R eindeutig. Offensichtlich folgt mit den Schröder-Äquivalenzen aus $\mathsf{I}R \subseteq R$ die Inklusion $\overline{R}R^\top \subseteq \overline{\mathsf{I}}$. Außerdem gilt $R\overline{\mathsf{I}} \subseteq \overline{R}$ wegen der Eindeutigkeit von R. Es folgt*

$$\begin{aligned}sb(R) &= R(B \cap \overline{\mathsf{I}})R^\top \\ &\subseteq R\overline{\mathsf{I}}R^\top \\ &\subseteq \overline{R}R^\top \\ &\subseteq \overline{\mathsf{I}}.\end{aligned}$$

Also ist $sb(R)$ irreflexiv. Wegen der Reflexivität von B gilt $\overline{B} \subseteq \bar{\mathsf{I}}$ und es folgt analog zu oben die Irreflexivität von $esb(R)$ durch

$$\begin{aligned} esb(R) &= R\overline{B}R^\top \\ &\subseteq R\bar{\mathsf{I}}R^\top \\ &\subseteq \overline{R}R^\top \\ &\subseteq \bar{\mathsf{I}}. \end{aligned}$$

Seien nun $sb(R)$ und $esb(R)$ irreflexiv. Wegen $\overline{B} \subseteq \bar{\mathsf{I}}$ gilt $\overline{B} \cup \bar{\mathsf{I}} = \bar{\mathsf{I}}$ und damit

$$\overline{B} \cup (\bar{\mathsf{I}} \cap B) = (\overline{B} \cup \bar{\mathsf{I}}) \cap (\overline{B} \cup B) = \bar{\mathsf{I}} \cap \mathsf{L} = \bar{\mathsf{I}}.$$

Es folgt

$$\begin{aligned} \bar{\mathsf{I}} &\supseteq esb(R) \cup sb(R) \\ &= R\overline{B}R^\top \cup R(B \cap \bar{\mathsf{I}})R^\top \\ &= R(\overline{B} \cup (B \cap \bar{\mathsf{I}}))R^\top \\ &= R\bar{\mathsf{I}}R^\top. \end{aligned}$$

Mit den Schröder-Äquivalenzen folgt aus $R\bar{\mathsf{I}}R^\top \subseteq \bar{\mathsf{I}}$ sofort $\bar{\mathsf{I}}R^\top \subseteq \overline{R}^\top$ und damit durch Transposition $R\bar{\mathsf{I}} \subseteq \overline{R}$. Die Relation R ist also eindeutig. Mit der eben gezeigten Äquivalenz lässt sich die Eindeutigkeit von S' jetzt einfach zeigen. Da wegen der Isomorphie der Relationen S und S' die Gleichung $sb(S) \cup esb(S) = sb(S') \cup esb(S')$ gilt, folgt sofort

$$\begin{aligned} S \text{ ist eindeutig} &\iff sb(S) \cup esb(S) \subseteq \bar{\mathsf{I}} \\ &\iff sb(S') \cup esb(S') \subseteq \bar{\mathsf{I}} \\ &\iff S' \text{ ist eindeutig.} \end{aligned}$$

Die Eindeutigkeit von S' wird verwendet, um die Eigenschaften (L1) und (L2) zu beweisen. Zu (L1): Wie schon gezeigt, gilt für Funktionen R die Gleichung $\overline{sb(R) \cup esb(R)} = RR^\top$. Für eindeutige Relationen gilt jedoch nur die Inklusion $RR^\top \subseteq \overline{sb(R) \cup esb(R)}$, wie sich leicht zeigen lässt. Da S eine Lösung ist, ist S insbesondere eine Funktion mit der Eigenschaft (L1). Es folgt:

$$\begin{aligned} \overline{N}S \subseteq \overline{S} &\iff SS^\top \subseteq N && (\textit{Schröder}) \\ &\iff \overline{sb(S) \cup esb(S)} \subseteq N && (S \textit{ Funktion}) \\ &\iff \overline{sb(S') \cup esb(S')} \subseteq N && (S \cong S') \end{aligned}$$

$$\Longrightarrow S'S'^{\top} \subseteq N \qquad (S' \textit{ eindeutig})$$
$$\Longleftrightarrow \overline{N}S' \subseteq \overline{S'} \qquad (\textit{Schröder})$$

Die Relation S' erfüllt also ebenfalls die Eigenschaft (L1). Zu (L2): Da S eine Lösung ist, erfüllt S insbesondere (L2). Es folgt

$$\begin{aligned}
JS \subseteq \overline{S\overline{B}} &\Longleftrightarrow JS \subseteq SB && (\overline{S\overline{B}} = SB, \textit{ da } S \textit{ Funktion})\\
&\Longrightarrow JSS^{\top} \subseteq SBS^{\top} &&\\
&\Longrightarrow J \subseteq SBS^{\top} && (S \textit{ ist total})\\
&\Longleftrightarrow J \subseteq S(B \cap \bar{\mathsf{I}})S^{\top} \cup S(B \cap \mathsf{I})S^{\top} &&\\
&\Longleftrightarrow J \subseteq S(B \cap \bar{\mathsf{I}})S^{\top} \cup SS^{\top} && (\mathsf{I} \subseteq B)
\end{aligned}$$

Wegen $J \cap N = \mathsf{O}$ und $SS^{\top} \subseteq N$ gilt $J \cap SS^{\top} = \mathsf{O}$ und es folgt $J \subseteq S(B \cap \bar{\mathsf{I}})S^{\top}$. Es gilt weiter:

$$\begin{aligned}
J \subseteq S(B \cap \bar{\mathsf{I}})S^{\top} &\Longleftrightarrow J \subseteq sb(S) &&\\
&\Longleftrightarrow J \subseteq sb(S') && (S \cong S')\\
&\Longleftrightarrow J \subseteq S'(B \cap \bar{\mathsf{I}})S'^{\top} &&\\
&\Longrightarrow J \subseteq S'BS'^{\top} &&\\
&\Longrightarrow JS' \subseteq S'BS'^{\top}S' &&\\
&\Longrightarrow JS' \subseteq S'B && (S' \textit{ eindeutig})\\
&\Longrightarrow JS' \subseteq \overline{S'\overline{B}} && (S'B \subseteq \overline{S'\overline{B}} \textit{ da } S' \textit{eindeutig})
\end{aligned}$$

Somit erfüllt S' ebenfalls die Forderung (L2).

Die Totalität von S' kann nicht ohne weiteres gefolgert werden, hier treten Spezialfälle auf, in denen totale und nicht-totale Relationen isomorph sind. Das einfachste Beispiel ist der Fall, dass J und N leere Relationen sind. Dann ist eine Relation S, in der alle Fächer der gleichen Gruppe zugeordnet sind, eine Lösung mit $sb(S) = esb(S) = \mathsf{O}$ und jede Relation S' mit $|\mathsf{L}S'| = 1$ ist zu S isomorph, wobei L vom Typ $[\mathbf{1} \leftrightarrow \mathcal{F}]$ vorausgesetzt sei.

Zu jeder Lösung S können isomorphe Lösungen durch die Multiplikation mit Permutationen des Typs $\mathcal{G} \leftrightarrow \mathcal{G}$ erzeugt werden, wobei an die Permutationen spezielle Anforderungen zu stellen sind. Gilt beispielsweise S_{fA} und $S_{f'B}$, und man erzeugt S' durch Vertauschung der Spalten A und U, so sind f und f' in S' nur noch eingeschränkt studierbar, während sie in der Lösung S studierbar sind. Die Permutation P muss also

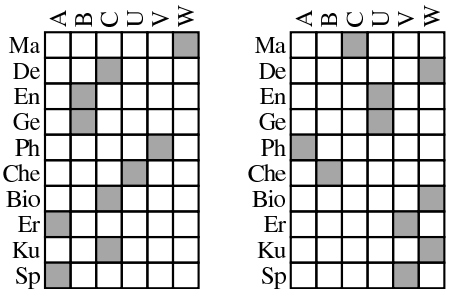


Abbildung 5.10: Zwei isomorphe Lösungen

in gewissem Sinne *blockerhaltend* sein, so dass zwei Gruppen, die sich im selben Block befinden, diese Eigenschaft auch nach Anwendung der Permutation behalten.

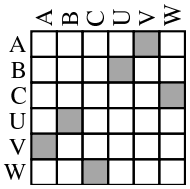


Abbildung 5.11: Eine blockerhaltende Permutation

5.2.5.5 Definition *Sei $P : \mathcal{G} \leftrightarrow \mathcal{G}$ eine Permutation. Wir nennen P blockerhaltend, falls die Inklusion*

$$B \subseteq PBP^\top$$

erfüllt ist.

Verwendet man zur Verdeutlichung die Abbildung b, die der Relation B zugrundeliegt, kann man die Eigenschaft der Blockerhaltung einer Permutation P durch die prädikatenlogische Formel

$$\forall g_1, g_2 \in \mathcal{G} : b(g_1) = b(g_2) \rightarrow b(P(g_1)) = b(P(g_2))$$

ausdrücken, wie die folgende Herleitung zeigt.

$$\begin{aligned}
&\forall g_1, g_2 : b(g_1) = b(g_2) \rightarrow b(P(g_1)) = b(P(g_2)) \\
&\iff \neg\exists g_1, g_2, g_1', g_2' : B_{g_1 g_2} \wedge P_{g_1 g_1'} \wedge P_{g_2 g_2'} \wedge \overline{B}_{g_1' g_2'} \\
&\iff \neg\exists g_1', g_2' : (P^\top BP)_{g_1' g_2'} \wedge \overline{B}_{g_1' g_2'} \\
&\iff \forall g_1', g_2' : (P^\top BP)_{g_1' g_2'} \rightarrow B_{g_1' g_2'} \\
&\iff P^\top BP \subseteq B \\
&\iff B \subseteq PBP^\top \qquad (P \text{ Permutation})
\end{aligned}$$

Um den Zusammenhang zwischen isomorphen Lösungen und blockerhaltenden Permutationen zu verdeutlichen, benötigen wir das folgende Lemma.

5.2.5.6 Lemma *Für jede blockerhaltende Permutation P gilt sogar*

$$B = PBP^\top.$$

Beweis: *Zum Beweis verwenden wir die Endlichkeit der Mengen $\mathcal{G}$ und $\mathcal{B}$. Für jede Permutation P gilt offensichtlich $|B| = |BP| = |PB|$, also insbesondere, da $P^\top$ ebenfalls eine Permutation ist, auch $|B| = |PBP^\top|$. Da für eine blockerhaltende Permutation die Inklusion $B \subseteq PBP^\top$ gilt, folgt mit $|B| = |PBP^\top|$ sofort $B = PBP^\top$.*

Im Folgenden wird gezeigt, dass durch die Multiplikation einer Lösung S des Stundenplanproblems mit einer blockerhaltenden Permutation eine zu S isomorphe Lösung erzeugt wird.

5.2.5.7 Satz *Für jede Lösung S des Stundenplanproblems (J, N, B) und jede blockerhaltende Permutation P ist die Relation SP ebenfalls eine Lösung für (J, N, B) und es gilt $S \cong SP$.*

Beweis: *Wir zeigen zunächst die Gleichheit der eingeschränkt studierbaren Kombinationen von S und SP, indem wir Lemma 5.2.5.6 verwenden. Aus $B = PBP^\top$ folgt mit Abschnitt 2.2*

$$\overline{B} = \overline{PBP^\top} = P\overline{B}P^\top$$

und damit

$$esb(SP) = SP\overline{B}P^\top S^\top = S\overline{B}S^\top = esb(S).$$

Da P eine Permutation ist, gilt, wegen $PP^\top = \mathsf{I}$, außerdem

$$\overline{\mathsf{I}} = \overline{PP^\top} = \overline{P\mathsf{I}P^\top} = P\overline{\mathsf{I}}P^\top$$

und es folgt

$$\begin{aligned} sb(SP) &= SP(B \cap \overline{\mathsf{I}})P^\top S^\top \\ &= S(PBP^\top \cap P\overline{\mathsf{I}}P^\top)S^\top && (\textit{Abschnitt 2.2}) \\ &= S(B \cap \overline{\mathsf{I}})S^\top \\ &= sb(S), \end{aligned}$$

wobei im zweiten Schritt die Eindeutigkeit von P verwendet wird. SP ist also isomorph zu S und damit, nach Lemma 5.2.5.4, eine eindeutige Relation, die (L1) und (L2)

erfüllt. Da aus der Totalität von S und P sofort auch die Totalität von SP folgt, ist SP eine Lösung von (J, N, B).

Man erhält also durch Multiplikation einer Lösung S mit einer blockerhaltenden Permutation eine zu S isomorphe Lösung. Im Folgenden wird gezeigt, dass man jede zu S isomorphe Lösung als Produkt von S mit einer blockerhaltenden Permutation darstellen kann.

5.2.5.8 Satz *Seien S und S' isomorphe Lösungen. Dann gibt es eine blockerhaltende Permutation P, so dass $S' = SP$ gilt.*

Beweis: *Für jede Lösung S sei $\mathcal{G}_1(S)$ die durch den Vektor $S^\top \mathsf{L}$ repräsentierte Teilmenge von $\mathcal{G}$, d.h., $\mathcal{G}_1(S) = \{g \mid (S^\top \mathsf{L})_g\} = \{g \mid \exists f : S_{fg}\}$. Seien nun S und S' isomorphe Lösungen. Dann gilt*

(1) $\forall g \in \mathcal{G}_1(S) \exists g' \in \mathcal{G}_1(S') \forall f : S_{fg} \leftrightarrow S' f g'$.
Beweis: Sei $g \in \mathcal{G}_1(S)$, dann existert ein $f \in \mathcal{F}$ mit S_{fg}. Da S' total und eindeutig ist, gibt es genau ein $g' \in \mathcal{G}$ mit $S'_{fg'}$. Sei nun ein beliebiges $\tilde{f} \in \mathcal{F}$ gegeben. Es ist die Äquivalenz $S_{\tilde{f}g} \leftrightarrow S'_{\tilde{f}g'}$ zu zeigen. Es gelte also zunächst $S_{\tilde{f}g}$. Dann gilt $SS^\top_{f\tilde{f}}$. Wegen der Isomorphie von S und S' gilt

$$SS^\top = \overline{(sb(S) \cup esb(S))} = \overline{(sb(S') \cup esb(S'))} = S'S'^\top$$

und damit auch $S'S'^\top_{f\tilde{f}}$. Da S' eindeutig ist, folgt $S'_{\tilde{f}g'}$. Die verbleibende Richtung der Äquivalenz zeigt man analog.

Da es für jedes $g \in \mathcal{G}_1(S)$ genau ein $g' \in \mathcal{G}_1(S')$, mit der oben erwähnten Eigenschaft gibt, können wir eine Abbildung $\psi : \mathcal{G}_1(S) \to \mathcal{G}_1(S')$ durch $g \mapsto g'$ definieren. ψ ist offensichtlich bijektiv. Außerdem ist ψ blockerhaltend, d.h:

(2) $\forall g, \tilde{g} \in \mathcal{G}_1(S) : b(g) = b(\tilde{g}) \to b(\psi(g)) = b(\psi(\tilde{g}))$
Beweis: Für $g = \tilde{g}$ gilt die Aussage offensichtlich. Seien also g und $\tilde{g}$ aus $\mathcal{G}_1(S)$ mit $g \neq \tilde{g}$ und $b(g) = b(\tilde{g})$ gegeben. Dann gilt $(B \cap \bar{\mathsf{I}})_{g\tilde{g}}$. Wegen $g, \tilde{g} \in \mathcal{G}_1(S)$ gibt es $f, \tilde{f}$ mit S_{fg} und $S_{\tilde{f}\tilde{g}}$ und es folgt $sb(S)_{f\tilde{f}}$ aus $(B \cap \bar{\mathsf{I}})_{g\tilde{g}}$ Damit gilt auch $sb(S')_{f\tilde{f}}$ und, wegen $S'_{f\psi(g)}$ und $S'_{\tilde{f}\psi(\tilde{g})}$, folgt $b(\psi(g)) = b(\psi(\tilde{g}))$.

Wir können also eine weitere Abbildung $\alpha : b(\mathcal{G}_1(S)) \to b(\mathcal{G}_1(S'))$ durch $\alpha(b(g)) = b(\psi(g))$ definieren. α ist offensichtlich bijektiv und damit existiert eine bijektive Abbildung $\beta : \mathcal{B} \leftrightarrow \mathcal{B}$ mit $\beta_{|b(\mathcal{G}_1(S))} = \alpha$. Sei nun $i \in \mathcal{B}$. Dann gilt

(3) $|\mathcal{G}_1(S) \cap b^{-1}(i)| = |\mathcal{G}_1(S') \cap b^{-1}(\beta(i))|$

Beweis: Sei $g \in \mathcal{G}_1(S) \cap b^{-1}(i)$*. Dann ist* $\psi(g) \in \mathcal{G}_1(S')$*, und aus* $b(g) = i$ *folgt* $b(\psi(g)) = \alpha(b(g)) = \alpha(i) = \beta(i)$*, also* $\psi(g) \in b^{-1}(\beta(i))$*. Es gilt also* $\psi(g) \in \mathcal{G}_1(S') \cap b^{-1}(\beta(i))$ *und damit* $\psi(\mathcal{G}_1(S) \cap b^{-1}(i)) \subseteq \mathcal{G}_1(S') \cap b^{-1}(\beta(i))$*. Wegen der Bijektivität von* ψ *folgt* $|\mathcal{G}_1(S) \cap b^{-1}(i)| = |\psi(\mathcal{G}_1(S) \cap b^{-1}(i))| \leq |\mathcal{G}_1(S') \cap b^{-1}(\beta(i))|$*. Durch Vertauschung von* S *und* S' *sowie Verwendung der inversen Abbildungen von* ψ *und* α *erhält man die Behauptung.*

Es folgt sofort, dass $|(\mathcal{G} \setminus \mathcal{G}_1(S)) \cap b^{-1}(i)| = |(\mathcal{G} \setminus \mathcal{G}_1(S')) \cap b^{-1}(\beta(i))|$ *für alle* $i \in \mathcal{B}$ *gilt. Also existiert für jedes* $i \in \mathcal{B}$ *eine Bijektion*

$$\psi^{(i)} : (\mathcal{G} \setminus \mathcal{G}_1(S)) \cap b^{-1}(i) \to (\mathcal{G} \setminus \mathcal{G}_1(S')) \cap b^{-1}(\beta(i))$$

Damit gibt es auch eine Permutation $P : \mathcal{G} \leftrightarrow \mathcal{G}$ *mit* $P_{|\mathcal{G}_1(S)} = \psi$ *und* $P_{|(\mathcal{G} \setminus \mathcal{G}_1(S)) \cap b^{-1}(i)} = \psi^{(i)}$ *für alle* $i \in \mathcal{B}$*. Die Permutation* P *ist nach Konstruktion blockerhaltend. Schließlich gilt noch:*

(4) $S' = SP$

Beweis: Sei $(SP)_{fg'}$*. Dann gibt es ein* $g \in \mathcal{G}$ *mit* S_{fg} *und* $P_{gg'}$*. Wegen* $g \in \mathcal{G}_1(S)$ *gilt* $\psi(g) = g'$ *und es folgt* $S'_{fg'}$ *nach Definition von* ψ*. Es gilt also* $SP \subseteq S'$*, und da sowohl* S' *als auch* SP *eindeutig und total sind, folgt* $SP = S'$*.*

Da hier eine Permutation konstruiert werden muss, und dabei insbesondere auf Mächtigkeitsargumente zurückgegriffen wird, scheint es nicht ohne weiteres möglich zu sein, den Beweis dieser Aussage in der abstrakten Relationenalgebra durchzuführen.

Nach den beiden vorangegangenen Sätzen sind zwei Lösungen S und S' also genau dann isomorph, wenn es eine blockerhaltende Permutation P mit $S' = SP$ gibt. Um für eine gegebene Lösung S die Menge aller zu S isomorphen Lösungen zu bestimmen, reicht es also aus, die Menge $\mathcal{P}$ aller blockerhaltenden Permutationen auf $\mathcal{G}$ zu bestimmen und $\{SP \mid P \in \mathcal{P}\}$ zu berechnen. Zunächst wird die Menge aller blockerhaltenden Permutationen in Form einer Relation mit Urbildbereich $\mathcal{G} \times \mathcal{G}$ bestimmt. Dafür wird im Folgenden ein Vektorprädikat φ_{bp} entwickelt, das für einen Vektor des Typs $[\mathcal{G} \times \mathcal{G} \leftrightarrow \mathbf{1}]$ entscheidet, ob die zugrundeliegende Relation eine blockerhaltende Permutation ist. φ_{bp} kann dann zu einer Testabbildung $\varphi_{bp}^{2^{\mathcal{G}\times\mathcal{G}}}$ erweitert werden, die, angewandt auf die Potenzmengenrelation $\mathsf{M} : \mathcal{G} \times \mathcal{G} \leftrightarrow 2^{\mathcal{G}\times\mathcal{G}}$, die Menge aller Spalten in M berechnet, die Vektordarstellungen von blockerhaltende Permutationen sind. Zunächst wird ein Vektorprädikat φ_{perm} benötigt, um zu bestimmen, ob die zugrundeliegende Relation $P : \mathcal{G} \leftrightarrow \mathcal{G}$ eines Vektors $p : \mathcal{G} \times \mathcal{G} \leftrightarrow \mathbf{1}$ eine Permutation ist. Hierbei verwenden wir die

in Abschnitt 4.2 entwickelten Vektorprädikate für Totalität, Eindeutigkeit, Injektivität und Surjektivität. Es gilt:

$$\begin{aligned}
\varphi_{perm}(p) &= \varphi_{to}(p) \cap \varphi_{sur}(p) \cap \varphi_{eind}(p) \cap \varphi_{inj}(p) \\
&= \overline{\mathsf{L}\overline{\pi^\top p}} \cap \overline{\mathsf{L}\overline{\rho^\top p}} \cap \overline{\mathsf{L}((\mathsf{I}\|\bar{\mathsf{I}})p \cap p)} \cap \overline{\mathsf{L}((\bar{\mathsf{I}}\|\mathsf{I})p \cap p)} \\
&= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top p} \cup \mathsf{L}\overline{\rho^\top p} \cup ((\mathsf{I}\|\bar{\mathsf{I}})p \cap p) \cup ((\bar{\mathsf{I}}\|\mathsf{I})p \cap p))} \\
&= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top p} \cup \mathsf{L}\overline{\rho^\top p} \cup (p \cap ((\mathsf{I}\|\bar{\mathsf{I}}) \cup (\bar{\mathsf{I}}\|\mathsf{I}))p))} \\
&= \overline{\mathsf{L}(\mathsf{L}(\overline{\pi^\top p \cap \rho^\top p}) \cup (p \cap ((\mathsf{I}\|\bar{\mathsf{I}}) \cup (\bar{\mathsf{I}}\|\mathsf{I}))p))}
\end{aligned}$$

Sei nun P eine Permutation des Typs $[\mathcal{G} \leftrightarrow \mathcal{G}]$ und $p = vec(P)$ die Vektordarstellung von P. Dann gilt:

$$\begin{aligned}
P \text{ ist blockerhaltend} &\iff B \subseteq PBP^\top \\
&\iff BP \subseteq PB && (P \text{ Funktion}) \\
&\iff B\overline{PB} \subseteq \overline{P} && (\text{Schröder, } B = B^\top) \\
&\iff BP\overline{B} \subseteq \overline{P} && (P \text{ Funktion}) \\
&\iff vec(BP\overline{B}) \subseteq vec(\overline{P}) && (vec \text{ monoton}) \\
&\iff (B\|\overline{B}^\top)vec(P) \subseteq \overline{vec(P)} && (\text{Theorem 4.1.10}) \\
&\iff (B\|\overline{B})p \subseteq \overline{p} && (B \text{ symmetrisch}) \\
&\iff \overline{\mathsf{L}(p \cap (B\|\overline{B})p)} = \mathsf{L} && (\text{Lemma 3.2.3.1})
\end{aligned}$$

Wir erhalten also das Vektorprädikat $\varphi_{be} : [\ G \times \mathcal{G} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$, definiert durch

$$\varphi_{be}(p) = \overline{\mathsf{L}(p \cap (B\|\overline{B})p)},$$

das für die Vektordarstellung einer Permutation P testet, ob P blockerhaltend ist. Durch $\varphi_{bp} = \varphi_{perm} \cap \varphi_{be}$ erhalten wir das Vektorprädikat $\varphi_{bp} : [\mathcal{G} \times \mathcal{G} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$, definiert durch

$$\varphi_{bp}(p) = \overline{\mathsf{L}(\mathsf{L}(\overline{\pi^\top p \cap \rho^\top p}) \cup (p \cap ((\mathsf{I}\|\bar{\mathsf{I}}) \cup (\bar{\mathsf{I}}\|\mathsf{I}) \cup (B\|\overline{B}))p))},$$

so dass $\varphi_{bp}(p) = \mathsf{L}$ genau dann gilt, wenn p eine blockerhaltende Permutation repräsentiert. Das Vektorprädikat kann nun zur Berechnung der Menge aller blockerhaltenden Permutationen verwendet werden. Sei dazu $\mathsf{M} : \mathcal{G} \times \mathcal{G} \leftrightarrow 2^{\mathcal{G}\times\mathcal{G}}$ die Potenzmengenrelation von $\mathcal{G} \times \mathcal{G}$. Die Relation M modelliert damit die Menge aller Relationen des Typs $[\mathcal{G} \leftrightarrow \mathcal{G}]$ in Vektordarstellung, und der Zeilenvektor $\varphi_{bp}^{2^{\mathcal{G}\times\mathcal{G}}}(\mathsf{M}) : \mathbf{1} \leftrightarrow 2^{\mathcal{G}\times\mathcal{G}}$ spezifiziert

alle blockerhaltenden Permutationen. Wie in Abschnitt 2.3 erörtert, erhalten wir durch

$$L = \mathsf{M}\, inj(\varphi_{bp}^{2^{\mathcal{G}\times\mathcal{G}}}(\mathsf{M})^\top)^\top$$

die Liste aller blockerhaltenden Permutationen auf $\mathcal{G}$ in Vektordarstellung. Mittels der Relation L kann nun für eine Lösung S die Liste I_S aller zu S isomorphen Lösungen berechnet werden.

5.2.5.9 Satz *Sei S eine Lösung von (J, N, B). Dann ist*

$$I_S := (S\|\mathsf{I})L$$

die Liste aller zu S isomorphen Lösungen.

Beweis: *Sei $P : \mathcal{G} \leftrightarrow \mathcal{G}$ eine blockerhaltende Permution und $p = vec(P)$. Dann ist nach Lemma 5.2.5.7 die Relation $S' := SP$ eine zu S isomorphe Lösung und nach Theorem 4.1.10 ist*

$$s' = vec(S') = vec(SP) = (S\|\mathsf{I})vec(P) = (S\|\mathsf{I})p$$

die Vektordarstellung von S'. Da die Abbildung $p \mapsto (S\|\mathsf{I})p$ offensichtlich ein Element der Menge $\mathcal{VA}$ (siehe Abschnitt 3.2.2) ist, und alle Spalten der Relation L Vektordarstellungen von blockerhaltenden Permutionen sind, ist $(S\|\mathsf{I})L$ eine Liste von Lösungen, welche zu S isomorph sind. Nach Lemma 5.2.5.8 existiert zu jeder Lösung S' mit $S' \cong S$ eine blockerhaltende Permutation P mit $S' = SP$. Da L die Liste aller blockerhaltender Permutationen ist, gibt es ein j mit $L^{(j)} = vec(P)$, und damit gilt

$$vec(S') = (S\|\mathsf{I})vec(P) = (S\|\mathsf{I})L^{(j)} = ((S\|\mathsf{I})L)^{(j)} = I_S^{(j)}.$$

Es gibt also eine Spalte in I_S, welche die Relation S' repräsentiert. Somit ist I_S die Liste aller zu S isomorphen Lösungen.

Die Relation I_S wird im Folgenden verwendet, um aus einer Menge von Lösungen diejenigen herauszufiltern, welche zu S isomorph sind. Sei also eine Relation $M : \mathcal{F} \times \mathcal{G} \leftrightarrow [1..n]$ und der Zeilenvektor $v = \varphi_{st}^{[1..n]}(M) : \mathbf{1} \leftrightarrow [1..n]$ gegeben, der die Lösungen des Stundenplanproblems (J, N, B) innerhalb von M spezifiziert. Für jeden Punkt $p \subseteq v^\top$ erhält man eine Lösung $S : \mathcal{F} \leftrightarrow \mathcal{G}$ durch $S = rel(Mp)$ und die Liste I_S der zu S isomorphen Lösungen durch $I_S = (S\|\mathsf{I})L$. Der Vektor

$$w = syq(M, I_S)\mathsf{L}$$

spezifiert dann die Menge aller Lösungen in M, welche zu S isomorph sind, d.h es gilt

$$w_i \Longleftrightarrow rel(M^{(i)}) \cong S,$$

denn

$$\begin{aligned}
w_i &\Longleftrightarrow syq(M, I_S)\mathsf{L}_i \\
&\Longleftrightarrow \exists j : syq(M, I_S)_{ij} \\
&\Longleftrightarrow \exists j : \forall x : M_{xi} \leftrightarrow S_{xj} && \text{(Abschnitt 2.5)} \\
&\Longleftrightarrow \exists j : M^{(i)} = I_S^{(j)} \\
&\Longleftrightarrow rel(M^{(i)}) \cong S && \text{(Satz 5.2.5.9)}
\end{aligned}$$

Insbesondere gilt also $w^\top \subseteq v$. Mithilfe des Vektors w können nun durch $(v \cap \overline{w^\top}) \cup p^\top$ alle zu S isomorphen Lösungen (außer S selber) aus v entfernt werden Durch sukzessive Anwendung dieses Verfahrens (siehe Anhang D.1) erhält man eine Menge von paarweise nicht isomorphen Lösungen. Bei unserem durch die Abbildungen 5.4 und 5.5 gegebenen Beispiel bestehen die zunächst ermittelten 144 Lösungen aus nur 2 Isomorphieklassen mit jeweils 72 Elementen. Abbildung 5.12 zeigt je einen Repräsentanten jeder Isomorphieklasse. Die beiden Lösungen S und $\tilde{S}$ unterscheiden sich beispielsweise

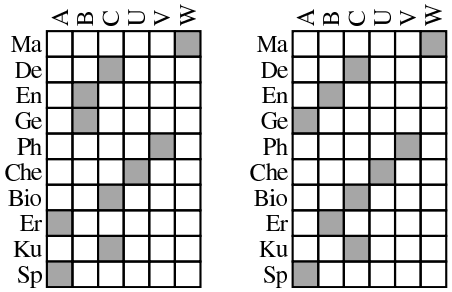


Abbildung 5.12: Zwei nicht isomorphe Lösungen S und $\tilde{S}$

darin, dass die Kombination Englisch/Erdkunde mit dem Stundenplan S überschneidungsfrei studierbar ist, da sich sich beide Fächer in verschiedenen Gruppen des ersten Blocks befinden. Im Stundenplan $\tilde{S}$ sind Englisch und Erdkunde der gleichen Gruppe zugeordnet und damit nicht gemeinsam studierbar. Genau umgekehrt verhält es sich mit der Kombination aus den Fächern Englisch und Geschichte.

Aufgrund organisatorischer Schwierigkeiten wurde das hier erörterte Stundenplanmodell für die Einführung des Bachelor-Studiengangs an der Universität Kiel wieder verworfen und ein einfacheres Modell entwickelt, welches im nächsten Abschnitt diskutiert

wird.

5.3 Abgewandeltes Universitäres Stundenplanproblem

In diesem Abschnitt wird das einfachere Modell behandelt, das bei der Einführung des eingangs erwähnten Bachelor-Studiengangs in Betracht gezogen und schließlich auch verwendet wurde. Dieses Modell ist wesentlich unkomplizierter als das im vorangehenden Abschnitt vorgestellte, und ähnelt eher dem eingangs behandelten klassischen Stundenplanproblem. Wie im vorangehenden Abschnitt soll auch hier wieder eine Menge von Fächern auf Zeitschienen verteilt werden. Hier spielen jedoch nur zwei Kategorien bezüglich der Kombinierbarkeit zweier Fächer eine Rolle, entweder eine Kombination ist wichtig, dann soll sie überschneidungsfrei studierbar sein, oder sie ist nicht wichtig, dann sind Überschneidungen ohne Einschränkungen erlaubt. Man geht hier von 4 zur Verfügung stehenden Zeitschienen aus, auf welche die einzelnen Fächer aufgeteilt werden sollen. Dabei gibt es einige Fächer, die aufgrund größeren Zeitaufwandes zwei Zeitschienen benötigen. Aufbauend auf eine relationale Beschreibung des Problems wird auch hier ein Vektorprädikat zur Berechnung von Stundenplänen entwickelt. Die hier vorgestellten Ergebnisse wurde in einer stark gekürzten Fassung bereits in [10] veröffentlicht.

5.3.1 Relationale Modellierung

Sei $\mathcal{F}$ die Menge der Fächer und $\mathcal{T}$ die Menge der Zeitschienen. Es stehen vier disjunkte Zeitschienen zur Verfügung, in welche die Fächer eingeordnet werden sollen. Dabei gibt es jedoch einige Fächer, die wegen eines höheren Zeitaufwandes zwei Zeitschienen benötigen. Um das Problem besser relational modellieren zu können, gehen wir von einer Menge $\mathcal{T} = \{z_1, \ldots, z_6\}$ von 6 Zeitschienen aus, wobei z_1, z_2, z_3, z_4 überschneidungsfrei sind und z_5 die Vereinigung von z_1 und z_2 darstellt, sowie z_6 die Vereinigung von z_3 und z_4. Diese Modellierung erfordert die Berücksichtigung von Verfügbarkeiten und Überschneidungen, wie im Folgenden erläutert wird. Das Stundenplanproblem ist durch drei Relationen

- $K : \mathcal{F} \leftrightarrow \mathcal{F}$,
- $A : \mathcal{F} \leftrightarrow \mathcal{T}$
- $Z : \mathcal{T} \leftrightarrow \mathcal{T}$

gegeben. K beschreibt die wichtigen Kombinationen, d.h es gilt

$$K_{ff'} \iff f \text{ und } f' \text{ müssen überschneidungsfrei studierbar sein.}$$

Die Relation A beschreibt, welche Zeitschienen für jedes Fach zur Verfügung stehen. Dies ist insbesondere wichtig für Fächer, die viel Zeit beanspruchen und daher nur in die großen Zeitschienen z_5 und z_6 eingeordnet werden können. Es gilt also

$$A_{fz} \iff f \text{ kann in die Zeitschiene } z \text{ eingeordnet werden.}$$

Die Modellierung mit den 6 Zeitschienen erfordert es, die Überschneidungen zwischen den Zeitschienen zu berücksichtigen. Dies wird durch die Relation Z realisiert, es gilt also

$$Z_{zz'} \iff z \text{ und } z' \text{ überschneiden sich.}$$

Insbesondere ist Z reflexiv und symmetrisch. Eine Lösung des durch K, A und Z ge-

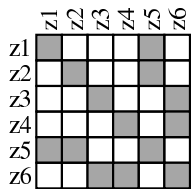


Abbildung 5.13: Die Relation Z

gebenen Stundenplanproblems ist eine Relation $S : \mathcal{F} \leftrightarrow \mathcal{T}$ mit den folgenden Eigenschaften:

(L1) $\forall f, z : S_{fz} \rightarrow A_{fz}$

(L2) $\neg\exists f, f', z, z' : K_{ff'} \wedge S_{fz} \wedge S_{f'z'} \wedge Z_{zz'}$

(L3) S ist total

(L4) S ist eindeutig

Die erste Formel beschreibt, dass jedes Fach nur in eine der durch A festgelegten geeigneten Zeitschiene eingeordnet werden kann. (L2) sagt aus, dass zwei Fächer, die kombinierbar sein sollen, nicht in Zeitschienen eingeordnet werden können, die sich überschneiden. Insbesondere können solche Fächer also nicht in die gleiche Zeitschiene eingeordnet werden. Da Eindeutigkeit und Totalität bereits in Abschnitt 4.2 behandelt wurden, konzentrieren wir uns im Folgenden auf die Eigenschaften (L1) und (L2), welche durch die folgenden relationalen Inklusionen ausgedrückt werden können.

(L1) $S \subseteq A$

(L2) $KSZ \subseteq \overline{S}$

Wir beweisen nur die Inklusion (L2). Es gilt:

$$\begin{aligned}\neg\exists f, f', z, z' : K_{ff'} \wedge S_{fz} \wedge S_{f'z'} \wedge Z_{z'z} &\iff \neg\exists f, z : (KSZ)_{fz} \wedge S_{fz} \\ &\iff \forall f, z : (KSZ)_{fz} \to \overline{S}_{fz} \\ &\iff KSZ \subseteq \overline{S}\end{aligned}$$

5.3.2 Entwicklung eines Vektorprädikats

Da in Abschnitt 5.1 bereits ein Vektorprädikat enwickelt wurde, das die Inklusion (L1) beschreibt, beschränken wir uns im Folgenden auf (L2). Zunächst gilt die Äquivalenz

$$KSZ \subseteq \overline{S} \iff (K \| Z)s \subseteq \overline{s}$$

für $s = vec(S)$, denn mit Theorem 4.1.10 erhält man

$$\begin{aligned}KSZ \subseteq \overline{S} &\iff (K \| Z^\top) vec(S) \subseteq vec(\overline{S}) \\ &\iff (K \| Z)s \subseteq \overline{s}\end{aligned}$$

Wir erhalten das Vektorprädikat $\varphi_{(L2)} : [\mathcal{F} \times \mathcal{T} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$ mit $\varphi_{(L2)}(s) = \overline{\mathsf{L}((K \| Z)s \cap s)}$ durch Lemma 3.2.3.1. Zusammen mit $\varphi_{(L1)}, \varphi_{(L3)}$ und $\varphi_{(L4)}$ aus Abschnitt 5.1.2 ergibt sich das Vektorprädikat $\varphi_{st} : [\mathcal{F} \times \mathcal{T} \leftrightarrow \mathbf{1}] \to [\mathbf{1} \leftrightarrow \mathbf{1}]$, definiert durch

$$\varphi_{st}(s) = \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup ((K \| Z) \cup (\mathsf{I} \| \bar{\mathsf{I}}))s) \cap s))},$$

denn es gilt

$$\begin{aligned}&s \text{ ist eine Lösung des Stundenplanproblems} \\ \iff\ &\varphi_{(L1)}(s) \cap \varphi_{(L2)}(s) \cap \varphi_{(L3)}(s) \cap \varphi_{(L4)}(s)\end{aligned}$$

und

$$\begin{aligned}&\varphi_{(L1)}(s) \cap \varphi_{(L2)}(s) \cap \varphi_{(L3)}(s) \cap \varphi_{(L4)}(s) \\ &= \overline{\mathsf{L}((K \| Z)s \cap s)} \cap \overline{\mathsf{L}(s \cap \overline{a})} \cap \overline{\mathsf{L}\overline{\pi^\top s}} \cap \overline{\mathsf{L}((\mathsf{I} \| \bar{\mathsf{I}})s \cap s)} \qquad \text{(Abschnitt 4.2 und 5.1.2)} \\ &= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s}) \cup (\overline{a} \cap s) \cup ((K \| Z)s \cap s) \cup ((\mathsf{I} \| \bar{\mathsf{I}})s \cap s))} \\ &= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup (K \| Z)s \cup (\mathsf{I} \| \bar{\mathsf{I}})s) \cap s)}\end{aligned}$$

$$= \overline{\mathsf{L}(\mathsf{L}\overline{\pi^\top s} \cup ((\overline{a} \cup ((K \| Z) \cup (\mathsf{I} \| \bar{\mathsf{I}}))s) \cap s))}$$

Auch hier kann das Vektorprädikat in unterschiedlicher Weise genutzt werden, um Lösungen für das Stundenplanproblem zu ermitteln, beispielsweise in einem exakten Verfahren unter Verwendung der Potenzmengenrelation.

5.3.3 Reduzierung der Problemgröße

In unserem speziellen Fall mit 34 Fächern und 6 Zeitschienen wurde für das exakte Verfahren eine Potenzmengenrelation der Größe 204×2^{204} benötigt, so dass es nicht möglich war, das Problem mittlels RelView in angemessener Zeit exakt zu lösen. Durch Besonderheiten in der Problemstruktur konnte jedoch die Problemgröße soweit reduziert werden, dass die Aufzählung aller Lösungen des Stundenplanproblems ermöglicht wurde. Es stellte sich heraus, dass nur eines der Fächer (Chemie, im Folgenden abgekürzt mit c) mehr als eine Zeitschiene beanspruchte, so dass das Modell mit den 6 Zeitschienen nicht mehr angemessen erschien. Stattdessen wurde das Fach c in zwei Fächer c_1 und c_2 aufgeteilt, die beide einer Zeitschiene zugeordnet werden mussten. Dies führte zu einer veränderten Eingabe (K', A', Z') für das Stundenplanproblem. Die Relation K' ist dabei vom Typ $[\mathcal{F}' \leftrightarrow \mathcal{F}']$ mit $\mathcal{F}' := \mathcal{F} \setminus \{c\} \cup \{c_1, c_2\}$ und es gilt

$$\begin{aligned} K'_{ff'} \Longleftrightarrow & (f, f' \neq c \ \wedge \ K_{ff'}) \ \vee \ (f \in \{c_1, c_2\} \ \wedge \ K_{cf'}) \\ & \vee \ (f' \in \{c_1, c_2\} \ \wedge \ K_{fc}) \ \vee \ (f \neq f' \ \wedge \ f, f' \in \{c_1, c_2\}). \end{aligned}$$

Das heisst, alle wichtigen Kombinationen, in denen Chemie keine Rolle spielt, werden übernommen, alle Fächer, die mit Chemie eine wichtige Kombination bilden, bilden auch mit c_1 und c_2 eine wichtige Kombination und $K'_{c_1c_2}$ stellt sicher, dass c_1 und c_2 in verschiedene Zeitschienen eingeordnet werden, so dass Chemie insgesamt zwei Zeitschienen erhält. Es werden also nur noch 4 Zeitschienen benötigt, und da zwischen ihnen keine Unterschiede existieren, wird die Verfügbarkeitsrelation A zur Allrelation $A' : \mathcal{F}' \leftrightarrow \mathcal{T}'$ mit $\mathcal{T}' = \{t_1, t_2, t_3, t_4\}$. Da keine Überschneidungen mehr zwischen verschiedenen Zeitschienen vorkommen, definieren wir zuletzt die Relation Z' als Identitätsrelation vom Typ $[\mathcal{T}' \leftrightarrow \mathcal{T}']$. Mit diesen Modifizierungen kann das in Abschnitt 5.3.2 entwickelte Vektorprädikat φ_{st} für die exakte Lösung des Stundenplanproblems verwendet werden, wobei sich die Größe der benötigten Potenzmengenrelation wegen $|\mathcal{F}'| = 35$ und $|\mathcal{T}'| = 4$ auf 140×2^{140} reduziert hat.

Durch eine zweite Eigenschaft des gegebenen Problems konnte die Problemgröße weiter reduziert werden. Die 4 romanischen Sprachen Spanisch, Portugisisch, Italienisch und Französisch (im Folgenden als s, p, i, f abgekürzt) bilden eine Clique von wichtigen Kombinationen. Jedes dieser Fächer muss mit den drei anderen kombinierbar sein, so dass die vier Fächer in jedem Fall vier verschiedene Zeitschienen benötigen. Indem man also jedem der Fächer aus $\mathcal{R} = \{s, p, i, f\}$ durch eine injektive Relation $R : \mathcal{R} \leftrightarrow \mathcal{T}$ eine Zeitschiene zuordnet, kann man die Menge der Fächer auf $\mathcal{F}'' := \mathcal{F}' \setminus \mathcal{R}$ reduzieren. Um die wichtige Kombinationen zwischen Fächern aus $\mathcal{R}$ und $\mathcal{F}' \setminus \mathcal{R}$ zu berücksichtigen, wird erneut die Verfügbarkeitsrelation verändert. Ein Fach f kann nicht eine Zeitschiene z eingeordnet werden, falls diese bereits für eine der romanischen Sprachen $r \in \mathcal{R}$ vorgesehen ist, die mit f eine wichtige Kombination bildet. Die neue Verfügbarkeitsrelation A'' wird also definiert durch

$$A''_{fz} \Longleftrightarrow \neg\exists r \in \mathcal{R} : K'_{fr} \ \wedge \ R_{rz}.$$

Zusammen mit $K'' : \mathcal{F}'' \leftrightarrow \mathcal{F}''$, definiert durch

$$K''_{ff'} \Longleftrightarrow K'_{ff'} \ \wedge \ f, f' \notin \mathcal{R}$$

und $Z'' = Z' = \mathsf{I}$ ergeben sich neue Eingaberelationen für das Vektorprädikat φ_{st}. Wegen $|\mathcal{F}''| = 31$ ist damit nur noch eine Potenzmengenrelation der Größe 124×2^{124} nötig, und das Problem damit auf eine moderate Größe reduziert, die es ermöglicht, innerhalb weniger Sekunden mittels RelView alle Lösungen des Stundenplanproblems zu berechnen. Da in den Ausgangsdaten jedoch zu viele wichtige Kombinationen angegeben waren, existierten für dieses Problem zunächst keine Lösungen. Schritt für Schritt mussten also Einträge aus der Relation K'' gelöscht werden, um die Lösbarkeit zu gewährleisten. Für eine zielgerichtete Modifikation wurden vor jedem Schritt unter Verwendung des in Beispiel 3.2.3.3 entwickelten Vektorprädikats φ_{cl} die größten Cliquen von K' berechnet und möglichst Kombinationen entfernt, die in mehreren großen Cliquen vorkommen. Die zunächst 133 wichtigen Kombinationen wurden so auf 119 reduziert, bevor Lösungen für das Stundenplanproblem gefunden werden konnten. Diese modifizierte Eingabe führte zu 32 verschiedenen Stundenplänen, von denen einer ausgewählt wurde, der es ermöglicht, 408 von 561 möglichen Kombinationen überschneidungsfrei zu studieren.

5.4 Spezielle Variationsoperatoren

In jedem der in hier vorgestellten Stundenplanmodelle werden Stundenpläne durch eindeutige und totale Relationen mit speziellen Eigenschaften repräsentiert. Setzt man zur Lösung dieser Probleme Suchheuristiken wie z.B. Evolutionäre Algorithmen ein, besteht ein mögliches Vorgehen darin, möglichst große Teillösungen zu erzeugen, um damit eventuell eine vollständige Lösung für das Problem zu finden. Dazu werden im Selektionsschritt ausschließlich gültige Teillösungen für die nächste Generation ausgewählt, die zusätzlich bezüglich ihrer Größe bewertet werden. Insbesondere ist man also grundsätzlich an eindeutigen Individuen interessiert. Verwendet man in einem Evolutionären Algorithmus randomisierte Variationsoperatoren, wie sie in den Abschnitten 3.3.1 und 3.3.2 eingeführt werden, kann man jedoch davon ausgehen, dass im Laufe des Algorithmus viele nicht eindeutige, und damit überflüssige Suchpunkte erzeugt werden, durch die kein Fortschritt im Suchprozess erzielt wird. Um eine zielgerichtetere Suche zu ermöglichen, können also spezielle Variationsoperatoren verwendet werden, welche die Eindeutigkeit aller generierten Individuen gewährleisten. Das folgende Lemma ist grundlegend für die Entwicklung solcher spezieller Variationsoperatoren.

5.4.1 Lemma *Seien A und B eindeutige Relationen sowie C eine beliebige Relation mit $A\mathsf{L} \subseteq C$. Dann ist die Relation $A \cup (B \cap \overline{C})$ ebenfalls eindeutig.*

Beweis: *Wegen der Eindeutigkeit von A gilt $A\mathsf{I} \subseteq \overline{A}$. Außerdem gilt $A\mathsf{I} \subseteq A\mathsf{L} \subseteq C \subseteq \overline{B} \cup C = \overline{B \cap \overline{C}}$, und damit*

$$A\mathsf{I} \subseteq \overline{A} \cap \overline{B \cap \overline{C}} = \overline{A \cup (B \cap \overline{C})}.$$

Wegen der Eindeutigkeit von B gilt weiter $(B \cap \overline{C})\mathsf{I} \subseteq B\mathsf{I} \subseteq \overline{B} \subseteq \overline{B} \cup C = \overline{B \cap \overline{C}}$. Aus $A\mathsf{I}^\top \subseteq A\mathsf{L} \subseteq C$ folgt mit den Schröder-Äquivalenzen $\overline{C}\,\mathsf{I} \subseteq \overline{A}$ und damit auch $(B \cap \overline{C})\mathsf{I} \subseteq \overline{A}$. Insgesamt gilt also

$$(B \cap \overline{C})\mathsf{I} \subseteq \overline{A} \cap \overline{B \cap \overline{C}} = \overline{A \cup (B \cap \overline{C})},$$

und damit

$$(A \cup (B \cap \overline{C}))\mathsf{I} = A\mathsf{I} \cup (B \cap \overline{C})\mathsf{I} \subseteq \overline{A \cup (B \cap \overline{C})}.$$

Die Relation $A \cup (B \cap \overline{C})$ ist also eindeutig.

Aufbauend auf dieses Lemma entwickeln wir im Folgenden einen Mutationsoperator, welcher die Eindeutigkeit der Individuen erhält. Wir beziehen uns dabei zunächst auf den $(\lambda+\lambda)$-EA, und verwenden die Notationen des Stundenplanmodells aus Abschnitts

5.1. Das heißt insbesondere, dass Suchpunkte Relationen des Typs $[\mathcal{M} \leftrightarrow \mathcal{H}]$ sind, welche als Vektoren des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ modelliert werden. Wir benötigen die folgenden Bezeichnungen.

5.4.2 Definition *Ein Vektor v des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ heiße $\mathcal{M}$-eindeutig, falls $rel(v)$ eindeutig ist. Eine Relation $P : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$ heiße spaltenweise $\mathcal{M}$-eindeutig, falls jede ihrer Spalten $\mathcal{M}$-eindeutig ist, d.h. wenn für jedes $i \in [1..\lambda]$ gilt, dass $rel(P^{(i)})$ eindeutig ist.*

In Abschnitt 4.2 wurde bei der Entwicklung des Vektorprädikats für Totalität bereits gezeigt, dass $vec(R\mathsf{L}) = \pi\pi^\top vec(R)$ für alle Relationen R gilt, sofern R und $R\mathsf{L}$ vom gleichen Typ sind. Wir verwenden diese Tatsache im folgenden Lemma.

5.4.3 Lemma *Seien $P, M : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$ spaltenweise $\mathcal{M}$-eindeutig. Dann ist*

$$M \cup (P \cap \overline{\pi\pi^\top M})$$

ebenfalls eine spaltenweise $\mathcal{M}$-eindeutige Relation.

Beweis: *Seien v und w $\mathcal{M}$-eindeutige Vektoren des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ und $V = rel(v)$ sowie $W = rel(w)$. Somit sind V und W eindeutig und es gilt nach Lemma 5.4.1, dass $V \cup (W \cap \overline{V\mathsf{L}})$ ebenfalls eindeutig ist. Es folgt*

$$\begin{aligned} vec(V \cup (W \cap \overline{V\mathsf{L}})) &= v \cup (w \cap \overline{vec(V\mathsf{L})}) \\ &= v \cup (w \cap \overline{\pi\pi^\top vec(V)}) \\ &= v \cup (w \cap \overline{\pi\pi^\top v}). \end{aligned}$$

Der Vektor $v \cup (w \cap \overline{\pi\pi^\top v})$ ist also $\mathcal{M}$-eindeutig. Es folgt für spaltenweise $\mathcal{M}$-eindeutige P und M, dass $M \cup (P \cap \overline{\pi\pi^\top M})$ ebenfalls spaltenweise $\mathcal{M}$-eindeutig ist, da für jedes $i \in [1..\lambda]$ die Gleichung

$$(M \cup (P \cap \overline{\pi\pi^\top M}))^{(i)} = M^{(i)} \cup (P^{(i)} \cap \overline{\pi\pi^\top M^{(i)}})$$

gilt, und damit jede Spalte von $M \cup (P \cap \overline{\pi\pi^\top M})$ einen $\mathcal{M}$-eindeutigen Vektor darstellt.

Geht man von einer Population aus, die aus λ eindeutigen Individuen besteht, also einer spaltenweise $\mathcal{M}$-eindeutigen Relation $P : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$, so kann man λ eindeutige Nachkommen erzeugen, indem man randomisiert eine spaltenweise $\mathcal{M}$-eindeutige Relation M erzeugt und

$$C = M \cup (P \cap \overline{\pi\pi^\top M})$$

berechnet. Zur Erzeugung der Relation M kann man beispielweise die injektive Einbettung verwenden. Für einen beliebigen Vektor $v : \mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}$ mit $|v| = \lambda$ und eine beliebige Permutation Q ist offensichtlich

$$M := inj(v)^\top Q$$

eine Relation des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]]$, die in jeder Spalte genau einen Eintrag enthält, und damit insbesondere eine spaltenweise $\mathcal{M}$-eindeutigen Relation ist. Bei der Verwendung eines solchen M wird allerdings jedes Individuum von P an höchstens einer Stelle verändert. Um größere Veränderungen zu erzielen, benötigt man eine Menge $\{v_1, \ldots, v_\ell\}$ von zufällig erzeugten Vektoren mit $|v_i| = \lambda$ sowie eine Menge zufällig erzeugter Permutationen $\{Q_1, \ldots, Q_\ell\}$ auf $[1..\lambda]$ und berechnet eine Folge von Relationen M_i durch

$$M_1 := inj(v_1)^\top Q_1$$

$$M_{i+1} := M_i \cup (inj(v_{i+1})^\top Q_{i+1} \cap \overline{\pi\pi^\top M_i}).$$

Alle diese M_i sind nach Konstruktion spaltenweise $\mathcal{M}$-eindeutig und damit auch $M := M_\ell$. Eine solche Relation M enthält bis zu ℓ Einträge pro Spalte und ermöglicht damit bis zu ℓ Veränderungen jedes Individuums aus P. Diese Art der Mutation kann ohne weiteres auf den $(1 + \lambda)$-EA übertragen werden, indem man $P = v\mathsf{L}$ setzt, wobei v das Eltern-Individuum und L vom Typ $[\mathbf{1} \leftrightarrow [1..\lambda]]$ ist.

In Experimenten hat sich gezeigt, dass bessere Ergebnisse erzielt werden, wenn man nur wenige Veränderungen zulässt. Wir verwenden daher das folgende RelView-Programm `uniqueMutation` für die eindeutige Mutation. Dabei ist v ein beliebiger Vektor des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ mit $|v| = \lambda$ und P eine spaltenweise $\mathcal{M}$-eindeutige Relation des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]]$. Den Parameter A, die Verfügbarkeitsrelation, benötigt man nur zur Erzeugung von $\pi : \mathcal{M} \times \mathcal{H} \leftrightarrow \mathcal{M}$.

```
uniqueMutation(P,v,A)
  DECL perm, M, pi, C
  BEG  perm = randomperm(L(v));
       M = inj(perm*v)^;
       M = M * randomperm(L1n(M)^);
       pi = pi(A);
       C = M | (P & -(pi*pi^*M))
       RETURN C
  END.
```

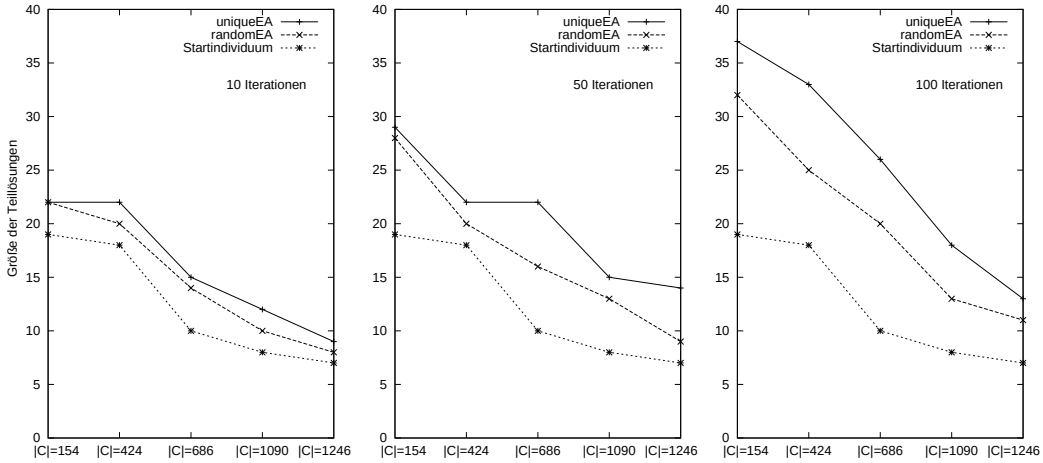


Abbildung 5.14: Vergleich der Mutationsoperatoren

Zunächst wurde dieser Mutationsoperator mit dem in Abschnitt 3.3.1 beschriebenen verglichen, wobei als Mutationswahrscheinlichkeit $p_M = \frac{1}{\mathcal{M} \times \mathcal{H}}$ verwendet wurde. Dafür wurden zwei bis auf den Mutationsoperator identische $(1 + \lambda)$-EAs auf verschiedene Instanzen des Stundenplanproblem angewendet. Die Abbildungen 5.14 und 5.15 zeigen die Ergebnisse für Instanzen mit 40 Meetings und 5 Zeitschienen. Dabei wurde mit Populationen der Größe 20 gearbeitet und nach jeweils 10, 50 oder 100 Iterationen das aktuelle Individuum zurückgegeben. In allen Fällen wurde ein zufälliges Startindividu-

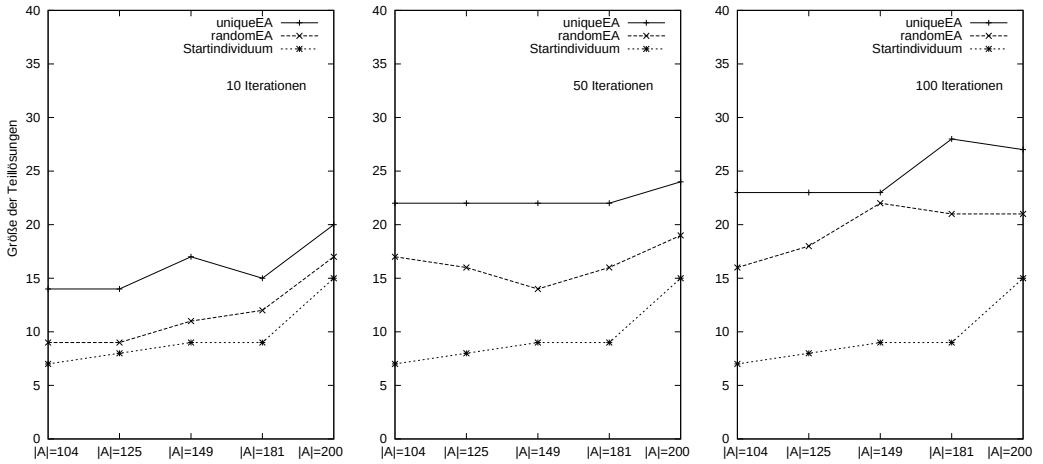


Abbildung 5.15: Vergleich der Mutationsoperatoren

um berechnet, welches eine Teillösung des Problems darstellt. Die genauen Algorithmen für diese Experimente sind in Anhang D.2 zu finden. Der in diesem Abschnitt entwickelte, speziellere Mutationsoperator erwies sich in allen betrachteten Fällen als überlegen, führte also stets zu größeren Teillösungen als die einfache Mutation. Abbildung 5.14 zeigt den Vergleich der Ergebnisse für 5 Instanzen, wobei jedesmal die gleiche Verfügbar-

keitsrelation A mit $|A| = 183$ verwendet, und die Konfliktmatrix C mit verschiedenen Füllgraden variiert wurde. Auf der x-Achse sind die verschiedenen Konfliktrelationen aufgelistet, die y-Achse beschreibt die Mächtigkeit der erzeugten Teillösungen. Man sieht, dass beide Algorithmen die erzeugte Startlösung erheblich verbessern, und die speziellere Art der Mutation die besseren Ergebnisse liefert. Die Überlegenheit dieser Art der Mutation tritt mit zunehmender Anzahl von Iterationen immer deutlicher hervor. Abbildung 5.15 zeigt die Ergebnisse von Experimenten mit einer festen Konfliktrelation C mit 658 Einträgen und verschiedenen Verfügbarkeitselationen A. Auch hier führt der neue Mutationsoperator zu deutlich größeren Teillösungen innerhalb der gleichen Anzahl von Iterationen. Die Laufzeiten des Algorithmus `uniqueEA` lagen einige Sekunden über denen von `randomEA`, insgesamt lagen jedoch alle Laufzeiten unter 30 Sekunden.

Bezüglich des Crossover-Operators stellt sich zunächst das folgende Problem. Erzeugt man eine Nachkommenschaft aus einer spaltenweise $\mathcal{M}$-eindeutigen Population P und einer beliebigen Relation M durch $(P \cap M) \cup (P' \cap \overline{M})$, wobei P' aus P durch Permutation der Spalten entsteht, so ist die resultierende Relation im Allgemeinen nicht spaltenweise $\mathcal{M}$-eindeutig. Man muss also speziellere Relationen verwenden, um die Eindeutigkeit der erzeugten Individuen zu gewährleisten, nämlich eine Relation M, die die Eigenschaft

$$(\exists h : M_{\langle m,h\rangle i}) \to (\forall h : M_{\langle m,h\rangle i})$$

für alle $m \in \mathcal{M}$ und alle $i \in [1..\lambda]$ erfüllt. Das heißt, jede Spalte von M muss eine zeilenkonstante Relation repräsentieren. Das folgende Lemma betrifft die Vektordarstellung zeilenkonstanter Relationen und ist entscheidend für die Erzeugung geeigneter Relationen M.

5.4.4 Lemma *Für $v : \mathcal{M} \leftrightarrow \mathbf{1}$ und $\mathsf{L} : \mathbf{1} \leftrightarrow \mathcal{H}$ gilt*

$$vec(v\mathsf{L}) = \pi v.$$

Beweis: *Es gilt:*

$$\begin{aligned}
vec(v\mathsf{L}) &= (\pi v\mathsf{L} \cap \rho)\mathsf{L} \\
&= \pi v\mathsf{L} \cap \rho\mathsf{L} && (\textit{Abschnitt 2.1}) \\
&= \pi v\mathsf{L} \cap \mathsf{L} && (\rho \textit{ ist total}) \\
&= \pi v\mathsf{L} \\
&= \pi v && (v \textit{ ist ein Vektor})
\end{aligned}$$

Mithilfe dieses Lemmas können wir nun eine geeignete Relation M für die Rekombination erzeugen, indem wir zufällige Relationen $\tilde{M}$ mit π multiplizieren und damit Relationen erhalten, deren Spalten zeilenkonstanten Relationen entsprechen, wie der folgende Satz beschreibt.

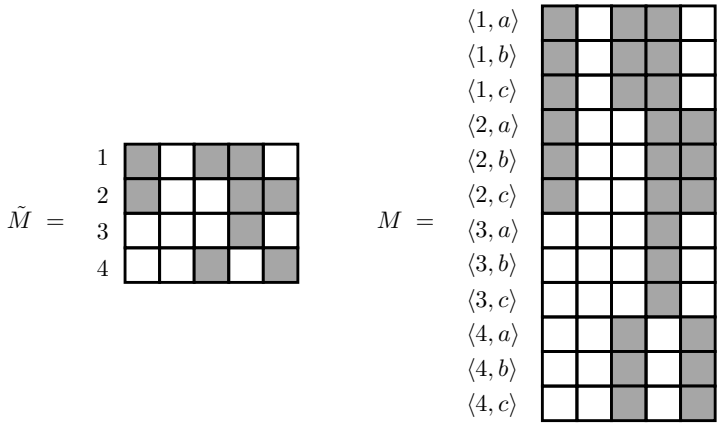


Abbildung 5.16: Die Relationen $\tilde{M}$ und M

5.4.5 Satz *Sei $\tilde{M}$ eine Relation des Typs $[\mathcal{M} \leftrightarrow [1..\lambda]]$ und*

$$M := \pi\tilde{M}.$$

Sei außerdem $P : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$ spaltenweise $\mathcal{M}$-eindeutig und $P' = PQ$, wobei Q eine Permutation auf $[1..\lambda]$ ist. Dann ist

$$C = (P \cap M) \cup (P' \cap \overline{M})$$

ebenfalls $\mathcal{M}$-eindeutig.

Beweis: *Seien zunächst zwei $\mathcal{M}$-eindeutige Vektoren s und s' des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ sowie $v : \mathcal{M} \leftrightarrow \mathbf{1}$ und $\mathsf{L} : \mathbf{1} \leftrightarrow \mathcal{H}$ gegeben. Die Relationen $S = rel(s)$ und $S' = rel(s')$ sind also eindeutig und damit nach Lemma 5.4.1 auch die Relation $(S \cap v\mathsf{L}) \cup (S' \cap \overline{v\mathsf{L}})$. Es folgt mit Lemma 5.4.4*

$$\begin{aligned} vec((S \cap v\mathsf{L}) \cup (S' \cap \overline{v\mathsf{L}})) &= (vec(S) \cap vec(v\mathsf{L})) \cup (vec(S') \cap vec(\overline{v\mathsf{L}})) \\ &= (vec(S) \cap \pi v) \cup (vec(S') \cap \overline{\pi v}) \\ &= (s \cap \pi v) \cup (s \cap \overline{\pi v}), \end{aligned}$$

also ist der Vektor $(s \cap \pi v) \cup (s \cap \overline{\pi v})$ $\mathcal{M}$*-eindeutig. Da für jedes* $i \in [1..\lambda]$

$$\begin{aligned} C^{(i)} &= ((P \cap M) \cup (P' \cap \overline{M}))^{(i)} \\ &= ((P \cap \pi\tilde{M}) \cup (P' \cap \overline{\pi\tilde{M}}))^{(i)} \\ &= (P^{(i)} \cap \pi\tilde{M}^{(i)}) \cup (P'^{(i)} \cap \overline{\pi\tilde{M}^{(i)}}) \end{aligned}$$

gilt, ist also jede Spalte von C $\mathcal{M}$*-eindeutig und damit* C *eine spaltenweise* $\mathcal{M}$*-eindeutige Relation.*

Diese Überlegungen führen sofort zum folgenden RelView-Programm zur Implementierung eines eindeutigkeits-erhaltenden Crossover-Operators.

```
uniqueCrossover(P,A,p)
  DECL Q, m, M, C, L, M1
  BEG L = L1n(P);
      Q = P*randomperm(L^);
      m = random(L,p);
      M1 =  random50(Ln1(A)*L) & Ln1(A)*m;
      M = pi(A)*M1;
      C = (P & M) | (Q & -M)
      RETURN C
  END.
```

Dabei wird, wie in Abschnitt 3.3.2 erklärt, durch den Parameter p die Crossover-Wahrscheinlichkeit p_C festgelegt. Durch verschiedene Experimente wurde untersucht, inwieweit sich der Einsatz eindeutigkeits-erhaltender Variationsoperatoren im $(\lambda + \lambda)$-EA auf die Qualität der erzielten Ergebnisse auswirkt. Dazu wurden zwei Algorithmen verglichen, von denen der eine durch die Verwendung des hier entwickelten Mutations- und Crossover-Operatoren ausschließlich mit Populationen aus eindeutigen Individuen arbeitet, während der andere beliebige Individuen erzeugt, indem Variationsoperatoren aus den Abschnitten 3.3.1 und 3.3.2 eingesetzt werden. Die entsprechenden RelView-Programme sind in Anhang D.2 angegeben. Es zeigte sich in den Experimenten meist nur ein geringer Vorteil für die in diesem Abschnitt entwickelten Operatoren. Insbesondere beim Einsatz von zufällig generierten Startpopulationen waren die erzielten Ergebnisse beider Algorithmen von ähnlicher Qualität. Die Abbildungen 5.17 und 5.18 illustrieren die Ergebnisse von Experimenten auf Instanzen mit 40 Meetings und 5 Zeitschienen, einer Populationsgröße von 20, einer Crossover-Wahrscheinlichkeit von 0.8 und einer Mutations-Wahrscheinlichkeit von $\frac{1}{200}$ im Fall der ursprünglichen Mutati-

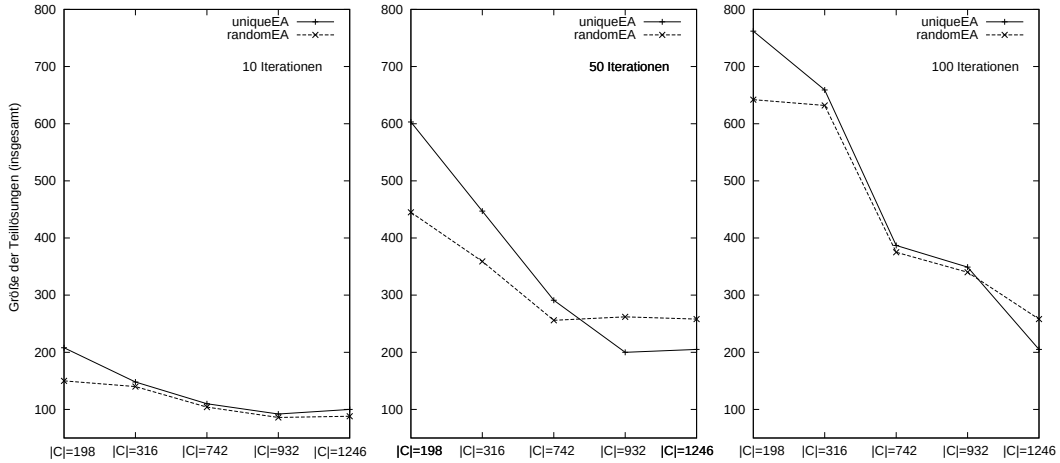


Abbildung 5.17: Vergleich der Variationsoperatoren

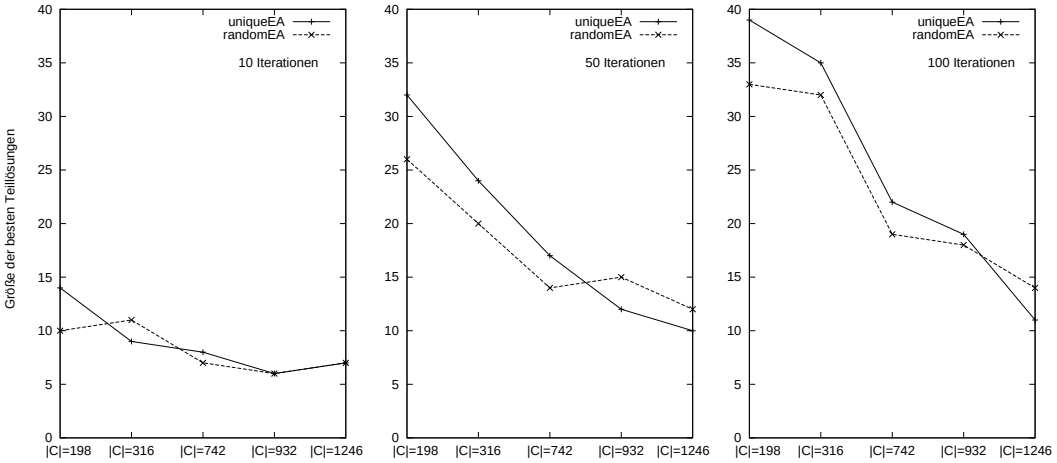


Abbildung 5.18: Vergleich von Variationsoperatoren

on. Wieder wurden die Ergebnisse nach 10, 50 und 100 Iterationen verglichen, wobei als Startpopulation die leere Relation verwendet wurde. Bei fester Verfügbarkeitsrelation A mit $|A| = 183$ und verschiedenen Konfliktrelationen wurden zum einen die erzeugten Populationen (siehe Abbildung 5.17) und zum anderen die besten erzeugten Individuen bezüglich ihrer Größe verglichen (siehe Abbildung 5.18). Enthält die Konfliktrelation weniger Einträge, so erzielt, besonders nach einer höheren Anzahl von Iterationen, der Algorithmus, welcher ausschließlich eindeutige Individuen erzeugt, bessere Ergebnisse also der Algorithmus mit den herkömmlichen Mutations- und Crossover-Operatoren. Bei stärker gefüllten Konfliktrelationen ergibt sich ein gegensätzliches Bild.

6 Anwendung 2: Petrinetze

Das Konzept der Petrinetze geht auf die Dissertation von C. A. Petri [36] zurück und dient und der Modellierung von Vorgängen, Organisationen und Geräten, bei denen geregelte Flüsse eine Rolle spielen. Mit Petrinetzen können Aufbau, Arbeitsweise und Eigenschaften von Systemen beschrieben und kausale Abhängigkeiten innerhalb einer Menge von Ereignissen dargestellt werden. Die Formaliserung durch ein Petrinetz ermöglicht es außerdem, Systemeigenschaften zu erkennen, nachzuweisen und Korrektheitsbeweise zu führen. Die Veranschaulichung durch Petrinetze wird unter anderem in der Softwareentwicklung, der nebenläufigen Programmierung und bei der Verwaltung von Arbeitsabläufen verwendet. Für eine weiterführende Einführung in die Theorie der Petrinetze siehe [38].

Da die Grundlage eines Petrinetzes ein gerichteter Graph ist, liegt es nahe, Petrinetze mit relationalen Methoden zu untersuchen, wie beispielsweise in [11] und [23]. In diesem Kapitel stützen wir uns auf die in [23] angegebene relationale Modellierung von Petrinetzen, überführen jedoch die dort verwendete Darstellung von Markierungen durch Relationen in eine wesentlich praktischere Modellierung durch Vektoren. Theorem 4.1.10 ermöglicht es, Vektorabbildungen zu entwickeln, um beispielsweise Mengen von aktivierten Transitionen oder Nachfolgemarkierungen zu berechnen. Als besonders vorteilhaft erweist sich die Verwendung von Vektoren bei der Berechnung des Zustandsraumes eines Petrinetzes, der aus allen erreichbaren Markierungen besteht. Hier vermeidet man durch die direkte Auswertung von Vektoren ein wiederholtes Umrechnen von Relationen in Vektoren und umgekehrt. Das in diesem Kapitel beschriebene Vorgehen wird ebenfalls in [24] erörtert. Nach der Einführung einiger Grundbegriffe der Petrinetztheorie beschäftigt sich der zweite Abschnitt dieses Kapitels mit der relationalen Modellierung von Petrinetzen. Hier werden Methoden diskutiert, mit denen zu einer Markierung die Menge der durch sie aktivierten Transitionen sowie die Menge der Nachfolgemarkierungen ermittelt werden kann. Auf dieser Grundlage werden Algorithmen zur Berechnung des Zustandsraumes eines Petrinetzes sowie der Erreichbarkeitsrelation zwischen den Zuständen angegeben. Im letzten Abschnitt wird anhand von einigen Beispielen gezeigt, wie die zuvor berechneten Relationen zur relationalen Analyse des

zugrundeliegenden Petrinetzes verwendet werden können.

6.1 Grundlagen

In diesem Abschnitt werden die einzelnen Komponenten eines Petrinetzes definiert und einige Notationen eingeführt. Des Weiteren wird geklärt, unter welchen Umständen eine Transition aktiviert ist und wie durch das Schalten einer aktivierten Transition aus einer Markierung eine Nachfolgemarkierung entsteht. Wir beschränken uns auf Petrinetze mit endlichen Kapazitäten. Ein *Netzgraph* ist ein Tripel $N = (\mathcal{P}, \mathcal{T}, F)$ von Mengen, wobei $F \subseteq (\mathcal{P} \times T) \cup (\mathcal{T} \times \mathcal{P})$ gilt. Die Elemente von $\mathcal{P}$ werden *Stellen*, die von $\mathcal{T}$ *Transitionen* genannt. F ist die sogenannte *Flussrelation*. Wir betrachten ausschließlich endliche Mengen von Transitionen und Stellen. Ein Petrinetz ist ein 6-Tupel $\mathcal{PN} = (N, C, f, M_0)$, wobei

- $N = (\mathcal{P}, \mathcal{T}, F)$ ein Netzgraph,
- $C : \mathcal{P} \to \mathbb{N}$ die Kapazitätsfunktion der Stellen,
- $f : F \to \mathbb{N}$ die Gewichtsfunktion des Flusses,
- $M_0 : \mathcal{P} \to \mathbb{N}_0$ die Startmarkierung, so dass $M_0(p) \leq C(p)$ für alle $p \in \mathcal{P}$ gilt.

Dabei sei $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. Mit $\mathcal{M}(\mathcal{PN})$ bezeichnen wir die Menge aller möglichen Startmarkierungen, also

$$\mathcal{M}(\mathcal{PN}) := \{M \mid M : \mathcal{P} \to \mathbb{N}, \forall p : M(p) \leq C(p)\}.$$

Für alle Knoten $x \in \mathcal{P} \cup \mathcal{T}$ bezeichne

$$\bullet x = \{y \mid (y, x) \in F\} \qquad \text{und} \qquad x\bullet = \{y \mid (x, y) \in F\}.$$

Eine Transition $t \in \mathcal{T}$ heisst *aktiviert* (oder auch *konzessioniert*) unter einer Markierung $M \in \mathcal{M}(\mathcal{PN})$, falls die folgenden beiden Bedingungen gelten:

(A1) $\forall p \in \bullet t : M(p) \geq f(p, t)$

(A2) $\forall p \in t\bullet : M(p) \leq C(p) - f(t, p)$

Eine unter M aktivierte Transition t kann *schalten* (oder auch *feuern*) d.h., sie produziert eine Nachfolgemarkierung M' (in Zeichen $M \overset{t}{\rightsquigarrow} M'$), welche durch die folgende *Schaltregel* definiert ist:

$$M'(p) := \begin{cases} M(p) - f(p,t) & : \ p \in {\bullet t} \setminus t{\bullet} \\ M(p) + f(t,p) & : \ p \in t{\bullet} \setminus {\bullet t} \\ M(p) - f(p,t) + f(t,p) & : \ p \in {\bullet t} \cap t{\bullet} \\ M(p) & : \ sonst \end{cases}$$

M' heisst dann unmittelbare Folgemarkierung von M. Wegen (A1) und (A2) ist M' wieder ein Element der Menge $\mathcal{M}(\mathcal{PN})$. Wir schreiben

$$M \rightsquigarrow M' :\Longleftrightarrow \exists t \in \mathcal{T} : M \overset{t}{\rightsquigarrow} M',$$

durch $\rightsquigarrow$ wird also eine Relation auf $\mathcal{M}(\mathcal{PN})$ definiert. Es bezeichne im Folgenden $M\rightsquigarrow := \{M' \,|\, M \rightsquigarrow M'\}$ die Menge der unmittelbaren Folgemarkierungen von M. Durch den Übergang zur reflexiv-transitiven Hülle $\overset{*}{\rightsquigarrow}$ der Relation $\rightsquigarrow$ erhalten wir die sogenannte *Erreichbarkeitsrelation* und den Begriff der *erreichbaren Markierungen*. Sei $M\overset{*}{\rightsquigarrow} := \{M' \,|\, M \overset{*}{\rightsquigarrow} M'\}$ die Menge der von M erreichbaren Markierungen. Von besonderem Interesse ist hier die Menge $M_0\overset{*}{\rightsquigarrow}$, also die Menge aller von der Startmarkierung erreichbarer Markierungen, welche den Zustandsraum des Petrinetzes definiert. Offensichtlich ist $M_0\overset{*}{\rightsquigarrow}$ eine Zusammenhangskomponente in der Erreichbarkeitsrelation, daher werden die Relationen $\rightsquigarrow$ und $\overset{*}{\rightsquigarrow}$ häufig auf der Menge $M_0\overset{*}{\rightsquigarrow}$ betrachtet.

6.2 Relationale Modellierung von Petrinetzen

Im Folgenden werden Petrinetze mittels einer relationalen Struktur definiert, und Vektorabbildungen entwickelt, mit denen aktivierte Transitionen und Nachfolgemarkierungen berechnet werden. Diese Vektorabbildungen werden verwendet, um die Menge $M_0\overset{*}{\rightsquigarrow}$ in Form einer Relation zu ermitteln und die Erreichbarkeitsrelation auf $M_0\overset{*}{\rightsquigarrow}$ zu berechnen. Dazu benötigen wir zunächst einige weitere Konstanten, beispielsweise zur relationalen Modellierung der Subtraktion und der Addition von natürlichen Zahlen. Für jedes $n \in \mathbb{N}$ seien im Folgenden die Relationen S und A des Typs $[[0..n] \times [0..n] \leftrightarrow [0..n]]$ definiert durch

$$\mathsf{S}_{\langle l,m\rangle k} \Longleftrightarrow l - m = k \qquad \text{und} \qquad \mathsf{A}_{\langle l,m\rangle k} \Longleftrightarrow l + m = k.$$

Es bezeichne außerdem G die Größer-Gleich-Relation auf $[0..n]$, also

$$\mathsf{G}_{km} \Longleftrightarrow k \geq m.$$

Wir beginnen mit der relationalen Modellierung eines Netzgraphen. Ein Netzgraph wird relational durch ein 4-Tupel $(\mathcal{P}, \mathcal{T}, R, S)$ dargestellt, wobei $\mathcal{P} \cap \mathcal{T} = \emptyset$ gilt und R und S Relationen mit $R : \mathcal{P} \leftrightarrow \mathcal{T}$ und $S : \mathcal{T} \leftrightarrow \mathcal{P}$ sind. Diese Struktur entspricht dem in Abschnitt 6.1 definierten Netzgraphen in dem Sinne, dass die Flussrelation F durch die beiden Relationen R und S modelliert wird, d.h. es gilt für alle $p \in \mathcal{P}, t \in \mathcal{T}$:

$$R_{pt} \Longleftrightarrow (p,t) \in F \qquad \text{und} \qquad S_{tp} \Longleftrightarrow (t,p) \in F.$$

Wir beschränken uns im Folgenden auf Petrinetze mit endlichen Kapazitäten. Ein Petrinetz mit maximaler Kapazität $n := max\{C(p) \,|\, p \in \mathcal{P}\} \in \mathbb{N}$ wird durch die Struktur $\mathcal{PN} = (N, C, M_0, V, W)$ beschrieben. Dabei ist $N = (\mathcal{P}, \mathcal{T}, R, S)$ ein Netzgraph im obigen relationalen Sinne, C und M_0 Relationen des Typs $[\mathcal{P} \leftrightarrow [0..n]]$ und V und W Relationen des Typs $[\mathcal{P} \times \mathcal{T} \leftrightarrow [0..n]]$, welche die folgenden Eigenschaften erfüllen.

1. C ist eine Funktion, und es gilt $C\vec{0} = \mathsf{O}$
2. M_0 ist eine Funktion, und es gilt $C\overline{\mathsf{G}} \subseteq \overline{M_0}$
3. V und W sind eindeutig, und es gilt $V\mathsf{L} = vec(R)$ sowie $W\mathsf{L} = vec(S^\top)$

Die Relationen C und M_0 modellieren die Kapazitätsfunktion bzw. die Startmarkierung. Wir verwenden im Folgenden sowohl die relationale als auch die funktionale Schreibweise. Die Gleichung in (1) bedeutet, dass alle Kapazitäten größer als 0 sind; es gilt also

$$C\vec{0} = \mathsf{O} \Longleftrightarrow \forall p : C(p) \neq 0.$$

Mit (2) wird beschrieben, dass die Markierungen die Kapazitäten nicht überschreiten, d.h.

$$C\overline{\mathsf{G}} \subseteq \overline{M_0} \Longleftrightarrow \forall p : M_0(p) \leq C(p).$$

Mithilfe der Relationen V und W wird die Gewichtsfunktion f des Flusses modelliert, und die Gleichung $V\mathsf{L} = vec(R)$ sichert zu, dass der Vorbereich von V ausschließlich aus Paaren der Relation R besteht, d.h

$$V\mathsf{L} = vec(R) \Longleftrightarrow (\exists k : V_{\langle p,t\rangle k}) \leftrightarrow R_{pt}.$$

Analog gilt

$$W\mathsf{L} = vec(S^\top) \Longleftrightarrow (\exists k : W_{\langle p,t\rangle k}) \leftrightarrow S_{tp},$$

und damit modellieren V und W die Funktion f in dem Sinne, dass

$$V_{\langle p,t\rangle k} \Longleftrightarrow f(p,t) = k \qquad \text{und} \qquad W_{\langle t,p\rangle k} \Longleftrightarrow f(t,p) = k$$

für alle Stellen p, Transitionen t und alle $k \in \mathcal{N}$.

Im Folgenden werden Abbildungen entwickelt, um zu einer gegebenen Markierung die Menge aller durch sie aktivierten Transitionen zu berechnen, sowie das Schalten einer Transition unter einer Markierung zu modellieren. Dabei gehen stets von einem Petrinetz im relationalen Sinne aus. Zur Vereinfachung schreiben wir $\mathcal{N}$ für $[0..n]$. Des Weiteren seien die natürlichen Projektionen (π, ρ) auf $\mathcal{P} \times \mathcal{T}$, (α, β) auf $\mathcal{P} \times \mathcal{N}$, sowie (γ, δ) auf $\mathcal{N} \times \mathcal{N}$ gegeben. Wir benötigen zunächst die folgende Konstruktion. Zu einer Relation Z des Typs $[\mathcal{P} \times \mathcal{T} \leftrightarrow \mathcal{N}]$ sei $\widetilde{Z} : \mathcal{P} \times \mathcal{N} \leftrightarrow \mathcal{T}$ definiert durch

$$\widetilde{Z} = (\alpha\pi^\top \cap \beta Z^\top)\rho.$$

Dann gilt $\widetilde{Z}_{\langle p,n\rangle t}$ genau dann wenn $Z_{\langle p,t\rangle n}$. Sei nun M eine zulässige Markierung, also eine Relation des Typs $[\mathcal{P} \leftrightarrow \mathcal{N}]$ mit der Eigenschaft $C\overline{\mathsf{G}} \subseteq \overline{M}$, und $m = vec(M)$ ihre Vektordarstellung. Dann gilt das folgende Lemma.

6.2.1 Lemma *Der Vektor* $\overline{\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)m}$ *des Typs* $[\mathcal{T} \leftrightarrow \mathbf{1}]$ *spezifiziert die Menge der Transitionen, welche die Bedingung (A1) aus Abschnitt 6.1 erfüllen.*

Beweis: *Zunächst gilt unter Verwendung von Theorem 4.1.10*

$$\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)m = \widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)vec(M) = \widetilde{V}^\top vec(M\overline{\mathsf{G}}),$$

und es folgt für alle Transitionen t die nachstehende Äquivalenz:

$$\begin{aligned}
(\overline{\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)m})_t &\Longleftrightarrow (\overline{\widetilde{V}^\top vec(M\overline{\mathsf{G}})})_t \\
&\Longleftrightarrow \neg\exists p,k : \widetilde{V}_{\langle p,k\rangle t} \ \wedge \ vec(M\overline{\mathsf{G}})_{\langle p,k\rangle} \\
&\Longleftrightarrow \neg\exists p,k : V_{\langle p,t\rangle k} \ \wedge \ (M\overline{\mathsf{G}})_{pk} \\
&\Longleftrightarrow \neg\exists p,k,k' : V_{\langle p,t\rangle k} \ \wedge \ M_{pk'} \ \wedge \ \overline{\mathsf{G}}_{k'k} \\
&\Longleftrightarrow \forall p,k,k' : (f(p,t) = k \ \wedge \ M(p) = k') \rightarrow \mathsf{G}_{k'k} \\
&\Longleftrightarrow \forall p \in {\bullet t} : M(p) \geq f(p,t)
\end{aligned}$$

Um auf ähnliche Weise die Menge der Transitionen zu bestimmen, die die verbleibende Bedingung (A2) erfüllen, benötigen wir eine weitere Hilfskonstruktion. Sei im Folgenden die Relation $Z : \mathcal{P} \times \mathcal{T} \leftrightarrow \mathcal{N}$ definiert durch

$$Z = (\pi C\gamma^\top \cap W\delta^\top)\mathsf{S}.$$

6.2.2 Lemma *Für alle $p \in \mathcal{P}$, $t \in \mathcal{T}$ und $k \in \mathcal{N}$ gilt die Äquivalenz*

$$Z_{\langle p,t\rangle k} \Longleftrightarrow S_{tp} \ \wedge \ C(p) - f(t,p) = k.$$

Beweis:

$$\begin{aligned}
Z_{\langle p,t\rangle k} &\Longleftrightarrow ((\pi C\gamma^\top \cap W\delta^\top)\mathsf{S})_{\langle p,t\rangle k} \\
&\Longleftrightarrow \exists k', k'' : (\pi C\gamma^\top \cap W\delta^\top)_{\langle p,t\rangle\langle k',k''\rangle} \ \wedge \ \mathsf{S}_{\langle k',k''\rangle k} \\
&\Longleftrightarrow \exists k', k'' : C_{pk'} \ \wedge \ W_{\langle p,t\rangle k''} \ \wedge \ \mathsf{S}_{\langle k',k''\rangle k} \\
&\Longleftrightarrow S_{tp} \ \wedge \ \exists k', k'' : C(p) = k' \ \wedge \ f(t,p) = k'' \ \wedge \ k' - k'' = k \\
&\Longleftrightarrow S_{tp} \ \wedge \ C(p) - f(t,p) = k
\end{aligned}$$

Mithilfe der modifizierten Relation $\widetilde{Z}$ kann mit dem folgenden Lemma nun auf die verbleibende Bedingung (A2) eingegangen werden.

6.2.3 Lemma *Der Vektor $\overline{\widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})m}$ des Typs $[\mathcal{T} \leftrightarrow \mathbf{1}]$ spezifiziert die Menge der Transitionen, welche die Bedingung (A2) aus Abschnitt 6.1 erfüllen.*

Beweis: *Zunächst gilt mit Theorem 4.1.10*

$$\widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})m = \widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})vec(M) = \widetilde{Z}^\top vec(M\overline{\mathsf{G}}^\top),$$

und es folgt unter Verwendung von Lemma 6.2.2 für alle Transitionen t:

$$\begin{aligned}
&(\overline{\widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})m})_t \\
&\Longleftrightarrow (\overline{\widetilde{Z}^\top vec(M\overline{\mathsf{G}}^\top)})_t \\
&\Longleftrightarrow \neg\exists p, k : \widetilde{Z}_{\langle p,k\rangle t} \ \wedge \ vec(M\overline{\mathsf{G}}^\top)_{\langle p,k\rangle} \\
&\Longleftrightarrow \neg\exists p, k : Z_{\langle p,t\rangle k} \ \wedge \ (M\overline{\mathsf{G}}^\top)_{pk} \\
&\Longleftrightarrow \neg\exists p, k, k' : Z_{\langle p,t\rangle k} \ \wedge \ M_{pk'} \ \wedge \ \overline{\mathsf{G}}_{kk'} \\
&\Longleftrightarrow \forall p, k, k' : (Z_{\langle p,t\rangle k} \ \wedge \ M_{pk'}) \rightarrow \mathsf{G}_{kk'} \\
&\Longleftrightarrow \forall p, k, k' : (M(p) = k' \ \wedge \ S_{tp} \ \wedge \ C(p) - f(t,p) = k) \rightarrow k \geq k' \\
&\Longleftrightarrow \forall p \in t\bullet : M(p) \leq C(p) - f(t,p) \\
&\Longleftrightarrow \forall p \in t\bullet : M(p) + f(t,p) \leq C(p)
\end{aligned}$$

Mithilfe dieser Vorbereitungen kann man nun eine Abbildung

$$\varphi_{act} : [\mathcal{P} \times \mathcal{T} \leftrightarrow \mathbf{1}] \rightarrow [\mathcal{T} \leftrightarrow \mathbf{1}]$$

durch

$$\varphi_{act}(m) = \overline{(\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top) \cup \widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}}))m}$$

definieren, die zu einer Markierung m in Vektordarstellung die durch m aktivierten Transitionen in Form eines Vektors des Typs $[\mathcal{T} \leftrightarrow \mathbf{1}]$ berechnet. Diese Tatsache halten wir im folgenden Korollar fest.

6.2.4 Korollar *Für alle Markierungen M mit $vec(M) = m$ gilt $\varphi_{act}(m)_t$ genau dann, wenn M die Transition t aktiviert.*

Beweis: *Es gilt*

$$\begin{aligned}\varphi_{act}(m) &= \overline{(\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top) \cup \widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}}))m} \\ &= \overline{\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)m \cup \widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})m} \\ &= \overline{\widetilde{V}^\top(\mathsf{I}\|\overline{\mathsf{G}}^\top)m} \cap \overline{\widetilde{Z}^\top(\mathsf{I}\|\overline{\mathsf{G}})m}\end{aligned}$$

und daraus folgt mit Lemma 6.2.1 und Lemma 6.2.3, dass $\varphi_{act}(m)_t$ genau dann gilt, wenn t die Bedingungen (A1) und (A2) erfüllt.

Offensichtlich ist die Abbildung φ_{act} ein Element der in Kapitel 3.2.2 definierten Menge $\mathcal{VA}$, so dass man sie zu einer Abbildung erweitern kann, die auf Listen von Markierungen definiert ist und die somit eine Liste von Transitionsmengen liefert. Sei also eine Relation L des Typs $[\mathcal{P} \times \mathcal{T} \leftrightarrow [1..k]]$ gegeben, wobei k eine natürliche Zahl sei. Seien weiter alle Spalten von L zulässige Markierungen, d.h. für alle $i \in [1..k]$ gelte $C\overline{\mathsf{G}} \subseteq \overline{rel(L^{(i)})}$. Dann repräsentiert jede Spalte $\varphi_{act}^{[1..k]}(L)^{(j)}$ der Relation $\varphi_{act}^{[1..k]}(L) : \mathcal{T} \leftrightarrow [1..k]$ genau die Menge der durch die Markierung $L^{(j)}$ aktivierten Transitionen; es gilt also

$$\varphi_{act}^{[1..k]}(L)_{tj} \Longleftrightarrow L^{(j)} \text{ aktiviert } t.$$

Für die relationale Modellierung der Schaltregel benötigen wir weitere Vorbereitungen. Seien die Relationen $F^1, F^2, F^3, F^4 : \mathcal{P} \times \mathcal{N} \leftrightarrow \mathcal{T}$ wie folgt gegeben:

$$F^1 = \alpha(R \cap \overline{S}^\top)$$

$$F^2 = \alpha(\overline{R} \cap S^\top)$$

$$F^3 = \alpha(R \cap S^\top)$$

$$F^4 = \alpha(\overline{R} \cap \overline{S}^\top)$$

Die Relationen dienen der Modellierung der Fallunterscheidungen in der Schaltregel, wie das folgende Lemma verdeutlicht.

6.2.5 Lemma *Für alle $t \in \mathcal{T}$ gelten die folgenden Äquivalenzen.*

1. $(F^1\vec{t})_{\langle p,k\rangle} \Longleftrightarrow p \in {\bullet t} \setminus t{\bullet}$
2. $(F^2\vec{t})_{\langle p,k\rangle} \Longleftrightarrow p \in t{\bullet} \setminus {\bullet t}$
3. $(F^3\vec{t})_{\langle p,k\rangle} \Longleftrightarrow p \in {\bullet t} \cap t{\bullet}$
4. $(F^4\vec{t})_{\langle p,k\rangle} \Longleftrightarrow p \in \mathcal{P} \setminus ({\bullet t} \cup t{\bullet})$

Beweis: *Für den ersten Fall gilt beispielsweise*

$$\begin{aligned}(F^1\vec{t})_{\langle p,n\rangle} &\Longleftrightarrow F^1_{\langle p,n\rangle t}\\ &\Longleftrightarrow (\alpha(R \cap \overline{S}^\top))_{\langle p,n\rangle t}\\ &\Longleftrightarrow (R \cap \overline{S}^\top)_{pt}\\ &\Longleftrightarrow R_{pt} \ \wedge \ \overline{S}_{tp}\\ &\Longleftrightarrow p \in {\bullet t} \ \wedge \ p \notin t{\bullet}\\ &\Longleftrightarrow p \in {\bullet t} \setminus t{\bullet},\end{aligned}$$

die verbleibenden Äquivalenzen folgen analog.

Des Weiteren definieren wir für jede Transition t zwei Relationen V^t und W^t des Typs $[\mathcal{P} \leftrightarrow \mathcal{T}]$ durch

$$V^t_{pn} \Longleftrightarrow V_{\langle p,t\rangle n} \qquad \text{und} \qquad W^t_{pn} \Longleftrightarrow W_{\langle p,t\rangle n}.$$

Mit diesen beiden Relationen kann nun das Schalten einer aktivierten Transition modelliert werden. Für die Entwicklung einer entsprechenden Vektorabbildung nutzen wir zunächst das folgende in [23] bewiesene Lemma.

6.2.6 Lemma *Seien M eine zulässige Markierung und t eine von M aktivierte Transition. Dann gelten die folgenden drei Äquivalenzen.*

1. $((M\gamma^\top \cap V^t\delta^\top)\mathsf{S})_{pk} \Longleftrightarrow k = M(p) - f(p,t)$
2. $((M\gamma^\top \cap W^t\delta^\top)\mathsf{A})_{pk} \Longleftrightarrow k = M(p) + f(t,p)$
3. $(((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}\gamma^\top \cap W^t\delta^\top)\mathsf{A})_{pk} \Longleftrightarrow k = M(p) - f(p,t) + f(t,p)$

Im Folgenden werden mithilfe von Theorem 4.1.10 die Vektordarstellungen der Relationen des vorangegangenen Lemmas berechnet und zur Entwicklung einer Abbildung verwendet, welche zu einer Markierung und einer aktivierten Transition die durch das Schalten der Transition erzeugte Nachfolgemarkierung berechnet.

6.2.7 Lemma *Seien M eine zulässige Markierung, $m = vec(M)$ und t eine von M aktivierte Transition. Dann gelten die folgenden Aussagen.*

1. $vec((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}) = (\mathsf{I}\|\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}\vec{t}\,)$
2. $vec((M\gamma^\top \cap W^t\delta^\top)\mathsf{A}) = (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{W}\vec{t}\,)$
3. $vec(((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}\gamma^\top \cap W^t\delta^\top)\mathsf{A})$
   $= (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}\vec{t}\,) \cap (\mathsf{I}\|\delta)\widetilde{W}\vec{t}\,)$

Beweis: *Es gilt zunächst*

$$vec(V^t) = \widetilde{V}\vec{t} \qquad und \qquad vec(W^t) = \widetilde{W}\vec{t}.$$

Damit folgt

$$\begin{aligned}
vec((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}) &= (\mathsf{I}\|\mathsf{S}^\top)vec(M\gamma^\top \cap V^t\delta^\top) && (\textit{Theorem 4.1.10})\\
&= (\mathsf{I}\|\mathsf{S}^\top)(vec(M\gamma^\top) \cap (vec(V^t\delta^\top))\\
&= (\mathsf{I}\|\mathsf{S}^\top)((\mathsf{I}\|\gamma)vec(M) \cap (\mathsf{I}\|\delta)vec(V^t)) && (\textit{Theorem 4.1.10})\\
&= (\mathsf{I}\|\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}\vec{t}\,)
\end{aligned}$$

Die zweite Behauptung folgt analog. Für (3) gilt

$$\begin{aligned}
&vec(((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}\gamma^\top \cap W^t\delta^\top)\mathsf{A})\\
&= (\mathsf{I}\|\mathsf{A}^\top)vec((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}\gamma^\top \cap W^t\delta^\top) && (\textit{Theorem 4.1.10}))\\
&= (\mathsf{I}\|\mathsf{A}^\top)(vec((M\gamma^\top \cap V^t\delta^\top)\mathsf{S}\gamma^\top) \cap vec(W^t\delta^\top))\\
&= (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma\mathsf{S}^\top)(vec(M\gamma^\top) \cap vec(V^t\delta^\top)) \cap (\mathsf{I}\|\delta)vec(W^t)) && (\textit{Theorem 4.1.10})\\
&= (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}\vec{t}\,) \cap (\mathsf{I}\|\delta)\widetilde{W}\vec{t}\,) && (\textit{Theorem 4.1.10})
\end{aligned}$$

Auf Grundlage der eben entwickelten Ausdrücke lassen sich nun vier Abbildungen

$$\varphi_1, \varphi_2, \varphi_3, \varphi_4 : [\mathcal{P} \times \mathcal{T} \leftrightarrow \mathbf{1}] \times [\mathcal{T} \leftrightarrow \mathbf{1}] \to [\mathcal{P} \times \mathcal{T} \leftrightarrow \mathbf{1}]$$

durch

$$\varphi_1(m,v) = F^1 v \cap (\mathsf{I}\|\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}v),$$

$$\varphi_2(m,v) = F^2 v \cap (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{W}v),$$

$$\varphi_3(m,v) = F^3 v \cap (\mathsf{I}\|\mathsf{A}^\top)((\mathsf{I}\|\gamma\mathsf{S}^\top)((\mathsf{I}\|\gamma)m \cap (\mathsf{I}\|\delta)\widetilde{V}v) \cap (\mathsf{I}\|\delta)\widetilde{W}v),$$

$$\varphi_4(m,v) = F^4 v \cap m$$

definieren, mit deren Hilfe man das Resultat des Schaltens einer Transition berechnen kann, wie im folgenden Korollar beschrieben wird.

6.2.8 Korollar *Seien M eine zulässige Markierung, $m = vec(M)$ und $t \in \mathcal{T}$ mit $\varphi_{act}(m)_t$, also eine durch M aktivierte Transition. Dann gelten die folgenden Äquivalenzen.*

1. $\varphi_1(m,\vec{t}\,)_{\langle p,k\rangle} \Longleftrightarrow p \in {\bullet t} \setminus t^{\bullet} \ \wedge \ k = M(p) - f(p,t)$
2. $\varphi_2(m,\vec{t}\,)_{\langle p,k\rangle} \Longleftrightarrow p \in t^{\bullet} \setminus {\bullet t} \ \wedge \ k = M(p) + f(p,t)$
3. $\varphi_3(m,\vec{t}\,)_{\langle p,k\rangle} \Longleftrightarrow p \in {\bullet t} \cap t^{\bullet} \ \wedge \ k = M(p) - f(p,t) + W(t,p)$
4. $\varphi_3(m,\vec{t}\,)_{\langle p,k\rangle} \Longleftrightarrow p \in \mathcal{P} \setminus ({\bullet t} \cup t^{\bullet}) \ \wedge \ k = M(p)$

Beweis: *Die Aussagen folgen sofort aus den Lemmata 6.2.5, 6.2.6 und 6.2.7.*

Durch das Korollar ist mit

$$\varphi_{\mathit{fire}} = \varphi_1 \cup \varphi_2 \cup \varphi_3 \cup \varphi_4$$

eine Abbildung gegeben, die zu einer Markierung m in Vektordarstellung und einer von m aktivierten Transition t die Markierung berechnet, die aus m durch das Schalten von t entsteht. Es gilt also

$$\mathit{rel}(m) \overset{t}{\rightsquigarrow} \mathit{rel}(\varphi_{\mathit{fire}}(m,\vec{t}\,)).$$

Analog zum Vorgehen in Abschnitt 3.2.3.1 definieren wir nun für ein festes m die Abbildung

$$\varphi_{\mathit{fire}_m} := \varphi_{\mathit{fire}}(m,\cdot) : [\mathcal{T} \leftrightarrow \mathbf{1}] \to [\mathcal{P} \times \mathcal{T} \leftrightarrow \mathbf{1}],$$

um sie zu einer Abbildung

$$\varphi^{\mathcal{A}}_{\mathit{fire}_m} : [\mathcal{T} \leftrightarrow \mathcal{A}] \to [\mathcal{P} \times \mathcal{T} \leftrightarrow \mathcal{A}]$$

in der üblichen Weise fortzusetzen. Dabei sei $\mathcal{A} = \{t \,|\, \varphi_{act}(m)_t^\top\}$ die durch den Vektor $\varphi_{act}(m)^\top$ repräsentierte Menge, also die Menge der von m aktivierten Transitionen.

Jede Spalte der Relation $A = inj(\varphi_{act}(m)^\top)^\top : \mathcal{T} \leftrightarrow \mathcal{A}$ entspricht genau einem Punkt $\vec{t}$ mit $t \in \mathcal{A}$, es gilt also für alle $t \in \mathcal{A}$

$$\varphi^{\mathcal{A}}_{\mathit{fire}_m}(A)^{(t)} = \varphi_{\mathit{fire}_m}(A^{(t)}) = \varphi_{\mathit{fire}}(m, \vec{t}),$$

und damit ist jede Spalte $\varphi^{\mathcal{A}}_{\mathit{fire}_m}(A)^{(t)}$ die Nachfolgemarkierung, die aus m durch Schalten der Transition t entsteht. Es gilt also

$$rel(m) \overset{t}{\rightsquigarrow} rel(\varphi^{\mathcal{A}}_{\mathit{fire}_m}(A)^{(t)}),$$

und damit repräsentiert die Relation $\varphi^{\mathcal{A}}_{\mathit{fire}_m}(A) : \mathcal{P} \times \mathcal{T} \leftrightarrow \mathcal{A}$ genau die Menge $rel(m)\rightsquigarrow$ aller unmittelbaren Folgemarkierungen von $rel(m)$.
Die hier entwickelten Abbildungen zur Bestimmung der aktivierten Transitionen sowie zur Erzeugung von unmittelbaren Nachfolgemarkierungen können nun verwendet werden, um den Zustandsraum eines Petrinetzes, also die Menge aller von der Startmarkierung erreichbaren Markierungen, zu berechnen. Wir benötigen dafür einige Vorbereitungen, insbesondere für den Umgang mit im Berechnungsprozess mehrfach auftretenden Markierungen. Hierbei werden zwei Abbildungen verwendet, die sich auf den symmetrischen Quotienten stützen. Nach Abschnitt 2.5 gilt für zwei Relationen A und B mit gleichem Urbildbereich die Äquivalenz

$$syq(A, B)_{ij} \iff A^{(i)} = B^{(j)},$$

insbesondere gibt also der Vektor $syq(A, B)\mathsf{L}$ die Spalten in A an, welche auch in der Relation B vorkommen. Daher wird durch

$$new(A, B) = \overline{syq(A, B)\mathsf{L}}$$

der Vektor bestimmt, der die Spalten von A angibt, welche nicht in B vorkommen. Die Abbildung *new* ist damit entscheidend für das Hinzufügen neuer, bisher noch nicht erzeugter Markierungen. Des Weiteren können in einer Relation doppelte Spalten vorkommen, von denen man jeweils nur eine übernehmen möchte. Dazu kann die Abbildung *singles* verwendet werden, die durch

$$singles(A) = \overline{(syq(A, A) \cap \overline{O})\mathsf{L}}$$

definiert ist, wobei O eine beliebige lineare Ordnungsrelation ist. Es ist $syq(A, A)$ eine Äquivalenzrelation, und der Vektor $\overline{(syq(A, A) \cap \overline{O})\mathsf{L}}$ enthält aus jeder Äquivalenzklasse genau ein Element, nämlich das kleinste Element bezüglich der Ordnung O (vergleiche

auch Abschnitt 3.4). Durch $\hat{A} := A\, inj(singles(A))^\top$ wird also eine Relation erzeugt, die die folgenden Bedingungen erfüllt.

$$\forall i \exists j : \hat{A}^{(i)} = A^{(j)}$$

$$\forall i \exists j : A^{(i)} = \hat{A}^{(j)}$$

$$\forall i, j : \hat{A}^{(i)} = \hat{A}^{(j)} \rightarrow i = j$$

Für den im Folgenden angegebenen Algorithmus in Pseudo-Code nehmen wir außerdem eine Funktion *length* an, die zu einer Relation die Anzahl ihrer Spalten liefert, sowie eine Funktion *conc*, die zwei Relationen gewissermaßen zusammensetzt (oder *konkateniert*), so dass aus zwei Relationen $A : X \leftrightarrow Y$ und $B : X \leftrightarrow Z$ durch $conc(A, B)$ eine Relation des Typs $[X \leftrightarrow Y + Z]$ entsteht, wobei $Y + Z$ die disjunkte Vereinigung der Mengen Y und Z beschreibt und $conc(A, B)_{xw}$ genau dann gilt, wenn $A_{xw} \ \wedge \ w \in Y$ oder $B_{xw} \ \wedge \ w \in Z$. Eine formale Definition der Konkatenation mittels der injektiven Einbettungen der direkten Summe wird in [8] gegeben. Nach diesen Vorbereitungen können wir nun einen Algorithmus angeben, der zu einer in Vektordarstellung gegebenen Startmarkierung m_0 die Menge $M_0 \overset{*}{\rightsquigarrow}$ aller von M_0 aus erreichbaren Markierungen als Spalten einer Relation M mit Urbildbereich $\mathcal{P} \times \mathcal{N}$ berechnet.

```
marks(m_0)
  M := m_0
  i := 1
  while i ≤ length(M)
    m := M^(i)
    a := φ_act(m)
    if a ≠ O then
      N := φ_fire_m^[1..|a|](inj(a)^⊤)
      v := new(N, M)
      if v ≠ O then
        N := N inj(v)^⊤
        N := N inj(singles(N))^⊤
        M := conc(M, N)
    i := i + 1
  return M
```

Hierbei werden ausgehend von der Startmarkierung m_0 sukzessive alle erreichbaren Markierungen berechnet, indem die Liste der bisher berechneten Markierungen abgearbeitet wird. Zu jeder Markierung m in der Liste M wird mithilfe der Abbildung

φ_{act} der Vektor a der durch m aktivierten Transitionen ermittelt und damit durch $\varphi_{\mathit{fire}_m}^{[1..|a|]}(\mathit{inj}(a)^\top)$ die Liste N der direkten Nachfolgemarkierungen von m berechnet. Mit $\mathit{new}(N, M)$ wird der Vektor v erzeugt, der angibt, welche der Markierungen in N noch nicht in M vorhanden sind. Schließlich werden durch Anwendung der Abbildung *singles* doppelt in N vorkommende Markierungen entfernt, so dass sichergestellt wird, dass eine Markierung nur dann als Spalte der Relation M hinzugefügt wird, wenn sie in dieser noch nicht enthalten ist, und dass jede neue Markierung nur genau einmal hinzugefügt wird. Durch $\mathit{conc}(M, N)$ werden die neuen Markierungen hinten an die Relation M angefügt, und die nächste Markierung wird abgearbeitet.

Die so berechnete Relation M kann man nun für weitere Analysen des Petrinetzes $\mathcal{PN}$ nutzen, beispielsweise zur Berechnung der Erreichbarkeitsrelation $\overset{*}{\rightsquigarrow}$ auf der Menge $M_0\overset{*}{\rightsquigarrow}$. Sei im Folgenden $|M_0\overset{*}{\rightsquigarrow}| = l$ und damit die Relation $M = \mathit{marks}(m_0)$ eine Relation des Typs $[\mathcal{P} \times \mathcal{N} \leftrightarrow [1..l]]$. Wir berechnen zunächst die unmittelbare Erreichbarkeitsrelation $\rightsquigarrow$ auf $M_0\overset{*}{\rightsquigarrow}$ in Form einer Relation $U : [1..l] \leftrightarrow [1..l]$, für welche die Äquivalenz

$$U_{ij} \Longleftrightarrow \mathit{rel}(M^{(i)}) \rightsquigarrow \mathit{rel}(M^{(j)})$$

gilt. Dabei wird ausgenutzt, dass es sich bei der Abbildung φ_{act}, mit der man die Menge der aktivierten Transitionen einer Markierung berechnet, um ein Vektorprädikat handelt. Man erhält daher durch $A = \varphi_{act}^{[1..l]}(M)$ eine Relation des Typs $[\mathcal{T} \leftrightarrow [1..l]]$, die spaltenweise die aktivierten Transitionen für jede Markierung in $M_0\overset{*}{\rightsquigarrow}$ auflistet, d.h. $A^{(i)}$ gibt die Menge der durch $M^{(i)}$ aktivierten Transitionen an. Um Positionen von Markierungen in M zu bestimmen, definieren wir mithilfe des symmetrischen Quotienten die Abbildung *positions* durch

$$\mathit{positions}(N, M) = \mathit{syq}(M, N)\mathsf{L},$$

dann gilt $\mathit{positions}(N, M)_i$ genau dann wenn ein j mit $M^{(i)} = N^{(j)}$ existiert. Die Abbildung gibt also die Spalten in M an, welche mit Spalten in N übereinstimmen. Der folgende Algorithmus berechnet die unmittelbare Erreichbarkeitsrelation, wobei M die durch $\mathit{marks}(m_0)$ erzeugte Relation ist, die die Menge aller von m_0 aus erreichbaren Markierungen modelliert. Ausgehend von der leeren Relation U des Typs $[[1..l] \leftrightarrow [1..l]]$ werden sukzessive alle Einträge der unmittelbaren Erreichbarkeitsrelation U ermittelt. Zunächst wird durch $\varphi_{act}^{[1..l]}(M)$ die Relation A berechnet, deren Spalten die Menge der von der entspechenden Markierung aktivierten Transitionen angibt. Für jede Markierung $m = M^{(i)}$ wird dann durch $\varphi_{\mathit{fire}_m}^{[1..|a|]}(\mathit{inj}(a)^\top)$ die Menge der von m direkt erreichbaren Markierungen in Form einer Relation N des Typs $[\mathcal{P} \times \mathcal{N} \leftrightarrow [1..|a|]]$ berechnet, wo-

bei $a = A^{(i)}$ der Vektor der von m aktivierten Transitionen ist. Durch $positions(N, M)$ wird ein Vektor p des Typs $[1..l] \leftrightarrow \mathbf{1}$ ermittelt, der angibt, welche Spalten von M den Markierungen in N entsprechen. Die Relation $\vec{i}\,p^\top$ drückt also aus, welche Markierungen von $M^{(i)}$ unmittelbar erreichbar sind, d.h. es gilt

$$(\vec{i}\,p^\top)_{ij} \iff rel(M^{(i)}) \rightsquigarrow rel(M^{(j)}).$$

Schließlich werden mit $U \cup \vec{i}\,p^\top$ die neu ermittelten Einträge der Relation U hinzugefügt.

$U(M)$
  $U := \mathsf{O}$
  $A := \varphi_{act}^{[1..l]}(M)$
  for $i = 1$ *to* l
    $m := M^{(i)}$
    $a := A^{(i)}$
    $N := \varphi_{fire_m}^{[1..|a|]}(inj(a)^\top)$
    $p := positions(N, M)$
    $U := U \cup \vec{i}\,p^\top$
  return U

Durch $E = U(M)^*$ erhalten wir aus der so berechneten direkten Erreichbarkeitsrelation sofort die Erreichbarkeitsrelation. Es gilt also

$$E_{ij} \iff rel(M^{(i)}) \overset{*}{\rightsquigarrow} rel(M^{(j)}).$$

Die in diesem Kapitel entwickelten Algorithmen *marks* und U lassen sich in relationale Programme übertragen und mithilfe des RelView-Systems auswerten. Da hierbei keine natürlichen Zahlen zur Verfügung stehen, wird die While-Schleife im Falle von *marks* wie folgt durch die Verwendung von Punkten verschiedenen Typs realisiert. Der Laufindex i wird durch einen Punkt $p = \vec{i} : [1..l] \leftrightarrow \mathbf{1}$ modelliert, wobei l die Anzahl der Spalten der aktuellen Relation M ist. Anfangs ist $M = m_0$ ein Vektor, es wird also `p` als Allrelation des Typs $[\mathbf{1} \leftrightarrow \mathbf{1}]$ initialisiert. Die Schleifenbedingung ist dann `-empty(p)`, und das „Hochzählen“ von `p` erfolgt durch die Zuweisungen

```
p = p + On1(N);
p = next(p);
```

nach der Berechnung von N. Die erste Zuweisung passt die Länge von `p` der Anzahl der Spalten der aktuellen Relation M an, während die zweite Zuweisung sicherstellt, dass im nächsten Schleifendurchlauf die nächste Spalte der Relation M bearbeitet wird. Ent-

spricht der Punkt `p` dann dem letzten Eintrag eines Vektors, so wird durch `p = next(p)` ein leerer Vektor zurückgeliefert, und die Schleife bricht ab. Die Zuweisung $m := M^{(i)}$ entspricht der Zuweisung `m = M*p` im relationalen Programm.

Im Fall der Berechnung der unmittelbaren Erreichbarkeitsrelation U wird die For-Schleife durch eine While-Schleife ersetzt, wobei auch hier der Laufindex i durch einen Punkt $p = \vec{i} : [1..l] \leftrightarrow \mathbf{1}$ modelliert wird, wobei l die Anzahl der Spalten der Eingaberelation M ist. Da l im Programm nicht verändert wird, kann `p` durch `p = next(p)` hochgezählt werden.

6.3 Relationale Analyse von Petrinetzen

Hat man die im vorangehenden Kapitel definierten Relationen $M : \mathcal{P} \times \mathcal{N} \leftrightarrow [1..l]$ zur Darstellung aller von der Startmarkierung erreichbaren Markierungen, $A : \mathcal{T} \leftrightarrow [1..l]$ zur Angabe der durch die entsprechenden Markierungen aktivierten Transitionen sowie die Erreichbarkeitsrelation $E : [1..l] \leftrightarrow [1..l]$ zur Verfügung, lassen sich viele Eigenschaften des zugrundeliegenden Petrinetzes sehr einfach relational ausdrücken. Dafür sollen in diesem Kapitel einige Beispiele angegeben werden. Zunächst geht es um die Identifikation von Markierungen mit gewissen Eigenschaften. Dabei wird eine Menge von Markierungen als Vektor v des Typs $[1..l] \leftrightarrow \mathbf{1}$ angegeben, welche die Menge $\{rel(M^{(i)}) \mid v_i\}$ repräsentiert. Jede Markierung $M' \in M_0 \overset{*}{\rightsquigarrow}$ entspricht einer Spalte $M^{(i)}$ in M und wird daher durch einen Punkt $\vec{i}$ des Typs $[1..l] \leftrightarrow \mathbf{1}$ repräsentiert.

Wir beginnen mit der Berechnung der Mengen $M' \overset{*}{\rightsquigarrow}$ für $M' \in M_0 \overset{*}{\rightsquigarrow}$. Wie man leicht sieht, wird die Menge $rel(M^{(i)}) \overset{*}{\rightsquigarrow}$ der von $rel(M^{(i)})$ aus erreichbaren Markierungen durch die Relation $E^\top \vec{i}$ dargestellt. Des Weiteren gilt $\vec{i}\,\vec{j}^\top \subseteq E$ genau dann, wenn $rel(M^{(j)})$ von $rel(M^{(i)})$ aus erreichbar ist. Eine Markierung wird *Home Marking* genannt, wenn sie von allen Markierungen in $M_0 \overset{*}{\rightsquigarrow}$ erreichbar ist. Der Vektor $\overline{\overline{E}^\top \mathsf{L}}$ gibt die Menge aller Home Markings an, denn es gilt:

$$\begin{aligned} rel(M^{(i)}) \text{ ist Home Marking} &\iff \forall M' \in M_0 \overset{*}{\rightsquigarrow} : M' \overset{*}{\rightsquigarrow} rel(M^{(i)}) \\ &\iff \forall j : rel(M^{(j)}) \overset{*}{\rightsquigarrow} rel(M^{(i)}) \\ &\iff \forall j : E_{ij} \\ &\iff (\overline{\overline{E}^\top \mathsf{L}})_i \end{aligned}$$

Von einer *toten Markierung* spricht man, wenn durch eine Markierung keine Transitio-

nen aktiviert werden. Es gilt also:

$$\begin{aligned}
&rel(M^{(i)}) \text{ ist eine tote Markierung} \\
&\Longleftrightarrow \neg\exists t : \varphi_{act}(M^{(i)})_t \\
&\Longleftrightarrow \neg\exists t : A_t^{(i)} \\
&\Longleftrightarrow \neg\exists t : A_{ti} \\
&\Longleftrightarrow (\overline{A^\top \mathsf{L}})_i
\end{aligned}$$

Die Menge aller toten Markierungen wird also durch den Vektor $\overline{A^\top \mathsf{L}}$ repräsentiert. Ein Petrinetz heisst *schwach lebendig*, wenn es keine toten Markierungen aufweisst, wenn also $\overline{A^\top \mathsf{L}} = \mathsf{O}$ gilt.

Auch Eigenschaften von Transitionen können einfach relational dargestellt werde. Hierbei werden Mengen von Transitionen wie gewohnt als Vektoren des Typs $[\mathcal{T} \leftrightarrow \mathbf{1}]$ angegeben. Eine Transition heisst *tot* (unter M_0), wenn sie von keiner Markierung in $M_0 \overset{*}{\rightsquigarrow}$ aktiviert wird. Der Vektor $\overline{A\mathsf{L}}$ gibt die Menge aller toten Transitionen an, denn es gilt:

$$\begin{aligned}
t \text{ ist eine tote Transition} &\Longleftrightarrow \neg\exists i : \varphi_{act}(M^{(i)})_t \\
&\Longleftrightarrow \neg\exists i : A_{ti} \\
&\Longleftrightarrow (\overline{A\mathsf{L}})_t
\end{aligned}$$

Eine Transition t heisst *schwach lebendig* (unter M_0), wenn sie nicht tot (unter M_0) ist, d.h. wenn $(A\mathsf{L})_t$ gilt. Sie heisst *lebendig*, wenn sie schwach lebendig unter allen Markierungen in $M_0 \overset{*}{\rightsquigarrow}$ ist. Es gilt:

$$\begin{aligned}
t \text{ ist lebendig} &\Longleftrightarrow \forall M' \in M_0 \overset{*}{\rightsquigarrow} \exists M'' \in M' \overset{*}{\rightsquigarrow} : \varphi_{act}(M'')_t \\
&\Longleftrightarrow \forall i \exists j : rel(M^{(i)}) \overset{*}{\rightsquigarrow} rel(M^{(j)}) \ \wedge \ \varphi_{act}(M^{(j)})_t \\
&\Longleftrightarrow \forall i \exists j : E_{ij} \ \wedge \ A_{tj} \\
&\Longleftrightarrow \forall i : (AE^\top)_{ti} \\
&\Longleftrightarrow (\overline{\overline{AE^\top}\mathsf{L}})_t
\end{aligned}$$

Somit repräsentiert der Vektor $\overline{\overline{AE^\top}\mathsf{L}}$ die Menge aller lebendigen Transitionen. Weitere durch relationale Ausdrücke gut zu formalisierende Eigenschaften von Petrinetzen werden in [24] vorgestellt.

7 Abschließende Bemerkungen

In dieser Arbeit haben wir uns mit der Entwicklung von relationalen Algorithmen beschäftigt, und uns dabei besonders auf zwei Konzepte gestützt. Zunächst wurde das Konzept der Vektorabbildungen und Vektorprädikate entwickelt. Diese ermöglichen es, auf einfache Art und Weise aus einer Abbildung auf Vektoren eine Abbildung auf Relationen zu entwickeln, die die Relation sozusagen "spaltenweise" abbildet. Es wird also durch relationale Umformungen aus einer gegebenen Spezifikation zunächst eine Abbildung auf Vektoren entwickelt. Erst im zweiten Schritt wird auf der Grundlage einer solchen Vektorabbildung eine Abbildung auf Relationen definiert, die zu einer Menge von Vektoren (modelliert als Relation) die Menge der Ergebnisvektoren (ebenfalls modelliert als Relation) berechnet. Im Fall der Vektorprädikate erhält man durch dieses Vorgehen Testabbildungen, mit denen man aus einer Menge von Vektoren diejenigen mit gewissen erwünschten Eigenschaften herausfiltern kann. Interessant für die weitere Forschung auf diesem Gebiet wäre die Untersuchung von mehrstelligen Vektorabbildungen, also Abbildungen der Form

$$[X_1 \leftrightarrow \mathbf{1}] \times \ldots \times [X_n \leftrightarrow \mathbf{1}] \rightarrow [Y \leftrightarrow \mathbf{1}].$$

In dieser Arbeit treten solche mehrstelligen Vektorabbildungen implizit im Bereich der Petrinetze und der Bedingungs-Ereignis-Netze auf, wobei aber in diesen Fällen aus 2-stelligen Abbildungen jeweils einstellige Abbildungen abgeleitet wurden.

Wie sich gezeigt hat, lässt sich das Konzept der Vektorprädikate und Testabbildungen im Selektionsprozess relationaler Evolutionärer Algorithmen einsetzen. Auch andere wichtige Module Evolutionärer Algorithmen wie Mutation und Rekombination können mit relationalen Methoden modelliert werden. Insgesamt kann man feststellen, dass sich die relationale Programmierung zumindest für einige Arten von Evolutionären Algorithmen eignet und zu einfachen und übersichtlichen Programmen führt. Als besonders vorteilhaft erweist sich die Verwendung von relationalen Methoden im Zusammenhang mit multikriterieller Optimierung, wie in Abschnitt 3.4 gezeigt wurde. Hier wurde ein einfaches und effizientes relationales Programm entwickelt, um die Anzahl der Op-

timierungskriterien eines Problems zu reduzieren. An dieser Stelle wäre interessant, inwieweit sich relationale Methoden auch für die Entwicklung von Programmen zur Lösung multikriterieller Optimierungsprobleme einsetzen lassen.

Einen weiteren Schwerpunkt dieser Arbeit stellt die Repräsentation von Relationen durch Vektoren, und hierbei besonders Theorem 4.1.10 dar. Das Theorem und die bekannten Eigenschaften der Abbildung *vec* ermöglichen die Transformation von relationalen Eigenschaften in entsprechende Eigenschaften des zugehörigen Vektors. Dieses Vorgehen wurde am Beispiel von Stundenplanproblemen in vielfältiger Art und Weise angewandt. Durch die Darstellung von möglichen Lösungen eines Stundenplanproblems als Vektoren kann mithilfe der Potenzmengenrelation die Menge aller möglichen Lösungen modelliert werden. Die Entwicklung einer Testabbildung aus einem Vektorprädikat führt dann zu einem exakten Lösungsverfahren, welches im Falle der Universitären Stundenplanprobleme erfolgreich in der Praxis angewandt werden konnte. Die formale und dennoch einfache Art der Programmentwicklung ermöglichte dabei ein schnelles Reagieren auf Veränderungen des Stundenplanmodells in der Planungsphase der neuen Bachelor-Studiengänge. Das Konzept der Vektorrepräsentation und Vektorprädikate wurde im Bereich der Stundenplanprobleme außerdem genutzt, um Mengen von isomorphen Lösungen zu bestimmen. Es könnte im Folgenden untersucht werden, ob sich dieser Ansatz auf weitere Isomorphie-Probleme übertragen lässt, beispielsweise auf die Isomorphie zwischen Graphen.

Am Beispiel von Petrinetzen wurde am Schluss dieser Arbeit eine weitere Einsatzmöglichkeit für die Verwendung von Vektorabbildungen erörtert. Zwei Vektorabbildungen für die Berechnung von aktivierten Transitionen und Nachfolgemarkierungen bilden die Grundlage für Algorithmen zur Erzeugung der Menge aller erreichbaren Markierungen und der Erreichbarkeitsrelation. Die Modellierung durch Relationen erlaubt eine einfache relationale Analyse wichtiger Netzeigenschaften wie beispielsweise der Untersuchungen bezüglich der Lebendigkeit. Es könnte weiterhin untersucht werden, inwieweit sich die Darstellung durch Relationen für die Untersuchung weiterer Eigenschaften von Petrinetzen eignet, beispielsweise zur Ermittlung von nebenläufigen Transitionsmengen.

Literaturverzeichnis

[1] Appleby J.S., Blake D.V., Newman E.A.: Techniques for producing school timetables on a computer and their application to other scheduling problems. Computer Journal 3, 237-245 (1960).

[2] Behnke R.: Transformationelle Programmentwickung im Rahmen relationaler und sequenzieller Algebren. Dissertation, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel (1998).

[3] Berghammer R.: A generic program for minimal subsets with applications. In: Leuschel M. (Hrsg): Logic-based Program Development and Transformation, LNCS 2664, Springer, 144-157 (2003).

[4] Berghammer R.: Relational-algebraic computation of fixed points with applications. The Journal of Logic and Algebraic Programming 66, 112-126 (2006).

[5] Berghammer R.: Computation of Cut Completions and Concept Lattices Using Relational Algebra and RelView. Journal on Relational Methods in Computer Science 1, 50-72 (2004).

[6] Berghammer R., Fronk A.: Exact computation of minimum feedback vertex sets with relational algebra. Fundamenta Informaticae 70, 301-316 (2006).

[7] Berghammer R., Hoffmann T.: Calculating a Relational Program for Transitive Reductions of Strongly Connected Graphs. In: de Swart H. (Hrsg): Relational Methods in Computer Science, LNCS 2561, Springer, 258-275 (2001).

[8] Berghammer R., Hoffmann T.: Modeling Sequences within the RelView System. Journal of Universal Computer Science 7, 107-123 (2001).

[9] Berghammer R., Kehden B.: Relation Algebra and RelViewin practical use: Construction of special university timetables. In: Berghammer R., Möller B., Struth G. (Hrsg): Relational Methods in Computer Science, LNCS 4988, Springer, 22-36 (2008).

[10] Berghammer R., Kehden B.: Relational Construction of Specific Timetables. In: Nickel S., Kalcsics J. (Hrsg): Operations Research Proceedings 2007, Springer, 397-402 (2008).

[11] Berghammer R., v. Karger B., Ulke C.: Relation-Algebraic Analysis of Petri Nets with RelView. In: Margaria T., Steffen B. (Hrsg): TACAS, LNCS 1055, Springer, 49-69 (1996).

[12] Berghammer R., Milanese U.: Relational approach to Boolean logic problems. In: MacCaull W., Winter M.,, Düntsch I. (Hrsg): Relational Methods in Computer Science, LNCS 3929, Springer, 48-59 (2006).

[13] Berghammer R., Neumann F.: RELVIEW - An OBDD-based Computer Algebra System For Relations. Ganzha G.V., Mayr E.W., Vorozhtsov E. V. (Hrsg): Computer Algebra in Scientific Computing, LNCS 3718, Springer, 40-51 (2005).

[14] Berghammer R., Zierer H.: Relational algebraic semantics of deterministic and nondeterministic programs. Theoretical Computer Science 43 (1986).

[15] Brockhoff D., Zitzler E.: On Objective Conflicts and Objective Reduction in Multiple Criteria Optimization. TIK-Report 243, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich (2006)

[16] Cormen T.T., Leiserson C.E., Rivest R.L.: Introduction to algorithms. The MIT Press (1990).

[17] Davey B.A., Priestley H.A.: Introduction to Lattices and Order, Cambridge University Press (1990).

[18] De Morgan A.: On the Syllogism: IV, and on the Logic of Relations. Transactions of the Cambridge Philosophical Society 10, 331-358 (1864).

[19] Diedrich F., Kehden B., Neumann F.: Multi-Objective Problems in Terms of Relational Algebra. In: Berghammer R., Möller B., Struth G. (Hrsg): Relational Methods in Computer Science, LNCS 4988, Springer, 84-98 (2008).

[20] Eiben A.E., Smith J.E.: Introduction to Evolutionary Computing. Springer (2003).

[21] Even S., Itai A., Shamir A.: On the Complexity of Timetable and Multicommodity Flow Problems. SIAM Journal of Computing 5 (1976).

[22] Fogel L.J., Owens A.J., Walsh M.J: Artificial Intelligence Through Simulated Evolution. John Wiley & Sons (1966).

[23] Fronk A.: Using relation algebra for the analysis of Petri nets in a CASE tool based approach. In: Software Engineering and Formal Methods (SEFM). IEEE Computer Society, 396-405 (2004).

[24] Fronk A., Kehden B.: State Space Analysis of Petri Nets with Relation-Algebraic Methods. Memorandum Nr. 163, Lehrstuhl fr Software-Technologie, Universität Dortmund (2006).

[25] Gotlieb C.C.: The Construction of Class-Teacher Time-Tables. IFIP Congress, 73-77 (1962).

[26] Holland J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press (1975).

[27] Kehden B.: Evaluating Sets of Search Points using Relational Algebra. In: Schmidt R.A. (Hrsg): Relational Methods in Computer Science, LNCS 4136, Springer, 266-280 (2006).

[28] Kehden, B., Neumann F.: A Relation-Algebraic View on Evolutionary Algorithms for Some Graph Problems. In: Gottlieb J., Raidl G.R.(Hrsg): Evolutionary Computation in Combinatorial Optimization, LNCS 3906, Springer, 147-158 (2006).

[29] Kehden B., Neumann F., Berghammer R.: Relational implementation of simple parallel evolutionary algorithms. In: MacCaull W., Winter M., Düntsch I.(Hrsg): Relational Methods in Computer Science, LNCS 3929, Springer, 161-172 (2006).

[30] Leoniuk B.: ROBDD-basierte Implementierung von Relationen und relationalen Operationen mit Anwendungen. Dissertation, Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel (2001).

[31] Leung J. Y.-T.: Handbook of Scheduling. Chapman & Hall/CRC (2004).

[32] Milanese U.: Zur Implementierung eines ROBDD-basierten Systems für die Manipulation und Visualisierung von Relationen. Logos Verlag (2004).

[33] Motwani R., Raghavan P.: Randomized Algorithms. Cambridge University Press (1995).

[34] Nissen V.: Einführung in Evolutionäre Algorithmen. Vieweg (1997).

[35] Peirce C.S.: Description of a Notation for the Logic of Relatives, Resulting from Amplification of the Conceptions of Boole's Calculus of Logic. Memoirs of the American Academy of Sciences 9, 317-378 (1870)

[36] Petri A.C.: Kommunikation mit Automaten. Schriften des Institutes für Instrumentelle Mathematik, Bonn (1962).

[37] Rechenberg I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog (1973).

[38] Reisig W.: Petrinetze. Eine Einführung. Springer (1982).

[39] Schmidt, G., Ströhlein, T.: Some aspects in the construction of timetables. IFIP Congress, 516-520 (1974).

[40] Schmidt G., Ströhlein, T.: A Boolean matrix iteration in timetable construction. Linear Algebra and Applications 15, 27-51 (1976).

[41] Schmidt G., Ströhlein, T.: Timetable Construction – An Annotated Bibliography. Computer Journal 23, 307-316 (1980).

[42] Schmidt G., Ströhlein T.: Relationen und Graphen. Springer (1989).
Englische Version: Relations and Graphs. Springer (1993)

[43] Schröder E.: Vorlesungen über die Algebra der Logik (Exakte Logik). Dritter Band: Algebra und Logik der Relative. B.G.Teubner (1895).

[44] Schwefel H.-P.: Evolution and Optimum Seeking. Wiley Interscience (1995).

[45] Tarski A.: On the Calculus of Relations. Journal of Symbolic Logic 6 (3), 73-89 (1941).

[46] Wegener I.: Complexity Theory. Springer (2005).

[47] Wren A.: Scheduling, Timetabling and Rostering – A special Relationship? In: Burke E., Ross P. (Hrsg): The Practice and Theory of Automated Timetabling, LNCS 1153, Springer, 46-75 (1995).

[48] Zierer H.: Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung, Dissertation, Technische Universität München (1988).

[49] Zitzler E., Laumanns M., Thiele L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Giannakoglou K.C. et al. (Hrsg): Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems, 95-100 (2002).

A Zu Kapitel 2

Durch `pi(A)` und `rho(A)` werden zu einer Relation $A : X \leftrightarrow Y$ die natürlichen Projektionen π und ρ auf $X \times Y$ erzeugt.

```
pi(A)
  DECL prod = PROD(A*A^,A^*A);
       pi
  BEG pi = p-1(prod)
      RETURN pi
  END.


rho(A)
  DECL prod = PROD(A*A^,A^*A);
       rho
  BEG rho = p-2(prod)
      RETURN rho
  END.
```

Mit `vec(R)` wird die Vektordarstellung der Relation R berechnet.

```
vec(R) = (pi(R)*R & rho(R))*L1n(R)^.
```

Mit `rel(r,S)` wird für einen Vektor $r : X \times Y \leftrightarrow \mathbf{1}$ die Relationendarstellung des Vektors r berechnet, dabei ist der Parameter S eine beliebige Relation vom Typ $[X \leftrightarrow Y]$, die den Typ der Ausgaberelation angibt.

```
rel(r,S) = pi(S)^*(rho(S) & r*L1n(S)).
```

B Zu Kapitel 3

Folgende Algorithmen werden zur Erzeugung von Startpopulationen für den $(\lambda + \lambda)$-EA zur Approximation von Minimalen Vertex Covern benötigt. Durch `RandomEdge(R)` wird randomisiert eine Kante eines ungerichteten Graphen gewählt, der durch die irreflexive, symmetrische Relation R repräsentiert wird. Eine Kante entspricht dann einer symmetrischen Teilrelation E von R mit $|E| = 2$. Dabei wird die Operation `randomatom` verwendet, die sich auf `randompoint` stützt und randomisiert ein Atom der nichtleeren Eingaberelation R liefert, also eine Relation A des Typs von R mit $A \subseteq R$ und $|A| = 1$

```
randomEdge(R)
  DECL A, E
  BEG A = randomatom(R);
      E = A | A^
      RETURN E
  END.
```

Mit `randomGavril(R)` wird mithilfe des Algorithmus von Gavril und Yannakakis ein Vertex Cover von R berechnet.

```
randomGavril(R)
  DECL G, E, c
  BEG c = On1(R);
      G = R;
      WHILE -empty(G) DO
        E = randomEdge(G);
        c = c | dom(E);
        G = G & -(E*L(E) & R) & -(L(E)*E & R)
      OD
      RETURN c
  END.
```

Das folgende Programm erzeugt eine Startpopulation in Form einer Relation $P : X \leftrightarrow [1..\lambda]$, deren Spalten Vertex Cover von $R : X \leftrightarrow X$ repräsentieren. Der Parameter l ist ein Zeilenvektor der Länge λ.

```
startPop(R,l)
  DECL P,z,p,v
  BEG P = 0(Ln1(R)*l);
      z = l^;
      WHILE -empty(z) DO
        p = point(z);
        v = randomGavril(R);
        P = P |(v*p^);
        z = z & -p
      OD
      RETURN P
  END.
```

C Zu Kapitel 4

Durch `pk(A,B)` wird die Parallele Komposition $(A \| B)$ der Relationen A und B berechnet.

```
pk(A,B)
  DECL X,Y,pk
  BEG  X = On1(A)*On1(B)^;
       Y = O1n(A)^*O1n(B);
       pk = pi(X)*A*pi(Y)^ & rho(X)*B*rho(Y)^
       RETURN pk
  END.
```

D Zu Kapitel 5

D.1 Zu Abschnitt 5.2

Mit `solutions(J,N,b,M)` werden die Spalten der Relation M bestimmt, die Lösungen des Stundenplanproblems (J, N, b) repräsentieren. Das Programm leitet sich aus dem Vektorprädikat φ_{st} ab.

```
solutions(J,N,b,M)
  DECL B, IJ, IB, L, L2, pi, sol
  BEG B = b*b^;
      IJ = I(J);
      IB = I(B);
      L = Ln1(M)^;
      L2 = L^*L1n(J);
      pi = pi(Ln1(J)*L1n(B));
      sol = -(L*( L2*-(pi^*M) |
           ((pk(IJ,-IB) | pk(-N,IB))*M & M) |
           (pk(J,IB)*M & pk(IJ,-B)*M)))
      RETURN sol
  END.
```

Mit `allSolutions(J,N,b)` werden alle Lösungen des Stundenplanproblems (J, N, b) in Form eines Zeilenvektors des Typs $[\mathbf{1} \leftrightarrow 2^{\mathcal{F}\times\mathcal{B}}]$ berechnet.

```
allSolutions(J,N,b)
  DECL S, s, M, sol
  BEG S = On1(J)*On1(b)^;
      s = vec(S);
      M = epsi(s);
      sol = solutions(J,N,b,M)
      RETURN sol
  END.
```

Zur Bestimmung aller Lösungen bis auf Isomorphie wird mit dem folgenden Programm zunächst die Liste aller blockerhaltenden Pemutionen auf $\mathcal{G}$ erzeugt. Dabei ist $B = bb^\top$.

```
bpList(B)
  DECL pi, rho, M, I, L1, L2, bpVec, bpList
  BEG pi = pi(B);
      rho = rho(B);
      M = epsi(Ln1(pi));
      I = I(B);
      L1 = Ln1(pi)^;
      L2 = L(pi);
      bpVec = -(L1*( L2*-(pi^*M & rho^*M) |
              (M & (pk(I,-I) | pk(-I,I) | pk(B,-B))*M)));
      bpList = M * inj(bpVec^)^
      RETURN bpList
   END.
```

Mithilfe dieser Liste wird zu einer Lösung S die Liste aller zu S isomorphen Lösungen erzeugt.

```
isoList(S,bpList) = pk(S,I(L1n(S)^*L1n(S))) * bpList.
```

Mit `removeIso(sol,B,R)` wird ein Teilvektor von *sol* berechnet, der aus jeder Isomorphieklasse von Lösungen genau eine Lösung repräsentiert. Dabei ist R eine beliebige Relation vom Typ $[\mathcal{F} \leftrightarrow \mathcal{G}]$.

```
removeIso(sol,B,R)
  DECL M, bpList,L,v, z, p, S, IS, w
  BEG M = epsi(vec(R));
      bpList  = bpList(B);
      L = L1n(bpList)^;
      v = sol^;
      z = O(v);
      WHILE -empty(v) DO
        p = point(v);
        S = rel(M*p,R);
        IS = isoList(S, bpList);
        w = syq(M,IS)*L;
        v = v & -w;
        z = z|p
        OD
      RETURN z
  END.
```

D.2 Zu Abschnitt 5.4

Durch `selection(A,C,v,P)` wird die Selektion im $(1 + \lambda)$-EA für das durch A und C gegebene Stundenplanproblem realisiert. Dabei ist v das aktuelle Elternindividuum und P die Offspringpopulaition. Der Algorithmus sucht in P nach einer Teillösung, die weniger oder gleich viele Einträge enthält wie v, und liefert, sofern eine solche existiert, diese zurück. Anderenfallls wird v zurückgeliefert.

```
selection(A,C,v,P)
  DECL w, p,d
  BEG  w = v;
       d =  isPartSol(A,C,P)^;
       WHILE -empty(d) DO
         p = point(d);
         IF cardlt(P*p,v) THEN d = d & -p
         ELSE w = P*p; d = 0(d)
         FI
       OD
       RETURN w
  END.
```

Durch `tournDecision(A,C,P,Ch)` und `tournSelection(P,Ch,d)` wird die Turnierselektion im $(\lambda + \lambda)$-EA für das Stundenplanproblem (A, C) realisiert. Dabei berechnet `tournDecision(A,C,P,Ch)` den Entscheidungsvektor d bezüglich der Elternpopulation P und der Offspringpopulation Ch.

```
tournDecision(A,C,P,Ch)
   DECL d, z, p
   BEG d = - isPartSol(A,C,Ch);
       z = -d^;
       WHILE -empty(z) DO
         p = point(z);
         IF cardlt(Ch*p, P*p) THEN d = d | p^ FI;
         z = z & -p
       OD
       RETURN d
   END.
```

Ist d der so berechnete Entscheidungsvektor, berechnet `tournSelection(P,Ch,d)` die neue Generation, aus der Elternpopulation P und der Offspringpopulation Ch.

```
tournSelection(P,Ch,d) = (P & Ln1(P)*d) | (Ch & -(Ln1(P)*d)).
```

Zur Erzeugung einer Startpopulation, die aus Teillösungen des Stundenplanproblems besteht, benötigen wir eine Hilfsfunktion, die zu einer Verfügbarkeitsrelation A, einer Konfliktrelation C und einer Teillösung S eine neue Verfügbarkeitsrelation `newA(A,C,S)` berechnet, die für die in S noch nicht berücksichtigten Meetings angibt, in welchen Zeitschienen (unter Berücksichtigung von S) sie stattfinden können. Es gilt

$$newA(A,C,S)_{mt} \iff A_{mt} \wedge \neg\exists m' : C_{mm'} \wedge S_{m't}.$$

Daraus leitet sich die fogende Funktion ab.

```
newA(A,C,S)= A & -(C*S).
```

Mit `startSol(A,C)` wird randomisiert eine Teillösung des Stundenplanproblems erzeugt. Sukzessive werden randomisiert Meetings gewählt und auf Zeitschienen verteilt, jedesmal wird A durch `newA(A,C,S)` angepasst. Wenn erstmals ein Meeting gewählt wird, für das keine verfügbaren Zeitschienen mehr existieren, stoppt der Algorithmus und gibt die berechnete Teillösung in Vektordarstellung zurück.

```
startSol(A,C)
  DECL dom, S, newA, m, h, s
  BEG dom = Ln1(A);
      S = 0(A);
      newA=A;
      WHILE -empty(dom) DO
        m = randompoint(dom);
        h = newA^*m;
```

```
      IF -empty(h) THEN
        h = randompoint(h);
        S = S | m*h^;
        newA = newA(newA,C,S);
        dom = dom & -m
      ELSE dom = O(dom)
      FI
    OD;
    s = vec(S)
    RETURN s
  END.
```

Sukzessive werden randomisiert Meetings gewählt und auf Zeitschienen verteilt, jedesmal wird A durch `newA(A,C,S)` angepasst. Wenn erstmals ein Meeting gewählt wird, für das keine verfügbaren Zeitschienen mehr existieren, stoppt der Algorithmus und gibt die berechnete Teillösung in Vektordarstellung zurück.

Das folgende Programm entspicht der Testabbilung, welche sich aus dem in Kapitel 5.1 entwickelten Vektorprädikat $\varphi_{(L1)} \cap \varphi_{(L2)}$ ergibt. Es wird also ein Zeilenvektor berechnet, der die Spalten in P repräsentiert, die Teillösungen des Stundenplanproblems sind.

```
isPartSol(A,C,P)
  DECL a, L1, L2, I1, I2, ps
  BEG a = vec(A);
      L1 = Ln1(a)^;
      L2 = L1n(P);
      I1 = I(A*A^);
      I2 = I(A^*A);
      ps = -(L1*((-a*L2 | (pk(C,I2) | pk(I1,-I2))*P ) &  P))
      RETURN ps
  END.
```

Der folgende Algorithmus implementiert den $(1 + \lambda)$-EA mit herkömmlicher Mutation (siehe Kapitel 3.3.1). Der Vektor v repräsentiert ein Startindividuum, p bestimmt die Mutationswahrscheinlichkeit, l ist die Allrelation vom Typ $[\mathbf{1} \leftrightarrow \lambda]$

und damit hat jede erzeugte Population die Grösse λ. Der Vektor t bestimmt die Anzahl der Iterationen.

```
randomEA(A,C,v,p,l,t)
   DECL z, w, P
   BEG z = t;
       w = v;
       WHILE -empty(z) DO
         P = mutation(w*l,p);
         w = selection(A,C,w,P);
         z = z & -point(z)
       OD
       RETURN w
    END.
```

Mit `uniqueEA(A,C,v,l,t)` wird der $(1 + \lambda)$-EA mit eindeutigkeitserhaltender Mutation, (siehe Kapitel 5.4) implementiert. Dabei ist v ein Startindividuum (also eine Teillösung), l ist ein Vektor des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ mit $|l| = \lambda$, so dass die Grösse der Populationen λ beträgt. Der Vektor t bestimmt die Anzahl der Iterationen.

```
uniqueEA(A,C,v,l,t)
   DECL  z, w, P, L
   BEG z = t;
       w = v;
       L=Ln1(inj(l))^;
       WHILE -empty(z) DO
         P = uniqueMutation(w*L,l,A);
         w = selection(A,C,w,P);
         z = z & -point(z)
       OD
       RETURN w
   END.
```

Für den $(\lambda + \lambda)$-EA wird eine Startpopulation, bestehend aus λ Teillösungen benötigt. Diese wird mit `startPop(A,C,l)` erzeugt, wobei l ein Zeilenvektor der

Länge λ ist.

```
startPop(A,C,l)
  DECL a,P, z, S, s,p
  BEG a = vec(A);
      P = O(a*l);
      z = l^;
      WHILE -empty(z) DO
        S = startSol(A,C);
        s = vec(S);
        p = point(z);
        z = z & -p;
        P = P | s*p^;
      OD
      RETURN P
  END.
```

Mit `randomEA2(A,C,S,pM,pC,t)` wird der $(\lambda + \lambda)$-EA für das Stundenplanproblem realisiert, mit herkömmlicher Mutation und Crossover, wie in Kapitel 3.3.1 und 3.3.2 beschrieben. $S : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$ ist die Startpopulation, bestehend aus λ Teillösungen des Stundenplanproblems in Vektordarstellung. Die Relationen pM und pC bestimmen Mutations- und Crossoverwahrscheinlichkeit, der Vektor t die Anzahl der Iterationen.

```
randomEA2(A,C,S,pM,pC,t)
  DECL P,z,Ch,d
  BEG P = S;
      z = t;
      WHILE -empty(z) DO
        Ch = crossover(P,pC);
        Ch = mutation(Ch, pM);
        Ch = Ch*randomperm(L1n(Ch)^);
        d = tournDecision(A,C,P,Ch);
        P = tournSelection(P,Ch,d);
        z = z & -point(z)
      OD
      RETURN P
  END.
```

Mit `uniqueEA2(A,C,S,l,pC,t)` wird der $(\lambda + \lambda)$-EA mit eindeutigkeitserhaltender Mutation und Crossover, wie in Kapitel 5.4 beschrieben druchgeführt. Dabei ist $S : \mathcal{M} \times \mathcal{H} \leftrightarrow [1..\lambda]$ eine Startpopulation wie oben, l ein Vektor des Typs $[\mathcal{M} \times \mathcal{H} \leftrightarrow \mathbf{1}]$ mit $|l| = \lambda$, pC und t wie oben.

```
uniqueEA2(A,C,S,l,pC,t)
  DECL P,z,Ch,d
  BEG P = S;
      z = t;
      WHILE -empty(z) DO
        Ch = uniqueCrossover(P,A,pC);
        Ch = uniqueMutation(Ch,l,A);
        Ch = Ch*randomperm(L1n(Ch)^);
        d = tournDecision(A,C,P,Ch);
        P = tournSelection(P,Ch,d);
        z = z & -point(z)
      OD
      RETURN P
  END.
```