Ein neuer Ansatz zur Fahrplanoptimierung im ÖPNV: Maximierung von zeitlichen Sicherheitabständen

Inaugural - Dissertation

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Zülfükar Genç

aus Hazar-Elazig/Türkei

April 2003

Berichterstatter:	Prof. Dr. Ewald Speckenmeyer
	Prof. Dr. Rainer Schrader
Tag der mündlichen Prüfung:	6. Juni 2003

Danksagung

Mein besonderer Dank geht an meinem Doktorvater Prof. Dr. Ewald Speckenmeyer für die engagierte Betreuung, seine Diskussionsbereitsschaft und die vielen wertvollen Ratschläge.

Des Weiteren bedanke ich mich bei der gesamten Arbeitsgruppe für die angenehme Arbeitsatmosphäre und die gute Gemeinschaft.

Meinen Kollegen Dr. Bert Randerath, Gero Lückemeyer und Dr. Stefan Porschen möchte ich noch danken für die kritische Durchsicht dieser Arbeit.

Schließlich danke ich ganz besonders meinen Eltern und Geschwistern sowie meinen Freunden, die mich das ganze Studium hindurch unterstützt und ermutigt haben, auf deren Rückhalt ich mich jederzeit verlassen konnte und die, auch in schwierigen Phasen, immer Verständnis zeigten.

Inhaltsverzeichnis

Zusammenfassung

Der Fahrplan bildet ein wichtiges Element der Verkehrsplanung. Er legt die Ankunfts- und Abfahrtszeiten der einzelnen Linien im Streckennetz fest.

Die bislang verwendete Software hilft bei der manuellen Erstellung am Bildschirm und ermöglicht die Visualisierung und Simulation von Fahrplänen. Zur besseren Unterstützung des Planers fehlen jedoch Systeme zur automatischen Generierung zulässiger und in Hinblick auf die Zielvorstellung optimierter Fahrpläne.

Die vorliegende Arbeit beschäftigt sich mit einem neuen Ansatz zur Optimierung von Fahrplänen. Das der Fahrplangestaltung zu Grunde liegende Streckennetzwerk besteht aus einer Menge von Stationen und Linien, denen Takte zugeordnet sind. Den Streckenabschnitten zwischen den Stationen sind Fahrzeiten zugeordnet.

Das Ziel der Fahrplanoptimierung, die in dieser Arbeit betrachtet wird, besteht darin, die Abfahrtszeiten der Linien an den Anfangsstationen so festzulegen, dass die Ankunftszeiten an den einzelnen Stationen möglichst gleichmäßig verteilt sind. Damit soll erreicht werden, dass kleine Störungen, wie sie häufig bei Nahverkehrssytemen auftreten, nur geringe, zeitlich begrenzte Auswirkungen auf den Linienverkehr haben.

Im ersten Kapitel werden die einzelnen Planungsprozesse im öffentlichen Personennahverkehr (ÖPNV) erläutert und es wird ein Überblick über einige aus der Literatur bekannte Ansätze zur Fahrplanoptimierung gegeben. Kapitel 2 stellt die Modellierung zur Fahrplanoptimierung, die in dieser Arbeit untersucht wird, dar.

In Kapitel 3 werden Reduktionen des Streckennetzwerks präsentiert, mit deren Hilfe die Zielfunktion und obere Schranken schneller ermittelt werden können. Die Reduktionen nutzen Relationen, die auf der Menge der Stationen und auf der Menge der Kanten eines speziellen noch zu definierenden Multi-Linienkonflikt-Graphen, der aus dem Streckennetzwerk gebildet wird, definiert sind. Darüber hinaus enthält Kapitel 3 Ergebnisse zur Zeitkomplexität des hier untersuchten Fahrplanoptimierungsproblems.

Ein Branch-and-Bound-Algorithmus zur Lösung des Fahrplanoptimierungsproblems wird in Kapitel 4 vorstellt. Es werden Heuristiken präsentiert, um mit deren Hilfe gute Anfangslösungen für den Branch-and-Bound-Algorithmus zu erhalten. Kapitel 4 enthält noch einen Ansatz zur Zerlegung des Streckennetzwerks in Teilnetze, so dass es ausreicht, optimale Fahrpläne für die Teilnetze zu berechnen, um daraus einen optimalen Fahrplan für

das gesamte Streckennetzwerk zu erhalten. Dazu werden die zweifachen Zusammenhangskomponenten des Linienkonflikt-Graphen, der aus dem Streckennetzwerk gebildet wird, betrachtet.

In Kapitel 5 wird das Fahrplanoptimierungsproblem als ganzzahliges lineares Programm dargestellt, um dann mittels CPLEX, einem Programmpaket zur Lösung linearer und gemischtganzzahliger Variablen, gelöst zu werden. Mit Hilfe von Reduktionen lässt sich die Zahl der Variablen und Gleichungen so verringern, dass CPLEX in vertretbarer Zeit gute Lösungen liefert.

Kapitel 6 präsentiert experimentelle Ergebnisse zur Lösung des Fahrplanoptimierungsproblems, die mit einem Branch-and-Bound-Verfahren und mit CPLEX auf Basis der ganzzahligen LP-Formulierung anhand der Daten des Streckennetzwerks der Kölner Verkehrs-Betriebe (KVB) einerseits und auf künstlich erzeugten Streckennetzwerken andererseits, erzielt worden sind. Die Arbeit schließt mit einer Zusammenfassung und einem Ausblick auf mögliche zukünftige Entwicklungen dieses Ansatzes.

Abstract

An important element of train scheduling is the timetable. It determines the arrival and departure times for the different train lines in a public transportation network. The software which has been hitherto employed features the manual creation, visualisation and simulation of timetables on the screen. There is, however, a lack of systems which allow the automatic generation of valid and teleologically optimized timetables.

This thesis deals with a new method of timetable optimization. The network which forms the basis of timetable creation consists of a set of stations and lines, to which certain time periods are assigned. The connections between stations correspond to travelling times. The goal of timetable optimization, as analyzed in this thesis, is to set the departure time of the respective lines at the first station in order to evenly distribute the arrival times. The intention behind that is to limit the temporal effect of small disturbances, as they happen quite often in public transport networks.

In the first chapter, we discuss the different planning processes in public transport systems and give an overview of existing models of timetable optimization. Chapter 2 presents the models of timetable optimization that will form the basis of our further work. In chapter 3, we introduce reductions of the network leading to faster calculation of the optimization function and upper bounds. The reductions make use of relations that are defined on the set of stations and on the set of the edges of a special yet-to-be-defined multi-line-conflict-graph formed out of the network. Moreover, chapter 3 consists of results explaining the time-complexity of the central timetable optimization problem of this thesis. A branch and bound algorithm to solve the timetable optimization problem is presented in chapter 4. Heuristics will be employed to get feasible solutions that can be used with the branch and bound algorithm. Chapter 4 contains a method to split the network into sub-networks, making it possible to calculate optimal timetables of the sub-networks and to combine them into an optimal timetable for the whole network. This is archieved by focusing on the bi-connected components of the line-conflict-graph that was formed out of the network. In chapter 5, the timetable optimization problem is described as an integer linear program, before solving it with the help of CPLEX, a program package to solve linear and mixed integer variables. As a result of the described reductions of the number of variables and equations, CPLEX can give good solutions in a considerably short amount of time. Chapter 6 discusses experimental results for solving the timetable optimization problem. These were produced by the branch and bound algorithm and CPLEX on the basis of integer LP-formulation of data of the cologne tram network on the one hand and through artificially created networks on the other hand. The thesis concludes with a summary and a suggestion for further research in this area.

Kapitel 1

Einleitung und Überblick

1.1 Einleitung

Angesichts knapper finanzieller Ressourcen und fortschreitender Privatisierung und Globalisierung der Märkte sind öffentliche Verkehrunternehmen einem steigenden Wettbewerbsdruck unterworfen. Um in Zukunft weiter wettbewerbsfähig gegenüber dem Individualverkehr und anderen Verkehrsträgern zu bleiben, setzen ÖPNV-Betriebe auf die Zufriedenkeit der Kunden und auf den effizienten Einsatz ihrer Ressourcen wie Bahnen und Personal. Eine entscheidende Rolle spielen dabei computerunterstützte Planungsmethoden. In den meisten Planungsschritten, mit denen sich Unternehmen im ÖPNV auseinandersetzen, kann durch mathematische Optimierungsmethoden die Arbeit des Planers im ÖPNV unterstützt werden.

1.1.1 Planungsprozesse im ÖPNV

In der Regel teilt man die Planung eines öffentlichen Nahverkehrssystems in drei Phasen:

- strategische Planung
- taktische Planung
- operative Planung

Die Aufgaben der drei Phasen werden im Folgenden kurz skizziert. Genaueres kann man aus [BWZ97], [Grö99] und [Lin00] entnehmen.

Strategische Planung

Bei der strategischen Planung geht es um die Ermittlung des Verkehrsbedarfs und die Entscheidung über das Systemangebot.

- *Ermittlung des Verkehrsbedarfs:*
  Als erstes wird der Planungsraum in Teilgebiete zerlegt. Diese Zerlegung hängt von bestehenden Verkehrsverbindungen, Bevölkerungszahl, Siedlungsstruktur, usw. ab. Nach der Aufteilung wird der Verkehrsbedarf geschätzt. Dafür wird üblicherweise eine sogenannte Quelle/Ziel-Matrix ermittelt. Jeder Eintrag (i, j) gibt die Anzahl der Personen an, die von Gebiet i nach j in einem festem Zeitraum (z.B. einer Stunde) fahren. Diese Werte müssen für jedes Gebiet erhoben werden, dabei arbeitet man mit statistischen Methoden.

- *Entscheidung über das Systemangebot:*
  Nachdem der Verkehrsbedarf ermittelt wurde, wird festgelegt, mit welchen Verkehrsmitteln (S-Bahn, U-Bahn, Bus usw.) der Bedarf gedeckt werden kann. Diese Überlegungen werden häufig von wirtschaftlichen Interessen und politischen Zielsetzungen einzelner Gruppen dominiert.

Taktische Planung

- *Netzstrukturplanung:*
  Bei der Netzstrukturplanung wird festgelegt, wo Betriebshöfe und Haltestellen eingerichtet werden.

- *Linienplanung:*
  Nach Festlegung des Streckennetzes und der Haltestellen mit Hilfe der Quelle/Ziel-Matrix wird das Liniennetz festgelegt. Dabei versucht man, aus der Menge möglicher Linienwege eine Teilmenge auszuwählen, so dass die Zahl der Direktfahrer maximiert wird und alle Passagiere von ihrer Start- zur Zielstation kommen können. ([BZ97],[BKZ95], [Bus98]).

- *Fahrplan-Planung:*
  In der Fahrplangestaltung werden die Ankunfts- und Abfahrtszeiten der Linien an den Stationen festgelegt. Dabei werden verschiedene Ziele betrachtet. Ein Ziel kann es sein, die Ankunfts- und Abfahrtszeiten so festzulegen, dass die Umsteigezeiten zwischen den Linien kurz sind, damit der Umsteigevorgang möglichst geringe Fahrzeitverzögerung für die Fahrgäste mit sich bringt. Dabei kann zusätzlich darauf geachtet werden, dass die Kosten für die Verkehrsunternehmen so gering wie möglich sind. Wir werden genauer auf die verschiedenen Ansätze zur Fahrplanoptimierung im Abschnitt 1.2 eingehen.

Operative Planung

- *Fahrzeugumlaufplanung/Fahrzeugeinsatzplanung (vehicle scheduling):*
  Bei der Fahrzeugumlaufplanung geht es um die Minimierung der benötigten Fahrzeuge. Die Fahrzeugeinsatzplanung beschäftigt sich mit der Zuordnung von Fahrzeugen

zu den einzelnen Fahrten. Ziel ist es, so wenige Fahrzeuge wie möglich einzusetzen und die Kosten für Leerfahrten so gering wie möglich zu halten. Dabei wird das Fahrzeugumlaufproblem als Mehrgüterflussproblem dargestellt und mit Hilfe von Branch-and-Cut-Verfahren gelöst ([Löb98],[LS95], [GLV96], [MK02]).

- *Dienstplanung(crew scheduling)/Dienstreihenfolgeplanung:*
  Die Aufgabe der Dienstplanung besteht in der Zuordnung von Personal zu den einzelnen Fahrten, mit dem Ziel, die Kosten zu minimieren, d.h. so wenig Personal wie möglich einzusetzen, um den Fahrplan zu erfüllen. Dabei sind gesetzliche und tarifliche Bestimmungen sowie betriebliche Vereinbarungen zu beachten. Die Ansätze zur Lösung der Dienstplanungsaufgaben beruhen auf Formulierungen als Set-Partitioning- bzw. Set-Covering-Probleme, die sich mit Hilfe ganzzahliger LP-Techniken lösen lassen. Ist die Dienstplanung erfolgt, so bildet den nächsten Schritt die Dienstreihenfolgeplanung. Dabei werden unterschiedliche Dienstarten (Früh- oder Spätdienst, geteilte Dienste, etc.) über Wochen so verteilt, dass eine faire Zuweisung von Diensten an Fahrer möglich ist ([FLO00], [BLSV99], [KF00], [CFT+97]).

1.1.2 Qualität im Nahverkehr

Bei dem Begriff der Qualität im Nahverkehr unterscheidet man zwischen Planungs- und Ausführungsqualität. Unter Planungsqualitäten versteht man kurze Reisezeiten, möglichst durchgängige Verbindungen, komfortable Umsteigebeziehungen, häufige Bedienung, sowie guter Zugang zum Verkehrsangebot, d.h. kurze Wege zu den Haltestellen. Unter Ausführungsqualität versteht man die Pünktlichkeit und Anschlusssicherung, Sicherheit in Zügen und auf Bahnhöfen, technische Funktionstüchtigkeit, positives Erscheinungsbild, gute Information der Fahrgäste in Regel- sowie bei Störfällen, Sauberkeit in Zügen und auf Bahnhöfen ([fWVuT01]).

Qualität im ÖPNV ist eine Mischgröße aus messbaren und subjektiv bewertbaren Kriterien. Am Verkehrwissenschaftlichen Institut der RWTH Aachen wurde eine Untersuchung durchgeführt, mit dem Ziel, die Qualität des ÖPNV rechenbar zu machen, um Marktwirksamkeit von Qualitätsverbesserungen insbesondere im Komfortbereich quantifizieren zu können. Dabei wurde aufgezeigt, dass die komplexen Wirkungszusammenhänge bei der Wertfindung für die ÖPNV-Qualität und ihre Marktwirtschaftlichkeit rechenbar gemacht werden können, also sich ohne die Notwendigkeit aufwendiger Befragungen ermitteln lassen ([Wal96]).

Um einen Mindeststandard für den Nahverker in NRW zu erreichen, hat das Ministerium für Wirtschaft und Mittelstand, Energie und Verkehr des Landes NRW eine Broschüre “Masterplan Qualität” herausgebracht, der eine “Qualitätscharta für den Nahverkehr in NRW” enthält und alle Verkehrsunternehmen aufgerufen, sich auf die Qualitätscharta zu verpflichten. Diese Qualitätscharta spiegelt in zehn Punkten die aus zahlreichen Markterhebungen bekannten Erwartungen der Kunden an einen qualitativ hochwertigen ÖPNV wider. Das sind z.B. pünktliche Verbindungen, schnelle Information über Verspätungen,

saubere Fahrzeuge oder zuverlässige Fahrplan- und Tarifauskünfte.

1.1.3 Anforderungen an einem Fahrplan

Der öffentlichen Personen-Nahverkehr (ÖPNV) wird in Zukunft weiter an Bedeutung gewinnen, da in dichtbesiedelten Regionen die Attraktivität des motorisierten Individualverkehrs aufgrund von Überlastungen des Straßennetzes abnimmt. Damit der ÖPNV auch angenommen wird, muss die Qualität stimmen. Zu den wichtigsten Faktoren gehören Reisezeit und Zuverlässigkeit – diese hängen vom Taktfahrplan ab.

Damit der ÖPNV gegenüber dem Individualverkehr konkurrenzfähig bleibt, braucht man einen Taktfahrplan, der zuverlässig und dicht ist. Dabei sind "Zuverlässigkeit" und "Dichtheit" Gegensätze. Denn je dichter ein Fahrplan ist, desto weniger Spielraum bleibt, um eventuelle Verspätungen aufzufangen. Weitere Anforderungen sind:

- *Kurze Reisezeiten*: Die Reisezeit zwischen zwei beliebigen Stationen sollte so kurz wie möglich sein, wobei beim Umsteigen, falls keine direkte Verbindung besteht, unnötige Wartezeiten vermieden werden sollen. Die Zahl der Fahrten ohne Umsteigen sollte möglicht groß sein.
- *Pünktlichkeit und Anschlusssicherung*: Die Pünktlichkeit spielt eine herausragende Rolle, da Unpünktlichkeit von den Fahrgästen nur bis zu einem bestimmten Grad akzeptiert wird. Wichtig im Falle von Verspätungen ist die umgehende und umfassende Information der Fahrgäste. Fehlt diese, sind die Fahrgäste besonders verärgert ([fWVuT01]).

1.2 Ein Überblick über die Ansätze zur Fahrplanoptimierung

Bei der Fahrplanerstellung stellen die wichtigsten Kriterien zur Beurteilung der Güte eines Fahrplans aus der Sicht der Verkehrsbetriebe die Wirtschaftlichkeit und die Akzeptanz durch die Fahrgäste dar. Die Akzeptanz betreffend müssen mehrere Punkte beachtet werden wie Anschlüsse zu anderen Linien, Umsteigebeziehungen, Taktdichte etc. Bei Veränderungen bestehender Linien wird dabei unter anderem auf Kundenanregungen (Beschwerden, Wünsche) reagiert, bei Neuplanungen werden Auswertungen wie Fahrgastzahlen, Bevölkerungsdichte und weitere Kriterien herangezogen.

Ein wichtiges Element der Verkehrsplanung ist der Fahrplan. Der Fahrplan legt fest, wo und wann sich die Bahnen der einzelnen Linien im Streckennetz befinden. Dabei müssen verschiedene Anforderungen erfüllt werden.

Die bislang verwendete Software hilft bei der manuellen Erstellung am Bildschirm und ermöglicht die Visualisierung und Simulation von Fahrplänen. Zur wirklichen Un-

terstützung des Planers fehlen jedoch Systeme zur automatischen Generierung zulässiger und in Hinblick auf die Zielvorstellung optimierter Fahrpläne.

Es gibt zwei unterschiedliche Ansätze in der Literatur. Der erste Ansatz versucht die Reisezeit der Passagiere so gering wie möglich zu halten, in dem die Summe aller Umsteigezeiten minimiert wird. Mit dem zweiten Ansatz möchte man unter Berücksichtigung der Betriebskosten die Umsteigezeiten minimieren. Die beiden Ansätze unterscheiden sich auch in der Modellierung. Im Folgenden wollen wir diese Ansätze näher vorstellen. Im weiteren wird ein dritter theoretischer Ansatz vorgestellt, den wir aber bei der Berechnung von oberen Schranken in einem Branch-and-Bound-Verfahren benutzen.

1.2.1 Minimierung der Summe aller Umsteigezeiten

Umsteigevorgänge sind nicht zu vermeiden, da nicht zu jedem Fahrtwunsch eine direkte Verbindung angeboten werden kann. Das Ziel dieses Ansatzes ist es, die von den Fahrgästen als störend empfundenen Umsteigevorgänge so attraktiv wie möglich zu gestalten. Die Umsteigezeit hängt dabei von der im Fahrplan vorgesehenen Ankunftszeit der Linie ab, mit der man im Umsteigebahnhof ankommt und der Abfahrtszeit derjenigen Linie, mit der man weiterfährt.

Bei der Minimierung der Summe aller Umsteigezeiten ist zusätzlich zum Streckennetzwerk eine Quelle-Ziel Matrix gegeben, die die Anzahl der Reisenden zwischen den Stationen angibt. Dieses Problem wird als QSAP (quadratic semi-assignment problem) formuliert.

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{h=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{m} c_{ihjk}\, x_{ih}\, x_{jk} && (1.2.1)\\ & \sum_{h=1}^{m} x_{ih} = 1 && \forall i \in L \\ & x_{ih} \in \{0,1\} && \forall i \in L, h \in S \end{aligned}$$

Dabei ist c_{ihjk} die Summe der Wartezeiten der Passagiere, die von Linie i zu Linie j umsteigen wollen, falls Linie i zum Zeitpunkt h startet und Linie j zum Zeitpunkt k.

$$x_{ih} = \begin{cases} 1 & \text{falls Abfahrtszeit } h \text{ der Linie } i \text{ zugeordnet wird} \\ 0 & sonst \end{cases}$$

Domschke, Forst und Voß ([DFV92], [DV95], [Dom89],[Voß92]) untersuchten den Einsatz von Tabu-Search-Techniken beim QSAP und verglichen diese mit Simulated Annealing Techniken. Dabei wurde für den Test als Steckennetzwerk ein Teil des U-Bahnnetzes von (West-)Berlin genommen und die symmetrische Kosten-Matrix (c_{ihjk}) wurde zufällig generiert mit Werten von 0 bis 99. In experimentellen Resultaten mit 10, 11 und 12 Linien

und 10-minütigem Takt waren die gemessenen Ergebnisse für die Tabu-Search-basierten Verfahren immer besser als die für Simulated-Annealing-basierte Verfahren.

M. Krista [Kri95] untersucht auch die Minimierung der Umsteigezeiten, aber dort wird jeder Umsteigebeziehung ein Gewicht zugeordnet (anstelle der Quelle-Ziel-Matrix), das die Bedeutung der Umsteigebeziehung angibt. Krista entwickelt genetische Algorithmen, um Anschlussoptimierungsprobleme zu lösen.

Ballungsräume werden in den Hauptverkehrszeiten mit einem dichten Takt bedient, so dass die Wartezeiten der Fahrgäste auf Anschlüsse gering sind. Anschlussoptimierung ist dort sinnvoll für das Umsteigen von einem Verkehrsmittel auf ein anderes, z.B. von Bahn auf Bus. Dieser Ansatz ist am sinnvollsten im Fernverkehr, wo die Bahnen stündlich verkehren. In Städten ist dieser Ansatz sinnvoll in Abendzeiten und am Wochenende, wenn die Linientakte größer sind.

1.2.2 Periodic Event Scheduling Problem (PESP)

Bei der Minimierung der Umsteigezeiten versucht man durch geschickte Wahl der Abfahrtszeiten die Umsteigezeit zwischen Linien mit starken Umsteigebeziehungen zu verkürzen. Dies liefert gute Fahrpläne oft jedoch um den Preis erhöhter Betriebskosten, da durch die Wahl der Abfahrtszeiten die Umlaufbildung und damit der Fahrzeugbedarf beeinflusst wird. In der Praxis müssen sowohl die Umsteigezeiten der Passagiere als auch die Anzahl der Fahrzeuge berücksichtigt werden ([LP01], [LN02]). Mit Hilfe des PESP-Ansatzes werden diese beiden Ziele kombiniert.

Das Periodic Event Scheduling Problem (PESP) wurde von Paolo Serafini and Walter Ukovich [SU89] zum ersten Mal vorgestellt. Es ist ein Ansatz zur Modellierung von periodischen Fahrplänen im öffentlichen Verkehr. Ein PESP besteht aus periodischen Ereignissen und periodischen Nebenbedingungen. Ein Ereignis ist ein Tripel aus Linie, Station und Ankunfts- oder Abfahrtszeit. In einem Graphen wird ein Ereignis als ein Knoten repräsentiert. Eine periodische Nebenbedingung (Restriktion) ist eine Bedingung an zwei Ereignisse, welchen zeitlichen Abstand diese beiden Linien einhalten müssen. Im Graphen wird eine periodische Restriktion durch eine gerichtete Kante repräsentiert mit einer oberen und unteren Schranke für die Differenz beider Ereignisse.

Die Ungleichung für zwei Ereignisse i und j und deren Ankunfts- oder Abfahrtszeiten π_i und π_j mit Abstand mindestens l_{ij} und höchstens u_{ij} sieht wie folgt aus:

$$l_{ij} \leq \pi_j - \pi_i + p_{ij} \cdot T \leq u_{ij}$$

wobei T die Periode aller Ereignisse darstellt und p_{ij} eine ganze Zahl ist.

Mit diesen Restriktionen können die Fahrzeit von einer Station zur nächsten, die Aufenthaltszeit an einer Station, die Umsteigezeit von einer Linie auf eine andere bei Berücksichtigung einer Mindestübergangszeit, die Vorgabe einer ausgeglichenen Zugfolge bei mehreren Linien auf gemeinsam befahrenen Strecken, das Begegnungsverbot auf eingleisigen

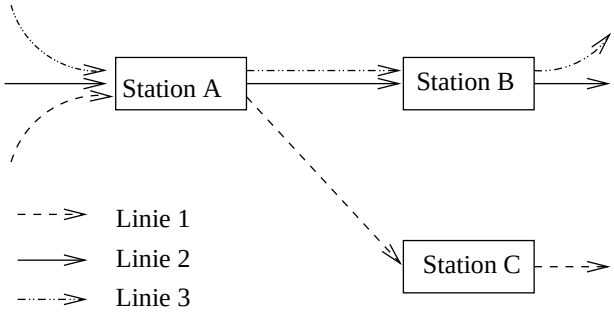


Abbildung 1.1: Beispiel für ein Streckennetzwerk

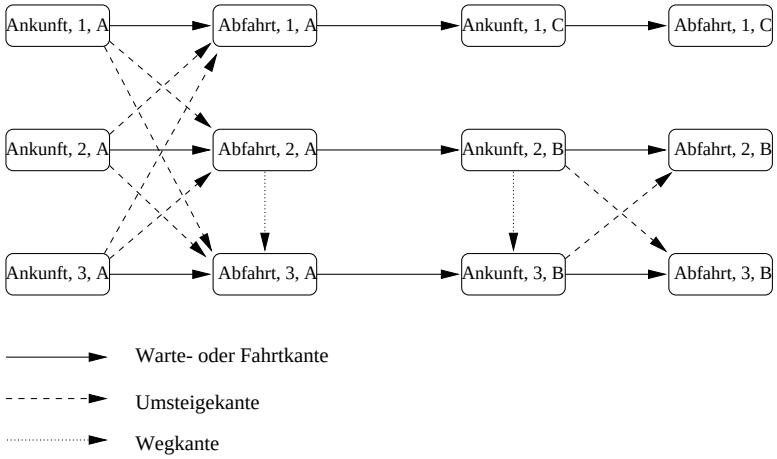


Abbildung 1.2: Ereignis-Graph zum Streckennetz in Abb. 1.1.

Sreckenabschnitten, sowie die Einhaltung von Mindestfolgezeiten für nacheinander verkehrende Fahrzeuge modelliert werden.

Das Entscheidungsproblem für eine gegebene PESP-Instanz, ob ein periodischer Fahrplan existiert, der alle Nebenbedingungen erfüllt, ist NP-vollständig [SU89] und [LP01].

Wenn man den Kanten des Ereignis-Graphen noch Gewichte hinzufügt, so lassen sich verschiedene zulässige Lösungen bewerten und miteinander vergleichen. Ein häufig verwendetes Gewicht stellt die Umsteigewartezeit aller Passagiere dar.

Um PESP-Instanzen zu lösen wurde mit verschiedenen Ansätzen gearbeitet. Serafini und Ukovich wählten Backtracking, um das Entscheidungsproblem zu lösen. Um gute Fahrpläne zu erzeugen entwickelten Nachtigall und Voget genetische Algorithmen [NV96]. Odijk und Nachtigall verwenden eine Branch-and-Bound-Methode um PESP zu lösen [Odi97], [Nac98]. Lindner entwickelt ein neues Branch-and-Cut-Verfahren zur Bearbeitung von PESP-Instanzen [Lin00].

1.2.3 Polygon Scheduling

Eine abstrakte Betrachtung dieser Fahrplan-Problematik stellt die Modellierung als Polygon-Scheduling-Problem dar. Vince, Cerny und Guldan ([Vin89], [CG87], [Gul80]) betrachten folgendes Fahrplanoptimierungsproblem: Gegeben ist eine Menge von Linien, die eine Station oder eine Strecke gemeinsam befahren. Jede Linie befährt die Station mit einer festen Periode. Wie lassen sich die Ankunftszeiten der Linien festlegen, so dass der zeitliche Abstand zweier aufeinander folgender Linien an einer Station oder auf der Strecke maximal wird?

Eine mathematische Beschreibung dieses Problems wird mit Polygonen in einem Kreis gegeben. Wenn man die gemeinsame Periode der Linien als einen Kreis repräsentiert, so ist jede Ankunftszeit einer Linie ein Punkt auf diesem Kreis. Wenn man für einen festen Fahrplan die Ankunftszeiten einer Linie verbindet, so bilden diese ein regelmässiges Polygon. Der zeitliche Abstand zwischen zwei aufeinander folgenden Bahnen an der gemeinsamen Station wird auf diesem Kreis repräsentiert durch den Abstand der Eckpunkte der Polygone. Jede Linie wird in diesem Modell repräsentiert durch ein regelmäßiges Polygon. Hier stellt sich die Frage, wie sich die Polygone in dem Kreis so platzieren lassen, dass der minimale Abstand zweier benachbarter Ecken maximal wird.

Guldan [Gul80] gibt eine algorithmische Lösungsmethode für dieses Problem an. Dazu wird der Lösungsraum in exponentiell viele Teile zerlegt und jedes Teilproblem durch einen gewichteten azyklischen Graphen dargestellt, in welchem sich eine Lösung als längste Wegeproblem ermitteln lässt. Einen anderen Lösungsansatz stellt Vince [Vin89] vor. Dazu zeigt er, dass es eine optimale Lösung gibt, deren Koordinaten rational mit beschränkten Zählern und Nennern sind. Dadurch kann die Suche in einem endlichen Lösungsraum fortgesetzt werden. Für jeden möglichen Nennerwert wird ein gewichteter vollständiger Graph erzeugt. Auf jedem wird das Polygonproblem gelöst. Vince gibt in seiner Arbeit

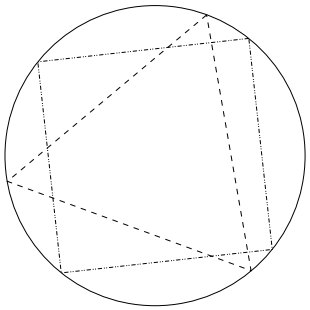


Abbildung 1.3: Zwei Polygone in einem Kreis

auch für zwei und drei Ereignisse (Linien, Polygone) eine optimale Lösung und zeigt die NP-Vollständigkeit des zugehörigen Entscheidungsproblems, indem er das Entscheidungsproblem auf das Graphfärbungsproblem zurückführt.

Burkard [Bur86] und Hurink [Hur92] verallgemeinern später das Zielkriterium, um bei der Betrachtung von Linien auf einer gemeisamen Strecke als Optimierungsziel durchschnittliche Wartezeit oder maximale Wartezeit minimieren zu können.

Im Folgenden wird das allgemeine Polygon-Problem genauer vorgestellt, da es in dieser Arbeit verwendet wird:

Gegeben sei ein Kreis C der Länge A, n reguläre Polygone $P = \{P_1, \ldots, P_n\}$ wobei jedes Polygon P_i m_i Ecken hat. Sei $m = \sum_{i=1}^{n} m_i$ die Anzahl aller Ecken aller Polygone im Kreis.

Auf C definiert man ein Koordinatensystem, in dem ein Punkt mit 0 bezeichnet wird, dieser entspricht dem Ursprung. Jedem Punkt x auf C ordnet man einen Wert x zu, welcher der Länge des Kreisbogens vom Ursprung in positiver Richtung entspricht.

Wird eine Ecke eines Polygons P_i auf den Wert a_i gesetzt, so liegen alle Ecken des Polygons auf den folgenden Punkten $\{a_i + k \cdot A/m_i | k = 0, \ldots, m_i - 1\}$.

Da sich stets eine Ecke des Polygons P_i im Koordinatenintervall $[0, A/m_i[$ befindet, benötigt man zur Beschreibung der Position von n-regulären Polygonen in einem Kreis nur ein n-Tupel $a = (a_1, a_2, \ldots, a_n)$ mit $0 \leq a_i < A/m_i$. a_i legt die Position einer Ecke des Polygons P_i fest, womit das Polygon P_i eindeutig bestimmt ist. Sei u_k $(k = 1, \ldots, m)$ der Abstand zwischen zwei benachbarten Ecken auf dem Kreis C. Der Spezialfall $u_k = 0$ bedeutet, dass zwei benachbarte Ecken zusammenfallen. Eine Ecke f heißt *benachbart* zu einer anderen Ecke e, wenn der Abstand in positiver Richtung (gegen den Uhrzeigersinn) zwischen den Ecken e und f kleiner ist als alle Abstände in positiver Richtung zwischen der Ecke e und allen anderen Ecken.

Hurink und Burkard ([Gul80], [Bur86], [Hur92]) betrachten dann folgende allgemeine

Zielfunktion:

$$(\|u\|_p)^p = \sum_{k=1}^{m} |u_k|^p \underset{u_k \geq 0}{=} \begin{cases} \max_{1 \leq k \leq m} u_k, \text{ falls } p = \infty \\ \sum_{k=1}^{m} u_k^p, \text{ falls } -\infty < p \leq 0 \text{ oder } 1 \leq p < \infty \\ \min_{1 \leq k \leq m} u_k, \text{ falls } p = -\infty \end{cases}$$

Dabei ist p ein beliebig fester Wert mit $-\infty \leq p \leq 0$ oder $1 \leq p \leq \infty$. Für $1 \leq p \leq \infty$ versucht man die Zielfunktion über alle n-Tupel $a = (a_1, a_2, \ldots, a_n)$ mit $0 \leq a_i < A/m_i$ zu minimieren und für $-\infty \leq p \leq 0$ zu maximieren. Spezialfälle dieses Optimierungsproblems sind die folgenden:

1. $p = \infty$: In diesem Fall wird die maximale Wartezeit minimiert.

$$\min_{a \in [\mathbb{R}/A]^n} \quad \max_{1 \leq k \leq m} u_k$$

2. $p = 2$: Hier wird die durchschnittliche Wartezeit minimiert.

$$\min_{a \in [\mathbb{R}/A]^n} \quad \sum_{k=1}^{m} u_k^2$$

3. $p = -\infty$: Dies ist der für uns interessante Fall. Es wird nämlich das minimale Sicherheitsintervall (Wartezeit) maximiert.

$$\max_{a \in [\mathbb{R}/A]^n} \quad \min_{1 \leq k \leq m} u_k$$

   Dieses Problem wurde von Vince, Cerny und Guldan untersucht.

Für zwei Polygone und für beliebiges p gibt Burkard ([Bur86]) eine optimale Lösung an.

Das Polygon-Problem wurde auch für irreguläre Polygone betrachtet. In einer Arbeit von Brucker, Burkard und Hurink [BBH90] wurde gezeigt, dass das Polygon-Problem bei n irregulären Polygonen für ein festes n polynomiell lösbar ist. Aber im Allgemeinen ist das Polygon-Problem für irreguläre Polygone NP-hart, die Laufzeit wächst exponentiell mit n. Der Beweis, dass das Polygon-Problem für irreguläre Polygone NP-hart ist, erfolgt durch die Reduktion des NP-vollständigen 3-Partition-Problems auf das irreguläre Polygon-Problem.

Einige Ergebnisse aus diesen Arbeiten werden beim Branch-and-Bound-Algorithmus in Kapitel 4 für die Berechnung von oberen Schranken verwendet, näheres wird in den dazugehörigen Kapiteln erläutert.

1.2.4 Motivation

Dass Verspätungen entstehen, lässt sich kaum verhindern, diese entstehen entweder wegen technischer Störungen oder wegen erhöhten Passagieraufkommens.

Bei erhöhtem Passagieraufkommen bleiben Fahrzeuge länger an den Haltestellen stehen und können nicht abfahren, da die Einsteigezeit der Passagiere sich verlängert. Es gibt viele technische Störungen: Gleisabschnitte sind unpassierbar wegen eines Unfalls oder wegen eines Falschparkers, Stromausfall in einem Streckenabschnitt oder in einem Zug, Defekte an den Türen usw.

Verspätungen passieren und können nicht ausgeschlossen werden, aber kleine Verspätungen (1-5 Minuten) sollten keine oder nur geringe Auswirkungen auf andere Linien haben.

Wie lassen sich ohne Änderung der Takte kleine Verspätungen ohne nennenswerte Folgen auffangen? Eine Möglichkeit besteht darin, die Ankunftszeiten der Linien an den einzelnen Stationen gleichmäßig zu verteilen, d.h. der Abstand zwischen den Ankunftszeiten einzelner Linien an den Stationen sollte so groß wie möglich sein. Dies hätte zu Folge, dass kleine Störungen sich nur auf eine begrenzte Zahl von Linien auswirken. Im Rahmen dieser Arbeit wollen wir möglichst gut die Ankunftszeiten der einzelnen Linien verteilen, damit kleine Verspätungen, die sich nicht vermeiden lassen, nur geringe Auswirkungen haben.

Im Nahverkehr spielt die Anschlussoptimierung eine untergeordnete Rolle, da die Takte der Linien klein sind.

Kapitel 2

Modellierung

In diesem Kapitel wird das Fahrplanoptimierungsproblem, das in dieser Arbeit betrachtet wird, modelliert. Dazu werden die Begriffe Streckennetzwerk und Sicherheitabstand definiert. Im Weiteren werden die Bewertung von Fahrplänen sowie die Modellierung eines realen Liniennetzwerks detailliert beschrieben.

2.1 Grundlegende Definitionen

Gegeben sei ein Streckennetzwerk $N(S, C, t, L, T)$ mit Stationen $S = \{s_1, \ldots, s_n\}$, Linien $L = \{l_1, \ldots, l_m\}$ und Verbindungen (Connections) $C = \{c_1, \ldots, c_v\}$ zwischen diesen Stationen. Jeder Verbindung wird durch $t : C \to \mathbb{N}_0$ eine Fahrzeit zugeordnet. Eine Linie $l_j = (s_{j_1} s_{j_2} \ldots s_{j_{k(j)}})$ ist ein einfacher gerichteter Weg im gerichteten Graphen $G = (S, C)$. Jede Linie l_j besitzt einen bestimmten Takt T_j. Die Menge der Takte bezeichnen wir mit $T = (T_1, T_2, \ldots, T_m)$. Die Abfahrtszeit der Linie l_j an der Anfangsstation s_{j_1} bezeichnen wir mit λ_j, den Vektor der m Abfahrtszeiten aller Linien, den Fahrplan, mit λ und die Menge aller Fahrpläne mit $\Lambda = \mathbb{N}_{T_1} \times \mathbb{N}_{T_2} \times \cdots \times \mathbb{N}_{T_m}$, wobei $\mathbb{N}_k = \{0, \ldots, k-1\}$ sei. Man kann sich bei den Abfahrtzeiten einer Linie l auf die Zeiten $\mathbb{N}_{T_l}$ beschränken, da die Linie den Takt T_l besitzt. Im Weiteren enthalte ein Streckennetzwerk keine Station, die nicht mindestens von einer Linie befahren wird.

Da die Fahrzeiten zwischen den Stationen fest gegeben sind, sind die Ankunftszeiten der Linie l_j an den einzelnen Stationen durch die Abfahrtzeit λ_j an der Startstation vollständig bestimmt. Die Ankunftszeit einer Linie an einer Station lässt sich dadurch berechnen, dass man die Abfahrtszeit der Linie an der Anfangsstation zu der Summe aller Zeiten der Verbindungen von der Anfangsstation bis zur jeweiligen Station addiert. Wenn also die Linie l_j an der Anfangsstation s_{j_1} zum Zeitpunkt λ_j abfährt, so ergibt sich die Abfahrtszeit

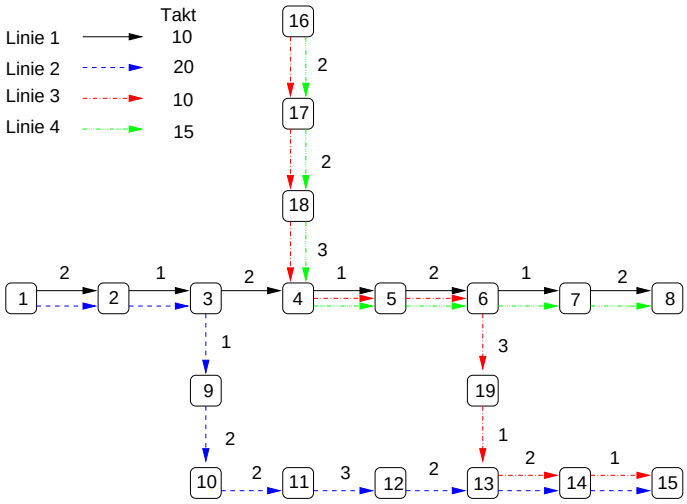


Abbildung 2.1: Beispiel Streckennetzwerk mit 4 Linien und 19 Stationen

(=Ankunftszeit) $a(s_{j_p}, l_j, \lambda)$ der Linie l_j an der Station s_{j_p} durch:

$$a(s_{j_p}, l_j, \lambda) := \lambda_j + \sum_{k=1}^{p-1} t(s_{j_k} s_{j_{k+1}}) \quad , 1 \leq p \leq s_{j_{k(j)}} \tag{2.1.1}$$

Nach der Festlegung der Abfahrtszeiten der einzelnen Linien lassen sich mit Hilfe von (2.1.1) und den jeweiligen Takten die Fahrpläne für die Linien leicht erstellen.

Bemerkung 2.1.1 In unserem Modell wird zwischen Abfahrtszeit und Ankunftszeit nicht unterschieden. Im Nahverkehr wird nicht wie im Fernverkehr zwischen Abfahrtszeit und Ankunftszeit unterschieden, da die Haltezeiten in der Regel weniger als eine Minute betragen und es keine feste Haltezeit gibt.

Definition 2.1.2 Sei $s \in S$ und $l \in L$.
Die Menge aller Linien, die durch die Station s fahren, bezeichnen wir mit $L(s)$. Die Menge aller Stationen, die von der Linie l befahren werden, bezeichnen wir mit $S(l)$. Für $\tilde{S} \subseteq S$ und $\tilde{L} \subseteq L$ definieren wir noch folgende Mengen:

- Linien, die durch mindestens eine der Stationen aus $\tilde{S}$ fahren:
  $L(\tilde{S}) := \bigcup_{s \in \tilde{S}} L(s)$
- Stationen, die von mindestens einer der Linien in $\tilde{L}$ befahren werden:
  $S(\tilde{L}) := \bigcup_{l \in \tilde{L}} S(l)$
- Takte der Linien in $\tilde{L}$:
  $T(\tilde{L}) := \{T_l | l \in \tilde{L}\}$

Definition 2.1.3 (Fahrzeit)
Seien s_i, $s_j \in S(l)$.

$$F(s_i, s_j, l) := a(s_j, l, \vec{0}) - a(s_i, l, \vec{0})$$

ist die Fahrzeit der Linie l von der Station s_i zu s_j. Die Fahrzeit kann auch negativ sein, wenn die Linie l die Station s_j vor s_i anfährt. Man beachte, dass die Fahrzeit zwischen zwei Stationen nicht vom Fahrplan abhängt, d.h. für jeden Fahrplan λ gilt:

$$F(s_i, s_j, l) = a(s_j, l, \lambda) - a(s_i, l, \lambda).$$

Definition 2.1.4 (Sicherheitsabstand zwischen zwei Linien)
Der Sicherheitsabstand $\delta(s, l_j, l_k, \lambda)$ an der Station $s \in S$ von Linie l_k zu Linie l_j ist definiert als minimale Differenz aller Ankunftszeiten der Linien l_j und l_k an der Station s:

$$\delta(s, l_j, l_k, \lambda) := \min_{r,t \in \mathbb{N}} |a(s, l_j, \lambda) + r \cdot T_j - a(s, l_k, \lambda) - t \cdot T_k| \tag{2.1.2}$$

Es gilt dann:

$$\delta(s, l_j, l_k, \lambda) = \delta(s, l_k, l_j, \lambda) \tag{2.1.3}$$

Definition 2.1.5 (Sicherheitsabstand an einer Station)
Der Sicherheitsabstand $\delta(s, \lambda)$ an der Station s ist definiert als das Minimum der paarweisen Sicherheitsabstände der Linien aus $L(s)$:

$$\delta(s, \lambda) := \begin{cases} \min_{l_j, l_k \in L(s), l_j \neq l_k} \delta(s, l_j, l_k, \lambda), & \text{falls } |L(s)| > 1 \\ T_l, & \text{falls } |L(s)| = 1, \text{ d.h. } L(s) = \{l\} \end{cases} \tag{2.1.4}$$

Der Sicherheitsabstand an einer Station für einen festen Fahrplan ist der kleinste Abstand der Ankunftszeiten aller Linien an dieser Station.

Definition 2.1.6 (Sicherheitsabstand in einem Streckennetzwerk)
Sei

$$\delta(\lambda) := \min_{s \in S} \delta(s, \lambda) \tag{2.1.5}$$

der Sicherheitsabstand an allen Stationen von Fahrplan λ.

$$\delta_\Sigma(\lambda) := \sum_{s \in S} \delta(s, \lambda) \tag{2.1.6}$$

$\delta_\Sigma(\lambda)$ ist die Summe der Sicherheitsabstände über alle Stationen von Fahrplan λ.

Der Sicherheitsabstand $\delta(\lambda)$ bezeichnet also für einen festen Fahrplan λ den kleinsten zeitlichen Abstand zweier aufeinander folgenden Linien an einer Station im ganzen Netz. Darum lässt er sich auch wie folgt definieren, falls jede Linie mit mindestens einer anderen Linie eine gemeinsame Station befährt.

Definition 2.1.7

$$\delta(\lambda) := \min_{s \in S, l_j, l_k \in L(s), l_j \neq l_k} \delta(s, l_j, l_k, \lambda)$$

2.2 Zielfunktionen

Wegen technische Störungen und erhöhtem Passagieraufkommens entstehen Verspätungen. die nicht ausgeschlossen werden können. Kleine Verspätungen sollten aber nur geringe Auswirkungen auf andere Linien haben.

In Rahmen meiner Diplomarbeit [Gen99] wurde deshalb das Sicherheitsabstand-Problem untersucht. Mit dem Sicherheitsabstand-Problem wollten wir den minimalen Sicherheitsabstand maximieren, damit an zentralen Punkten des Liniensystems kleine Störungen nur geringe Verzögerungen bei anderen Linien bewirken, so dass nach Behebung der Störung der Normalbetrieb sich binnen kurzer Zeit wieder einstellt.

Definition 2.2.1 (Sicherheitsabstand-Problem)
Gegeben sei ein Streckennetzwerk $N(S, C, t, L, T)$. Bestimme ein $\lambda_0 \in \Lambda$ mit $\delta(\lambda_0) = \delta^*$, wobei δ^* den optimalen Sicherheitsabstand bezeichnet und wie folgt definiert ist:

$$\delta^* := \max_{\lambda \in \Lambda} \delta(\lambda) \tag{2.2.7}$$

Das Sicherheitsabstand-Problem konnte mit einem Branch-and-Bound-Ansatz für die untersuchten Instanzen der Kölner Verkehrs-Betriebe in akzeptabler Zeit gelöst werden.

Wir beobachteten aber, dass das Sicherheitsabstand-Problem, dessen Optimierungsziel die Maximierung des minimalen Sicherheitsabstandes ist, im wesentlichen auf die optimale Benutzung stark befahrener Stationen abzielt. Allerdings werden dabei oft wenig befahrene Stationen in Randbezirken vernachlässigt.

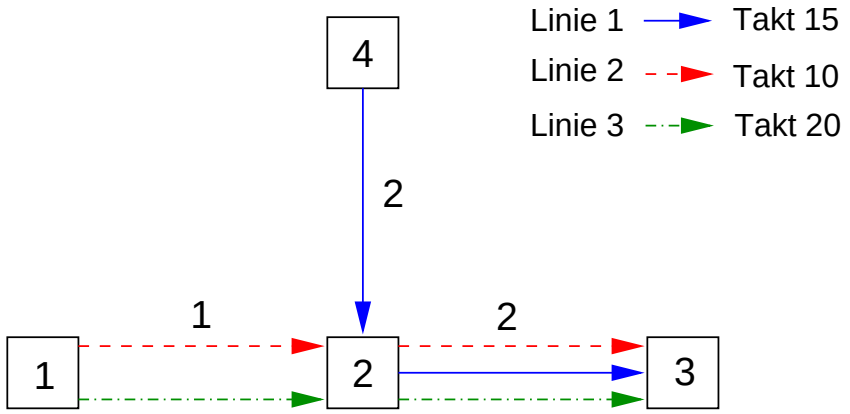
Es ist deshalb sinnvoll, unter den Lösungen, die den geforderten Sicherheitsabstand einhalten, ein weiteres Kriterium anzusetzen. Durch dieses sollte eine möglichst gleichmäßige Verteilung der Ankunftszeiten auf das Taktintervall erreicht werden, d.h. der Sicherheitsabstand sollte möglichst dem optimalen Einzelabstand entsprechen. Dies kann man durch das folgende Optimierungsproblem erreichen:

Definition 2.2.2 (Fahrplan-Problem)
Gegeben sei ein Streckennetzwerk $N(S, C, t, L, T)$. Bestimme ein $\lambda_0 \in \Lambda$ mit $\delta_\Sigma(\lambda_0) = \delta^*_\Sigma$, wobei δ^*_Σ den optimalen Sicherheitsabstand bezeichnet und wie folgt definiert ist:

$$\delta^*_\Sigma := \max_{\lambda \in \Lambda : \delta(\lambda) = \delta^*} \delta_\Sigma(\lambda) \tag{2.2.8}$$

2.3 Beispiel

Betrachte ein Streckennetzwerk mit 3 Linien und 4 Stationen.



An diesem Beispiel werden wir noch einmal einige Begriffe und unsere Zielfunktion erläutern.

Wenn als Fahrplan $\lambda = (2, 5, 1)$ gewählt wird, so ergeben sich für die Linien an den jeweiligen Stationen die Ankunftszeiten wie in Tabelle 2.1.

	Linie 1	Linie 2	Linie 3
Station 1		5 15 25 35 45 55	1 21 41
Station 2	4 19 34 49	6 16 26 36 46 56	2 22 42
Station 3	6 21 36 51	8 18 28 38 48 58	4 24 44
Station 4	2 17 32 47		

Tabelle 2.1: Ankunftszeiten der Linien an den Stationen für den Fahrplan $\lambda = (2, 5, 1)$.

Wenn wir Ankunftszeiten der Linien an den einzelnen Stationen in einen Kreis einzeichnen und diese miteinander verbinden, so erhalten wir je ein reguläres Polygon für eine Linie, siehe Abbildung 2.2. Der Sicherheitsabstand an einer Station ist genau der minimale Abstand zweier benachbarter Ecken der Polygone.

Wie wir gesehen haben, wurden für die Linien verschiedene Anzahlen von Ankunftszeiten (entspricht der Anzahl der Ecken des Polygons) an den Stationen berechnet, für Linie 1 wurden vier Ankunftszeiten berechnet und für Linie 2 sechs. Diese Anzahl hängt von dem kgV der Takte der Linien ab. Allgemein braucht man für die Linie i genau $\frac{kgV(T_1,..,T_n)}{T_i}$ aufeinander folgende Ankunftszeiten. Wenn weniger als $\frac{kgV(T_1,..,T_n)}{T_i}$ aufeinander folgende Ankunftszeiten für die Linie i betrachtet werden, so gilt im Allgemeinen nicht, dass der Sicherheitsabstand der kleinste Abstand zweier benachbarter Ecken der Polygone für einen festen Fahrplan ist. Die Fahrzeit von einer Station zur nächsten für eine Linie im Fahr-

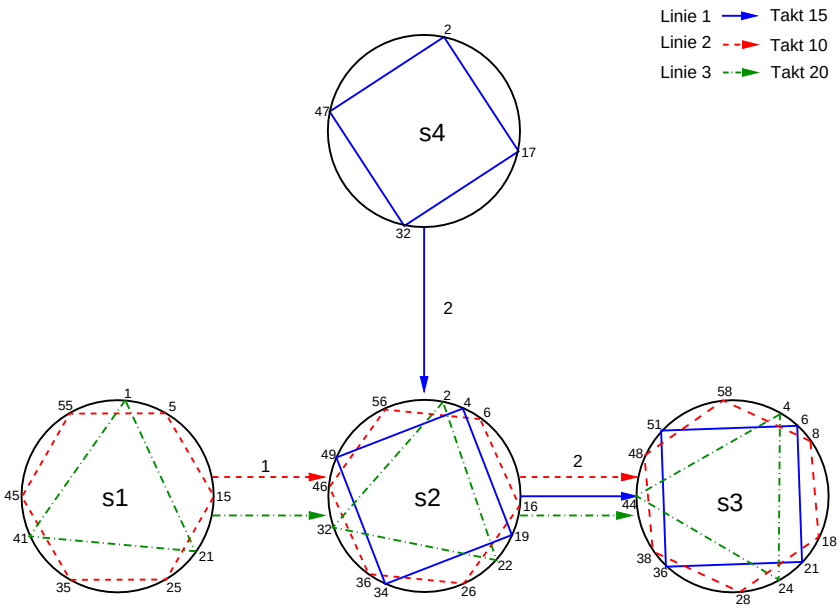


Abbildung 2.2: Fahrplangraph für Fahrplan $\lambda = (2, 5, 1)$

plangraphen entspricht einer Drehung des zugehörigen Polygons um die Fahrzeit dieser Linie.

Bildlich kann man sich das Fahrplan-Problem wie folgt vorstellen. Jeder Station entspricht ein Kreis der Länge $kgV(T_1, .., T_n)$ auf dem die Fahrzeiten $0, .., kgV(T_1, .., T_n) - 1$ eingetragen sind. Jede Linie i wird durch ein reguläres Polygon mit genau $\frac{kgV(T_1,..,T_n)}{T_i}$ Ecken repräsentiert. Die Fahrzeit einer Linie auf λ_i setzen bedeutet, dass sich eine der Ecken des Polygons auf dem Kreis an der Position λ_i befindet. Die Zielfunktion besagt, dass die Polygone an die Anfangsstationen so gesetzt werden, dass die Polygone an den Stationen gleichmäßig verteilt sind. Dabei werden durch die Festlegung des Polygons der Linie an der Anfangsstation, alle Polygone an den einzelnen Stationen, die zu dieser Linie gehören, durch Drehung um die entsprechende Fahrzeit festgelegt.

2.4 Zulässige Lösungen des Fahrplan-Problems

Beim Fahrplan-Problem sucht man unter den Fahrplänen, die eine optimale Lösung des Sicherheitsabstand-Problems sind, denjenigen Fahrplan, dessen Summe der Sicherheitsabstände maximal ist. Damit sieht die Menge der zulässigen Fahrpläne für das Fahrplan-Problem wie folgt aus:

$$\Lambda_0 = \{\lambda \in \Lambda : \delta(\lambda) = \delta^*\}$$

Eine zulässige Lösung zu bestimmen ist bereits NP-schwierig, da die Bestimmung von δ^* NP-schwer ist (siehe 3.3.2). Darum ist es nicht sinnvoll, zuerst das Sicherheitsabstand-Problem optimal zu lösen um δ^* zu erhalten, da unter Umständen bei bestimmten Streckennetzwerken die Berechnung von δ^* sehr lange dauern könnte.

Um die Berechnung von δ^* zu vermeiden, kann man eine Ordnung auf der Menge aller Fahrpläne Λ für ein gegebenes Streckennetzwerk definieren. Mit dieser Ordnung lassen sich Fahrpläne vergleichen.

Definition 2.4.1 Sei ein Streckennetzwerk $N(S, C, t, L, T)$ gegeben und seien $\lambda_1, \lambda_2 \in \Lambda$ zwei beliebige Fahrpläne.

Fahrplan λ_2 ist mindestens so gut wie der Fahrplan λ_1 ($\lambda_1 \leq_\Lambda \lambda_2$)

$$\Leftrightarrow (\delta(\lambda_1) < \delta(\lambda_2)) \quad \vee \quad (\delta(\lambda_1) = \delta(\lambda_2) \wedge \delta_\Sigma(\lambda_1) \leq \delta_\Sigma(\lambda_2))$$

Offensichtlich ist $\leq_\Lambda$ in Λ reflexiv und transitiv, also eine reflexive Quasiordnung. Die reflexive Quasiordnung $\leq_\Lambda$ in Λ induziert in Λ eine Äquivalenzrelation $=_\Lambda$, die für λ_1, $\lambda_2 \in \Lambda$ durch

$$\begin{aligned} \lambda_1 =_\Lambda \lambda_2 &\Leftrightarrow (\lambda_1 \leq_\Lambda \lambda_2) \wedge (\lambda_2 \leq_\Lambda \lambda_1) \\ &\Leftrightarrow \delta(\lambda_1) = \delta(\lambda_2) \wedge \delta_\Sigma(\lambda_1) = \delta_\Sigma(\lambda_2) \end{aligned}$$

definiert wird.

In der Faktormenge $\Lambda/=_\Lambda$ bzgl. der Äquivalenzrelation $=_\Lambda$ definieren wir eine Relation $\leq^*_\Lambda$ wie folgt: $[\lambda_1]_{=_\Lambda} \leq^*_\Lambda [\lambda_2]_{=_\Lambda} \Leftrightarrow \lambda_1 \leq_\Lambda \lambda_2$ wobei λ_1, $\lambda_2 \in \Lambda$ beliebig. Die Relation $\leq^*_\Lambda$ ist eine reflexive Ordnung in $\Lambda/=_\Lambda$.

Satz 2.4.2 *Sei ein Streckennetzwerk $N(S, C, t, L, T)$ gegeben.*
Sei $\lambda_0 \in \Lambda$ und sei $[\lambda_0]_{=_\Lambda}$ das maximale Element in $\Lambda/=_\Lambda$ bzgl. der reflexive Ordnung $\leq^*_\Lambda$.
Dann gilt: Jeder Fahrplan $\lambda \in [\lambda_0]_{=_\Lambda}$ ist eine optimale Lösung des Fahrplan-Problems und umgekehrt.

Beim Lösen des Fahrplan-Problems mit Hilfe des Branch-and-Bound-Verfahrens werden wir die reflexive Quasiordnung $\leq_\Lambda$ benutzen, um zwei Fahrpläne miteinander zu vergleichen.

2.5 Linienkonflikt- und Multi-Linienkonflikt-Graph

Im diesem Abschnitt wollen wir zwei Graphen definieren, die wir aus einem Streckennetzwerk bilden können. Mit Hilfe des einen Graphen werden wir in 3.2.3 das Streckennetzwerk geeignet reduzieren, um die Optimierungsschritte beim Branch-and-Bound-Verfahren

deutlich zu beschleunigen. Mit dem zweiten Graphen lässt sich das Streckennetzwerk in Teilnetze aufteilen. Für diese Teilnetze lassen sich Teilfahrpläne erzeugen, aus denen ein Fahrplan für das gesamte Streckennetzwerk gebildet werden kann, der die Eigenschaften der Teilfahrpläne besitzt (siehe 4.5).

Definition 2.5.1 Der *Linienkonflikt-Graph* (LK-Graph) zum Streckennetzwerk $N(S, C, t, L, T)$ ist definiert als LK-$G(N) = (L, E)$. Eine Kante existiert zwischen zwei Linien (=Knoten), genau dann, wenn die Linien mindestens eine gleiche Station befahren, d.h. es gilt:

$$e = (l_i, l_j) \in E \Leftrightarrow S(l_i) \cap S(l_j) \neq \emptyset$$

Definition 2.5.2 Der *Multi-Linienkonflikt-Graph* (MLK-Graph) zum Strekkennetzwerk $N(S, C, t, L, T)$ ist definiert als MLK-$G(N) = (L, M)$ und zwischen je zwei Linien l_i, l_j existiert eine Kante $(s, l_i, l_j) \in M$, genau dann wenn die Linien die Station s befahren, d.h. es gilt:

$$(s, l_i, l_j) \in M \Leftrightarrow s \in S(l_i) \cap S(l_j)$$

Der Linienkonflikt-Graph entsteht folglich aus dem Multi-Linienkonflikt-Graphen durch Eliminierung von parallelen Kanten.

Beispiel 2.5.3 In Abbildung 2.3 ist ein Streckennetzwerk mit 4 Linien und 19 Stationen dargestellt. Den zugehörigen LK-Graphen und MLK-Graphen sieht man in den Abbildungen 2.4 und 2.5.

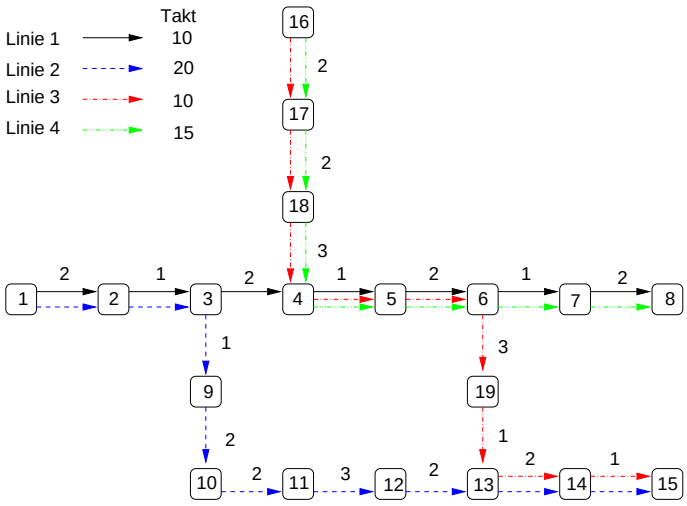


Abbildung 2.3: Ein Streckennetzwerk mit 4 Linien und 19 Stationen

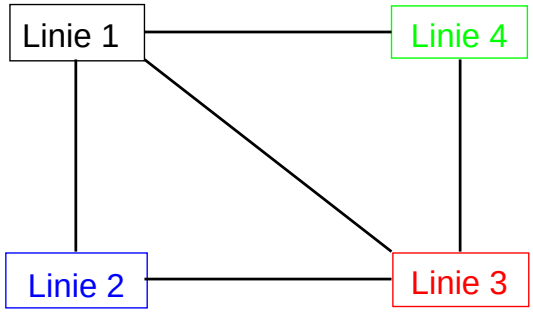


Abbildung 2.4: Der LK-Graph zum Streckennetzwerk aus Abbildung 2.3.

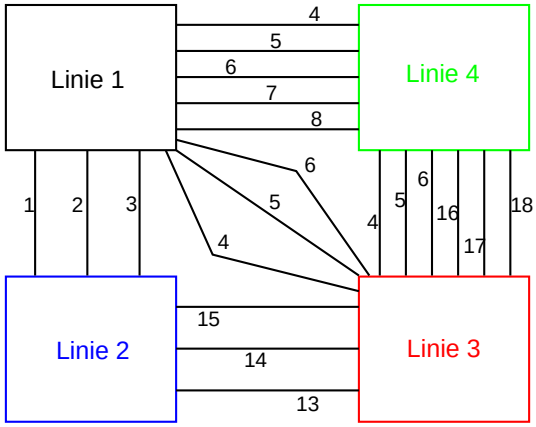


Abbildung 2.5: Der MLK-Graph zum Streckennetzwerk aus Abbildung 2.3.

2.6 Umwandlung des realen Netzwerks in unser Modell

Bei der Umwandlung eines realen Streckennetzes (z.B. eines Straßenbahnnetzes) in unser Modell muss man einige Besonderheiten beachten. In unserem Modell ist eine Linie ein einfacher gerichteter Weg. Darum muss eine Linie im realen Netzwerk durch zwei Linien in unserem Modell modelliert werden. Dabei wird eine Linie in Hin- und Rückrichtung aufgeteilt. Wenn die Aufspaltung der Linien durchgeführt wird, müssen dabei auch die Stationen aufgespalten werden. Weiterhin muss bei der Aufspaltung der Stationen darauf geachtet werden, zwischen Stationen zu unterscheiden, bei denen sowohl oberirdisch als auch unterirdisch Linien fahren. Stationen, die nur oberirdisch oder nur unterirdisch verlaufen, werden in unserem Modell in genau zwei Stationen aufgeteilt. Stationen, bei denen sowohl oberirdisch als auch unterirdisch Linien fahren, werden dann in vier Stationen aufgeteilt. Durch diese Konstruktion werden die Anzahl der Linien im realen Netz und die Anzahl der Stationen verdoppelt.

Im Allgemeinen lässt sich das erzeugte Streckennetz nicht in zwei getrennte Netze bzgl. Hin- und Rückrichtung unterteilen, siehe dazu Abbildung 2.6 und 2.7. Um festzustellen, ob man das Streckennetzwerk in getrennte Teilnetze unterteilen kann, reicht es aus, im zugehörigen LK-Graphen die Zusammenhangskomponenten zu berechnen. Jede Zusammenhangskomponente bildet ein Teilnetz, das von den anderen Teilnetzen unabhängig ist, weil sie keine gemeinsamen Linien besitzen. Daher kann man diese Teilnetze auch getrennt optimieren.

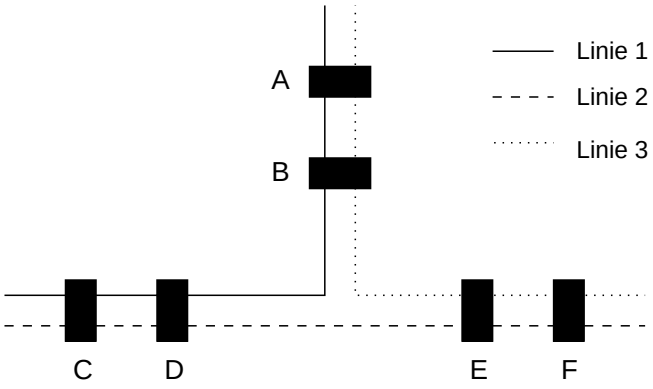


Abbildung 2.6: Ein Netz, bei dem man Hin- und Rückrichtung nicht entkoppeln kann.

2.7 Andere Zielfunktionen

In diesem Abschnitt wollen wir andere Zielfunktionen vorstellen und begründen, warum diese nicht sinnvoll sind.

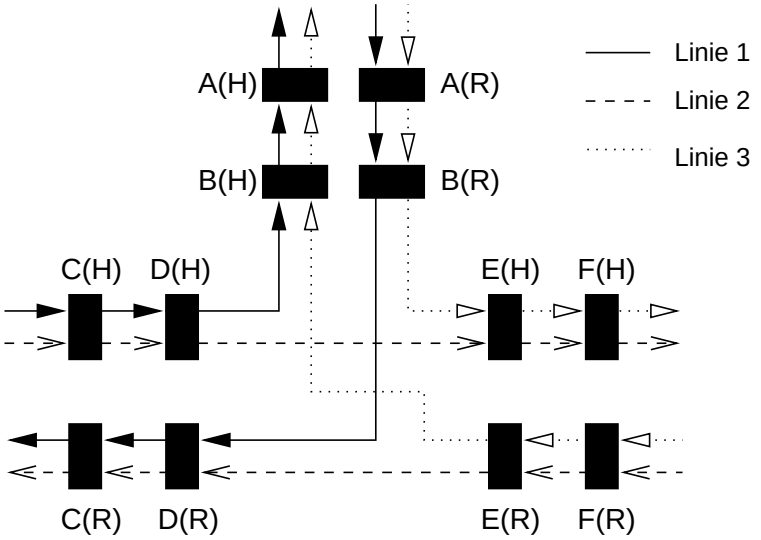


Abbildung 2.7: Unser Modell zum Streckennetzwerk in Abbildung 2.6.

2.7.1 Maximierung der Summe aller Sicherheitsabstände ohne Berücksichtigung von δ^*

Wenn wir die Summe aller Sicherheitsabstände maximieren ohne den minimalen Sicherheitsabstand δ^* zu berücksichtigen, so sieht das Problem wie folgt aus:

$$\delta_\Sigma := \max_{\lambda \in \Lambda} \delta_\Sigma(\lambda) \tag{2.7.9}$$

Da $\{\lambda \in \Lambda : \delta(\lambda) = \delta^*\} \subset \Lambda$ ist, gilt offensichtlich:

$$\max_{\lambda \in \Lambda : \delta(\lambda) = \delta^*} \delta_\Sigma(\lambda) \leq \max_{\lambda \in \Lambda} \delta_\Sigma(\lambda)$$

Da der Sicherheitsabstand nicht betrachtet wird, kann es bei der Optimierung der Summe der Sicherheitsabstände gemäß 2.7.9 vorkommen, dass der optimale Fahrplan den Sicherheitsabstand 0 hat.

Beispiel 2.7.1 Ein optimaler Fahrplan für das Streckennetzwerk in Abbildung 2.8 bzgl. des Fahrplan-Problems und damit auch des Sicherheitsabstand-Problems ist z.B. $\lambda_1 = (0, 2, 4, 1, 3, 5)$. Für die Sicherheitsabstände des Fahrplans λ_1 gilt dann: $\delta(\lambda_1) = 1 = \delta^*$ und $\delta_\Sigma(\lambda_1) = 9 = \delta^*_\Sigma$.
Ein optimaler Fahrplan für Zielfunktion (2.7.9) ist z.B. $\lambda_2(0, 2, 4, 0, 3, 6)$ mit $\delta_\Sigma(\lambda_2) = 10$ und $\delta(\lambda_2) = 0$.

Das folgende einfache Beispiel zeigt, dass die Differenz zwischen den optimalen Lösungen gemäß den Zielfunktionen (2.7.9) und (2.2.8) beliebig groß sein kann.

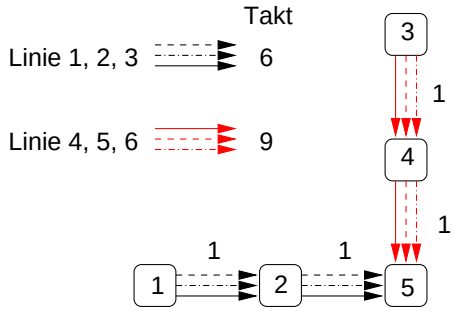


Abbildung 2.8: Beispiel eines Streckennetzwerks, für das bezüglich der Zielfunktion (2.7.9) der optimale Fahrplan den Sicherheitsabstand 0 hat.

Beispiel 2.7.2 Die Abbildung 2.9 stellt ein Streckennetzwerk mit 3 Linien und k+3 Stationen dar.

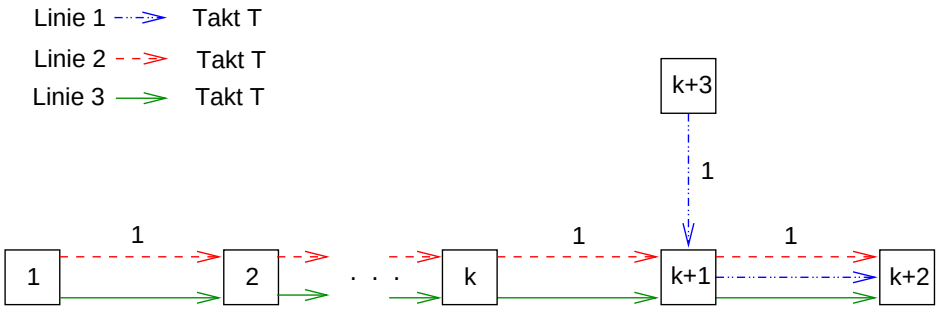


Abbildung 2.9: Beispiel, für das δ_Σ und δ^*_Σ sehr weit auseinander liegen.

Sei $T \geq 3$.

Der Fahrplan $\lambda_1 = (0, \lfloor \frac{T}{3} \rfloor, T - \lfloor \frac{T}{3} \rfloor)$ ist ein optimaler Fahrplan für das Fahrplan-Problem. An den Stationen $i \in \{1, \ldots, k\}$ gilt für den Sicherheitsabstand beim Fahrplan λ_1:

$$\delta(\lambda_1, i) = T - 2 \cdot \left\lfloor \frac{T}{3} \right\rfloor$$

und an den anderen Stationen gilt:
$\delta(\lambda_1, k+1) = \delta(\lambda_1, k+2) = \lfloor \frac{T}{3} \rfloor$ und $\delta(\lambda_1, k+3) = T$.

Daraus folgt: $\delta^* = \lfloor \frac{T}{3} \rfloor$ und $\delta^*_\Sigma = k \cdot (T - 2 \cdot \lfloor \frac{T}{3} \rfloor) + 2 \cdot \lfloor \frac{T}{3} \rfloor + T$.

Ein optimaler Fahrplan für das Problem 2.7.9 ist $\lambda_2 = (0, \lfloor \frac{T}{2} \rfloor, \lfloor \frac{T}{4} \rfloor)$. Für den Fahrplan λ_2 betragen die Sicherheitsabstände an den Stationen: $\delta(\lambda_2, i) = \lfloor \frac{T}{2} \rfloor$, für $i \in \{1, \ldots, k\}$ und $\delta(\lambda_2, k+1) = \delta(\lambda_2, k+2) = \lfloor \frac{T}{4} \rfloor$ und $\delta(\lambda_2, k+3) = T$.

Also gilt: $\delta(\lambda_2) = \lfloor \frac{T}{4} \rfloor$ und $\delta_\Sigma = k \cdot \lfloor \frac{T}{2} \rfloor + 2 \cdot \lfloor \frac{T}{4} \rfloor + T$

$$\delta_\Sigma - \delta_\Sigma^* = k \cdot \left[\left\lfloor \frac{T}{2} \right\rfloor - T + 2 \cdot \left\lfloor \frac{T}{3} \right\rfloor \right] + 2 \cdot \left[\left\lfloor \frac{T}{4} \right\rfloor - \left\lfloor \frac{T}{3} \right\rfloor \right] \approx (k-1) \cdot \frac{T}{6},$$

für genügend große T.

Im Folgenden wollen wir noch kurz erläutern, warum es sinnvoll ist, das Fahrplan-Problem zu betrachten.

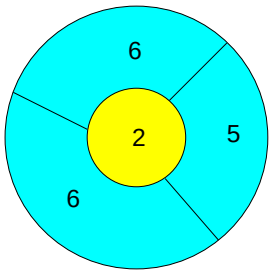


Abbildung 2.10: Obere Schranken für den Sicherheitsabstand in einem Streckennetzwerk für verschiedene Bezirke.

Abbildung 2.10 stellt die Sicherheitsabstände, die in einem Streckennetzwerk erreichbar wären, dar. In der Innenstadt ist ein Sicherheitsabstand von 2, in den drei Randbereichen ist ein Sicherheitsabstand von 6, 6 und 5 erreichbar. Ein optimaler Fahrplan für das Sicherheitsabstand-Problem wäre gefunden, wenn wir einen Fahrplan finden würden, der in allen Bezirken den Sicherheitsabstand 2 hat. Ein optimaler Fahrplan für das Fahrplan-Problem müsste in der Innenstadt einen Sicherheitsabstand von 2 erreichen und einen möglichst großen Sicherheitsabstand auch in den Randbezirken erreichen.

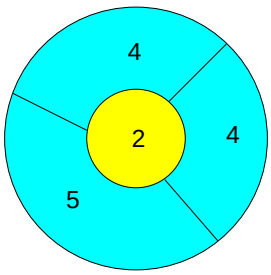


Abbildung 2.11: Beispiel für eine optimale Lösung des Fahrplan-Problems.

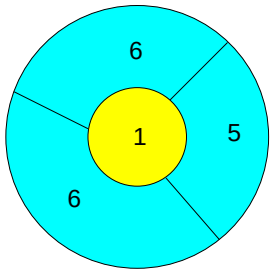


Abbildung 2.12: Beispiel für eine optimale Lösung des Problems (2.7.9).

Wenn man die Summe aller Sicherheitsabstände maximiert ohne den minimalen Sicherheitsabstand δ^* zu berücksichtigen, so kann man nicht garantieren, dass im Zentrum,

wo die Stationen stark befahren werden, der optimale Sicherheitsabstand erreicht wird. Es kann sogar passieren, dass der Sicherheitsabstand im Zentrum Null wird, d.h. zwei Linien haben dieselbe Ankunftszeit.

2.7.2 Erweiterung der Zielfunktion des Fahrplan-Problems

Ein optimaler Fahrplan für das Fahrplan-Problem erreicht den optimalen Sicherheitsabstand an allen Stationen unter Berücksichtigung des Sicherheitsabstandes im Zentrum, aber verteilt nicht immer die Ankunftszeiten an den Stationen gleichmäßig, wie man es am folgenden Beispiel sehen kann.

Beispiel 2.7.3 In Abbildung 2.7.3 sehen wir m Linien, die durch dieselben Stationen fahren und den gleichen Takt T haben.

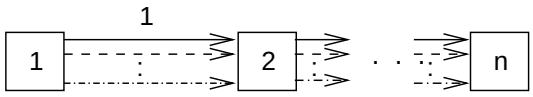


Abbildung 2.13: Ein Streckennetzwerk mit m Linien und n Stationen.

Offensichtlich gilt:

$$\delta^* = \left\lfloor \frac{T}{m} \right\rfloor$$

$$\delta_\Sigma = \delta^*_\Sigma = n \cdot \left\lfloor \frac{T}{m} \right\rfloor$$

Die folgenden zwei Lösungen für das Fahrplan-Problem sind beide optimal:

1. $\lambda_1 = (0, \delta^*, 2 \cdot \delta^*, 3 \cdot \delta^*, \ldots, (m-1) \cdot \delta^*)$
2. $\lambda_2 = \left(0, \left\lfloor \frac{T}{m} \right\rfloor, \left\lfloor \frac{2 \cdot T}{m} \right\rfloor, \ldots, \left\lfloor \frac{(m-1) \cdot T}{m} \right\rfloor \right)$

Was ist der Unterschied zwischen den beiden Lösungen?

Die erste Lösung λ_1 verteilt die Ankunftszeiten auf den ersten Teil des durch den Takt T gegebenen Zeitintervalls, dabei kann unter Umständen ein beträchtlicher Rest des Zeitintervalls ungenutzt bleiben. Die zweite Lösung λ_2 verteilt die Ankunftszeiten gleichmäßig auf das Zeitintervall.

Hier ein konkretes Beispiel: $T = 10; m = 6; n = 2$

1. $\lambda_1 = (0, 1, 2, 3, 4, 5)$
2. $\lambda_2 = (0, 1, 3, 5, 6, 8)$

Wie lassen sich solche ungleichmäßigen Verteilungen der Ankunftszeiten vermeiden? Wenn an den Stationen anstelle der Maximierung des minimalen Sicherheitsabstandes die Minimierung der Differenz zwischen maximaler und minimaler Wartezeit aufeinanderfolgender Linien an derselben Station betrachtet wird, würden die Ankunftszeiten gleichmäßiger verteilt.

$$\min_{\lambda \in \Lambda : \delta(\lambda) = \delta^*} \sum_{s \in S} (\delta_{max}(s, \lambda) - \delta_{min}(s, \lambda)) =: \delta_{gv} \tag{2.7.10}$$

wobei $\delta_{max}(s, \lambda)$ (bzw. $\delta_{min}(s, \lambda)$) die maximale (minimale) Wartezeit zweier aufeinander folgender Linien an der Station s ist.

Diese Zielfunktion braucht man nicht zu betrachten, da solche Strukturen selten auftauchen und normalerweise die ungleichmäßige Verteilung durch die Randbezirke verhindert wird.

Kapitel 3

Zeit-Komplexität

Dieses Kapitel untersucht die Zeit-Komplexität einiger Probleme bei der Fahrplanoptimierung. Zuerst wird die Zielfunktion betrachtet, die von den Sicherheitsabständen an den einzelnen Stationen abhängt. Dafür wird die Berechnung des Sicherheitsabstandes an einer Station und im gesamten Streckennetzwerk untersucht. Für die Berechnung des Sicherheitsabstandes im gesamten Streckennetzwerk werden Relationen definiert, mit deren Hilfe die Anzahl der Stationen, die betrachtet werden müssen, erheblich verringert werden kann. Danach wird die Komplexität des Sicherheitsabstand- und Fahrplan-Problems betrachtet. Es wird durch Reduktion des k-Färbungsproblems auf das Entscheidungsproblem zum Sicherheitsabstand-Problem gezeigt, dass das Sicherheitsabstand-Problem NP-vollständig ist. Zuletzt wird untersucht, ob die Dimension des Suchraums verringert werden kann.

3.1 Sicherheitsabstand an einer Station

In diesem Abschnitt wollen wir untersuchen, wie wir den Sicherheitsabstand an einer Station am besten berechnen können. Zuerst wird die Berechnung des Sicherheitsabstandes zwischen zwei Linien an einer Station betrachtet. Danach wird dieser Schritt auf die Berechnung des Sicherheitsabstandes an einer Station erweitert.

3.1.1 Sicherheitsabstand zwischen zwei Linien an einer Station

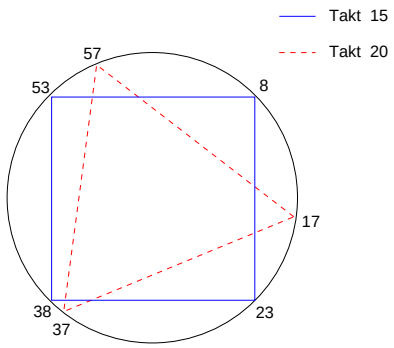
Nach Definition 2.1.4 ist der Sicherheitsabstand $\delta(s, l_j, l_k, \lambda)$ zwischen zwei Linien l_j, l_k an einer Station s für einen vorgegebenen Fahrplan λ definiert als der kleinste zeitliche Abstand der Ankunftszeiten der beiden Linien an der Station s.

$$\delta(s, l_j, l_k, \lambda) := \min_{r,t \in \mathbb{N}} |a(s, l_j, \lambda) + r \cdot T_j - a(s, l_k, \lambda) - t \cdot T_k|$$

Man kann zeigen, dass man zur Berechnung des Sicherheitsabstandes $\delta(s, l_j, l_k, \lambda)$ an

einer Station s für einen festen Fahrplan λ nur ein Zeitintervall der Länge $kgV(T_{l_j}, T_{l_k})$ betrachten muss. Für beide Linien berechnet man die Ankunftszeiten in diesem Zeitintervall und sortiert diese Ankunftszeiten aufsteigend. Der kleinste Abstand zweier aufeinander folgender Ankunftszeiten in der aufsteigend sortierten Ankunftszeitenfolge bildet den Sicherheitsabstand.

Beispiel 3.1.1 Wenn zwei Linien mit Takt 15 und Takt 20 zum Zeitpunkt 8 und 17 an derselben Station ankommen. So sind die Ankunftszeiten der Linien im Zeitintervall $kgV(15, 20) = 60$ die folgenden: 8, 17, 23, 37, 38, 53, 57. D.h. Sicherheitsabstand an der Station beträgt $38 - 37 = 1$



Der Sicherheitsabstand ist der kleinste Abstand zwischen den Ankunftszeiten.

Diese Idee beruht auf Arbeiten über reguläre Polygone von Guldan und Burkard([Bur86], [Gul80]). Sie untersuchten reguläre Polygone in einem Kreis. Dabei versucht man Polygone so in einen Kreis einzuzeichnen, dass die Abstände zwischen den Ecken der Polygone bestimmte Zielfunktionen maximieren oder minimieren.

Ein anderes Verfahren zur Sicherheitsabstandsbestimmung zweier Linien, das in einer Arbeit von Vince [Vin89] genannt wird, besagt:

Satz 3.1.2 *Sei $\lambda \in \Lambda$, $s \in S$ und $l_j, l_k \in L(s)$. Dann gilt:*

$$\delta(s, l_j, l_k, \lambda) = \min\{d, ggT(T_j, T_k) - d\}$$

wobei $d = |a(s, l_j, \lambda) - a(s, l_k, \lambda)| \bmod ggT(T_j, T_k)$ ist.

Um den Sicherheitsabstand zwischen zwei Linien zu berechnen, braucht man nach Satz 3.1.2 also nur je eine Ankunftszeit der beiden Linien und den größten gemeinsamen Teiler der Takte dieser Linien.

Angewandt auf Beispiel 3.1.1 ergibt sich wegen $ggT(15, 20) = 5$ und
$d = |8 - 17| \bmod 5 = 4$ für den Sicherheitsabstand $\min\{4, 5 - 4\} = 1$.

Die Laufzeiten der beiden Verfahren:

- Laufzeit $kgV(T_i, T_j)$-Berechnung + $\frac{kgV(T_i,T_j)}{T_i}$ + $\frac{kgV(T_i,T_j)}{T_j}$
- Laufzeit $ggT(T_i, T_j)$-Berechnung + C

Offensichtlich ist das Verfahren aus Satz 3.1.2 besser beeignet für die Berechnung des Sicherheitsabstandes zwischen zwei Linien als das andere Verfahren.

3.1.2 Berechnung des Sicherheitsabstandes an einer Station

Nach Definition 2.1.5 ist der Sicherheitsabstand $\delta(s, \lambda)$ an einer Station s, der minimale Sicherheitsabstand zweier Linien für einen festen Fahrplan λ:

$$\delta(s,\lambda) := \begin{cases} \min_{l_j, l_k \in L(s), l_j \neq l_k} \delta(s, l_j, l_k, \lambda), & \text{falls } |L(s)| > 1 \\ T_l, & \text{falls } |L(s)| = 1, \text{ d.h. } L(s) = \{l\} \end{cases}$$

Um den Sicherheitabstand an einer Station zu berechnen, gibt es zwei Möglichkeiten. Bei der ersten Möglichkeit werden die Sicherheitsabstände für je zwei Linien berechnet und später wird das Minimum darüber bestimmt, genau wie in der Definition. Man berechnet bei der zweiten Möglichkeit den Sicherheitsabstand direkt, in dem eine bestimmte Anzahl von Ankunftszeiten der Linien berechnet wird.

Berechnung von $\delta(s, \lambda)$ mittels Sicherheitsabständen für alle Linienpaare

Wir benutzen die Idee aus dem vorherigen Abschnitt zur Berechnung des Sicherheitsabstandes zwischen zwei Linien an einer Station und berechnen mit deren Hilfe den Sicherheitsabstand für jedes Linienpaar. Der minimale Sicherheitsabstand zwischen zwei Linien wäre dann der Sicherheitsabstand an der Station.

Direkte Berechnung von $\delta(s, \lambda)$

Die direkte Berechnung des Sicherheitsabstandes $\delta(s, \lambda)$ an einer Station s für einen Fahrplan λ erfolgt, in dem zuerst alle Ankunftszeiten der Linien im Intervall $[0, kgV(T(L(s))) - 1]$ berechnet werden. Für jedes $l \in L(s)$ wären es genau $\frac{kgV(T(L(s)))}{T_l}$ Ankunftszeiten. Danach werden alle Ankunftszeiten geordnet und der kleinste Abstand zur nächsten Ankunftszeit berechnet, dabei ist auch der Abstand von letzter Ankunftszeit zur ersten Ankunftszeit zu beachten, unter Beachtung der Periodizität der Ankunftszeiten. Der Algorithmus zur direkten Berechnung des Sicherheitsabstandes $\delta(s, \lambda)$ an der Station s für einen vorgegebenen Fahrplan λ sieht wie folgt aus:

Algorithmus 1: Algorithmus zur direkten Berechnung des Sicherheitsabstandes $\delta(s, \lambda)$ an einer Station s für einen Fahrplan λ

Gegeben: Fahrplan λ;
Sei $T_s = kgV\{T_l | l \in L(s)\}$ und $m = \sum_{l \in L(s)} \frac{T_s}{T_l}$;
Sei A ein Array mit m Feldern und $0 \leq a_l(s) < T_l$ die Ankunftszeit der Linie l an der Station s ;
$i := 1$;
foreach $l \in L(s)$ **do**
  for $j \leftarrow 0$ ***to*** $\frac{T_s}{T_l} - 1$ **do**
    $A[i] := a_l(s) + j \cdot T_l$;
    $i++$;
Sortiere Array A;
$\delta(s, \lambda) = T_s$;
for $j \leftarrow 1$ ***to*** $m - 1$ **do**
  if $(A[j+1] - A[j]) < \delta(s, \lambda)$ **then**
    $\delta(s, \lambda) := A[j+1] - A[j]$;
if $(A[1] + T_s - A[m]) < \delta(s, \lambda)$ **then**
  $\delta(s, \lambda) := A[1] + T_s - A[m]$;

3.2 Berechnung des Sicherheitsabstand und die Summe der Sicherheitsabstände im gesamten Streckennetzwerk

Nach Definition gilt für den Sicherheitsabstand und die Summe der Sicherheitsabstände:

$$\delta(\lambda) := \min_{s \in S} \delta(s, \lambda) \qquad \delta_\Sigma(\lambda) := \sum_{s \in S} \delta(s, \lambda)$$

Wie man sieht, geht hier die Anzahl der Stationen entscheidend ein. Da viele Stationen gleiche Eigenschaften haben, kann man diese zusammenfassen.

Im Branch-and-Bound-Algorithmus, der im Kapitel 4 vorgestellt wird, werden häufig der Sicherheitsabstand und die Summe der Sicherheitsabstände berechnet. Da diese Berechnungen oft die meiste Rechenleistung benötigen, wird in diesem Abschnitt untersucht, in wieweit eine Reduktion des Streckennetzwerks durchgeführt werden kann, so dass die Branch-and-Bound-Schritte erheblich beschleunigt werden.

Wie man an der Definition des Sicherheitsabstands und der Summe der Sicherheitsabstände sehen kann, hängen diese sehr stark von der Stationsmenge ab. Viele Stationen eines Streckennetzwerks sind aber überflüssig für die Berechnung der Sicherheitsabstände, weil sie nur von einer Linie oder von gleichen Linien befahren werden oder deren Infor-

mationen in anderen Stationen enthalten sind. Ziel der Reduktion wird es sein, die Menge der Stationen so zu reduzieren, dass mit der reduzierten Stationsmenge gearbeitet werden kann, ohne irgendwelche Information zu verlieren.

Es stellt sich die Frage, welche Stationen wirklich nötig sind, um den Sicherheitsabstand (bzw. die Summe der Sicherheitsabstände) für einen gegebenen Fahrplan zu berechnen.

Zuerst wird der Sicherheitsabstand untersucht und mit Hilfe einer Hitting-Set-Betrachtung eine Menge von Stationen gefunden, die das Gewünschte leistet. Da sich leider diese Art der Reduktion nur für das Sicherheitsabstand-Problem eignet, wurde ein anderes Reduktionsverfahren entwickelt, das auf der Konstruktion geeigneter Relationen über der Menge der Stationen beruht. Am Ende dieses Abschnitts wird eine weitere Äquivalenzrelation auf den Kanten des MLK-Graphen vorgestellt, die sich zur Berechnung des Sicherheitsabstandes als geeigneter erweist. Eine Kombination der Ergebnisse aus den Relationen auf der Menge der Stationen und den Kanten des MLK-Graphen liefert eine erhebliche Beschleunigung der Branch-and-Bound-Schritte.

3.2.1 Reduktion der Stationenmenge mittels des Hitting-Set-Problems

Im Folgenden wollen wir beschreiben, wie sich die Streckennetzwerk-Reduktion auf das Hitting-Set-Problem übertragen lässt und warum man dabei keine optimale Lösung verliert.

Hitting-Set-Problem

Definition 3.2.1 (Hitting-Set-Problem)
Gegeben sei eine endliche Menge S, und eine Teilmenge $\mathcal{C}$ der Potenzmenge von S, d.h. $\mathcal{C} \subset 2^S$.

Eine Teilmenge $S_0 \subset S$ wird *Hitting-Set* von $\mathcal{C}$ genannt, wenn S_0 mindestens ein Element von jeder Menge in $\mathcal{C}$ enthält, d.h. $\forall c \in \mathcal{C} : c \cap S_0 \neq \emptyset$. Das Problem besteht darin, ein Hitting-Set minimaler Größe zu finden.

Hitting-Set-Problem für ein Streckennetzwerk

Die Menge S einer Instanz des Hitting-Set-Problems wird durch die Menge der Stationen S gebildet.

Die Menge $\mathcal{C}$ wird wie folgt gebildet:

- Definiere für alle $l_i, l_j \in L$ mit $l_i \neq l_j$ die Menge der Stationen, die sowohl von der Linie l_i als auch von der Linie l_j befahren werden:

$$S_{ij} = S(l_i) \cap S(l_j)$$

- Sei $s_0 \in S_{ij}$.
  $S_{ij}(s_0) = \{s \in S_{ij} | F(s_0, s, l_i) = F(s_0, s, l_j)\} \cup \{s_0\}$
  $S_{ij}(s_0)$ enthält die Stationen s aus S_{ij}, die von s_0 mit den Linien l_i und l_j in gleicher Fahrzeit erreichbar sind.
- $\mathcal{C} = \{S_{ij}(s_0) | l_i, l_j \in L, l_i \neq l_j, s_0 \in S_{ij}\}$

Satz 3.2.2 *Sei $N(S, C, t, L, T)$ ein Streckennetzwerk und sei $S_0 \subset S$ ein Hitting-Set von $\mathcal{C}$, wobei $\mathcal{C}$ wie oben gebildet wird. Dann gilt für jeden Fahrplan λ:*

$$\delta(\lambda) = \min_{s \in S} \delta(s, \lambda) = \min_{s \in S_0} \delta(s, \lambda)$$

D.h. für die Berechnung des Sicherheitsabstandes reicht die Menge der Stationen S_0.

Beweis: Sei $S_0 \subset S$ ein Hitting-Set von $\mathcal{C}$ und sei $\lambda \in \Lambda$ beliebig. Seien l_i, $l_j \in L$, $l_i \neq l_j$ und $s \in S_{ij}$.

Beh.: $\exists s_0 \in S_{ij} \cap S_0 : \delta(s_0, l_i, l_j, \lambda) = \delta(s, l_i, l_j, \lambda)$

Sei $C \in \mathcal{C}$ mit $s \in C$ und $C \subset S_{ij}$. Solch eine Menge C aus $\mathcal{C}$ existiert, da $\mathcal{C}$ disjunkte Teilmengen von S_{ij} enthält, deren Vereinigung S_{ij} ergibt.

Da S_0 ein Hitting Set von $\mathcal{C}$ ist, gilt: $S_0 \cap C \neq \emptyset$. Sei $s_0 \in S_0 \cap C$ beliebig, dann gilt:

$$F(s_0, s, l_i) = F(s_0, s, l_j)$$

$$\begin{aligned}
& \delta(s_0, l_i, l_j, \lambda) = \min_{r,t \in \mathbb{N}} |a(s_0, l_i, \lambda) + r \cdot T_i - a(s_0, l_j, \lambda) - t \cdot T_j| \\
= \; & \min_{r,t \in \mathbb{N}} |a(s_0, l_i, \lambda) + F(s_0, s, l_i) + r \cdot T_i - a(s_0, l_j, \lambda) - F(s_0, s, l_i) - t \cdot T_j| \\
= \; & \min_{r,t \in \mathbb{N}} |a(s_0, l_i, \lambda) + F(s_0, s, l_i) + r \cdot T_i - a(s_0, l_j, \lambda) - F(s_0, s, l_j) - t \cdot T_j| \\
= \; & \min_{r,t \in \mathbb{N}} |a(s, l_i, \lambda) + r \cdot T_i - a(s, l_j, \lambda) - t \cdot T_j| \\
= \; & \delta(s, l_i, l_j, \lambda)
\end{aligned}$$

Aus der Behauptung folgt mit der Definition von $\delta(s, \lambda)$ der Satz. □

Greedy-Algorithmus für das Hitting-Set-Problem

Die Bestimmung eines bzgl. der Kardinalität minimalen Hitting-Sets ist NP-hart [GJ79]. Darum ist es nicht sinnvoll das Hitting-Set-Problem optimal zu lösen, da unter Umständen bei bestimmten Streckennetzwerken die Berechnung eines minimalen Hitting-Sets sehr lange dauern könnte. Deshalb haben wir einen Greedy-Algorithmus implementiert.

Der Algorithmus setzt am Anfang alle Mengen in $\mathcal{C}$ unmarkiert. Der Algorithmus wählt in jedem Schritt eine Station, die in den meisten unmarkierten Mengen auftaucht. Dies wird solange wiederholt bis alle Mengen in $\mathcal{C}$ markiert sind.

Algorithmus 2: Greedy-Algorithmus für das Hitting-Set-Problem

Seien S und $\mathcal{C}$ gegeben;
$S_0 := \emptyset$;
while $\mathcal{C} \neq \emptyset$ **do**
  Wähle eine Station $s_0 \in S \backslash S_0$, für die gilt: $|\mathcal{C}_{s_0}| := \max_{s \in S \backslash S_0} |\mathcal{C}_s|$, wobei $\mathcal{C}_s = \{c \in \mathcal{C} : s \in c\}$;
  $S_0 := S_0 \cup \{s_0\}$;
  $\mathcal{C} := \mathcal{C} \backslash \mathcal{C}_{s_0}$;

Instanzen	$\|L\|$	ohne Red. $\|S\|$	HitSet-Red. $\|S_0\|$
I_2	20	461	16
I_{12}	28	543	28
I_{20}	26	525	22

Tabelle 3.1: Ergebnisse der Stationsreduktion für die KVB-Testinstanzen mit dem Greedy-Algorithmus 2.

Die Tabelle 3.1 gibt die Größe der Streckennetzwerke für die drei verschiedenen Instanzen für das Straßenbahnnetz der Kölner Verkehrs-Betriebe (KVB) vor und nach der Daten-Reduktion an. Genaueres zu den Instanzen ist im Abschnitt 6.2.1 angegeben. Wie man an der Tabelle 3.1 sieht, wird durch die Datenreduktion die Datenmenge um 95% reduziert.

Aus der Arbeit von Chvatal [Chv79] folgt eine Güteabschätzung für S_0. Der Beweis nutzt eine Transformation des Hitting-Set-Problems in das Set-Covering-Problem.

Satz 3.2.3 *Sei S_0^{opt} eine optimale Lösung des Hitting-Set-Problems und S_0 die von dem Greedy-Algorithmus 2 bestimmte Lösung. Es gilt:*

$$|S_0| \leq \sum_{j=1}^{d} \frac{1}{j} \cdot |S_0^{opt}|$$

wobei

$$d = \max_{s \in S} |\{c \in \mathcal{C} : s \in c\}|.$$

Bemerkung 3.2.4 Bei den oben angegebenen Instanzen ergibt sich jeweils der Wert $d = 10$, woraus mit Satz 3.2.3 folgt, dass die Greedy-Lösung die optimale Lösung mit dem Faktor 2.93 approximiert.

3.2.2 Reduktion der Stationenmenge mittels Relationen

In diesem Abschnitt werden eine Äquivalenzrelation und eine Halbordnung auf der Menge der Stationen vorgestellt, mit deren Hilfe die Summe der Sicherheitsabstände bzw. der minimale Sicherheitsabstand schneller bestimmt werden können.

Äquivalenzrelation zur Berechnung der Summe der Sicherheitsabstände

Wir wollen Stationen zusammenfassen, die denselben Sicherheitsabstand haben, gleichgültig welchen Fahrplan man auswählt.

Ein Fahrplan an einer Station ist die Gesamtheit aller Ankunftszeiten an dieser Station. Man kann ihn durch reguläre Polygone darstellen, wobei je ein reguläres Polygon die Ankunftszeiten einer Linie darstellt. Die Abstände zwischen den Polygonecken ändern sich nicht, wenn man alle Polygone um die gleiche Einheit dreht (siehe Abb. 3.1). Der minimale Abstand zwischen den Polygonecken ist der Sicherheitsabstand, d.h. der Sicherheitsabstand an zwei Stationen ist gleich, wenn diese von denselben Linien befahren werden und die Fahrzeit aller Linie zwischen den beiden Stationen gleich ist (modulo ihres Taktes).

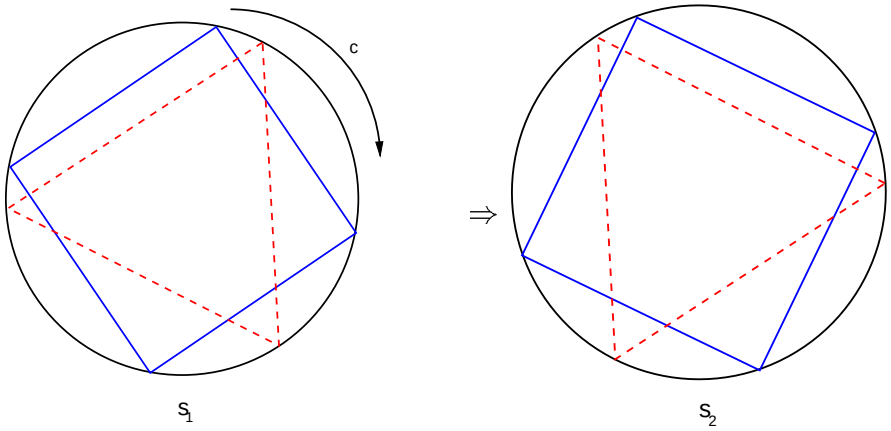


Abbildung 3.1: Zwei Stationen, die bzgl. $\sim_{R_\Sigma}$ äquivalent sind.

Im Folgenden definieren wir eine Relation, mit deren Hilfe wir eine geeignete Teilmenge der Stationen ermitteln werden, die ausreichend für die Berechnung der Summe der Sicherheitsabstände ist.

Definition 3.2.5 Sei S die Menge der Stationen eines Streckennetzwerks. Definiere eine Relation R_Σ auf der Menge S wie folgt:

$$s_1 \sim_{R_\Sigma} s_2 \Leftrightarrow$$

- $L(s_1) = L(s_2)$ und

- $\exists c \in \mathbb{Z}\ \forall l \in L(s_1) : F(s_1, s_2, l) = c$

D.h. zwei Stationen stehen in Relation zueinander, wenn beide durch die gleichen Linien befahren werden und die Fahrzeiten aller Linien zwischen den beiden Stationen gleich sind.

Aus der Definition 3.2.5 folgt offensichtlich der folgende Satz:

Satz 3.2.6 *Die Relation R_Σ in der obigen Definition ist eine Äquivalenzrelation.*

Lemma 3.2.7 *Seien s_1, $s_2 \in S$. Wenn $s_1 \sim_{R_\Sigma} s_2$ ist, so ist der Sicherheitsabstand an den Stationen s_1 und s_2 für jeden Fahrplan gleich, d.h.*

$$\forall \lambda \in \Lambda : \delta(s_1, \lambda) = \delta(s_2, \lambda)$$

Lemma 3.2.7 lässt sich leicht mit der Definition des Sicherheitsabstands beweisen.

Definition 3.2.8 Sei $S_\Sigma \subseteq S$. Wir bezeichnen S_Σ als Repräsentantenmenge der Faktormenge S/R_Σ wenn gilt:

- $|S_\Sigma| = |S/R_\Sigma|$ und
- $\bigcup_{s \in S_\Sigma} [s]_{R_\Sigma} = S$

Mit Hilfe von Lemma 3.2.7 kann der folgende Satz leicht bewiesen werden, der uns zeigt, dass wir für die Berechnung der Summe der Sicherheitsabstände nur die Faktormenge und die Kardinalitäten der Äquivalenzklassen brauchen.

Satz 3.2.9

$$\forall \lambda \in \Lambda : \delta_\Sigma(\lambda) = \sum_{s \in S} \delta(s, \lambda) = \sum_{s \in S_\Sigma} |[s]_{R_\Sigma}| \cdot \delta(s, \lambda)$$

Relation zur Berechnung des Sicherheitsabstands

Wir könnten den Sicherheitsabstand bereits mit der Stationsmenge S_Σ berechnen. Die Zahl der für die Berechnung des Sicherheitsabstands benötigten Stationen lässt sich aber noch weiter reduzieren. Da man bei der Sicherheitsabstandberechnung auf dem gesamten Streckennetzwerk das Minimum der Sicherheitsabstände an den einzelnen Stationen bildet, können wir Stationen, deren Sicherheitsabstand für jeden Fahrplan größer oder gleich dem Sicherheitsabstand an einer anderen Station ist, eliminieren. Die Abbildung 3.2 zeigt ein Beispiel. Wir haben zwei Stationen, alle Linien der Station s_1 fahren durch beide Stationen und jede Linie braucht die gleiche Fahrzeit. Dann ist offensichtlich, dass der Sicherheitsabstand an der Station s_1 größer oder gleich dem Sicherheitsabstand an der anderen Station s_2 ist. Da das Minimum der Sicherheitsabstände über alle Stationen beim Sicherheitsabstand berechnet wird, sind solche Stationen wie s_1 nicht von Bedeutung.

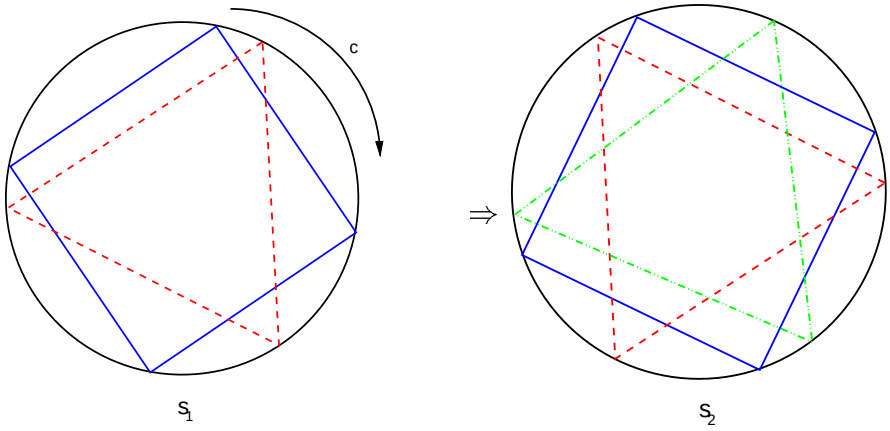


Abbildung 3.2: Zwei Stationen, die bzgl. $\leq_{R_\delta}$ in Beziehung stehen.

Definition 3.2.10 Sei S die Menge der Stationen eines Streckennetzwerks. Definiere eine Relation R_δ auf der Menge S wie folgt:

$$s_1 \leq_{R_\delta} s_2 \Leftrightarrow$$

- $L(s_1) \subseteq L(s_2)$ und
- $\exists c \in \mathbb{Z}\ \forall l \in L(s_1) : F(s_1, s_2, l) = c$

Satz 3.2.11 *Die Relation R_δ ist reflexiv und transitiv, aber weder symmetrisch noch antisymmetrisch.*

Satz 3.2.12 *Die Relation R_δ eingeschränkt auf die Menge S_Σ ist reflexiv, transitiv und antisymmetrisch, d.h. die Relation R_δ ist eine reflexive Halbordnung in S_Σ.*

Die Sätze 3.2.11 und 3.2.11 lassen sich leicht aus den Definition beweisen. Für die Relation R_δ gilt auch ein ähnliches Lemma wie in Lemma 3.2.7:

Lemma 3.2.13 *Seien s_1, $s_2 \in S$. Wenn $s_1 \leq_{R_\delta} s_2$, so gilt:*

$$\forall \lambda \in \Lambda : \delta(s_1, \lambda) \geq \delta(s_2, \lambda)$$

Definiere im Folgenden eine Menge, die zur schnellen Berechnung des Sicherheitsabstandes hilfreich sein wird.

Definition 3.2.14

$$S_\delta = \{s \in S_\Sigma | \forall s_1 \in S_\Sigma, s_1 \neq s : s \not\leq_{R_\delta} s_1\}$$

S_δ sind die maximalen Elemente von S_Σ bzgl. R_δ.

Mit Hilfe von Lemma 3.2.13 und der Definition 3.2.14 kann der folgende Satz leicht bewiesen werden, der uns zeigt, dass wir für die Berechnung der Sicherheitsabstände nur die Stationsmenge S_δ brauchen.

Satz 3.2.15

$$\forall \lambda \in \Lambda : \delta(\lambda) = \min_{s \in S} \delta(s, \lambda) = \min_{s \in S_\delta} \delta(s, \lambda)$$

Bemerkung 3.2.16 Die Relationen R_δ und R_Σ lassen sich leicht verallgemeinern, in dem man die folgende Zeile in der Definition:

$$\exists c \in \mathbb{Z}\ \forall l \in L(s_1) : F(s_1, s_2, l) = c$$

durch

$$\exists c \in \mathbb{Z}\ \forall l \in L(s_1) \exists t \in \mathbb{Z} : F(s_1, s_2, l) = c + t \cdot T_l$$

ersetzt. Alle Eigenschaften der beiden Relationen bleiben erhalten.

Instanzen	$\|L\|$	ohne Red. $\|S\|$	Relation $\|S_\Sigma\|$	Relation $\|S_\delta\|$	Greedy-Alg. $\|S_0\|$
I_2	20	461	28	16	16
I_{12}	28	543	48	28	28
I_{20}	26	525	44	22	22

Tabelle 3.2: Ergebnisse der Stationsreduktion für die KVB-Testinstanzen.

Die Tabelle 3.2 gibt die Anzahl der Stationen für die Instanzen vor und nach der Daten-Reduktion an. Wie man an der Tabelle 3.2 sieht, wird durch die Reduktion die Anzahl der Stationen, die für die Berechnung des Sicherheitsabstandes (bzw. Summe der Sicherheitsabstände) benötigt werden, um mindestens 95% (bzw. 90%) reduziert. Betrachtet man den Greedy-Algorithmus für das Hitting-Set-Problem, so erkennt man, dass dieser genau die Relation R_δ benutzt. Darum ist es nicht überraschend, dass die Anzahl der Stationen von S_0 und S_δ in der Tabelle 3.2 gleich ist.

Bemerkung 3.2.17 Eine andere Art von Datenreduktion stellt Weihe [Wei98] vor. Dabei geht es um die Überdeckung von Linien durch Stationen. Dort ist eine Menge von Linien gegeben. Es ist eine minimale Anzahl von Stationen gesucht, so dass jede Linie von mindestens einer Station überdeckt wird, d.h. jede Linie befährt mindestens eine Station aus den ausgewählten Stationen. Die Datenreduktion in seiner Arbeit nutzt Relationen aus, die auf den Linien und Stationen eines Streckennetzwerks definiert sind.

3.2.3 Reduktion durch eine Äquivalenzrelation auf Kanten des MLK-Graphen

Die bisher vorgestellten Reduktionen versuchen die Anzahl der Stationen zu verringern, um so die Berechnung des Sicherheitsabstandes und der Summe der Sicherheitsabstände zu beschleunigen. Nach diesen Reduktionen werden trotzdem manche Sicherheitsabstände zwischen zwei Linien mehrfach an verschiedenen Stationen berechnet. Um dies zu vermeiden, betrachtet man die Kanten des Multi-Linienkonflikt-Graphen (MLK-Graphen), die die Konflikte zwischen den Linien an den einzelnen Stationen darstellen.

Dazu wird ähnlich zu den bisher definierten Relationen auf der Menge der Stationen, eine Relation auf der Menge der Kanten des MLK-Graphen definiert.

Mit Hilfe dieser Relation konnten auch die Anzahl der Variablen und (Un-)Gleichungen in den ganzzahligen linearen Programm-Darstellungen des Fahrplan-Problems in Kapitel 5 erheblich verringert werden.

Nach Definition gilt für den Sicherheitsabstand:

$$\begin{aligned}\delta(\lambda) &= \min_{s\in S}\delta(s,\lambda)\\ &= \min\{\min_{l\in L} T_l, \min_{s\in S, l_j,l_k\in L(s), l_j\neq l_k}\delta(s,l_j,l_k,\lambda)\}\end{aligned}$$

In der Regel gilt für jeden Fahrplan λ:

$$\min_{l\in L} T_l \geq \min_{s\in S, l_j,l_k\in L(s), l_j\neq l_k}\delta(s,l_j,l_k,\lambda)$$

Damit diese Bedingung verletzt wird, muss mindestens eine Linie existieren, die keine gemeinsame Station mit anderen Linien hat.

Sei jetzt o.B.d.A:

$$\delta(\lambda) = \min_{s\in S, l_j,l_k\in L(s), l_j\neq l_k}\delta(s,l_j,l_k,\lambda)$$

Nach Definition 2.5.2 ist die Menge der Kanten im MLK-Graph wie folgt definiert:

$$\begin{aligned}M &= \{(s,l_j,l_k)|l_j\neq l_k \wedge l_j,l_k\in L(s)\}\\ &= \{(s,l_j,l_k)|l_j\neq l_k \wedge s\in S(l_j)\cap S(l_k)\}\end{aligned}$$

Offensichtlich gilt dann für den Sicherheitsabstand:

$$\delta(\lambda) = \min_{m\in M, m=(s,l_j,l_k)}\delta(s,l_j,l_k,\lambda)$$

Schreibe ab jetzt kurz:

$$\delta(\lambda) = \min_{m\in M}\delta(m,\lambda) \tag{3.2.1}$$

Um den Sicherheitsabstand zu berechnen, benötigen wir nicht die ganze Kantenmenge M des MLK-Graphen. Die Elemente von M, die zur Berechnung des Sicherheitsabstands nötig sind, werden durch eine Relation bestimmt.

Definition 3.2.18 Sei M die Menge wie oben beschrieben. Definiere eine Relation R_M auf der Menge M wie folgt:

$$m_1 = (s_1, l_{11}, l_{12}) \sim_{R_M} m_2 = (s_2, l_{21}, l_{22})$$
$$\Leftrightarrow \{l_{11}, l_{12}\} = \{l_{21}, l_{22}\} \wedge F(s_1, s_2, l_{11}) = F(s_1, s_2, l_{12})$$

Aus der Definition 3.2.18 ergibt sich offensichtlich das nächste Ergebnis:

Satz 3.2.19 *Die Relation R_M ist eine Äquivalenzrelation.*

Satz 3.2.20 *Seien m_1,$m_2 \in M$. Wenn $m_1 \sim_{R_M} m_2$ gilt, so ist der Sicherheitsabstand auf m_1 und m_2 für jeden Fahrplan gleich, d.h.*

$$\forall \lambda \in \Lambda : \delta(m_1, \lambda) = \delta(m_2, \lambda)$$

Der Satz 3.2.20 folgt sofort aus der Definition des Sicherheitsabstandes zweier Linien. Aus dem Satz 3.2.20 ergibt sich:

Korollar 3.2.21 *Sei $M_0 \subseteq M$ die Repräsentantenmenge von M. Dann gilt:*

$$\delta(\lambda) = \min_{m \in M} \delta(m, \lambda) = \min_{m \in M_0} \delta(m, \lambda)$$

D.h. wir benötigen zur Berechnung des Sicherheitsabstands ein Element aus jeder Äquivalenzklasse der Äquivalenzrelation $\sim_{R_M}$.

Instanzen	$\|L\|$	$\|S\|$	$\|M\|$	$\|M_0\|$	$\frac{\|M_0\|}{\|M\|}$ %
I_2	20	461	202	46	22.77
I_{12}	28	543	383	78	20.37
I_{20}	26	525	320	68	21.25

Tabelle 3.3: Ergebnisse der Reduktion der Kanten des MLK-Graphen für die KVB-Testinstanzen.

Definition 3.2.22 Sei $M_0 \subseteq M$ die Repräsentantenmenge von M. Definiere eine Abbildung $r : M \to M_0$, die jedem $m \in M$ seinen Repräsentanten zuordnet, d.h.

$$r(m) = m_0 \Leftrightarrow m \sim_{R_M} m_0 \wedge m \in M \wedge m_0 \in M_0$$

$$M(s) := \{m \in M | \exists l_j, l_k \in L(s) : m = (s, l_j, l_k)\}$$
$$M_0(s) := \{m_0 \in M_0 | \exists m \in M(s) : r(m) = m_0\}$$

Im Folgenden wollen wir erläutern, wie man aus der Menge M_0 die Summe der Sicherheitsabstände berechnet. Dazu muss zuerst geklärt werden, wie man den Sicherheitsabstand an einer Station berechnet.

Aus der Definition 3.2.22 und dem Satz 3.2.20 ergibt sich für die Berechnung des Sicherheitsabstandes an einer Station, die von mehr als zwei Linien befahren wird, das folgende Lemma:

Lemma 3.2.23 *Sei λ ein beliebiger Fahrplan und sei $s \in S$ eine Station, die von mindestens zwei Linien befahren wird, d.h. $|L(s)| >= 2$, so gilt für den Sicherheitsabstand an der Station s:*

$$\delta(s,\lambda) = \min_{m \in M(s)} \delta(m,\lambda) = \min_{m \in M_0(s)} \delta(m,\lambda)$$

Lemma 3.2.23 und Satz 3.2.15 führen zu folgender Aussage, die uns einen Algorithmus zur Berechnung der Sicherheitsabstände liefert.

Satz 3.2.24

$$\forall \lambda \in \Lambda : \delta(\lambda) = \min_{s \in S_\delta} \delta(s,\lambda) = \min_{s \in S_\delta} \min_{m \in M_0(s)} \delta(m,\lambda)$$

Der folgende Satz, der sich aus Lemma 3.2.23 und Satz 3.2.9 ergibt, liefert uns einen Algorithmus, mit dem wir schnell die Summe der Sicherheitsabstände berechnen können.

Satz 3.2.25

$$\forall \lambda \in \Lambda : \delta_\Sigma(\lambda) = \sum_{s \in S_\Sigma} |[s]_{R_\Sigma}| \cdot \min_{m \in M_0(s)} \delta(m,\lambda)$$

Der nächste Satz sagt aus, dass man mit Hilfe der Menge M_0, siehe Korollar 3.2.21, mindestens genauso schnell den Sicherheitsabstand berechnen kann wie mit Hilfe von S_δ, siehe Satz 3.2.15.

Satz 3.2.26 *Sei $M_1 = \bigcup_{s \in S_\delta} M(s)$. Dann gilt:*

$$|M_0| \leq |M_1|$$

Beweis: Angenommen: $|M_0| > |M_1|$
Dann ist M_1 keine Repräsentantenmenge von M bzgl. der Äquivalenzrelation $\sim_{R_M}$, da alle Repräsentantenmengen gleiche Kardinalität haben.
Daraus folgt: $\exists m_0 \in M_0 : \forall m \in [m_0]_{R_M} : m \notin M_1$.
Das ist äquivalent zu:
$\exists m_0 \in M_0 : \forall m \in [m_0]_{R_M} : \forall m_1 \in M_1 : m \not\sim_{R_M} m_1$. (*)
Sei $m_0 = (s_0, l_1, l_2)$.
Da S_δ die maximalen Elemente aus S bzgl. der Relation $\leq_{R_\delta}$ enthält, gilt:
$\exists s_1 \in S_\delta : s_0 \leq_{R_\delta} s_1 \Leftrightarrow L(s_0) \subseteq L(s_1) \wedge \exists c \in \mathbb{Z}\ \forall l \in L(s_0) : F(s_0, s_1, l) = c$.
Daraus folgt: $\exists m = (s_1, l_1, l_2) \in M_1 : m_0 \sim_{R_M} m_1$. Dies ist aber ein Widerspruch zu (*). □

Der Kardinalitätsunterschied zwischen der Menge M_0 und M_1 kann sehr groß werden, zur Verdeutlichung betrachten wir ein Streckennetzwerk, wo eine Linie einen anderen Linienweg nimmt als alle anderen Linien. Genaueres wird im folgenden Beispiel dargestellt.

Beispiel 3.2.27 Zuerst betrachten wir ein Streckennetzwerk mit drei Stationen und m Linien wie in Abbildung 3.3.

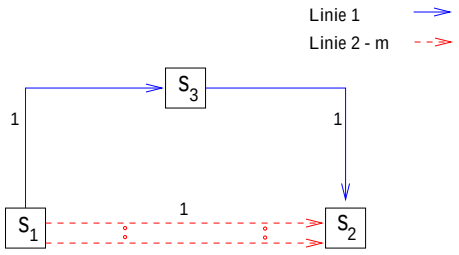


Abbildung 3.3: Ein Streckennetzwerk mit 3 Stationen und m Linien.

Die Menge M für das Streckennetzwerk in Abbildung 3.3 sieht dann wie folgt aus:

$$M = \{(s, l_j, l_k) | l_j \neq l_k \wedge l_j, l_k \in L, s \in \{s_1, s_2\}\}$$

Da die Fahrzeiten aller Linien von Station s_1 zu s_2 gleich sind, bis auf die Linie l_1, gilt für die Menge M_0:

$$M_0 = \{(s_1, l_j, l_k) | j < k \wedge l_j, l_k \in L\} \cup \{(s_3, l_1, l_k) | l_k \in L, l_k \neq l_1\}$$

Dann gilt für die Anzahl der Elemente in der Menge M_0 folgendes:

$$|M_0| = \frac{(m-1) \cdot m}{2} + m - 1$$

Da die Linie l_1 einen von anderen Linien verschiedenen Weg nimmt und die Fahrzeit von Station s_1 zu Station s_2 verschieden von denen der anderen Linien ist, gibt es keine $\leq_{R_\delta}$-Beziehung zwischen diesen beiden Stationen. Deshalb enthält die Menge S_δ beide Stationen, d.h. $S_\delta = \{s_1, s_2\}$.

$$M_1 = \{(s, l_j, l_k) | j < l_k \wedge l_j, l_k \in L, s \in \{s_1, s_2\}\}$$

$$|M_1| = 2 \cdot \frac{(m-1) \cdot m}{2} = (m-1) \cdot m$$

$$|M_1| - |M_0| = \frac{(m-1) \cdot m}{2} - (m-1) = \frac{(m-1) \cdot (m-2)}{2} \geq 0$$

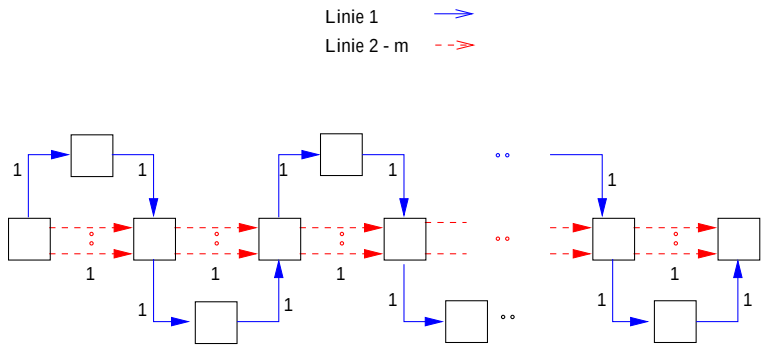


Abbildung 3.4: Erweiterung des Streckennetzwerks aus Abbildung 3.3 auf n Stationen.

Der Größenunterschied zwischen M_0 und M_1 kann stark steigen, wenn man das Streckennetzwerk aus Abbildung 3.3 wie in Abbildung 3.4 erweitert.

Mit den gleichen Überlegungen wie vorhin beschrieben erhält man für die Anzahl der Elemente von M_0 und M_1 für das Streckennetzwerk in Abbildung 3.4:

- $|M_0| = \frac{(m-1)\cdot m}{2} + \left(\frac{n-3}{2} + 1\right) \cdot (m-1)$
- $|M_1| = \left(\frac{n-3}{2} + 2\right) \cdot \frac{(m-1)\cdot m}{2}$

$$|M_1| - |M_0| = \left(\frac{n-3}{2} + 1\right) \cdot \left(\frac{(m-1)\cdot m}{2} - (m-1)\right) = O(m^2 \cdot n)$$

Bemerkung 3.2.28 Im Branch-and-Bound-Algorithmus, den wir implementiert haben, wurden alle Möglichkeiten der Berechnung der Sicherheitsabstande bzw. der Summe der Sicherheitsabstände implementiert. Es stellte sich heraus, dass die Berechnung mit Hilfe von Korollar 3.2.21 bzw. Satz 3.2.25 am schnellsten ist.

3.3 Sicherheitsabstand- und Fahrplan-Problem

In diesem Abschnitt wird gezeigt, dass das Entscheidungsproblem zum Sicherheitsabstand-Problem schon im einfachen Fall, wenn die Takte aller Linien gleich sind, NP-vollständig ist. Die NP-Vollständigkeit wird durch Reduktion des k-Färbungsproblems auf das Entscheidungsproblem bewiesen. Der NP-Vollständigkeitsbeweis wurde in meiner Diplomarbeit [Gen99] und in [HMS97] bereits behandelt.

3.3.1 Grundbegriffe der NP-Vollständigkeitstheorie

Um Unklarheiten zu vermeiden, wiederholen wir an dieser Stelle die von uns verwendeten Definitionen und Sätze. Diese sind aus [Weg93] entnommen.

Definition 3.3.1 P ist die Klasse aller Probleme, für die es eine deterministische Turingmaschine M gibt, deren Rechenzeit polynomiell beschränkt ist.

Definition 3.3.2 NP (nichtdeterministisch polynomiell) ist die Klasse aller Probleme, die von einer nichtdeterministischen Turingmaschine M in polynomieller Zeit akzeptiert werden.

Definition 3.3.3 Es seien L_1 und L_2 Sprachen über den Alphabeten Σ_1 und Σ_2. Dann heißt L_1 polynomiell auf L_2 reduzierbar, Notation $L_1 \leq_p L_2$, wenn es eine polynomielle Transformation von L_1 nach L_2 gibt, d.h. wenn es eine von einer deterministischen Turingmaschine in polynomieller Zeit berechenbare Funktion

$$f : \Sigma_1^* \to \Sigma_2^*$$

gibt, so dass für alle $\omega \in \Sigma_1^*$ gilt:

$$\omega \in L_1 \Leftrightarrow f(\omega) \in L_2$$

Definition 3.3.4 Eine Sprache L heißt NP-vollständig, wenn $L \in$ NP ist und für alle $\tilde{L} \in$ NP gilt: $\tilde{L} \leq_p L$

Satz 3.3.5 *Sei $L_2 \in$ NP und $L_1 \leq_p L_2$ für ein NP-vollständiges Problem L_1. Dann ist L_2 NP-vollständig.*

3.3.2 NP-Vollständigkeitsbeweis des Sicherheitsabstand-Problems

Wir zeigen, dass das Entscheidungsproblem zum Sicherheitsabstand-Problem NP-vollständig ist. Beim Beweis werden wir das als NP-vollständig bekannte k-Färbungsproblem, [GJ79], auf das Entscheidungsproblem reduzieren. Es wird sich herausstellen, dass die Reduktion genauso möglich ist, wenn wir uns auf das Entscheidungsproblem zum Sicherheitsabstand-Problem mit gleichem Takt beschränken.

Definition 3.3.6 Das k-Färbungsproblem:
Input: Ein ungerichteter, einfacher Graph G=(V,E) und eine Zahl $k \in \mathbb{N}$.
Frage: Ist G k-färbbar, d.h. gibt es eine Färbung $c : V \to \{0, 1, \ldots, k-1\}$, so dass $c(u) \neq c(v), \forall (u, v) \in E$?

Definition 3.3.7 Das Entscheidungsproblem des Sicherheitsabstand-Problems:
Input: Ein Streckennetzwerk $N(S, C, t, L, T)$ und eine Zahl $\Delta \in \mathbb{N}$.
Frage: Gibt es ein Fahrplan $\lambda \in \Lambda$, so dass der Sicherheitsabstand $\delta(\lambda) \geq \Delta$ ist?.

Satz 3.3.8 *Das Entscheidungsproblem des Sicherheitsabstand-Problems ist NP-vollständig.*

Beweis: Das Entscheidungsproblem des Sicherheitsabstand-Problems ist offensichtlich aus NP, da wir für einen Fahrplan λ in polynomieller Zeit entscheiden können, ob $\delta(\lambda) \geq \Delta$ ist.

Sei eine Instanz des k-Färbungsproblems gegeben, also ein einfacher, ungerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$. Aus dieser konstruieren wir eine Instanz für das Entscheidungsproblem zum Sicherheitsabstand-Problem. Die Stationen S seien die Kanten E im Färbungsgraphen ($S := E$). Die Verbindungen C im Streckennetzwerk ergeben sich aus der Definition der Linien (s.u.). Die Fahrzeit der einzelnen Verbindungen sei immer gleich 0 (oder ein Vielfaches der Färbungszahl k): $t \equiv 0$. Die Linien L sind die Knoten V des Färbungsgraphen. Die Linienwege werden wie folgt konstruiert: Zuerst nummeriert man die Kanten E in einer beliebigen Reihenfolge. Es ist wichtig, dass keine zwei Kanten die gleiche Nummer besitzen. Eine Linie l, d.h. ein Knoten im Färbungsgraphen, fährt über alle Stationen (Kanten im Färbungsgraphen), die inzident zur Linie l sind. Die Reihenfolge, wie die Stationen befahren werden, wird durch die Kantennummerierung festgelegt. Eine Linie befährt die Stationen in aufsteigender Reihenfolge ihrer Kantennummerierung. D.h. die erste Station, durch die die Linie fährt, hat die kleinste Kantennummer und die zweite Station durch die die Linie fährt, hat die zweit kleinste Kantennummer, usw.. Der Takt aller Linien sei immer gleich der Färbungszahl k ($T \equiv k$). Zuletzt definieren wir den Mindestsicherheitsabstand $\Delta = 1$.

Eine Färbung c ist gleichzeitig Lösung des Entscheidungsproblems im konstruierten Streckennetzwerk für $\Delta = 1$, da für jede Station (Kante) gilt, dass je zwei Linien (adjazente Knoten), die sich in einer Station kreuzen, unterschiedliche Ankunftszeiten (Farben) haben müssen, da wegen $t \equiv 0$ die Ankunftszeit gleich der Abfahrtszeit der Linien in jeder Station ist. Dies gilt auch, wenn die Zeiten für die Verbindungen ein Vielfaches des Taktes k sind.

Die Umkehrung gilt auch:

Ein Fahrplan λ, der den Sicherheitsabstand Δ erfüllt, ist gleichzeitig Lösung des Färbungsproblems, da für jede Kante (Station) gilt, dass die beiden zu ihr inzidenten Knoten (Linien) unterschiedliche Farben (Abfahrtszeiten) haben müssen.

Die Reduktion des k-Färbungsgraphen auf das Entscheidungsproblem des Sicherheitsabstand-Problems ist offensichtlich polynomiell. Nach Satz 3.3.5 gilt daher: Das Entscheidungsproblem des Sicherheitsabstand-Problems ist NP-vollständig.

□

Wir haben die NP-Vollständigkeit des Entscheidungsproblems zum Sicherheitsabstand-Problem gezeigt. Für den Beweis benötigten wir lediglich Instanzen des Sicherheitsabstand-

Problems mit einheitlichem Takt auf allen Linien, d.h. schon diese Klasse ist NP-vollständig.

Das folgende Beispiel zeigt die Transformation eines 3-Färbungproblems auf ein Streckennetzwerk.

Beispiel 3.3.9 Gegeben sei das 3-Färbungsproblem in Abbildung 3.5 mit 5 Knoten und 7 Kanten. Wie man leicht sieht, ist dieser Graph 3-färbbar. Definiere z.B. Färbung $c : V \to \{0, 1, 2\}$ wie folgt: $c(v_1) = 2$, $c(v_2) = 1$, $c(v_3) = 0$, $c(v_4) = 1$, $c(v_5) = 2$.

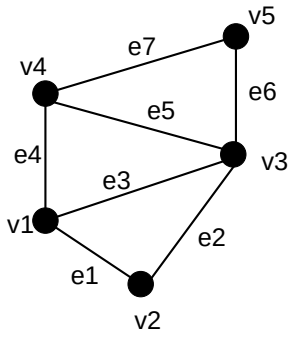


Abbildung 3.5: Färbungsgraph

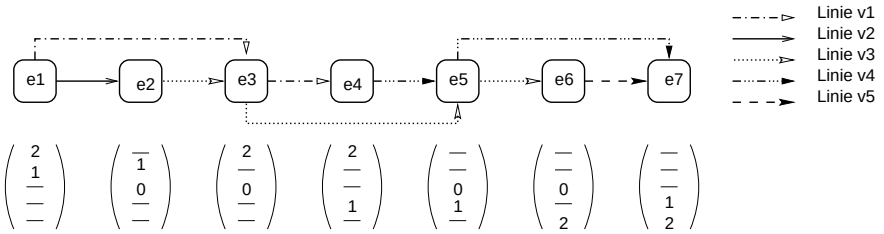


Abbildung 3.6: Nach Transformation des 3-Färbungsgraphen in Abbildung 3.5 entstandenes Streckennetzwerk.

In den Vektoren unter den Stationen in Abbildung 3.6 stehen die Ankunftszeiten der Linien, wenn man als Fahrplan λ die Färbung c nimmt, d.h., wenn $\lambda = (2, 1, 0, 1, 2)$ gilt.

Wie man leicht in Abbildung 3.6 sieht, kommen in einer Station im Streckennetzwerk, die einer Kante im Färbungsgraphen entspricht, immer genau zwei Linien an. Diese beiden Linien u und v einer Station e sind genau die beiden Knoten, durch die die Kante e im Färbungsgraphen definiert ist, d.h. $e = u - v$.

Das Entscheidungsproblem zum Sicherheitsabstand-Problem ist auch NP-vollständig, wenn man nur eine Station betrachtet und die Linien unterschiedliche Takte haben. In einer Arbeit von Vince [Vin89] ist ein Beweis für ein äquivalentes Problem für den Sicherheitsabstand mit einer Station angegeben. Beim Beweis wird auch das Färbungsproblem

auf dieses Problem zurückgeführt. Dort werden reguläre Polygone in Kreisen betrachtet, genauer wurde dieses Problem im Abschnitt 1.2.3 beschrieben.

Bemerkung 3.3.10 Das Fahrplan-Problem ist offensichtlich auch NP-schwer, da unter den Lösungen des Sicherheitsabstand-Problems eine mit maximaler Summe der Sicherheitsabstände gesucht wird und das Sicherheitsabstand-Problem an sich schon NP-vollständig ist.

3.4 Verringerung der Dimension des Suchraums

Der Suchraum (=Menge aller Fahrpläne) hat folgende Gestalt:

$$\Lambda = \mathbb{N}_{T_1} \times \mathbb{N}_{T_2} \times \cdots \times \mathbb{N}_{T_m}$$

In diesen Abschnitt werden wir zeigen, dass die Dimension des Suchraumes um eine Dimension verringert werden kann. Diese Beobachtung wurde auch z.B. von Voget [Vog95] angewandt. Dazu benötigen wir den folgenden Satz. Dieser besagt, dass der Sicherheitsabstand und die Summe der Sicherheitsabstände gleich bleiben bei Verschiebung der Fahrzeiten um den gleichen Betrag.

Satz 3.4.1 *Sei $c \in \mathbb{N}_0$. Sei $\lambda \in \Lambda$ beliebig und $\mu \in \Lambda$ mit $\mu_i = (\lambda_i + c) \bmod T_i$, $1 \leq i \leq m$, dann gilt:*

$$\delta(\lambda) = \delta(\mu) \quad \textit{und} \quad \delta_\Sigma(\lambda) = \delta_\Sigma(\mu)$$

Dieser Satz lässt sich mit der Definition des Sicherheitsabstandes leicht beweisen.

Sei $\Lambda_2 = \{0\} \times \mathbb{N}_{T_2} \times \cdots \times \mathbb{N}_{T_m}$. Die Dimension von Λ_2 ist um eins geringer als die Dimension von Λ, da bei den Fahrplänen von Λ_2 die Linie 1 die feste Abfahrtzeit 0 hat.

Wegen Satz 3.4.1 gibt es zu jedem Fahrplan $\lambda \in \Lambda$ einen Fahrplan $\mu \in \Lambda_2$ mit denselben Eigenschaft. D.h. wir können den Suchraum Λ durch den Suchraum Λ_2 im Fahrplan-Problem ersetzen, ohne etwas zu verlieren.

Bemerkung 3.4.2 Λ_2 hätte auch so aussehen können:

$$\Lambda_2 = \mathbb{N}_{T_1} \times \cdots \times \mathbb{N}_{T_{k-1}} \times \{0\} \times \mathbb{N}_{T_{k+1}} \times \cdots \times \mathbb{N}_{T_m}$$

mit $k \in \{1, \ldots, m\}$. Die Wahl derjenigen Linie, deren Abfahrtszeit man auf 0 setzt, kann bei der Zerlegung des Streckennetzwerks in Abschnitt 4.5 eine wichtige Rolle spielen.

Kapitel 4

Ein Branch-and-Bound-Algorithmus zur Lösung des Fahrplan-Problems

In diesem Kapitel wollen wir den Lösungsansatz mit Branch-and-Bound vorstellen. Mit dessen Hilfe versuchen wir, das Fahrplan-Problem zu lösen. Dabei werden wir erläutern, was die Begriffe "Branching" und "Bounding" bedeuten und die Anwendung auf das Fahrplan-Problem diskutieren. Wir werden ferner Heuristiken für das Fahrplan-Problem vorstellen. Schließlich wird noch untersucht, wie sich der Branch-and-Bound-Algorithmus beschleunigen lässt, wenn sich das Streckennetzwerk in signifikant kleinere Teilnetze mit Hilfe der zweifachen Zusammenhangskomponenten des Linienkonflikt-Graphen zerlegen lässt.

4.1 Branch-and-Bound-Verfahren

Die Methode Branch-and-Bound ist ein Verfahren zur Lösung von kombinatorischen Optimierungsproblemen.

Dabei wird ein Suchbaum rekursiv aufgebaut und systematisch nach Lösungen durchsucht. Jeder Knoten K in dem Suchbaum entspricht einem Teilproblem. Unter einem Teilproblem versteht man die Einschränkung des Lösungsraums bei gleich bleibender Zielfunktion, wobei die Lösungsraumbeschränkung durch Fixieren einzelner Entscheidungen erzielt wird. Der Weg von der Wurzel des Suchbaumes zum Knoten K kodiert die getroffene Entscheidungsfolge. Die Lösungen befinden sich an den Blättern des Suchbaumes. Das Treffen von Entscheidungen und das Erzeugen von Teilproblemen werden mit Hilfe der Operationen *branch* und *bound* erzielt.

Beim Branching wird ein Problem P in kleinere Probleme $P_1, \ldots, P_k$ aufgeteilt. Branching ist ein rekursiver Prozess. Jeder Branching-Schritt lässt sich durch einen Branching-Baum darstellen, wobei P der Vater von $P_1, \ldots, P_k$ ist. Die Probleme P_i $(i = 1, \ldots, k)$ nennt man Unterprobleme.

Beim Bounding wird für ein Teilproblem P eine obere Schranke $O(P)$ (bei Maximierungproblemen) berechnet. Wenn die obere Schranke $O(P)$ nicht größer als die bisherige aktuelle beste Lösung ist, so kann man dieses Problem verwerfen, da es keine bessere Lösung liefern kann, d.h. es ist nicht mehr nötig von dem Problem Unterprobleme zu betrachten. Falls die obere Schranke $O(P)$ größer als die bisherige aktuelle beste Lösung ist, so wird diese in eine Liste aufgenommen. Mit Hilfe dieser Operation kann entschieden werden, welches Problem weiter aufgespalten wird.

Bei der Best-First-Search Methode wird das Teilproblem mit der größten oberen Schranke expandiert, dies basiert auf der Vermutung, dass Elemente mit der größten oberen Schranke am schnellsten zu einer optimalen Lösung führen. Die Depth-First-Search-Methode geht bei der Suche immer zuerst in die Tiefe. Dieses Verfahren nennt man auch Backtracking. Der Vorteil des Verfahrens besteht darin, dass der benötigte Speicherplatz linear mit der Tiefe des Baumes wächst.

4.1.1 Branching

Branching bezieht sich auf das Zerlegen des Lösungsraums. Beim Fahrplan-Problem und Sicherheitsabstand-Problem werden der Menge aller möglichen Fahrpläne Λ aufgespalten. Sei $\{i_1, i_2, \ldots, i_m\}$ eine Linienreihenfolge, nach der die Abfahrtszeiten der Linien gewählt werden.

1. Im ersten Branching-Schritt wählt man die Variable λ_{i_1}, d.h. man wählt alle möglichen Abfahrtszeiten der Linie l_{i_1}, und zerlegt den Lösungsraum Λ in T_{i_1} Teile. Sei $r_{i_1} \in \{0, \ldots, T_{i_1} - 1\}$.

$$\Lambda[r_{i_1}] \quad = \quad \{\lambda \in \Lambda | \lambda_{i_1} = r_{i_1}\}$$

2. Wenn die Variablen $\lambda_{i_1}, \ldots, \lambda_{i_k}$ bereits festgelegt sind, so zerlegt man die Menge $\Lambda[r_{i_1}, \ldots, r_{i_k}]$ in:

$$\Lambda[r_{i_1}, \ldots, r_{i_k}, r_{i_{k+1}}] \quad = \quad \{\lambda \in \Lambda[r_{i_1}, \ldots, r_{i_k}] | \lambda_{i_{k+1}} = r_{i_{k+1}}\}$$

   wobei $r_{i_{k+1}} \in \{0, \ldots, T_{i_{k+1}} - 1\}$ ist.

Der so konstruierte Baum hat die Tiefe $|L|$. Die in der Menge $\Lambda[r_{i_1}, \ldots, r_{i_k}]$ in ihrem Wert $\lambda_j = r_j$ festgelegte Variable bezeichnet man auch als *fixierte* (gebundene) Variable, die übrigen bezeichnet man als *freie* Variablen.

4.1.2 Bounding und Pruning

Durch die Verzweigungsstruktur allein ist bzgl. des benötigten Rechenaufwands wenig gewonnen, da die Lösung der Teilprobleme im Allgemeinen genauso schwierig sein kann wie

die Lösung des Ausgangsproblems. Daher versucht man Teilprobleme zu bewerten (bounding), um mit Hilfe dieser Bewertung Teilmengen von Λ zu eliminieren (pruning), die nicht mehr zu optimalen Lösungen führen können, und die Teilprobleme mit bester Bewertung zuerst zu expandieren, da diese am schnellsten zu einer optimalen Lösung führen könnten.

Die Bewertung der Teilprobleme erfolgt durch obere Schranken, da wir Maximierungsprobleme betrachten.

4.2 Obere Schranken

In diesem Abschnitt werden obere Schranken für die Sicherheitsabstände hergeleitet. Als erstes werden einige Sätze vorgestellt, die eine a priori Aussage erlauben, wie gut der Sicherheitsabstand höchstens sein kann - nur in Abhängigkeit von den Daten des Streckennetzwerks. Diese gelten für alle Fahrpläne. Danach wird die Güte dieser oberen Schranken für Takte, die üblich für Linien im ÖPNV sind, untersucht. Anschliessend werden obere Schranken für den Sicherheitsabstand für das gesamte Streckennetzwerk in Abhängigkeit von Teilfahrplänen untersucht.

Definition 4.2.1 (obere Schranke)
Gegeben ist ein Streckennetzwerk $N(S, C, t, L, T)$. Seien $\overline{\delta}, \overline{\delta_\Sigma} \in \mathbb{N}$. Wenn $\forall \lambda \in \Lambda : \delta(\lambda) \leq \overline{\delta}$ (bzw. $\forall \lambda \in \Lambda : \delta_\Sigma(\lambda) \leq \overline{\delta_\Sigma}$) gilt, so heißt $\overline{\delta}$ (bzw. $\overline{\delta_\Sigma}$) obere Schranke für den Sicherheitsabstand (bzw. obere Schranke für die Summe der Sicherheitsabstände).

Eine Zahl $\overline{\delta(s)} \in \mathbb{N}$ heißt obere Schranke für den Sicherheitsabstand an der Station $s \in S$, wenn für jeden Fahrplan $\lambda \in \Lambda$ gilt: $\delta(\lambda, s) \leq \overline{\delta(s)}$.

Bemerkung 4.2.2 Eine obere Schranke für die Summe der Sicherheitsabstände liefert z.B. die Summe der oberen Schranken für die Sicherheitsabstände an den einzelnen Stationen, d.h. $\overline{\delta_\Sigma} = \sum_{s \in S} \overline{\delta(s)}$.

Wenn obere Schranken für die Sicherheitsabstände an den einzelnen Stationen bereits vorliegen, so können wir eine obere Schranke für den Sicherheitsabstand durch Minimumbildung berechnen. $(\overline{\delta} = \min_{s \in S} \overline{\delta(s)})$

Eine einfache obere Schranke für den Sicherheitsabstand an einer Station ist das Minimum der Takte der Linien, die durch diese Station fahren. $(\overline{\delta(s)} = \min_{l \in L(s)} T_l)$

4.2.1 Obere Schranken für den Sicherheitsabstand an einer Station

In Abschnitt 1.2.3 haben wir das Polygon-Scheduling-Problem vorgestellt. Der Fall $p = -\infty$ ist für uns von Interesse, wir bezeichnen diesen mit Polygon-Problem.

Definition 4.2.3 (Polygon-Problem)
Gegeben sei ein Kreis C der Länge A, n reguläre Polygone $P = \{P_1, \ldots, P_n\}$ wobei jedes Polygon P_i m_i Ecken hat. Sei $m = \sum_{i=1}^{n} m_i$ die Anzahl aller Ecken aller Polygone im Kreis.

$$t^* = \max_{a \in [\mathbb{R}/A]^n} \min_{1 \leq k \leq m} u_k$$

Dabei ist u_k der Abstand zwischen zwei benachbarten Ecken auf dem Kreis C für einen festen Positionsvektor a.

Wenn wir ein Streckennetzwerk $N(S, C, t, L, T)$ betrachten, bei dem alle Linien durch die gleichen Stationen fahren, so braucht man zur Lösung des Sicherheitsabstand-Problems nur eine Station zu betrachten, d.h.

$$\delta^* = \max_{\lambda \in \Lambda} \min_{s \in S} \delta(s, \lambda) = \max_{\lambda \in \Lambda} \delta(s_0, \lambda)$$

für ein beliebiges $s_0 \in S$.

Bei solchen Streckennetzwerken ist ein optimaler Fahrplan für das Sicherheitsabstand-Problem auch ein optimaler Fahrplan für das Fahrplan-Problem, da die Sicherheitsabstände an allen Stationen für jeden Fahrplan gleich sind.

Das Sicherheitsabstand-Problem für solche Streckennetzwerke lässt sich einfach in das Polygon-Problem überführen. Beim Polygon-Problem hat man eine Menge von regulären Polygonen, die man in einem Kreis mit konstanten Umfang so platzieren soll, dass der minimale Abstand zweier benachbarter Ecken maximal ist.

Dabei gibt es einen wichtigen Unterschied:

Beim Polygon-Problem sind als Positionen der Ecken für die Polygone reelle Zahlen erlaubt. Deshalb ist jede optimale Lösung des Polygon-Problems eine obere Schranke für die Lösung des Sicherheitsabstand-Problems.

Die Transformation des Polygon-Problems auf das Sicherheitsabstand-Problem und umgekehrt wurde bereits in meiner Diplomarbeit [Gen99] behandelt. Die Transformationen sehen wie folgt aus:

- **Die Transformation des Polygon-Problems auf das Sicherheitsabstand-Problem**

  Seien P_i reguläre Polygone mit m_i Ecken $(i = 1, \ldots, n)$ und sei A die Kreislänge. Sei $T = kgV(m_i | i \in \{1, \ldots, n\})$.

  Wir definieren den Takt der Linie l_i wie folgt: $T_i = \frac{T}{m_i}$ $(i = 1, \ldots, n)$. Es gibt nur eine Station und alle Linien fahren durch diese. Da wir nur eine Station haben, gibt es keine Verbindungen, d.h. $C = \emptyset$. Wenn wir keine Verbindungen haben, hat man auch keine Fahrzeitenfunktion t der Verbindungen. Man kann auch ein Streckennetzwerk

konstruieren, in dem alle Linien durch die gleiche Station fahren, wobei die Fahrzeit zwischen den Stationen beliebig gewählt werden kann. Dieses Streckennetzwerk hätte dieselben Eigenschaften bzgl. des Sicherheitsabstands wie das Streckennetzwerk mit einer Station.

Da die Lösungen des Sicherheitsabstand-Problems nur Linienabfahrtszeiten aus $\mathbb{N}_0$ erlauben, stellt jede optimale Lösung des Polygon-Problems t^* eine obere Schranke für die optimale Lösung des Sicherheitsabstand-Problems δ^* dar. Für den optimalen Abstand gilt die Beziehung:

$$\delta^* \leq \frac{T}{A} t^*$$

Da der optimale Abstand des Sicherheitsabstand-Problems δ^* aus $\mathbb{N}_0$ ist, gilt sogar:

$$\delta^* \leq \left\lfloor \frac{T}{A} t^* \right\rfloor$$

Eine zulässige Lösung des Polygon-Problems ist nicht unbedingt eine zulässige Lösung für das Sicherheitsabstand-Problem, da die Ecken des Polygons nicht notwendig auf ganzzahligen Werten im Kreis liegen.

- **Die Transformation des Sicherheitsabstand-Problems auf das Polygon-Problem**
  Seien l_i Linien mit Takten T_i $(i = 1, \ldots, n)$ und sei $A = kgV(T_i | i \in \{1, \ldots, n\})$. Das Streckennetzwerk bestehe nur aus einer Station, d.h. es gibt keine Verbindungen C und deshalb auch keine Fahrzeitenfunktion t. Streckennetzwerke mit mehreren Stationen können auch betrachtet werden, dabei müssen alle Linien den gleichen Weg nehmen.

  Wir definieren die Anzahl der Ecken der regulären Polygone P_i wie folgt: $m_i = \frac{A}{T_i}$ $(i = 1, \ldots, n)$. Hier gilt dann die folgende Beziehung zwischen den beiden optimalen Lösungen:

$$\delta^* \leq \lfloor t^* \rfloor$$

Für zwei Polygone hat Burkard ([Bur86]) eine optimale Lösung angeben können.

Satz 4.2.4 *Seien P_1 und P_2 zwei reguläre Polygone mit der Eckenanzahl m_1 bzw. m_2. Eine optimale Lösung von*

$$\min_{a \in [\mathbb{R}/A]^2} \quad \sum_{k=1}^{m} u_k^p$$

für beliebiges p mit $-\infty \leq p \leq 0$ oder $1 \leq p \leq \infty$, in einem Kreis der Länge A wird für $t = (0, t^*)$ mit $t^* = \frac{A}{2kgV(m_1, m_2)}$ angenommen und der optimale Wert ist t^*.

Aus diesem Satz ergibt sich durch Transformation und einige Umformungen die folgende obere Schranke für den Sicherheitsabstand zwischen zwei Linien:

Satz 4.2.5 *Sei s eine Station, und seien l_1, l_2 zwei Linien mit Takt T_1, T_2, die durch die Station s fahren. Dann gilt für jeden Fahrplan $\lambda \in \Lambda$:*

$$\delta(s, l_1, l_2, \lambda) \leq \left\lfloor \frac{ggt(T_1, T_2)}{2} \right\rfloor$$

Die obere Schranke wird auch angenommen, wenn der Fahrplan λ so gewählt wird, dass für die Ankunftszeiten der beiden Linien an der Station s gilt: $a(s, l_1, \lambda) \bmod T_1 = 0$, $a(s, l_2, \lambda) \bmod T_2 = \left\lfloor \frac{ggt(T_1,T_2)}{2} \right\rfloor$.

Mit Satz 4.2.5 erhält man leicht eine obere Schranke für den Sicherheitsabstand an einer Station.

Korollar 4.2.6 *Sei s eine Station und $L(s)$ die Menge der Linien, die durch die Station s fahren. Falls $|L(s)| \geq 2$ ist, so gilt für jeden Fahrplan $\lambda \in \Lambda$:*

$$\delta(s, \lambda) \leq \left\lfloor \min_{l_i, l_j \in L(s), l_i \neq l_j} \frac{ggT(T_i, T_j)}{2} \right\rfloor$$

Vince formuliert ein Problem (Scheduling Periodic Events), das äquivalent zu dem Polygon-Problem ist ([Vin89]).

Definition 4.2.7 (Scheduling Periodic Events)
Gegeben n periodische Ereignisse $E_1, \ldots, E_n$ mit Perioden $m_1, \ldots, m_n$. Sei $0 \leq x_i < m_i$ der Zeitpunkt des Eintretens von Ereignis E_i.

$$M(m_1, \ldots, m_n) := \max_{x \in I\!R^n} \quad \min_{1 \leq i < j \leq n} d_{ij}(x_i, x_j)$$

wobei $d_{ij}(x_i, x_j)$ den kleinsten Abstand zwischen den Ereignissen E_i und E_j bezeichnet.

Vince gibt in seiner Arbeit [Vin89] auch für drei Ereignisse (Linien, Polygone) eine optimale Lösung. Er zeigt auch die NP-Vollständigkeit des zugehörigen Entscheidungsproblems, in dem er das Entscheidungsproblem auf das Graphfärbungsproblem zurückführt.

Satz 4.2.8 *Seien E_1, E_2 und E_3 drei periodische Ereignisse mit Perioden m_1, m_2 und m_3. Sei $m_i' = \frac{m_i}{ggT(m_1, m_2, m_3)}$, $i = 1, 2, 3$. Dann gilt:*

- *1. Fall:* $ggT(m_1', m_2') = ggT(m_1', m_3') = ggT(m_2', m_3') = 1$
  $M(m_1, m_2, m_3) = \frac{1}{3} ggT(m_1, m_2, m_3)$ *und*

- *2. Fall: sonst:*

$$M(m_1, m_2, m_3) = \frac{1}{2} \min_{1 \leq i < j \leq 3} ggT(m_i, m_j)$$

Aus diesem Satz ergibt sich eine obere Schranke für den Sicherheitsabstand.

Korollar 4.2.9 *Sei s eine Station und $L(s)$ die Menge der Linien, die durch die Station s fahren. Falls $|L(s)| \geq 3$ ist, so gilt für jeden Fahrplan $\lambda \in \Lambda$:*

$$\delta(s,\lambda) \leq \left\lfloor \min_{l_i,l_j,l_k \in L(s), l_i \neq l_j, l_j \neq l_k, l_i \neq l_k} \delta(i,j,k) \right\rfloor$$

Wobei $\delta(i,j,k)$ wie folgt definiert ist: Sei $T'_t = \frac{T_t}{ggT(T_i,T_j,T_k)}$, $t = i,j,k$. Dann gilt:

- *1. Fall: $ggT(T'_i,T'_j) = ggT(T'_i,T'_k) = ggT(T'_j,T'_k) = 1$*
  $\delta(i,j,k) = \frac{1}{3} ggT(T_i,T_j,T_k)$ und

- *2. Fall: sonst:*

$$\delta(i,j,k) = \frac{1}{2} \min\{ggT(T_i,T_j), ggT(T_i,T_k), ggT(T_j,T_k)\}$$

Bemerkung 4.2.10 Korollar 4.2.9 lässt sich vereinfachen: Falls zwei Linien l_i und l_j aus $L(s)$ existieren mit $ggT(T_i,T_j) = 1$, so gilt für jeden Fahrplan λ: $\delta(s,\lambda) = 0$. Wenn für jedes Linienpaar l_i und l_j aus $L(s)$ $ggT(T_i,T_j) > 1$ ist, so folgt Korollar 4.2.6.

Die Sätze 4.2.11 und 4.2.12, die in meiner Diplomarbeit [Gen99] behandelt worden sind, werden in Satz 4.2.13 verallgemeinert.

Satz 4.2.11 $\forall \lambda \in \Lambda \; \forall s \in S : \delta(s,\lambda) \leq \left\lfloor \frac{1}{\sum_{i \in L(s)} \frac{1}{T_i}} \right\rfloor$

Beweis: Sei $s \in S$ und $\lambda \in \Lambda$ ein beliebiger Fahrplan.
Wir betrachten ein Zeitintervall der Länge $kgV(T(L(s)))$. In diesem Zeitintervall kommt jede Linie $l \in L(s)$ genau $\frac{kgV(T(L(s)))}{T_l}$-mal an, d.h. im Zeitintervall der Länge $kgV(T(L(s)))$ kommen genau $\sum_{i \in L(s)} \frac{kgV(T(L(s)))}{T_i}$ viele Linien an der Station s an. Wenn die Ankunftszeiten aller Linien gleichmäßig auf dem Zeitintervall der Länge $kgV(T(L(s)))$ verteilt werden, so kann der Sicherheitsabstand an der Station s höchstens $\frac{kgV(T(L(s)))}{\sum_{i \in L(s)} \frac{kgV(T(L(s)))}{T_i}}$ sein.

$$\delta(s,\lambda) \leq \frac{kgV(T(L(s)))}{\sum_{i \in L(s)} \frac{kgV(T(L(s)))}{T_i}} = \frac{1}{\sum_{i \in L(s)} \frac{1}{T_i}}$$

Da der Sicherheitsabstand ganzzahlig ist, gilt:

$$\delta(s,\lambda) \leq \left\lfloor \frac{1}{\sum_{i \in L(s)} \frac{1}{T_i}} \right\rfloor$$

□

Satz 4.2.12 *Sei s eine Station und $L(s)$ die Menge der Linien, die durch die Station s fahren. Sei m_i die Anzahl der Linien, die den Takt T_i besitzen und durch die Station s fahren. Falls $0 < m_i < |L(s)|$ so gilt:*

$$\forall \lambda \in \Lambda : \delta(s, \lambda) \leq \left\lfloor \frac{T_i}{m_i + 1} \right\rfloor$$

Wenn alle Linien denselben Takt haben, d.h. $m_i = |L(s)|$, so gilt:

$$\forall \lambda \in \Lambda : \delta(s, \lambda) \leq \left\lfloor \frac{T_i}{m_i} \right\rfloor$$

Beweis: Sei $T_i \in T(L(s))$. Sei m_i die Anzahl der Linien, die den Takt T_i besitzen und durch die Station s fahren.

In jedem Zeitintervall der Länge T_i kommen genau m_i Linien mit dem Takt T_i an. Da $m_i < |L(s)|$ ist, gibt es eine Linie l_j, die nicht den Takt T_i hat. In einem Zeitintervall der Länge $kgV(T_i, T_j)$ kommt die Linie l_j mit Takt T_j mindestens einmal vor. Teilen wir das Zeitintervall der Länge $kgV(T_i, T_j)$ in Zeitintervalle der Länge T_i, so wissen wir, dass in einem dieser Zeitintervalle der Länge T_i die Linie l_j ankommt. Also gibt es ein Zeitintervall der Länge T_i, in dem mindestens $m_i + 1$ Linien ankommen. Für einen beliebigen Fahrplan λ, ergibt sich dadurch eine obere Schranke für den Sicherheitsabstand von $\left\lfloor \frac{T_i}{m_i+1} \right\rfloor$. □

Der folgende Satz stellt eine Verallgemeinerung von Satz 4.2.11 und Satz 4.2.12 dar.

Satz 4.2.13 *Sei s eine Station und $L(s)$ die Menge der Linien, die durch die Station s fahren. Sei m_i die Anzahl der Linien, die den Takt T_i besitzen und durch die Station s fahren. Dann gilt für jeden Fahrplan $\lambda \in \Lambda$:*

$$\delta(s, \lambda) \leq \left\lfloor \min_{T_i \in T(L(s))} \frac{T_i}{m_i + \left\lceil \sum_{T_j \in T(L(s)), j \neq i} m_j \frac{T_i}{T_j} \right\rceil} \right\rfloor = \left\lfloor \min_{T_i \in T(L(s))} \frac{T_i}{\left\lceil \sum_{j \in L(s)} \frac{T_i}{T_j} \right\rceil} \right\rfloor$$

Beweis: Sei $T_i \in T(L(s))$ und sei $\lambda \in \Lambda$ ein beliebiger Fahrplan. Wenn man alle Zeitintervalle der Länge T_i betrachtet, so kommen im Durchschnitt $\sum_{j \in L(s)} \frac{T_i}{T_j}$ viele Linien im Zeitintervall der Länge T_i an. Da die Anzahl der Ankunftszeiten ganzzahlig ist, gibt es auch Zeitintervalle der Länge T_i, in denen $\left\lceil \sum_{j \in L(s)} \frac{T_i}{T_j} \right\rceil$ viele Linien ankommen. Wenn man diese Ankunftszeiten gleichmäßig auf das Intervall verteilt, so erhält man höchstens einen Sicherheitsabstand von $\left\lfloor \frac{T_i}{\left\lceil \sum_{j \in L(s)} \frac{T_i}{T_j} \right\rceil} \right\rfloor$. □

Bemerkung 4.2.14 Im Folgenden wollen wir kurz zeigen, dass die obere Schranke aus Satz 4.2.13 stärker ist als die in den Sätzen 4.2.11 und 4.2.12.

Zuerst zur Schranke in Satz 4.2.11.

$$\left\lfloor \frac{T_i}{\left\lceil \sum_{j \in L(s)} \frac{T_i}{T_j} \right\rceil} \right\rfloor \leq \left\lfloor \frac{T_i}{\sum_{j \in L(s)} \frac{T_i}{T_j}} \right\rfloor = \left\lfloor \frac{1}{\sum_{j \in L(s)} \frac{1}{T_j}} \right\rfloor$$

In Satz 4.2.12 ist $0 < m_i < |L(s)|$, also:

$$\left\lceil \sum_{j \neq i} m_j \frac{T_i}{T_j} \right\rceil \geq 1$$

und daraus folgt:

$$\left\lfloor \frac{T_i}{m_i + \left\lceil \sum_{j \neq i} m_j \frac{T_i}{T_j} \right\rceil} \right\rfloor \leq \left\lfloor \frac{T_i}{m_i + 1} \right\rfloor$$

Bemerkung 4.2.15 Die obere Schranke 4.2.13 ist nicht immer besser als 4.2.5, hier ein Beispiel.
Sei s eine Station, $|L(s)| = 2$ und $T_1 = 10$, $T_2 = 15$. Der Satz 4.2.5 liefert als obere Schranke 2, da $\left\lfloor \frac{ggT(10,15)}{2} \right\rfloor = 2$ ist, der Satz 4.2.13 hingegen 5, da $\left\lfloor \frac{10}{1+\left\lceil \frac{10}{15} \right\rceil} \right\rfloor = 5$ und $\left\lfloor \frac{15}{1+\left\lceil \frac{15}{10} \right\rceil} \right\rfloor = 5$ ist.

Beispiel 4.2.16 Sei s eine Station, durch die drei Linien mit Takten 5, 10 und 15 fahren. Die Sätze 4.2.5 und 4.2.13 liefern eine obere Schranke von 2, aber es ist nur ein Sicherheitsabstand von 1 erreichbar, siehe Abbildung 4.1.

Sei s eine Station, durch die drei Linien mit Takten 10, 20 und 30 fahren. Die Sätze 4.2.5 und 4.2.13 liefern eine obere Schranke von 5, aber es ist nur ein Sicherheitsabstand von 3 erreichbar.

Diese Beispiele geben Anlass zu einer neuen oberen Schranke.

Satz 4.2.17 *Sei s eine Station mit $L(s) \geq 3$. Seien T_1, T_2 und T_3 die Takte dreier Linien durch die Station s. Wenn $ggT(T_1, T_2) \leq T_3$ ist, so gilt:*

$$\delta(s, \lambda) \leq \left\lfloor \frac{T_3}{3} \right\rfloor$$

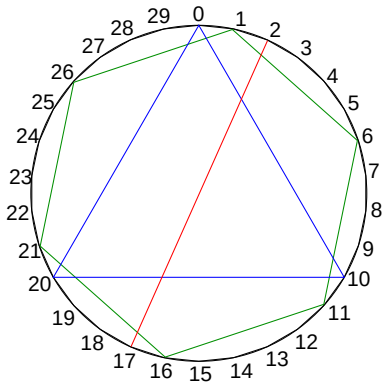


Abbildung 4.1: Beispiel, bei dem die obere Schranke nicht angenommen wird.

Beweis: Seien T_1, T_2 und T_3 die Takte dreier Linien l_1, l_2 und l_3 durch die Station s. Sei $\lambda \in \Lambda$ beliebig. Seien a_1, a_2 und a_3 die Ankunftszeiten der drei Linien an der Station s für den Fahrplan λ ($a_i := a(s, l_i, \lambda)$, $1 \leq i \leq 3$). Nach Satz 4.2.5 gilt:

$$\delta(s, l_1, l_2, \lambda) \leq \left\lfloor \frac{ggT(T_1, T_2)}{2} \right\rfloor$$

Daraus folgt:

$$\exists r_0, s_0 \in \mathbb{N}_0 : |a_1 + r_0 T_1 - a_2 - s_0 T_2| \leq \left\lfloor \frac{ggT(T_1, T_2)}{2} \right\rfloor \quad (*)$$

Ohne Einschränkung können wir annehmen, dass $a_1 + r_0 T_1 \leq a_2 + s_0 T_2$ ist. Sei $t_0 \in \mathbb{Z}$ so gewählt, dass gilt:

$$a_3 + t_0 T_3 \leq a_1 + r_0 T_1 \leq a_3 + (t_0 + 1) T_3$$

Wegen $ggT(T_1, T_2) \leq T_3$ gilt:

$$\text{(I)} \quad a_3 + t_0 T_3 \leq a_2 + s_0 T_2 \leq a_3 + (t_0 + 1) T_3$$

oder

$$\text{(II)} \quad a_3 + (t_0 + 1) T_3 \leq a_2 + s_0 T_2 \leq a_3 + (t_0 + 2) T_3$$

Im ersten Fall (I) hätten wir drei Linien, die in einem Zeitintervall von T_3 fahren. Daraus ergibt sich der Satz. Aus dem zweiten Fall (II) ergibt sich:

$$a_1 + r_0 T_1 \leq a_3 + (t_0 + 1) T_3 \leq a_2 + s_0 T_2$$

und daraus folgt, dass

$$\delta(s, \lambda) \leq \left\lfloor \frac{\left\lfloor \frac{ggT(T_1, T_2)}{2} \right\rfloor}{2} \right\rfloor \leq \left\lfloor \frac{T_3}{3} \right\rfloor .$$

Da die beste Möglichkeit, die Ankunftszeit zwischen den zwei Ankunftszeiten $a_1+r_0T_1$, $a_2+s_0T_2$ zu platzieren, die Mitte ist, und wegen (*) lässt sich höchstens einen Sicherheitsabstand von $\left\lfloor \frac{\lfloor \frac{ggT(T_1,T_2)}{2} \rfloor}{2} \right\rfloor$ erreichen. □

Bemerkung 4.2.18 Satz 4.2.17 gilt auch, wenn $\frac{3}{4} \cdot ggT(T_1, T_2) \leq T_3$ gilt.

Aus der Arbeit von Burkard [Bur86] folgt folgende obere Schranke:

Satz 4.2.19 *Sei s eine Station und $L(s)$ die Menge der Linien, die durch die Station s fahren. Sei m_i die Anzahl der Linien, die den Takt T_i besitzen und durch die Station s fahren. Seien $T_1 \neq T_2$ die Takte zweier Linien aus $T(L(s))$. Seien n_1, n_2, d, m und q wie folgt definiert: $n_1 := \frac{kgV(T_1,T_2)}{T_1}$, $n_2 := \frac{kgV(T_1,T_2)}{T_2}$, $d = ggT(n_1, n_2)$, $m = kgV(n_1, n_2)$ und $q = \left\lceil \frac{m_1 d}{n_2} \right\rceil + \left\lceil \frac{m_2 d}{n_1} \right\rceil$. Dann gilt für den Sicherheitsabstand an der Station s:*

$$\delta(s, \lambda) \leq \left\lfloor \frac{kgV(T_1, T_2)}{q \cdot m} \right\rfloor$$

Es stellt sich die Frage, ob der Satz 4.2.19 gelegentlich bessere Schranken liefert als die Sätze 4.2.5, 4.2.13 und 4.2.17. Das folgende Beispiel zeigt, dass die obere Schranke für den Sicherheitsabstand in Satz 4.2.19 bessere Werte liefern kann als die Schranken aus anderen Sätzen.

Beispiel 4.2.20 Sei s eine Station, durch die fünf Linien mit Takten 20, 30, 30, 30, 30 fahren. Die Sätze 4.2.5, 4.2.13 und 4.2.17 liefern eine obere Schranke von 5, aber es ist nur eine Sicherheitsabstand von 3 erreichbar. Satz 4.2.19 liefert als obere Schranke 3. $T_1 = 20, T_2 = 30$, $m_1 = 1$, $m_2 = 4$, $n_1 = 3$, $n_2 = 2$, $d = 1$, $m = 6$, $q = \left\lceil \frac{1 \cdot 1}{2} \right\rceil + \left\lceil \frac{4 \cdot 1}{3} \right\rceil = 3$. Daraus folgt die obere Schranke $\left\lfloor \frac{60}{3 \cdot 6} \right\rfloor = 3$.

4.2.2 Güte der oberen Schranken

Um zu untersuchen, wie gut die oben genannten oberen Schranken sind, wurden für typische Takte von Linien im ÖPNV (also für die Takte 5, 10, 15, 20, 30, 60), obere Schranken berechnet und später mit den optimalen Lösung verglichen. Im Falle, dass eine obere Schranke nicht mit der optimalen Lösung übereinstimmt, wurde diese gespeichert. Der folgende Algorithmus zeigt genauer, wie der Test durchgeführt wurde.

Dabei wurden nur 15 Fälle gefunden, bei denen die obere Schranke nicht mit der optimalen Lösung übereinstimme, siehe Tabelle 4.1.

Wie man an der Tabelle 4.1 sieht, ist der Unterschied zwischen dem optimalen Sicherheitsabstand und der oberen Schranke höchstens eins.

Algorithmus 3: Güte der oberen Schranken

for $n \leftarrow 3$ ***to*** 6 **do**
  foreach $T \in \{5, 10, 15, 20, 30, 60\}^n$ **do**
    Berechne obere Schranke $\overline{\delta}$ für den Sicherheitsabstand;
    Erzeuge ein Streckennetzwerk mit n Linien, die durch dieselben Stationen fahren und T als Taktvektor haben;
    Berechne die optimale Lösung δ^*;
    if $\overline{\delta} > \delta^*$ **then**
      $Q := Q \cup \{T\}$;
foreach $a, b \in Q$ *und* $a \neq b$ **do**
  if $\forall x \in \{5, 10, 15, 20, 30, 60\}$ *gilt :* $|a|_x \leq |b|_x$ *und* $\delta_a^* = \delta_b^*$ *und* $\overline{\delta}_a = \overline{\delta}_b$ **then**
    // wobei $|a|_x$ die Anzahl der Linien mit Takt x in a ist;
    Eliminiere a aus Q;

Anzahl der Linien	optimale Lösung	obere Schr.	Takte
4	2	3	10 10 20 30
4	1	2	10 10 15 20
4	1	2	5 10 20 30
5	1	2	5 20 20 20 30
5	2	3	10 20 20 20 30
5	1	2	10 15 20 20 20
6	3	4	20 20 30 30 30 60
6	1	2	5 20 30 30 30 30
6	0	1	5 5 5 5 15 20
6	1	2	10 10 10 10 20 30
6	1	2	10 10 15 15 15 60
6	1	2	10 10 15 15 15 30
6	2	3	10 20 30 30 30 30
6	1	2	10 15 15 15 20 30
6	0	1	5 5 5 5 10 15

Tabelle 4.1: 15 Fälle, in denen obere Schranke und optimaler Sicherheitsabstand nicht übereinstimmen

4.2.3 Obere Schranken für den Sicherheitsabstand über alle Stationen

Wir möchten für Teilfahrpläne, d.h. für Fahrpläne, bei denen nur für Teil der Linien die Abfahrtszeiten festgelegt sind, obere Schranken für Sicherheitsabstände im gesamten Streckennetzwerk berechnen. Alle Fahrpläne, die aus diesen Teilfahrplänen erzeugt werden können, können keinen besseren Sicherheitsabstand haben als eine obere Schranke für den jeweiligen Teilfahrplan.

Obere Schranke auf Basis bereits getroffener Entscheidungen

Der Branch-and-Bound-Algorithmus baut Fahrpläne gemäß der Best-First-Strategie auf, indem sukzessiv Abfahrtszeiten noch nicht behandelter Linien fixiert werden. Die jeweiligen Teilfahrpläne liefern kanonisch obere Schranken für den Sicherheitsabstand.

Der Sicherheitsabstand für einen Fahrplan λ, bei dem alle Linien fest sind, ist wie folgt definiert:

$\delta(\lambda) = \min_{s \in S} \delta(s, \lambda)$ mit $\delta(s, \lambda) = \min_{l_i, l_j \in L(s), l_i \neq l_j} \delta(s, l_i, l_j, \lambda)$

Wenn man die Menge L durch eine Teilmenge L_0 ersetzen würde, so bekommt man eine obere Schranke.

$$\forall L_0 \subset L: \ \delta(s, \lambda) \leq \min_{l_i, l_j \in L(s) \cap L_0, l_i \neq l_j} \delta(s, l_i, l_j, \lambda)$$

Durch Benutzung der oberen Schranken für die einzelnen Stationen $\overline{\delta(s)}$ ergibt sich:

$$\forall L_0 \subset L: \ \delta(s, \lambda) \leq \min\{\overline{\delta(s)}, \min_{l_i, l_j \in L(s) \cap L_0, l_i \neq l_j} \delta(s, l_i, l_j, \lambda)\}$$

Definition 4.2.21 Sei λ ein Teilfahrplan während eines Branch-and-Bound-Schritts. Mit $L(\lambda)$ bezeichnen wir die Menge der festen Linien im Teilfahrplan λ. Definiere eine obere Schrankenfunktion $\overline{\delta(\lambda)}$ für den Sicherheitsabstand wie folgt:

$$\overline{\delta(\lambda)} = \min_{s \in S} \overline{\delta(s, \lambda)}$$

wobei

$$\overline{\delta(s, \lambda)} = \min\{\overline{\delta(s)}, \min_{l_i, l_j \in L(s) \cap L(\lambda), l_i \neq l_j} \delta(s, l_i, l_j, \lambda)\}$$

Obere Schranke durch Vervollständigung von Teilfahrplänen

Eine Verbesserung der vorstehend genannten oberen Schranken auf Basis von Teilfahrplänen lässt durch geeignete Vervollständigung derselben erzielen.

Definition 4.2.22 Sei λ ein Teilfahrplan während eines Branch-and-Bound-Schritts. Sei $L_1 \subset L \setminus L(\lambda)$.
Definiere die Menge A aller Teilfahrpläne, die aus dem Teilfahrplan λ erzeugt werden können, wobei nur die Linien der Menge L_1 festgelegt werden.

$$A = \{\lambda_0 \in \Lambda | L(\lambda_0) = L(\lambda) \cup L_1,\ \forall l \in L(\lambda) : \lambda_{0l} = \lambda_l\}$$

Definiere eine obere Schrankenfunktion $\overline{\delta_2(\lambda)}$ für den Sicherheitsabstand wie folgt:

$$\overline{\delta_2(\lambda)} = \min_{\lambda_0 \in A} \overline{\delta(\lambda_0)}$$

Wie man sieht, muss man exponentiell viele obere Schranken bzgl. $|L_1|$ berechnen, um eine obere Schranke zu erhalten. Daher kann man nur eine sehr kleine Linienmenge L_1 auswählen. Diese Auswahl ist auch besonders wichtig, da sie die obere Schranken stark beeinflusst.

Bemerkung 4.2.23 Auf genau dieselbe Weise wie bei der Berechnung der oberen Schranke für den Sicherheitsabstand kann man eine obere Schranke für die Summe der Sicherheitsabstände berechnen. Sei λ ein Teilfahrplan während eines Branch-and-Bound-Schritts. Durch getroffene Entscheidungen erhalten wir die folgende obere Schrankenfunktion $\overline{\delta_\Sigma(\lambda)}$ für die Summe der Sicherheitsabstände wie in der Definition 4.2.21: $\overline{\delta_\Sigma(\lambda)} = \sum_{s \in S} \overline{\delta(s, \lambda)}$ mit $\overline{\delta(s, \lambda)} = \min\{\overline{\delta(s)}, \min_{l_i, l_j \in L(s) \cap L(\lambda), l_i \neq l_j} \delta(s, l_i, l_j, \lambda)\}$.

Durch Vervollständigung der Fahrpläne erhalten wir als obere Schrankenfunktion $\overline{\delta_\Sigma(\lambda)}$ für die Summe der Sicherheitsabstände wie in der Definition 4.2.22: $\overline{\delta_\Sigma(\lambda)} = \sum_{s \in S} \overline{\delta_2(s, \lambda)}$ mit $\overline{\delta_2(s, \lambda)} = \min_{\lambda_0 \in A} \delta(s, \lambda_0)$.

4.2.4 Obere Schranken für den Sicherheitsabstand zweier Linien mit verschiedenen Linienwegen zwischen zwei gemeinsamen Stationen

Im Folgenden wird untersucht, wie sich der Sicherheitsabstand ändert, wenn sich zwei Linien an einer gemeinsamen Station trennen und an einer anderen Stationen wieder zusammentreffen, wobei beide verschiedene Wege benutzen wie z.B. in Abbildung 4.2. In dem Streckennetzwerk der Kölner Verkehrs-Betriebe tritt dies zwischen den Linien 15 und 16, und 6 und 12 auf.

Satz 4.2.24 *Seien zwei Linien l_1 und l_2 wie in Abbildung 4.2 gegeben. Sei der Takt beider Linien gleich T. Seien $F(s_1, s_2, l_i)$ $(i = 1, 2)$ die Fahrzeiten der Linien von Station s_1 zu s_2. Für alle Fahrpläne $\lambda \in \Lambda$ gilt:*

$$\delta(\lambda) \leq \left\lfloor \frac{\max\{T - c, c\}}{2} \right\rfloor$$

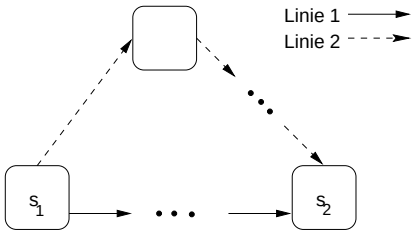


Abbildung 4.2: Ein Streckennetzwerk mit 2 Linien.

mit $c = (F(s_1, s_2, l_1) - F(s_1, s_2, l_2)) \bmod T$. *D.h.* $\left\lfloor \frac{\max\{T-c,c\}}{2} \right\rfloor$ *ist eine obere Schranke für den Sicherheitsabstand. Diese wird auch angenommen, wenn man als Fahrplan* $\lambda = (\lambda_1, 0)$ *wählt, wobei* λ_1 *folgende Bedingung erfüllt:* $\lambda_1 + (\lambda_1 + c) \bmod T = T$ *oder* $T + 1$ *wählt.*

Beweis: Sei Fahrplan $\lambda = (\lambda_1, \lambda_2) \in \Lambda$ beliebig. Ohne Einschränkung der Allgemeinheit kann $\lambda_2 = 0$ und $0 \leq \lambda_1 \leq T - 1$ gewählt werden, da der Suchraum um eine Dimension verringern werden kann (siehe Abschnitt 3.4). Dann gilt für die Sicherheitsabstände an den beiden Stationen s_1 und s_2 nach Satz 3.1.2:

$$\delta(s_1, \lambda) = \min\{|\lambda_1 - \lambda_2|, T - |\lambda_1 - \lambda_2|\} = \min\{\lambda_1, T - \lambda_1\}$$

und

$$\delta(s_2, \lambda) = \min\{A, T - A\}$$

mit $A = |\lambda_1 + F(s_1, s_2, l_1) - F(s_1, s_2, l_2)| \bmod T$. Daraus ergibt sich für den Sicherheitsabstand im ganzen Streckennetzwerk:

$$\delta(\lambda) = \min\{\delta(s_1, \lambda), \delta(s_2, \lambda)\} = \min\{\lambda_1, T - \lambda_1, A, T - A\}$$

Sei $c = (F(s_1, s_2, l_1) - F(s_1, s_2, l_2)) \bmod T$.
Für zwei ganze Zahlen a, b gilt: $(a + b) \bmod T = (a + b \bmod T) \bmod T$ $(*)$
Aus $(*)$ folgt, dass $A = (\lambda_1 + c) \bmod T$ oder $A = T - (\lambda_1 + c) \bmod T$ gilt und daraus ergibt sich für den Sicherheitsabstand:

$$\delta(\lambda) = \min\{\lambda_1, T - \lambda_1, (\lambda_1 + c) \bmod T, T - (\lambda_1 + c) \bmod T\}$$

Im Idealfall lässt sich λ_1 so wählen, dass der Sicherheitsabstand an beiden Stationen sich um höchstens um eins unterscheidet. Damit $\delta(s_1, \lambda) = \delta(s_2, \lambda)$ oder $\delta(s_2, \lambda) + 1$ gilt, muss $\lambda_1 + (\lambda_1 + c) \bmod T = T$ oder $T + 1$ sein.

Hier wird nur der Fall betrachtet, dass die Sicherheitsabstände an den beiden Stationen gleich sind. Der andere Fall funktioniert analog.
O.b.d.A. gelte $\lambda_1 + (\lambda_1 + c) \bmod T = T$ $(**)$.

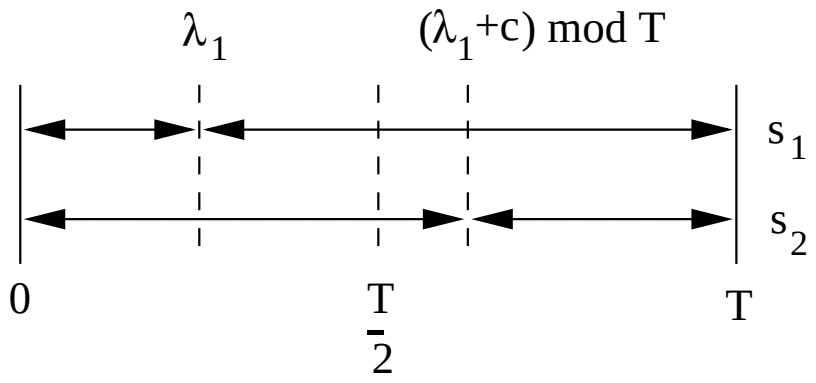


Abbildung 4.3: Sicherheitsabstände an den beiden Stationen.

1. Fall: $\lambda_1 + c = (\lambda_1 + c) \bmod T$
   Aus (**) folgt dann: $\lambda_1 = \frac{T-c}{2}$. Da $\lambda_1 \in \mathbb{N}_0$ ist, gilt dann: $\lambda_1 = \lfloor \frac{T-c}{2} \rfloor$ oder $\lambda_1 = \lceil \frac{T-c}{2} \rceil$
   Daraus ergibt sich für den Sicherheitsabstand: $\delta(\lambda) = \lfloor \frac{T-c}{2} \rfloor$.

2. Fall: $\lambda_1 + c \neq (\lambda_1 + c) \bmod T$
   Dann gilt :$\lambda_1 + c - T = (\lambda_1 + c) \bmod T$. Aus (**) folgt dann: $\lambda_1 = T - \frac{c}{2}$. Da $\lambda_1 \in \mathbb{N}_0$ ist, gilt dann: $\lambda_1 = \lfloor T - \frac{c}{2} \rfloor$ oder $\lambda_1 = \lceil T - \frac{c}{2} \rceil$.
   Daraus ergibt sich für den Sicherheitsabstand: $\delta(\lambda) = \lfloor \frac{c}{2} \rfloor$.

Aus beiden Fällen zusammen ergibt sich folgende obere Schranke für den Sicherheitsabstand:

$$\delta(\lambda) \leq \left\lfloor \frac{\max\{T-c, c\}}{2} \right\rfloor$$

□

4.3 Branch-and-Bound-Algorithmus

Am Anfang wird eine Startlösung für das Branch-and-Bound-Verfahren mit Hilfe eines Backtracking-Verfahrens bestimmt. Dieses Backtracking-Verfahren wurde in Rahmen meiner Diplomarbeit [Gen99] für das Sicherheitsabstand-Problem entwickelt. Der Suchbaum Q, der als Prioritätsschlange verwaltet wird, wird mit dem leeren Fahrplan, d.h. mit einem Fahrplan, bei dem für keine Linie eine Abfahrtszeit festgelegt worden ist, initialisiert. In jedem Branch-and-Bound-Schritt wird ein Teilfahrplan λ aus dem Suchbaum Q ausgewählt. Diese Auswahl erfolgt durch die Quasiordnung $\leq_\Lambda$ in Definition 2.4.1 auf den Fahrplänen, wobei der Sicherheitsabstand und die Summe der Sicherheitsabstände durch obere Schranken ersetzt werden. Für den Teilfahrplan λ werden eine obere Schranke für den Sicherheitsabstand $\overline{\delta(\lambda)}$ und eine obere Schranke für die Summe der Sicherheitsabstände

$\overline{\delta_\Sigma(\lambda)}$ berechnet. Aufgrund von Speicherplatzproblemen bei reiner Anwendung des Branch-and-Bound-Verfahren wird in regelmäßigen Abständen von dem ausgewählten Fahrplan ein Backtracking-Verfahren für das Fahrplan-Problem für wenige Sekunden gestartet, um bessere Lösungen zu erhalten.

Der im Rahmen dieser Arbeit implementierte Algorithmus hat folgenden Aufbau:

Algorithmus 4: Branch-and-Bound-Algorithmus

```
Bestimme mit Backtracking eine (suboptimale) Lösung λ_H des Sicherheitsabstand-
Problems;
Setze λ_best := λ_H und δ := δ(λ_H);
Setze λ := (−1, −1, ..., −1)^T und Q := {λ};
// λ ist die Wurzel des Suchbaumes Q. Noch ist keine Linie gesetzt;
while (Q ≠ ∅ und Zeitlimit nicht überschritten) do
    Wähle bzgl. der Quasiordnung ≤_Λ in Definition 2.4.1 den besten Fahrplan λ aus
    der Menge Q aus;
    Berechne obere Schranke für den Sicherheitsabstand δ(λ);
    if (δ(λ) < δ) then
        Eliminiere λ aus Q;
    else
        // d.h. δ(λ) ≥ δ;
        if ((δ(λ) > δ) ∨ (δ_Σ(λ) > δ_Σ(λ_best))) then
            Wähle eine Linie l im Fahrplan λ aus, die noch freie Variablen ist;
            foreach t ∈ {0, 1, ..., T_l − 1} do
                λ[l] := t;
                if ((δ(λ) > δ) ∨ (δ(λ) = δ ∧ δ_Σ(λ) > δ_Σ(λ_best))) then
                    if (alle Variablen λ_i fixiert) then
                        // bessere Lösung gefunden ;
                        Setze λ_best := λ und δ := δ(λ);
                    else
                        Füge Q den Fahrplan λ hinzu;
```

4.4 Heuristiken

In diesem Abschnitt werden verschiedene Heuristiken vorgestellt, um gute Anfangslösungen für das Branch-and-Bound-Verfahren zu erhalten. Als erste Heuristik wird ein einfacher probabilistischer Algorithmus verwendet, der nur Mutationen erlaubt. Als zweites wählen wir die Greedy-Heuristik. Die letzten beiden Heuristiken verwenden das Branch-

and-Bound-Verfahren eingeschränkt auf einige Linien. Dabei unterscheiden sie sich in der Auswahl der freien Linien.

4.4.1 Probabilistischer Algorithmus

Bei diesem Verfahren wird ein Fahrplan als "Gene" eines Fahrplans aufgefasst. Die Population besteht aus nur einem Individuum, auf das natürlich nur Mutationen angewandt werden können. Ausgehend von einem gegebenen oder zufällig erzeugten Fahrplan wird per Zufall eine Linie ausgewählt und die Abfahrtszeit um einen zufälligen Wert von +1, +2, -1, -2 Minuten verändert ("mutiert"). Falls sich die Bewertung des Fahrplans nicht verschlechtert, wird der neue Fahrplan weiterverfolgt, sonst der alte. Daraus wird zufällig eine Linie ausgewählt und die Abfahrtszeit mutiert etc.. Die Optimierung wird abgebrochen, wenn sich bei $k \cdot m$ aufeinander folgenden Schritten ("Mutationen") keine echte Verbesserung der Bewertung des Fahrplans ergibt. Dabei ist m die Anzahl der zu optimierenden Linien und k ein frei wählbarer Parameter, der in diesem Kontext mit $k = 100$ gewählt wurde.

Genetische Algorithmen werden im Bereich der Fahrplanoptimierung zur Minimierung der Gesamtwartezeiten beim Umsteigen eingesetzt ([Rie99],[KM00],[NV95]).

Algorithmus 5: Probabilistischer Algorithmus

```
while (Zeitlimit nicht überschritten) do
    Wähle zufälligen Fahrplan: λ;
    AnzS := 0; // AnzS bezeichnet die Anzahl der Schritte ohne Verbesserung;
    while (AnzS < 100 * |L|) do
        Wähle zufällig eine Linie aus und ändere deren Abfahrtszeit um einen zufällig
        gewählten Wert von +1, +2, -1, -2 Minuten;
        if (Fahrplan besser) then
            AnzS:=0;
        else
            Mache Änderung rückgängig;
            AnzS++;
```

4.4.2 Greedy-Algorithmus

Ein Fahrplan wird zufällig erzeugt. Dann wird dieser an einer Position geändert, dies geschieht für jede Linie und jede Abfahrtszeit. Die beste Verbesserung wird dann weiterverfolgt, bis entweder keine Verbesserung mehr erzielt wird oder eine Zeitlimit überschritten ist. Falls die Zeit noch nicht abgelaufen ist, so wird dies wiederholt.

In jedem while-Schritt wird lediglich die Abfahrtszeit einer Linie geändert, nämlich der Linie, die die beste Verbesserung bzgl. unserer Zielfunktion liefert.

Algorithmus 6: Greedy-Algorithmus

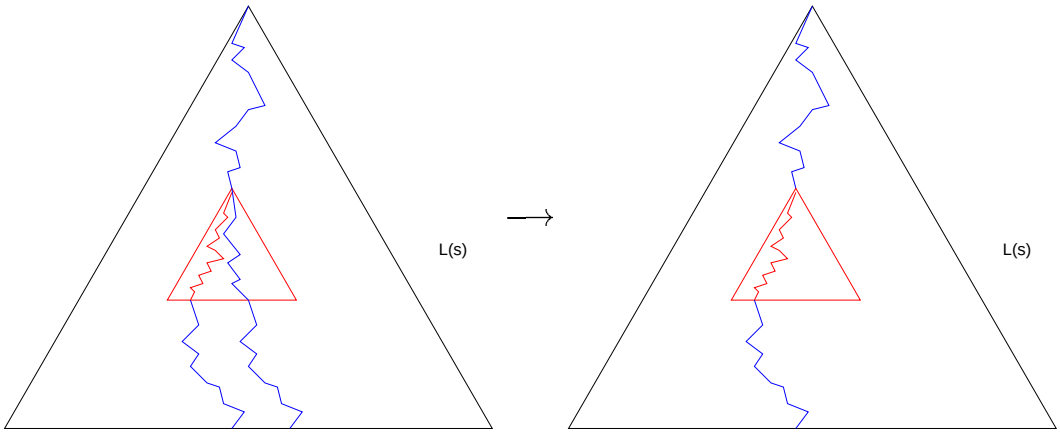
$\lambda_{greedy} := \vec{0}$;
while *(Zeitlimit nicht überschritten)* **do**
  Wähle zufälligen Fahrplan: λ;
  $\lambda_{best} := \lambda$;
  repeat
    foreach $l \in L$ **do**
      $\lambda_2 := \lambda$;
      foreach $t \in \{0, 1, \ldots, T_l - 1\}$ **do**
        $\lambda_2[l] := t$;
        if *(Fahrplan λ_2 besser als λ_{best})* **then**
          $\lambda_{best} := \lambda_2$;
    $\lambda := \lambda_{best}$;
  until *(keine Verbesserung von Fahrplan λ erzielt worden)*;
  if *(Fahrplan λ_{best} besser als λ_{greedy})* **then**
    $\lambda_{greedy} := \lambda_{best}$;

4.4.3 Lokale Anwendung des Branch-and-Bound-Algorithmus

Dieser Algorithmus versucht durch lokales Einsetzen des Branch-and-Bound Algorithmus eine bessere Lösung zu finden. Ausgehend von einem zufällig erzeugten Fahrplan wird zufällig eine Station ausgewählt. Dann werden die Linien, die durch die ausgewählte Station fahren, als veränderbare Variablen im Branch-and-Bound-Algorithmus festgelegt, d.h. während der Branch-and-Bound-Optimierung werden nur die Abfahrtszeiten der Linien geändert, die durch diese Station fahren. Der Branch-and-Bound-Algorithmus wird nur für eine Sekunden gestartet, die beste Lösung wird übernommen. Es wird wieder eine Station zufällig gewählt usw..

Algorithmus 7: Lokale Anwendung des Branch-and-Bound-Algorithmus

Wähle zufälligen Fahrplan: λ;
while *(Zeitlimit nicht überschritten)* **do**
  Wähle zufällig eine Station $s \in S_\Sigma$ aus;
  Starte Branch-and-Bound-Algorithmus mit freien Linien $L(s)$ für eine Sekunde: λ_{neu};
  if *(Fahrplan λ_{neu} besser als λ)* **then**
    $\lambda := \lambda_{neu}$;

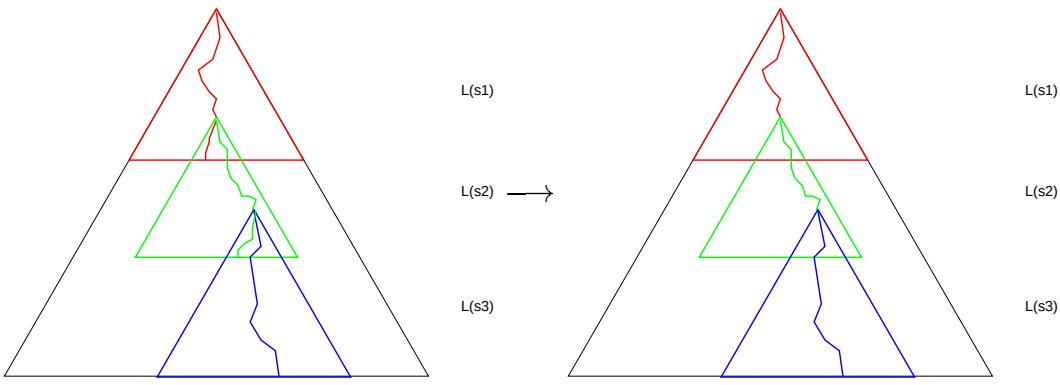


4.4.4 Langsamer Aufbau des Fahrplans

Dieses Verfahren erzeugt schrittweise den Fahrplan, indem eine Folge von Stationen durchlaufen wird. Am Anfang sind im Fahrplan alle Abfahrtszeiten nicht festgelegt. In jedem Schritt wählt man zufällig eine Station. Mit dem aktuellen Fahrplan startet man den Branch-and-Bound-Algorithmus für k Sekunden, wobei die Linien, die durch die ausgewählte Station fahren, als veränderbare Variablen im Branch-and-Bound-Algorithmus festgelegt sind. Dies wird solange durchgeführt bis alle Linien eine Abfahrtszeit haben.

Algorithmus 8: Langsamer Aufbau des Fahrplans

$L_0 := \emptyset$ // Menge der fixierten Linien ;
Setze $\lambda := (-1, -1, \ldots, -1)^T$; //Noch ist keine Linie gesetzt;
while $(L_0 \neq L)$ **do**
  Wähle zufällig eine Station $s \in S_\Sigma$ aus;
  Starte Branch-and-Bound-Algorithmus mit freien Linien $L(s)$ auf Streckennetzwerk mit Linienmenge L_0 für k Sekunden;
  $L_0 := L_0 \cup L(s)$;



4.5 Zerlegung des Streckennetzwerks

In diesem Abschnitt wird untersucht, wie sich der Branch-and-Bound-Algorithmus beschleunigen lässt, wenn sich das Streckennetzwerk in signifikant kleinere Teilnetze zerlegen lässt.

Dazu wird der Linienkonflikt-Graph (LK-Graph) betrachtet. Wenn der LK-Graph in zweifach zusammenhängende Teile zerfällt, so lässt sich aus den optimalen Fahrplänen der Teilnetze ein optimaler Fahrplan für das gesamte Streckennetzwerk erstellen, wobei die Teilnetze aus den zweifachen Zusammenhangskomponenten des LK-Graphen entstehen.

4.5.1 Zerlegung mit Hilfe der zweifachen Zusammenhangskomponenten des LK-Graph

Wenn der LK-Graph nicht zusammenhängend ist, so bilden die Linien einer Zusammenhangskomponente ein Teilnetz, das von den anderen Teilnetzen unabhängig ist, weil sie keine gemeinsamen Linien besitzen. Darum kann man die Teilnetze auch getrennt optimieren. Der optimale Fahrplan des gesamten Streckennetzwerks lässt sich aus den optimalen Fahrplänen der Teilnetze zusammensetzen.

Da der LK-Graph meistens zusammenhängend ist, werden im Weiteren die zweifachen Zusammenhangskomponenten untersucht. Wir werden sehen, dass auch aus den optimalen Fahrplänen der Teilnetze, die aus den zweifachen Zusammenhangskomponenten (2-Zhgskomp) des LK-Graph entstehen, ein optimaler Fahrplan für das Gesamtnetz in linearer Zeit zusammengesetzt werden kann.

Für jede zweifache Zusammenhangskomponente wird für deren zugehöriges Teilnetz ein optimaler Fahrplan erstellt. Die zweifachen Zusammenhangskomponenten bilden einen Wald, wenn man jede zweifache Zusammenhangskomponente als einen Knoten auffasst. Zwischen je zwei Knoten existiert eine Kante, wenn die beiden Knoten eine Linie gemeinsam haben. Diesen Wald bezeichnen wir als *Superstrukturgraph* des LK-Graphen. Im Superstrukturgraphen wird eine Tiefensuche durchgeführt (Breitensuche ist auch möglich). Die Knoten (2-Zhgskomp.=Teilnetze) werden bzgl. dieser Tiefensuchereihenfolge durchlaufen und aus den Fahrplänen der Teilnetze berechnet man den optimalen Fahrplan für das gesamte Netz. Dabei wird zwischen zwei Arten von Knoten unterschieden. Die erste Art von Knoten sind solche, die zum ersten Mal in einer Zusammenhangskomponente des Superstrukturgraphen besucht worden sind, d.h. sie sind Wurzel einer Zusammenhangskomponente des Superstrukturgraphen. Die zweite Art von Knoten bildet den Rest, d.h. diese Knoten haben einen Vorgängerknoten, mit dem sie eine Linie gemeinsam haben.

Wenn die Knoten also bzgl. der Tiefensuchereihenfolge durchlaufen werden und der Knoten von der ersten Art ist, so kann man dessen Fahrplan ohne Veränderung in den Gesamtfahrplan übernehmen. Ansonsten wird derjenige Vorgängerknoten gesucht, mit dem der aktuelle Knoten eine gemeinsame Linie hat. Für diese gemeinsame Linie wurde im

Vorgängerknoten eine Abfahrtszeit schon festgesetzt. Diese Abfahrtszeit wird beibehalten. Darum werden die Abfahrtszeiten im optimalen Fahrplan des Teilnetzes, den man in den Fahrplan des gesamten Streckennetzwerks einfügen will, so verschoben, dass die Abfahrtszeiten des Teilnetzes und mit den Abfahrtszeiten des Fahrplans für das gesamte Streckennetzwerk auf der gemeinsamen Linie übereinstimmen.

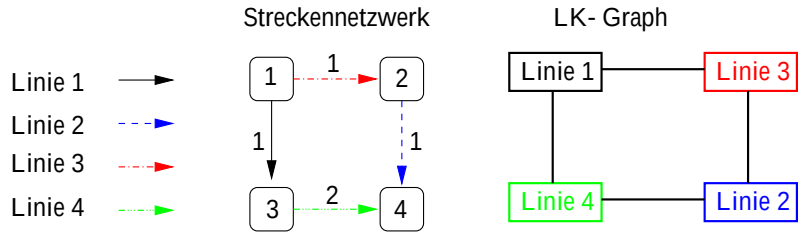
Der Algorithmus, der dieses berechnet, sieht wie folgt aus:

Algorithmus 9: Optimaler Fahrplan aus den Teilnetzen

```
Gegeben: LK-Graph;
Berechne die zweifachen Zusammenhangskomponenten des LK-Graphen (2-ZKen);
Seien z_1, ..., z_k die 2-ZKen;
foreach 2-ZK z_i do
    Berechne optimalen Fahrplan λ^i für das Teilnetz von 2-ZK z_i;
Bilde aus den 2-ZKen einen Superstrukturgraphen;
Führe im Superstrukturgraphen eine Tiefensuche durch;
Seien p_1, p_2, ..., p_k die Reihenfolge der 2-ZKen bzgl. der Tiefensuche;
Setze λ := (−1, −1, ..., −1); // Fahrplan für das Gesamtnetz;
for i ← 1 to k do
    if (Vater(p_i)=p_i) then
        //d.h. p_i ist Wurzel einer 2-ZKe;
        foreach (l ∈ L(p_i)) do
            λ_l := λ^i_l;
    else
        Sei l_0 := L(p_i) ∩ L(Vater(p_i));
        // d.h. l_0 ist die gemeinsame Linie von Knoten p_i und Vater(p_i);
        foreach (l ∈ L(p_i)) do
            λ_l := (λ^i_l + λ_{l_0} − λ^i_{l_0} + T_{l_0}) mod T_l;
```

Jedes Teilnetz behält seinen optimalen Fahrplan im Gesamtnetz, da sich durch die Verschiebung des Fahrplans nach Satz 3.4.1 nichts an den Eigenschaften des Fahrplans ändert und jede Linie höchstens einmal verändert wird, da im Superstrukturgraphen keine Kreise existieren.

Bei dreifachen Zusammenhangskomponenten kann man diese nicht getrennt optimieren. Dazu ein einfaches Beispiel mit 4 Linien und 4 Stationen.



Der LK-Graph im Bespiel hat vier dreifache Zusammenhangskomponenten $\{l_1, l_3\}$, $\{l_1, l_4\}$, $\{l_2, l_3\}$ und $\{l_2, l_4\}$. Bei einem festen Linientakt von 10 für alle Linien würde für jede dreifache Zusammenhangskomponente der optimale Sicherheitsabstand 5 betragen, aber für das gesamten Streckennetzwerk ist der optimale Sicherheitsabstand 4.

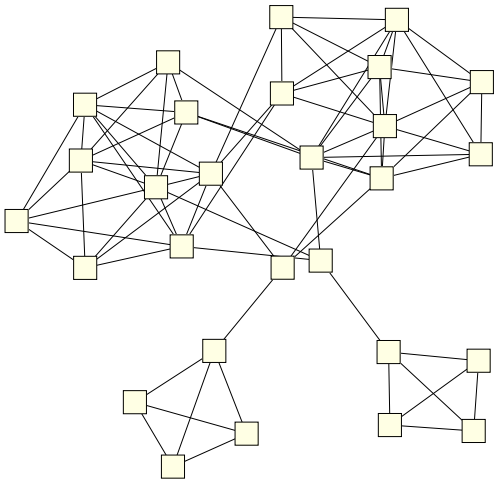


Abbildung 4.4: Der Linienkonfliktgraph für den KVB-Testinstanz I_{12} mit 28 Linien und 543 Stationen. (28 Knoten, 76 Kanten)

4.5.2 Unterteilung des LK-Graphen durch Festlegen der Abfahrtszeiten einiger Linien

Viele Streckennetzwerke im ÖPNV haben eine sehr große zweifache Zusammenhangskomponente, darum bringt der obige Ansatz meist keine große Verbesserung.

Wenn wir die Abfahrtszeiten einiger Linien festlegen, so können die zugehörigen Linienknoten im Linienkonfliktgraph entfernt werden. Dadurch kann bei geeigneter Wahl der Linien das Streckennetzwerke in viele Teile bzgl. der zweifache Zusammenhangskomponenten des LK-Graphen zerteilt werden.

Hier sehen wir zwei mögliche Ansätze:
Sei der LK-Graph $G = (L, E)$ für ein Streckennetzwerk gegeben.

1. Ansatz: Bestimme $S_0 \subset S$ für die gilt:
   (a) Graph $\tilde{G}$ hat nur zweifache Zusammenhangskomponenten der Größe $O(log|L|)$, wobei $\tilde{G} = (\tilde{L}, \tilde{E})$ mit $\tilde{L} = L \setminus L(S_0)$ und $\tilde{E} = E|_{\tilde{L}}$.
   (b) $|L(S_0)|$ ist minimal
2. Ansatz: Bestimme $L_0 \subset L$ für die gilt:
   (a) Graph $\tilde{G}$ hat nur zweifache Zusammenhangskomponentender Größe $O(log|L|)$, wobei $\tilde{G} = (\tilde{L}, \tilde{E})$ mit $\tilde{L} = L \setminus L_0$ und $\tilde{E} = E|_{\tilde{L}}$.
   (b) $|L_0|$ ist minimal

Für jeden Teilfahrplan der ausgewählten Linien ist eine neue Berechnung der optimalen Fahrpläne für die Teilnetze notwendig, da die optimalen Fahrpläne der Teilnetze vom Teilfahrplan der fixierten Linien abhängen. Der Vorteil des ersten Ansatzes ist, dass viele Teilfahrpläne nicht betrachtet werden müssen, da die Linien gemeinsame Stationen befahren und dadurch einige mögliche Fahrzeiten nicht sinnvoll sind. Unter Umständen kann die Menge der ausgewählten Linien $L(S_0)$ viel größer sein als mit dem zweiten Ansatz. Im zweiten Ansatz kann es vorkommen, dass Linien ausgewählt werden, die keine gemeinsamen Stationen haben und dadurch kann man keine Abfahrtszeiten ausschliessen.

Bemerkung 4.5.1 Sei $L_0 \subset L$ so gewählt, dass die zweifachen Zusammenhangskomponenten des Graphen $\tilde{G}$ die Größe $O(log|L|)$ haben, wobei $\tilde{G} = (\tilde{L}, \tilde{E})$ mit $\tilde{L} = L \setminus L_0$ und $\tilde{E} = E|_{\tilde{L}}$. Der Algorithmus 9 muss im worst-case $\prod_{l \in L_0} T_l$-mal aufgerufen werden, da es $\prod_{l \in L_0} T_l$-viele verschiedene Teilfahrpläne für die Linien L_0 gibt. In Abschnitt 3.4 wurde gezeigt, dass die Dimension des Suchbaums um eins verringert werden kann, indem die Abfahrszeit einer beliebigen Linie auf 0 gesetzt wird. Wenn wir eine Linie aus L_0 wählen, so kann die Anzahl der Teilfahrpläne verringert werden.

Kapitel 5

Darstellung des Fahrplan-Problems als ganzzahliges lineares Programm

In diesem Kapitel wollen wir das Fahrplan-Problem als ganzzahliges lineares Programm (ILP) darstellen, um es mittels CPLEX, einem Programmpaket zur Lösung linearer und quadratischer Optimierungsprobleme mit kontinuierlichen und gemischt ganzzahligen Variablen, zu lösen.

5.1 Lineare und ganzzahlige lineare Programme

Probleme der Form

$$\begin{aligned} \max \quad & c^T x \\ Ax &\leq b \\ x &\in \mathbb{R}^n \end{aligned}$$

nennt man Lineare Programmierungs-Probleme, kurz Lineare Programme (linear program, LP). Die Menge $\{x \in \mathbb{R}^n | Ax \leq b\}$ nennt man Lösungsraum, wobei A eine $m \times n$ Matrix ist, $c \in \mathbb{R}^n$ ein n-dimensionaler und $b \in \mathbb{R}^m$ ein m-dimensionaler Vektor ist. x ist der Vektor der Variablen oder Unbekannten. Die Funktion $k : \mathbb{R}^n \to \mathbb{R}$ mit $k(x) = c^T x$ heißt Kostenfunktion.

Wenn einige Variablen nur ganzzahlige Werte annehmen können, nennt man das Problem gemischt ganzzahliges Lineares Programm (mixed integer linear programm, MIP). Dieses lässt sich wie folgt formulieren:

$$\begin{aligned} \max \quad & c^T x + h^T y \\ Ax + By &\leq b \\ x &\in \mathbb{R}^n \\ y &\in \mathbb{Z}^n \end{aligned}$$

Wenn alle Variablen ganzzahlig sein müssen, so bezeichnet man es als ganzzahliges lineares Programm (integer linear program, ILP). Für Literatur zur Theorie von linearen Programmen und Lösungsmethoden verweisen wir auf [Sch98], [Chv83] und [NW88].

5.2 ILP-Darstellung des Sicherheitsabstand-Problems

Sei ein Streckennetzwerk $N(S, C, t, L, T)$ gegeben.

Im Folgenden wollen wir zuerst das Sicherheitsabstand-Problem als ILP darstellen. Ausgehend von der Darstellung des Problems werden die Definitionen erweitert und umformuliert, bis wir eine ILP-Darstellung erhalten. Aus den Definitionen erhalten wir folgende Programm-Darstellung für das Sicherheitsabstand-Problem:

$$\begin{gathered} \max \quad \delta(\lambda) \\ \delta(s,\lambda) = \min_{l_i,l_j \in L(s)} \delta(s, l_i, l_j, \lambda) \ \forall s \in S \\ \delta(\lambda) = \min_{s \in S} \delta(s, \lambda) \\ \lambda \in \Lambda \end{gathered}$$

Der Sicherheitsabstand zweier Linien an einer Station lässt sich berechnen, indem die Ankunftszeiten der beiden Linien im Zeitintervall von 0 bis zum kleinsten gemeinsamen Vielfachen aller Takte betrachtet wird.

Definiere folgende Konstanten für die ILP-Darstellung des Sicherheitsabstand-Problems.

- $T_{kgV} = kgV\{T_1, T_2, \ldots, T_m\}$
- $T_{min} = \min\{T_l | l \in L\}$, $T_{max} = \max\{T_l | l \in L\}$
- $m_i = \frac{T_{kgV}}{T_i} + 1$, $1 \leq i \leq |L|$
  m_i gibt die Anzahl der Ankunftszeiten der Linie l_i im Intervall T_{kgV} an, die für die Berechnung des Sicherheitsabstands benötigt wird.
- $\overline{\delta}$: obere Schranke für den Sicherheitsabstand
- $\overline{\delta_\Sigma}$: obere Schranke für die Summe der Sicherheitsabstände

- $\overline{\delta}_s$: obere Schranke für den Sicherheitsabstand an der Station $s \in S$
- $\overline{\delta}_m$: $m = (s, l_i, l_j)$ obere Schranke für den Sicherheitsabstand an der Station s zwischen den Linien l_i und l_j

Die Konstanten für die oberen Schranken werden bestimmt mit Hilfe der Sätze aus Kapitel 4.2.

$$\max \quad \delta \tag{5.2.1}$$

$$\begin{aligned}
a_{ij} &= \lambda_i + j \cdot T_i && \forall i \in L \quad \forall j \in \mathbb{N}_{m_i} \\
a^s_{ij} &= (a_{ij} + a(s, l_i, \vec{0})) \bmod (T_{kgV} + T_i) && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
\delta &\leq |a^s_{ij} - a^s_{kl}| && \forall s \in S \quad \forall i,k \in L(s), i \neq k \\
& && \quad \forall j \in \mathbb{N}_{m_i}, l \in \mathbb{N}_{m_k} \\
0 &\leq a^s_{ij} \leq T_{kgV} + T_i - 1 && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta \leq \overline{\delta} \\
a^s_{ij}, \lambda_i, \delta &\quad integer
\end{aligned}$$

Bedeutung der einzelnen Variablen:

- δ : Sicherheitsabstand
- λ_i : Abfahrtszeit der Linie i
- a_{ij} : Abfahrtszeiten der Linie i für m_i aufeinander folgende Bahnen $(0 \leq j \leq m_i - 1)$
- a^s_{ij} : Ankunftszeiten der Linie i an der Station s für m_i Bahnen im Zeitintervall $[0, T_{kgV} + T_i - 1]$.

In der Darstellung stören die Betrag- und Modulo-Funktion. Wir wollen die Gleichungen bzw. Ungleichungen, die Betrag- bzw. Modulo-Funktionen enthalten, im Folgenden durch äquivalente lineare Gleichungen bzw. Ungleichungen ersetzen.

- Modulo-Funktion: Seien $a, b \in \mathbb{Z}$ und $K \in \mathbb{N}$.

$$\begin{aligned}
x &= (a + b) \bmod K \\
\Leftrightarrow \exists z \in \mathbb{Z} : \; x - (a + b) &= z \cdot K, \; 0 \leq x \leq K - 1
\end{aligned}$$

Wie man sieht, muss man, um die Modulo-Funktion aus einer Gleichung zu eliminieren, eine weitere ganzzahlige Variable einführen.

- Betrag-Funktion: Seien $x, y, \delta \in \mathbb{Z}$ und $\overline{\delta}, T \in \mathbb{N}$.

$$\begin{cases} \delta & \leq |x-y| \\ 0 & \leq x, y \leq T-1 \\ 0 & \leq \delta \leq \overline{\delta} \end{cases}$$

$$\Leftrightarrow \begin{cases} \delta & \leq x - y + H(1-\alpha) \\ \delta & \leq -x + y + H\alpha \\ 0 & \leq x, y \leq T-1 \\ 0 & \leq \delta \leq \overline{\delta} \\ \alpha & \in \{0,1\} \end{cases}$$

Wobei H eine Konstante ist, für die gilt:

$$\forall\, 0 \leq x, y \leq T-1\ \forall\, 0 \leq \delta \leq \overline{\delta} : |x-y| + \delta \leq H$$

Darum reicht es $H = T + \overline{\delta}$ zu wählen.
Falls $\alpha = 0$ gewählt wird, so bedeutet dies, dass y größer als x ist. Die Konstante H garantiert dann, dass die andere Gleichung erfüllt wird. Bei $\alpha = 1$ gilt es umgekehrt.[NW88]

Durch Eliminierung der Modulo-und Betragfunktionen aus den (Un-)Gleichungen im ILP 5.2.1 ergibt sich die folgende ILP-Darstellung für das Sicherheitsabstand-Problem:

$$\max \quad \delta \tag{5.2.2}$$

$$\begin{aligned}
z_{ij}^s \cdot (T_{kgV} + T_i) &= a_{ij}^s - (\lambda_i + j \cdot T_i + a(s, l_i, 0)) && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
\delta &\leq a_{ij}^s - a_{kl}^s + H(1 - \alpha_{ij,kl}) && \forall s \in S \quad \forall i,k \in L(s), i \neq k \\
& && \forall j \in \mathbb{N}_{m_i}, l \in \mathbb{N}_{m_k} \\
\delta &\leq -a_{ij}^s + a_{kl}^s + H\alpha_{ij,kl} \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq a_{ij}^s \leq T_{kgV} + T_i - 1 && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
0 &\leq \alpha_{ij,kl} \leq 1 && \forall s \in S \quad \forall i,k \in L(s), i \neq k \\
& && \forall j \in \mathbb{N}_{m_i}, l \in \mathbb{N}_{m_k} \\
\alpha_{ij,kl}, a_{ij}^s, z_{ij}^s, \lambda_i, \delta &\quad integer
\end{aligned}$$

Wobei $H = T_{kgV} + T_{max} + T_{min}$ gewählt werden kann.

5.3 ILP-Darstellung des Fahrplan-Problems

Das Sicherheitsabstand-Problem hatte als Zielfunktion den Sicherheitabstand. Beim Fahrplan-Problem wurde im Branch-and-Bound-Ansatz mit einer Ordnung gearbeitet, aber wir brauchen für die ILP-Darstellung eine Zielfunktion.

Definition 5.3.1 (Zielfunktion)
Gegeben sei ein Streckennetzwerk $N(S, C, t, L, T)$. Sei $\overline{\delta_\Sigma} \in \mathbb{N}$ eine obere Schranke für die Summe der Sicherheitsabstände. Definiere die folgende Zielfunktion auf der Menge der Fahrpläne Λ:

$$\mu(\lambda) := \delta(\lambda) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda)$$

mit $\lambda \in \Lambda$.

Definition 5.3.2 (Fahrplan-Problem2)
Gegeben sei ein Streckennetzwerk $N(S, C, t, L, T)$. Bestimme ein $\lambda_0 \in \Lambda$ mit $\mu(\lambda_0) = \mu^*$, wobei μ^* wie folgt definiert ist:

$$\mu^* := \max_{\lambda \in \Lambda} \mu(\lambda) \tag{5.3.3}$$

Satz 5.3.3 *Jede optimale Lösung des Fahrplan-Problems ist auch eine optimale Lösung des Fahrplan-Problem2 und umgekehrt, d.h. für alle $\lambda_0 \in \Lambda$ gilt:*

$$\mu(\lambda_0) = \mu^* \Leftrightarrow \delta_\Sigma(\lambda_0) = \delta_\Sigma^*$$

Beweis:

- "$\Leftarrow$": Seien $\lambda_0 \in \Lambda$ und $\delta_\Sigma(\lambda_0) = \delta_\Sigma^*$. (*)
  Daraus folgt: $\delta(\lambda_0) = \delta^*$.
  Angenommen es existiert ein $\lambda_1 \in \Lambda : \mu(\lambda_1) > \mu(\lambda_0)$. (**)

  1. Fall: $\delta(\lambda_1) = \delta(\lambda_0) = \delta^*$

$$\begin{aligned} \mu(\lambda_1) &> \mu(\lambda_0) \\ \Leftrightarrow \delta(\lambda_1) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) &> \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_0) \\ \Leftrightarrow \delta_\Sigma(\lambda_1) &> \delta_\Sigma(\lambda_0) \end{aligned}$$

  Dies ist ein Widerspruch zur Optimalität von λ_0 (*).

2. Fall: $\delta(\lambda_1) < \delta(\lambda_0) = \delta^*$
Daraus folgt: $\delta(\lambda_1) - \delta(\lambda_0) + 1 \leq 0$, da $\delta(\lambda) \in \mathbb{N}$.

$$\begin{aligned}
\mu(\lambda_1) &= \delta(\lambda_1) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) \\
&= \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) + (\delta(\lambda_1) - \delta(\lambda_0)) \cdot \overline{\delta_\Sigma} \\
&< \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \overline{\delta_\Sigma} + (\delta(\lambda_1) - \delta(\lambda_0)) \cdot \overline{\delta_\Sigma} \\
&= \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + (\delta(\lambda_1) - \delta(\lambda_0) + 1) \cdot \overline{\delta_\Sigma} \\
&\leq \delta(\lambda_0) \cdot \overline{\delta_\Sigma} \\
&\leq \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_0) = \mu(\lambda_0)
\end{aligned}$$

Das ist ein Widerspruch zu Annahme (**).

- "$\Rightarrow$": Seien $\lambda_0 \in \Lambda$ und $\mu(\lambda_0) = \mu^*$.(***)
Zu zeigen $\delta(\lambda_0) = \delta^*$ und $\delta_\Sigma(\lambda_0) = \delta^*_\Sigma$.
Beweis von $\delta(\lambda_0) = \delta^*$ funktioniert analog zum 2.Fall in "$\Leftarrow$".
Es ist noch zu zeigen, dass $\delta_\Sigma(\lambda_0) = \delta^*_\Sigma$ gilt.
Angenommen es gilt $\delta_\Sigma(\lambda_0) < \delta^*_\Sigma$, d.h. es existiert ein $\lambda_1 \in \Lambda : \delta(\lambda_1) = \delta^*$ und $\delta_\Sigma(\lambda_1) > \delta_\Sigma(\lambda_0)$.

$$\begin{aligned}
\mu(\lambda_1) &= \delta(\lambda_1) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) \\
&= \delta^* \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) \\
&= \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_1) \\
&> \delta(\lambda_0) \cdot \overline{\delta_\Sigma} + \delta_\Sigma(\lambda_0) = \mu(\lambda_0)
\end{aligned}$$

Dies ist aber ein Widerspruch zu Optimalität von λ_0 (***).

□

Das Fahrplan-Problem2 (siehe Def. 5.3.2), das äquivalent zum Fahrplan-Problem ist, lässt sich als ILP dann wie folgt darstellen:

$$\max \quad \delta \cdot \overline{\delta_\Sigma} + \sum_{s \in S} \delta_s \tag{5.3.4}$$

$$\begin{aligned}
z^s_{ij} \cdot (T_{kgV} + T_i) &= a^s_{ij} - (\lambda_i + j \cdot T_i + a(s, l_i, 0)) && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
\delta_s &\leq a^s_{ij} - a^s_{kl} + H(1 - \alpha_{ij,kl}) && \forall s \in S \quad \forall i,k \in L(s), i \neq k \\
& && \forall j \in \mathbb{N}_{m_i}, l \in \mathbb{N}_{m_k} \\
\delta_s &\leq -a^s_{ij} + a^s_{kl} + H\alpha_{ij,kl} \\
\delta &\leq \delta_s && \forall s \in S \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta_s \leq \overline{\delta_s} && \forall s \in S \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq a^s_{ij} \leq T_{kgV} + T_i - 1 && \forall s \in S \quad \forall i \in L(s) \quad \forall j \in \mathbb{N}_{m_i} \\
0 &\leq \alpha_{ij,kl} \leq 1 && \forall s \in S \quad \forall i,k \in L(s), i \neq k \\
& && \forall j \in \mathbb{N}_{m_i}, l \in \mathbb{N}_{m_k} \\
& \alpha_{ij,kl}, a^s_{ij}, z^s_{ij}, \lambda_i, \delta_s, \delta \quad \textit{integer}
\end{aligned}$$

Wenn man die Reduktion in Abschnitt 3.2.2 benutzt und S durch S_Σ ersetzt und $\sum_{s \in S} \delta_s$ in der Zielfunktion durch $\sum_{s \in S_\Sigma} |[s]_{R_\Sigma}| \cdot \delta_s$ ersetzt, so wird die Anzahl der Variablen und Ungleichungen reduziert. Die Tabelle 5.1 gibt die Größe der ILPs (5.3.4) für die drei Instanzen des Straßenbahnnetzes der KVB an. Genaueres zu den Instanzen ist im Abschnitt 6.2.1 angegeben. Wie man in Tabelle 5.1 sehen kann, werden die Anzahl der Variablen um 70% und die Anzahl der Ungleichungen um 60% verringert.

			ILP (5.3.4)		ILP (5.3.4) nach Reduktion	
Inst.	$\|L\|$	$\|S\|$	$\|Var.\|$	$\|Ungl.\|$	$\|Var.\|$	$\|Ungl.\|$
I_2	20	461	4526	4907	961	1829
I_{12}	28	543	18060	26159	5125	9115
I_{20}	26	525	10332	13488	2871	5075

Tabelle 5.1: Größe von ILPs, die gemäß ILP (5.3.4) und deren Reduktion für die KVB-Testinstanzen entstehen.

Die ILP-Darstellung (5.3.4) und deren reduzierte Darstellung des Fahrplan-Problems ist nicht geeignet für CPLEX. CPLEX liefert nach einer Stunde Rechenzeit für die drei Instanzen der Straßenbahnnetzes der KVB sehr schlechte Ergebnisse. Um eine andere ILP-Darstellung zu bekommen, wollen wir den Sicherheitsabstand zweier Linien anders berechnen. Mit Hilfe von Satz 3.1.2 wurde eine alternative Berechnungsmethode des Sicher-

heitsabstandes zwischen zwei Linien vorgestellt. Damit sieht die Darstellung des Fahrplan-Problems wie folgt aus:

$$\max \quad \delta \cdot \overline{\delta_\Sigma} + \sum_{s \in S} \delta_s \tag{5.3.5}$$

$$\begin{aligned}
a_i^s &= \lambda_i + a(s, l_i, 0) && \forall s \in S \quad \forall i \in L(s) \\
x_{ij}^s &= |a_i^s - a_j^s| \bmod ggT(T_i, T_j) && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
\delta_s &\leq \min\{x_{ij}^s, ggT(T_i, T_j) - x_{ij}^s\} && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
\delta &\leq \delta_s && \forall s \in S \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta_s \leq \overline{\delta_s} && \forall s \in S \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq x_{ij}^s \leq ggT(T_i, T_j) - 1 && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
x_{ij}^s, \lambda_i, \delta_s, \delta &\quad integer
\end{aligned}$$

wobei die Variablen folgende Bedeutung haben:

- δ_s : Sicherheitsabstand an der Station s
- a_i^s : Ankunftszeit der Linie i an der Station s mit Fahrzeit λ_i
- x_{ij}^s : Hilfsvariable, mit deren Hilfe der Sicherheitsabstand berechnet wird.

Im Folgenden werden äquivalente Schritte gezeigt, wie man den Betrag, die Modulo- und die Min-Funktion eliminieren kann.

I	II
$\max \quad d$	$\max \quad d$
$a = \lvert x - y \rvert \bmod D$	$a = \lvert x - y \rvert - z \cdot D$
$d \leq \min\{a, D - a\}$	$d \leq \min\{a, D - a\}$
	$0 \leq a \leq D - 1$
	$z \in \mathbb{Z}$

III	IV
$\max \quad d$	$\max \quad d$
$a = x - y - z \cdot D$	$a = x - y - z \cdot D$
$d \leq \min\{a, D - a\}$	$d \leq a$
	$d \leq D - a$
$0 \leq a \leq D - 1$	$0 \leq a \leq D - 1$
$z \in \mathbb{Z}$	$z \in \mathbb{Z}$

Von Schritt I zu II wird die Modulo-Funktion wie bekannt entfernt. Beim Übergang von Schritt II zu III kann der Betrag weggelassen werden, da durch die Einschränkung $0 \leq a \leq D-1$ sichergestellt wird, dass die Menge $\{a, D-a\}$ sich nicht verändert. Im letzten Schritt wird durch zwei Ungleichungen die Ungleichung $d \leq \min\{a, D-a\}$ ersetzt.

Durch Entfernen von Modulo und Betrag ergibt sich aus ILP 5.3.5 das ILP 5.3.6 :

$$\max \quad \delta \cdot \overline{\delta_\Sigma} + \sum_{s \in S} \delta_s \tag{5.3.6}$$

$$\begin{aligned}
a_i^s &= \lambda_i + a(s, l_i, 0) && \forall s \in S \quad \forall i \in L(s) \\
x_{ij}^s &= a_i^s - a_j^s - z_{ij}^s \cdot ggT(T_i, T_j) && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
\delta_s &\leq x_{ij}^s && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
\delta_s &\leq ggT(T_i, T_j) - x_{ij}^s && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
\delta &\leq \delta_s && \forall s \in S \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta_s \leq \overline{\delta_s} && \forall s \in S \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq x_{ij}^s \leq ggT(T_i, T_j) - 1 && \forall s \in S \quad \forall i, j \in L(s), i \neq j \\
x_{ij}^s, z_{ij}^s, \lambda_i, \delta_s, \delta &\quad integer
\end{aligned}$$

			ILP (5.3.6)	
Ins.	$\|L\|$	$\|S\|$	$\|Var.\|$	$\|Ungl.\|$
I_2	20	461	886	1067
I_{12}	28	543	1338	1692
I_{20}	26	525	1192	1485

Tabelle 5.2: Größe von ILPs, die gemäß ILP (5.3.6) für die KVB-Testinstanzen entstehen.

Wir können die Anzahl der Variablen und Gleichungen reduzieren, indem wir die Stationen eliminieren, durch die nur eine Linie fährt.

Bemerkung 5.3.4 Falls eine Station s nur von einer Linie l befahren wird, so ist der Sicherheitsabstand an dieser Station für jeden Fahrplan λ gleich dem Takt der Linie, d.h. $\delta(s, \lambda) = T_l$.

Definition 5.3.5 Sei $S_{>1} = \{s \in S | 1 < |L(s)|\}$, $S_{=1} = \{s \in S | 1 = |L(s)|\}$ und $C = \sum_{s \in S_{=1}} T_{L(s)}$.

Wenn wir im ILP 5.3.6 die Menge der Stationen S durch $S_{>1}$ ersetzen, so erhalten wir ILP 5.3.7:

$$\max \quad \delta \cdot \overline{\delta_\Sigma} + \sum_{s \in S_{>1}} \delta_s + C \tag{5.3.7}$$

$$\begin{aligned}
a_i^s &= \lambda_i + a(s, l_i, 0) && \forall m = (s, l_i, l_j) \in M \\
x_{ij}^s &= a_i^s - a_j^s - z_{ij}^s \cdot ggT(T_i, T_j) && \forall m = (s, l_i, l_j) \in M \\
\delta_s &\leq x_{ij}^s && \forall m = (s, l_i, l_j) \in M \\
\delta_s &\leq ggT(T_i, T_j) - x_{ij}^s && \forall m = (s, l_i, l_j) \in M \\
\delta &\leq \delta_s && \forall s \in S_{>1} \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta_s \leq \overline{\delta_s} && \forall s \in S_{>1} \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq x_{ij}^s \leq ggT(T_i, T_j) - 1 && \forall m = (s, l_i, l_j) \in M \\
x_{ij}^s, z_{ij}^s, \lambda_i, \delta_s, \delta &\quad integer
\end{aligned}$$

Sei $S_{\Sigma,>1} = S_{>1} \cap S_\Sigma$. Wir ersetzen M durch M_0, indem wir die Sätze 3.2.24 und 3.2.25 benutzen, und erhalten:

$$\max \quad \delta \cdot \overline{\delta_\Sigma} + \sum_{s \in S_{\Sigma,>1}} |[s]_{R_\Sigma}| \cdot \delta_s + C \tag{5.3.8}$$

$$\begin{aligned}
a_i^s &= \lambda_i + a(s, l_i, 0) && \forall m = (s, l_i, l_j) \in M_0 \\
x_{ij}^s &= a_i^s - a_j^s - z_{ij}^s \cdot ggT(T_i, T_j) && \forall m = (s, l_i, l_j) \in M_0 \\
\delta_m &\leq x_{ij}^s && \forall m = (s, l_i, l_j) \in M_0 \\
\delta_m &\leq ggT(T_i, T_j) - x_{ij}^s && \forall m = (s, l_i, l_j) \in M_0 \\
\delta_s &\leq \delta_m && \forall s \in S_{\Sigma,>1}\ \forall m \in M_0(s) \\
\delta &\leq \delta_s && \forall s \in S_{\Sigma,>1} \\
0 &\leq \lambda_i \leq T_i - 1 && \forall i \in L \\
0 &\leq \delta_s \leq \overline{\delta_s} && \forall s \in S_{\Sigma,>1} \\
0 &\leq \delta \leq \overline{\delta} \\
0 &\leq \delta_m \leq \left\lfloor \frac{ggT(T_i, T_j)}{2} \right\rfloor && \forall m = (s, l_i, l_j) \in M_0 \\
0 &\leq x_{ij}^s \leq ggT(T_i, T_j) - 1 && \forall m = (s, l_i, l_j) \in M_0 \\
& x_{ij}^s, z_{ij}^s, \lambda_i, \delta_m, \delta_s, \delta \quad integer
\end{aligned}$$

wobei δ_m den Sicherheitsabstand zwischen zwei Linien darstellt und nach Satz 4.2.5 wissen wir, dass für den Sicherheitsabstand zwischen zwei Linien gilt: $\delta_m \leq \left\lfloor \frac{ggT(T_i,T_j)}{2} \right\rfloor$.

			ILP (5.3.7)		ILP (5.3.8)	
Ins.	$\|L\|$	$\|S\|$	$\|Var.\|$	$\|Ungl.\|$	$\|Var.\|$	$\|Ungl.\|$
I_2	20	461	533	714	187	242
I_{12}	28	543	984	1338	311	416
I_{20}	26	525	837	1130	275	370

Tabelle 5.3: Größe von ILPs, die gemäß ILP (5.3.8) für die KVB-Testinstanzen entstehen.

Tabelle 5.3 zeigt, dass wir die Anzahl der Ungleichungen und Variablen für die ILP-Darstellung des Fahrplan-Problems um 75% verringert haben.

Kapitel 6

Experimentelle Ergebnisse

In diesen Kapitel wollen wir anhand verschiedener Streckennetzwerke unsere Branch-and-Bound-Lösungen mit den CPLEX-Lösungen vergleichen. Da wir mit dem Schienennetzwerk der KVB aber nur ein reales Streckennetzwerk haben, haben wir auch künstliche Streckennetzwerke erzeugt, um zu sehen, wie sich die beiden Ansätze verhalten und welcher besser geeignet ist, das Fahrplan-Problem zu lösen.

6.1 Software und Hardware

Wir haben das Programm unter dem Betriebssystem Sun Solaris 2.6 in der Programmiersprache C++ entwickelt. Als Compiler wurde der GNU C++ Compiler in Version gcc 2.95.2 eingesetzt. Für die Lösung des ganzzahligen linearen Programms wurde die kommerzielle Software CPLEX in der Version 6.6.1 eingesetzt, näheres zu CPLEX findet man in [ILO98].

Die Tests wurden auf einer SUN Workstation Ultra-4 mit vier Prozessoren mit je 296Mhz und einem Arbeitsspeicher von 1024 MB durchgeführt.

6.2 Instanzen

In Folgenden wollen wir uns näher mit den Instanzen befassen, an denen wir die Güte unserer Lösungsansätze messen wollen.

6.2.1 Kölner Verkehrs-Betriebe (KVB)

Von der Kölner Verkehrs-Betriebe AG (KVB) stammen unsere realen Daten. Die Daten für die Linien, Stationen, Verbindungen und Takte sind dem Fahrplan vom 30.5.2001 der KVB entnommen. Für jede Stunde an Werktagen haben wir aus diesen Daten eine

Streckennetzwerk-Instanz für unser Programm erzeugt. Dabei entstanden 10 verschiedene Streckennetzwerke, da einige Linien nur zu bestimmten Zeiten oder die Takte einiger Linien wechseln. Aus diesen 10 Instanzen haben wir drei repräsentative ausgewählt, da sich die anderen Instanzen wenig von diesen drei Instanzen unterscheiden.

Abbildung 6.1 zeigt das Straßenbahnnetz der KVB.

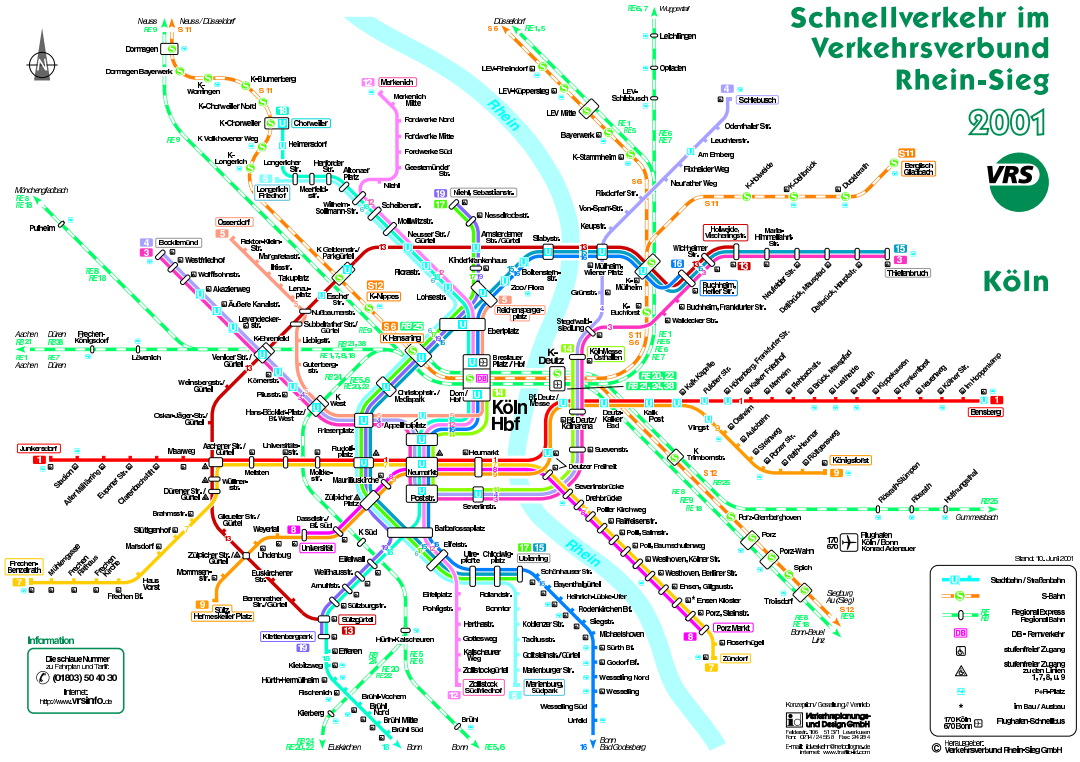


Abbildung 6.1: Straßenbahnnetz der Kölner Verkehrs-Betriebe (KVB).

Die Tabelle 6.1 gibt die Größe der Streckennetzwerke für drei verschiedene Instanzen an.

6.2.2 Gitter-Streckennetzwerke

Gitter-Streckennetzwerke sind Streckennetzwerke, die zufällig erzeugt werden. Als Grundstruktur für das Streckennetzwerk dient ein Stationsgitter mit Verbindungen wie in der Abbildung 6.2.

Auf einem Stationsgitter werden Linien zufällig erzeugt. Dazu wird für jede Linie eine

Instanzen	$\|L\|$	$\|S\|$	$\|C\|$
I_2	20	461	424
I_{12}	28	543	508
I_{20}	24	525	486

Tabelle 6.1: Größe der drei KVB-Testinstanzen

zufällige Startstation in der ersten Spalte des Gitters ausgewählt. Von den Stationen im Gitter dürfen sich die Linien nur in zwei Richtungen bewegen, entweder parallel zu den Zeilen von links nach rechts oder je nach Position der Stationen nach oben (gerade Spalten) oder nach unten (ungerade Spalten). Welchen Weg die Linien nehmen, wird zufällig entschieden. Eine Linie endet, wenn sie letzte Spalte im Gitter erreicht hat. Die Fahrzeiten zwischen den Stationen im Gitter werden zufällig aus $\{1, 2, 3\}$ gewählt, und die Takte der Linien zufällig aus $\{5, 10, 15, 20, 30, 60\}$.

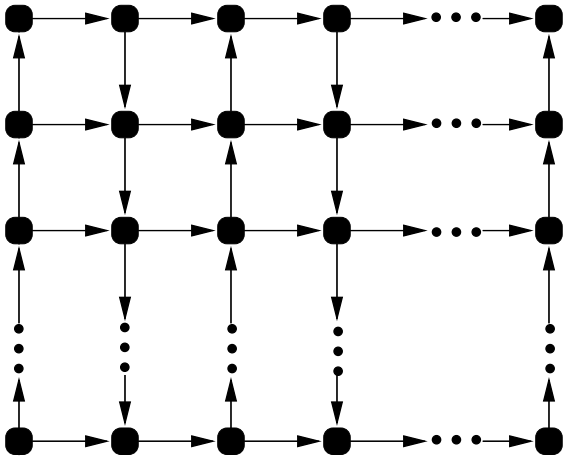


Abbildung 6.2: Grundstruktur für Gitter-Streckennetzwerke.

Ein Beispiel für ein Zufallsstreckennetzwerk mit 3 Linien auf einem 4x5 Gitter findet sich in Abbildung 6.3.

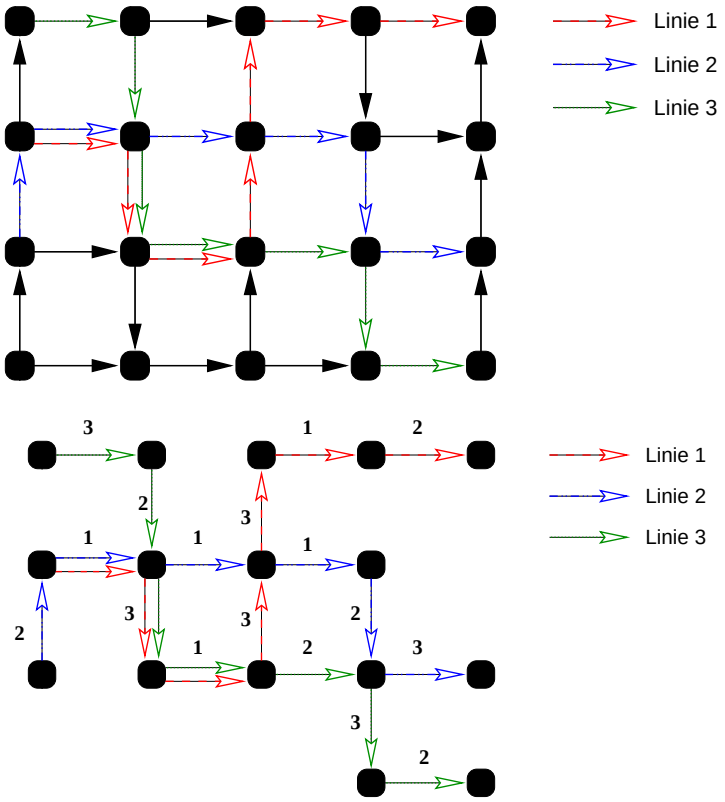


Abbildung 6.3: Beispiel für drei Linien in einem 4x5 Gitter.

6.3 Ergebnisse

6.3.1 Kölner Verkehrs-Betriebe (KVB)

Branch-and-Bound Ergebnisse

Im Branch-and-Bound-Algorithmus werden bei jedem Branch-and-Bound-Schritt der Sicherheitsabstand δ und die Summe der Sicherheitsabstände δ_Σ für einen Teilfahrplan berechnet. Mit Hilfe der in Abschnitt 3.2 vorgestellten Relationen lässt sich die Berechnungzeit von δ und δ_Σ deutlich verbessern.

Zuerst wollen wir untersuchen, wie stark sich diese Reduzierungen auf die Laufzeit des Branch-and-Bound-Algorithmus auswirken.

		ohne Reduktion		mit Reduktion		
Instanzen	$\|L\|$	$\|S\|$	$\|M\|$	$\|S_\Sigma\|$	$\|S_\delta\|$	$\|M_0\|$
I_2	20	461	202	28	16	46
I_{12}	28	543	383	48	28	78
I_{20}	26	525	320	44	22	68

Tabelle 6.2: Ergebnisse der Reduktion mit Hilfe von Relationen aus Kapitel 3.2 für die KVB-Testinstanzen.

Wenn im Branch-and-Bound-Algorithmus an Stellen, wo der Sicherheitsabstand $\delta(\lambda)$ (bzw. Summe der Sicherheitsabstände $\delta_\Sigma(\lambda)$) berechnet wird, für S jetzt die Menge S_δ (bzw. S_Σ) verwendet wird, so braucht der Branch-and-Bound-Algorithmus für unsere Instanzen nur ein Viertel der Zeit, um die gleiche Lösung zu finden.

Die erzielte Laufzeitverbesserung ist gegenüber der Stationsreduktion sehr gering. Dies liegt daran, dass die Berechnung des Sicherheitsabstandes quadratisch mit der Anzahl der Linien zunimmt und die meisten redundanten Stationen von sehr wenigen Linien befahren werden.

In der Tabelle 6.3 sehen wir die Backtracking-Ergebnisse nach 5 Minuten Rechenzeit mit und ohne Ausnutzung der Reduktion der drei Instanzen der KVB.

Aus der Tabelle 6.3 ergibt sich, dass der Laufzeitgewinn bei der Reduktion mittels der MLK-Graphen(M_0) am größten ist. Darum wird in den weiteren Experimenten im Branch-and-Bound-Algorithmus nur diese Reduktion betrachtet.

Tabelle 6.4 zeigt die Ergebnisse des Branch-and-Bound-Verfahrens angewendet auf die Instanzen der KVB nach 5 Minuten und einer Stunde Rechenzeit. Wie man erkennt, verändert sich die Lösung kaum.

		ohne Reduktion			mit Red. (S_δ, S_Σ)			mit Red. (M_0)		
Inst.	$\|L\|$	δ	δ_Σ	Zeit	δ	δ_Σ	Zeit	δ	δ_Σ	Zeit
I_2	20	12	23516	253.44	12	23561	119.39	12	23561	70.97
I_{12}	28	2	4596	166.33	2	4596	41.58	2	4596	15.45
I_{20}	26	**2**	6532	89.01	**2**	6532	24.04	**2**	6542	41.67

Tabelle 6.3: Backtracking-Ergebnisse für die KVB-Testinstanzen nach 5 Minuten Rechenzeit.

		Ergebnis nach 5 Minuten				Ergebnis nach einer Stunde			
Inst.	$\|L\|$	δ	δ_Σ	gap	Zeit	δ	δ_Σ	gap	Zeit
I_2	20	12	23548	570	156.74	12	23548	445	271.80
I_{12}	28	2	4613	128	16.80	2	4612	112	92.38
I_{20}	26	3	6518	173	233.17	3	6544	143	1858.54

Tabelle 6.4: Branch-and-Bound-Ergebnisse für die KVB-Testinstanzen nach 5 Minuten und einer Stunde Rechenzeit.

Ergebnisse mit CPLEX

In Kapitel 5 wurde das Fahrplan-Problem als ganzzahliges lineares Programm dargestellt. Die Anzahl der Gleichungen und Variablen wurden mit Hilfe der Relationen, die bei Branch-and-Bound eingesetzt wurden, erheblich verringert. Die Ergebnisse die CPLEX für die ILP-Darstellung der drei Instanzen erzielte, zeigt Tabelle 6.5.

	nach 5 Minuten				nach einer Stunde			
Ins.	δ	δ_Σ	gap	time	δ	δ_Σ	gap	time
I_2	12	23214	857	297.02	12	23675	12	1894.54
I_{12}	2	4541	224	37.56	2	4651	99	1043.50
I_{20}	3	6452	262	20.59	3	6570	116	1452.05

Tabelle 6.5: CPLEX Ergebnisse für die KVB-Testinstanzen gemäß ILP (5.3.8) nach 5 Minuten und einer Stunde Rechenzeit.

Wenn man die Ergebnisse von unseren Branch-and-Bound-Algorithmus und dem Programmpaket CPLEX vergleicht, so erkennt man, dass die Lösungen nach 5 Minuten von Branch-and-Bound-Algorithmus besser sind als die von CPLEX, aber nach einer Stunde liefert CPLEX bessere Lösungen und bessere obere Schranken.

Ergebnisse mit Hilfe der Heuristiken

In diesen Abschnitt werden die Ergebnisse, die mit Hilfe der Heuristiken aus Abschnitt 4.4 erzielt worden sind, präsentiert und verglichen.

	Algor. 5		Algor. 6		Algor. 7		Algor. 8	
Alg:GA	δ	δ_Σ	δ	δ_Σ	δ	δ_Σ	δ	δ_Σ
I_2	**6**	23220	**4**	23752	**7**	23894	**10**	23533
I_{12}	**1**	4606	**1**	4671	**1**	4667	**1**	4686
I_{20}	**1**	6551	**1**	6621	**1**	6622	**2**	6595

Tabelle 6.6: Ergebnisse der Heuristiken für die KVB-Testinstanzen nach 5 Minuten.

Wie man in Tabelle 6.6 sehen kann, liefern die Heuristiken gute Lösungen für die Summe der Sicherheitsabstände, aber leider sind die minimalen Sicherheitsabstände sehr schlecht. Für die Verwendung im Branch-and-Bound-Algorithmus sind diese Heuristiken nicht geeignet, da Backtracking-Verfahren nach einigen Sekunden viel bessere Lösungen als die Heuristiken liefern. Die Heuristiken, die in Algorithmen 6 und 7 beschrieben werden, kann man zur Nachbesserung von schon berechneten Fahrplänen nutzen.

6.3.2 Ergebnisse für Gitter-Streckennetzwerke

Bei Gitter-Streckenetzwerken treten oft Linienverläufe auf, die in realen Streckennetzwerken kaum auftauchen. In wirklichen Streckennetzwerken kommt es selten vor, dass zwei Linienwege zusammenlaufen, sich trennen und später wieder zusammen kommen. In den erzeugten Gitter-Streckennetzwerken geschieht dies oft. Im Beispiel in Abbildung 6.3 erkennt man, dass dies zwischen den Linie 1 und 2 und zwischen 2 und 3 auftritt. Dies macht die Optimierung viel schwieriger, da die Fahrzeiten der beiden Linien von der Station, wo die Linien sich verzweigen, zu der anderen Station, wo die Linien wieder zusammenlaufen, in der Regel unterschiedlich sind, und dadurch die oberen Schranken sehr schlecht werden, weil solche Strukturen gar nicht berücksichtigt werden.

Für jedes Gitter und für eine feste Anzahl von Linien wurden je 10 verschiedene Streckennetzwerke zufällig erzeugt und für jedes Streckennetzwerk wurde das Fahrplan-Problem mit dem Branch-and-Bound-Algorithmus und die zugehörige ILP-Formulierung mittels CPLEX gelöst. Nach einer Stunde wurde die Rechnung abgebrochen.

In den Abbildungen 6.4 und 6.6 werden prozentual der durchschnittliche Unterschied zwischen bester Lösung und der oberen Schranke für jedes Verfahren dargestellt. In den Abbildungen 6.5 und 6.7 werden prozentual die Differenz der Summe der Sicherheitsabstände für die besten Lösungen beider Verfahren für jede Instanz dargestellt.

Für keine Instanz war die Lösung, die CPLEX lieferte, besser als die Lösung unseres Branch-and-Bound-Algorithmus. Die obere Schranke, die CPLEX liefert, war immer

schlechter als die obere Schranke von unserem Branch-and-Bound-Algorithmus. Bei 10x10-Gitter berechnete der Branch-and-Bound-Algorithmus immer optimale Lösungen für die Instanzen (siehe Abbildung 6.4). Bei 20x20-Gitter liefert CPLEX ab 10 Linien etwa eine 3-4% schlechtere Lösung bzgl. der Summe der Sicherheitsabstände als unser Branch-and-Bound-Algorithmus (siehe Abbildung 6.7).

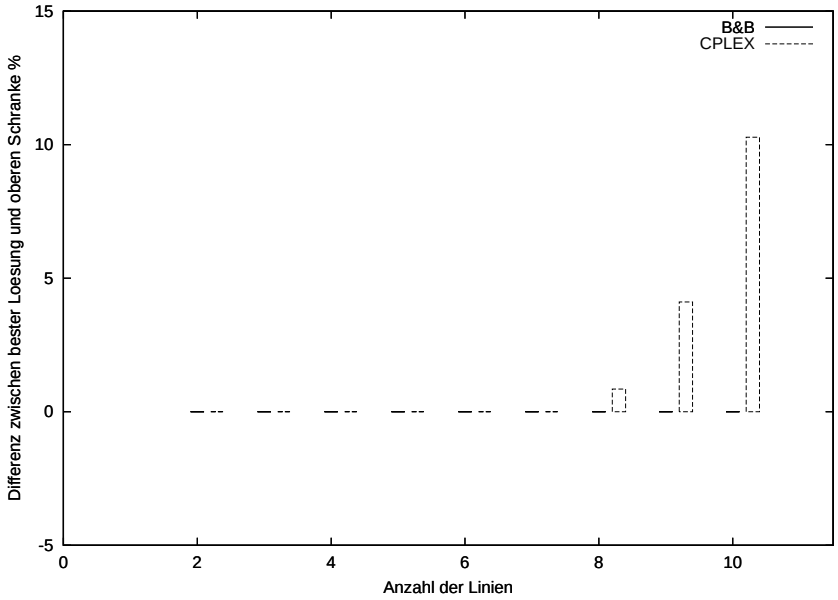


Abbildung 6.4: Durchschnittliche Unterschied zwischen bester Lösung und oberen Schranke für beide Verfahren bei 10x10 Gitter-Streckennetzwerken.

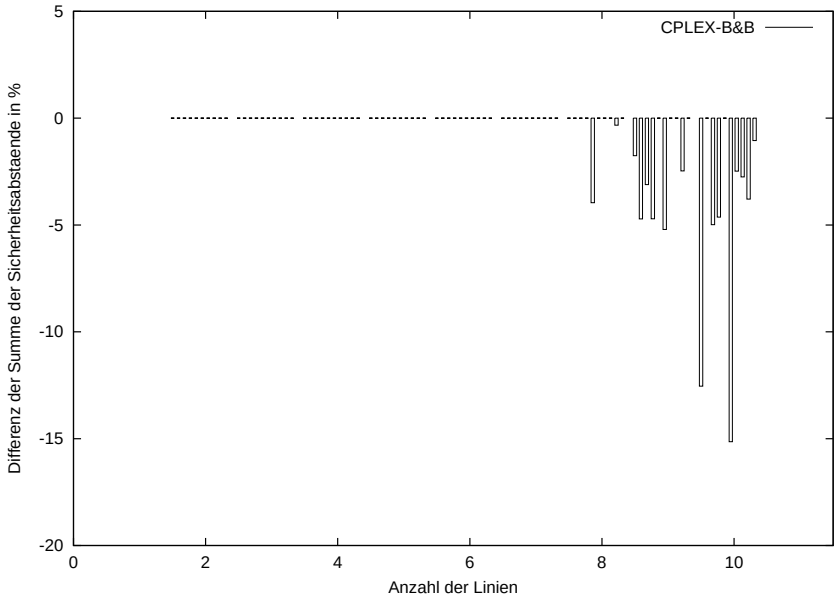


Abbildung 6.5: Differenz der Summe der Sicherheitsabstände der besten Lösungen von CPLEX und Branch-and-Bound-Algorithmus für jede Instanz bei 10x10 Gitter-Streckennetzwerke.

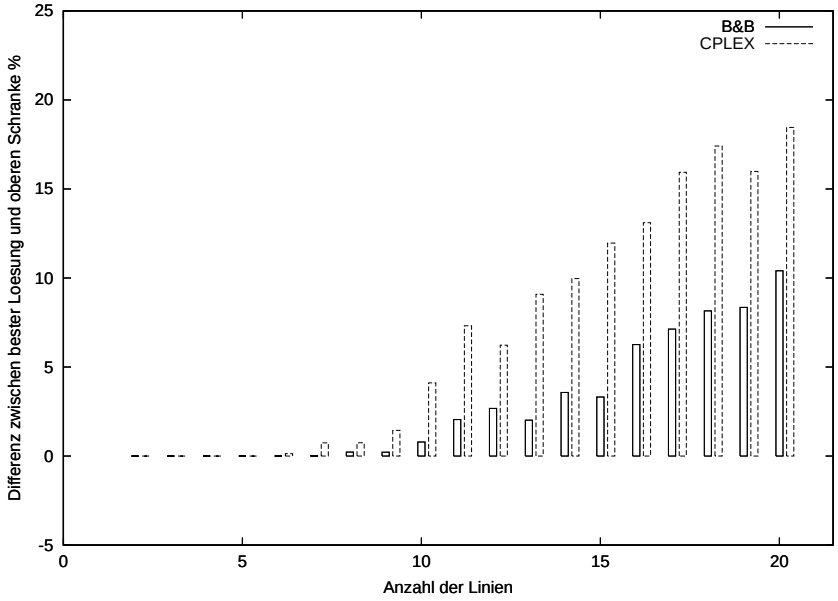


Abbildung 6.6: Durchschnittliche Unterschied zwischen bester Lösung und oberen Schranke für beide Verfahren bei 20x20 Gitter-Streckennetzwerken.

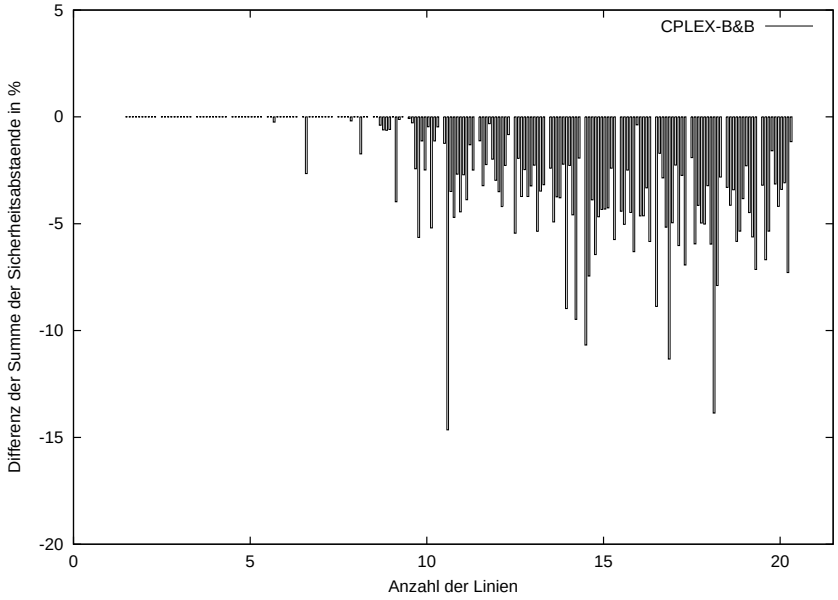


Abbildung 6.7: Differenz der Summe der Sicherheitsabstände der besten Lösungen von CPLEX und Branch-and-Bound-Algorithmus für jede Instanz bei 20x20 Gitter-Streckennetzwerke.

Kapitel 7

Zusammenfassung und Ausblick

In dieser Arbeit wurde ein neuer Ansatz der Fahrplanoptimierung untersucht, bei dem es um Maximierung von zeitlichen Sicherheitabständen geht, wobei das gesamte Streckennetz berücksichtigt wird. Das Ziel dieses Ansatzes war es die Ankunftszeiten der Linien auf den einzelnen Stationen so gleichmäßig wie möglich zu verteilen, so dass kleine Verspätungen, wie sie häufig vorkommen, nur geringe Auswirkungen auf andere Linien haben.

Das Fahrplan-Problem ist NP-schwer, da unter den Lösungen des NP-vollständigen Sicherheitsabstand-Problems, diejenige gesucht wird, deren Summe der Sicherheitsabstände maximal ist. Des Weiteren haben wir untersucht, ob sich das Streckennetz so aufteilen lässt, dass jedes einzelne Teilnetz getrennt optimiert wird und aus den optimalen Fahrplänen der Teilnetze ein optimaler Fahrplan für das gesamte Streckennetz konstruiert werden kann. Dafür haben wir den LK-Graphen des Streckennetzes betrachtet. Es stellte sich heraus, dass man das Streckennetz bzgl. der zweifachen Zusammenhangskomponenten des LK-Graphen in Teilnetze zerlegen kann. Anhand eines Beispiels wurde gezeigt, dass sich dieser Ansatz nicht auf dreifache Zusammenhangskomponenten erweitern lässt, da die Eigenschaften der Teilfahrpläne im Fahrplan nicht für das gesamte Streckennetzwerk gelten.

Um das Fahrplan-Problem exakt zu lösen, haben wir einen Branch-and-Bound-Algorithmus vorgestellt. Für das Branch-and-Bound-Verfahren haben wir obere Schranken für den Sicherheitsabstand untersucht. Da die Laufzeit des Branch-and-Bound-Verfahrens von der Zeit zur Berechnung des Sicherheitsabstands und der Summe der Sicherheitsabstände dominiert wird, und diese von der Anzahl der Stationen abhängt, haben wir mit Hilfe von in Kapitel 3.2 eingeführten Relationen die Anzahl der Stationen reduziert. Beim Liniennetz der Kölner Verkehrs-Betriebe AG (KVB) konnte die Anzahl der benötigten Stationen zur Berechnung von Sicherheitsabständen um 90-95% reduziert werden und zur Berechnung der Summe der Sicherheitsabstände um 85%. Diese Reduktion lieferte eine Laufzeitverbeserung für das Branch-and-Bound-Verfahren auf den Instanzen der KVB von etwa 70%. Eine zweite Reduktion der Kanten des MLK-Graphen lieferte sogar eine Verbesserung der Laufzeit von 85%. Es darf hier nicht vergessen werden, dass diese Laufzeitverbesserung wesentlich von der Reduktion des Streckennetzwerks abhängt. Für experimentelle Unter-

suchungen haben wir Daten des Streckennetzwerks der Kölner Verkehrs-Betriebe AG und künstlich erzeugte Streckennetzwerke verwendet. Leider lieferte nach einer Stunde Laufzeit das Branch-and-Bound-Verfahren keine optimale Lösung für das Fahrplan-Problem für die drei ausgewählten Beispielinstanzen. Das Branch-and-Bound-Verfahren lieferte zwar nach einigen Sekunden bereits gute Lösungen, deren Verbesserung jedoch hohen Rechenbedarf erforderte.

Das Fahrplan-Problem konnten wir auch als ganzzahliges lineares Programm darstellen und durch Anwendung der Reduktion die Anzahl der Variablen und Gleichungen so weit reduzieren, dass es mit CPLEX gut gelöst werden konnte.

Bei dem Vergleich der Lösungen, die von dem von uns implementierten Branch-and-Bound-Verfahren und denen durch CPLEX auf Basis einer ganzzahligen LP-Formulierung gefunden wurden, stellte sich heraus, dass das Branch-and-Bound-Verfahren schnell gute Lösungen findet, aber die Lösungen sich nur mit hohem Rechenaufwand verbessern lassen. Auf künstlich erzeugten Streckennetzwerken waren die Lösungen und oberen Schranken, die CPLEX lieferte, immer schlechter als die des Branch-and-Bound-Verfahrens. Auf den drei Instanzen der KVB lieferte das Branch-and-Bound-Verfahren in den ersten Minuten bessere Lösungen als CPLEX, aber nach längerer Laufzeit wurden die Lösungen und oberen Schranken von CPLEX besser als die mit dem Branch-and-Bound-Verfahren erzielten.

Es wurden auch verschiedene Heuristiken entwickelt, um gute Anfangslösungen für das Branch-and-Bound-Verfahren zu erhalten. Die Heuristiken lieferten keine gute Lösungen für die untersuchten Instanzen der KVB. Einfache Backtracking-Verfahren lieferten nach sehr kurzer Zeit bessere Lösungen als die durch Heuristiken erzielten. Einige Heuristiken können zur Nachbesserung von Fahrplänen die das Branch-and-Bound-Verfahren liefert eingesetzt werden.

Insgesamt liefert der Branch-and-Bound-Ansatz bereits nach kurzer Zeit gute Fahrpläne. Aber Fahrpläne, die mit diesem Ansatz erzeugt werden, berücksichtigen nur eine bestimmte Tageszeit. Damit erhält man Fahrpläne für einzelne Tagesabschnitte, jedoch nicht für die Übergänge zwischen diesen.

Der im Teilabschnitt 4.5.2 vorgestellte, vielversprechende Zerlegungsansatz könnte auch zur Berechnung optimaler Lösungen des Fahrplan-Problems für große Streckennetzwerke herangezogen werden.

Wir haben den Einfluss verschiedener Fahrpläne auf die Betriebskosten nicht berücksichtigt. Dieser Kostenaspekt hat aus Sicht von Verkehrsunternehmen natürlich einen hohen Stellenwert. Darum wäre eine Erweiterung, die den Kostenaspekt einschliesst, wichtig. Dies könnte durch Kombination des PESP-Ansatzes, der in Abschnitt 1.2.2 vorgestellt wurde, mit unserem Ansatz erreicht werden.

Literaturverzeichnis

[BBH90] P. Brucker, R.E. Burkard, and J. Hurink. Cyclic schedules for r irregularly occurring events. *J. Comput. Appl. Mat.*, 30(2):173–189, 1990.

[BKZ95] M. R. Bussieck, P. Kreuzer, and U. T. Zimmermann. Optimal Lines for Railways Systems. Technical Report TR-95-01, Technische Universität Braunschweig, 1995.

[BLSV99] R. Bornhöfer, A. Löbel, U. Strubbe, and M. Völker. Zielorienterte Dienstplanoptimierung. *Heureka '99: Optimierung in Verkehr und Transport, Tagungsbericht*, pages 171–194, 1999.

[Bur86] R. E. Burkard. Optimal schedules for periodically recurring events. *Discrete Applied Mathematics*, 15:167–180, 1986.

[Bus98] M. Bussieck. *Optimal Lines in Public Rail Transport.* PhD thesis, Technische Universität Braunschweig, 1998.

[BWZ97] M. Bussieck, T. Winter, and U. T. Zimmermann. Discrete optimization in public rail transport. *Mathematical Programming 79(3)*, pages 415–444, 1997.

[BZ97] M. R. Bussieck and U. T. Zimmermann. Schlussbericht Optimale Linienführung und Routenplanung in Verkehrssystemen (Schienenverkehr). Technische Universität Braunschweig, 1997.

[CFT+97] A. Caprara, M. Fischetti, P. Toth, D. Vigo, and P. L. Guida. Algorithms for railway crew management. Technical Report OR-97-2, DEIS University of Bologna, Italy, 1997.

[CG87] J. Cerny and F. Guldan. Location of polygon vertices on circles and its application in transport studies. *Aplikace Matematiky*, 32:81–95, 1987.

[Chv79] V. Chvatal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[Chv83] V. Chvatal. *Linear programming.* Freeman, 1983.

[DFV92] W. Domschke, P. Forst, and S. Voß. Tabu search techniques for the quadratic semi-assignment problem. In G. Fandel, T. Gullegde, and A. Jones, editors, *New Directions for Operations Research in Manufacturing*, pages 389–405. Springer, 1992.

[Dom89] W. Domschke. Schedule synchronization for public transit networks. *OR Spektrum 11*, pages 17–24, 1989.

[DV95] J. R. Daduna and S. Voß. Practical experiences in schedule synchronization. In Joachim R. Daduna, Isabel Branco, and Jose M. Pinto Paixao, editors, *Computer-Aided Transit Scheduling*, pages 39–55. Springer, 1995.

[FLO00] R. Freling, R. M. Lentink, and M. A. Odijk. Scheduling train crews: a case study for the dutch railways. Technical Report EI2000-17/A, Economic Institute, Erasmus University Rotterdam, 2000.

[fWVuT01] Bayerische Staatsministerium für Wirtschaft Verkehr und Technologie. Schienennahverkehrsplan 2001/2002, 2001. http://text.stmwvt.bayern.de/pdf/verkehr/Schienennahverkehrsplan.pdf.

[Gen99] Z. Genc. Ein Aspekt der Fahrplanoptimierung im ÖPNV : Maximierung minimaler Sicherheitsabstände. Master's thesis, Universität zu Köln, 1999.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.

[GLV96] M. Grötschel, A. Löbel, and M. Völker. Optimierung des Fahrzeugumlaufs im öffentlichen Nahverkehr. *Heureka '96: Optimierung in Verkehr und Transport, Tagungsbericht*, pages 341–355, 1996.

[Grö99] M. Grötschel. Verkehrsplanung: Bessere Lösungen mit Mathematik. In Dr. Dietmar Hömberg, editor, *Forschungspolitische Dialoge in Berlin: Angewandte Mathematik –Die verborgene Schlüsseltechnologie*, pages 11–22. Veranstaltungsforum der Verlagsgruppe Georg von Holtzbrinck GmbH, 1999.

[Gul80] F. Guldan. Maximization of distances of regular polygons on a circle. *Aplikace Matematiky*, 25:182–195, 1980.

[HMS97] P. Heusch, F. Meisgen, and E. Speckenmeyer. CATS - Computer Aided Tram Scheduling. Technical Report 262, Informatik, Universität zu Köln, 1997.

[Hur92] J. Hurink. *Polygon scheduling.* PhD thesis, Fachbereich Mathematik/Informatik, Universität Osnabrück, 1992.

[ILO98] ILOG. *ILOG CPLEX 6.0.* Reference Manual, 1998.

[KF00] L. Kroon and M. Fischetti. Crew scheduling for netherlands railways - destination: Customer. Technical Report ERS-2000-56-LIS, Erasmus Research Institut of Managment(ERIM), 2000.

[KM00] I. Kontas and J. Mühlenkamp. Fahrplanoptimierung im öffentlichen Nahverkehr. Master's thesis, Universität Dortmund, 2000.

[Kri95] M. Krista. *Verfahren zur Fahrplanoptmierung dargestellt am Beispiel der Synchronzeiten.* PhD thesis, Technische Universität Braunschweig, 1995.

[Lin00] T. Lindner. *Train Schedule Optimization in Public Rail Transport.* PhD thesis, Technische Universität Braunschweig, 2000.

[LN02] C. Liebchen and K. Nökel. Überführung eines mathematischen Modells zur Taktversatzoptimierung in die Praxis. *Heureka 2002: Optimierung in Verkehr und Transport, Tagungsbericht*, pages 49–61, 2002.

[Löb98] A. Löbel. *Optimal Vehicle Scheduling in Public Transit.* PhD thesis, Technische Universität Berlin, 1998.

[LP01] C. Liebchen and L. Peeters. Some practikal aspects of periodic timetabling. In Operations Research Proceedings, 2001.

[LS95] A. Löbel and U. Strubbe. Wagenumlaufoptimierung - Methodischer Ansatz und praktischer Anwendung. Technical Report SC-95-38, Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), 1995.

[MK02] T. Mellouli and N. Kliewer. Umlaufplanung im öffentlichen Verkehr mit mehreren Depots und Fahrzeugtypen: Neue Lösungsmodelle und praktische Aspekte. *Heureka 2002: Optimierung in Verkehr und Transport, Tagungsbericht*, pages 63–75, 2002.

[Nac98] K. Nachtigall. Periodic netword optimization and fixed interval timetables. Habilitationsschrift, Universität Hildensheim, 1998.

[NV95] K. Nachtigall and S. Voget. Optimierung von periodischen Netzplänen mit genetischen Algorithmen. Informatik-Berichte, Universität Hildensheim, 1995.

[NV96] K. Nachtigall and S. Voget. A genetic algorithm approach to periodic railway synchronization. *Computers and Operations Research*, 23:453–463, 1996.

[NW88] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[Odi97] M. A. Odijk. *Railway Timetable Generation.* PhD thesis, Delft University of Technology, 1997.

[Rie99] E. H. Riedemann. Fahrplan - Fahrzeiten verkürzendes Programm durch Linienabstimmung im öffentlichen Nahverkehr. *Heureka '99: Optimierung in Verkehr und Transport, Tagungsbericht*, pages 159–169, 1999.

[Sch98] A. Schrijver. *Theory of linear and integer programming.* Wiley, 1998.

[SU89] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIAM J. Discrete Math.*, 2(4):550–581, 1989.

[Vin89] A. Vince. Scheduling periodic events. *Discrete Appl. Math.*, 25(3):299–310, 1989.

[Vog95] S. Voget. *Aspekte genetischer Optimierungsalgorithmen: mathematishce Modellierung und Einsatz in der Fahrplanerstellung.* PhD thesis, Universität Hildensheim, 1995.

[Voß92] S. Voß. Network design formulations in schedule synchronization. In M. Desrochers and J.M. Rousseau, editors, *Computer-Aided Transit Scheduling*, volume 386, pages 137–152. Springer, 1992.

[Wal96] K. Walther. Die Marktwirksamkeit von ÖPNV-Qualität. *Der Nahverkehr*, 12, 1996.

[Weg93] I. Wegener. *Theoretische Informatik.* B.G. Teubner, 1993.

[Wei98] K. Weihe. Covering trains by stations or the power of data reduction, 1998. In OnLine Proceedings of the 1st Workshop on Algorithms and Experiments (ALEX'98).

Tabellenverzeichnis

Abbildungsverzeichnis

Algorithmenverzeichnis

Symbolverzeichnis

$\sim_{R_\Sigma}$	Äquivalenzrelation auf der Menge S	36
$\sim_{R_M}$	Äquivalenzrelation auf der Menge M	41
$[\cdot]_X$	Äquivalenzklasse der Äquivalenzrelation X	
$\leq_{R_\delta}$	reflexive Halbordnung in S_Σ	38
$\leq_\Lambda$	Halbordnung auf der Menge aller Fahrpläne	19
$a(s_i, l_j, \lambda)$	Ankunftszeit der Linie l_j an der Station s_i bei Fahrplan λ	14
c_k	Verbindung k	13
l_j	Linie j	13
m	Anzahl der Linien	13
n	Anzahl der Stationen	13
s_i	Station i	13
t	Fahrzeitenfunktion für die Verbindungen	13
λ_i	Abfahrtszeit der Linie i	13
λ	Fahrplan	13
Λ	Menge aller Fahrpläne	13
$\delta(s_i, l_j, l_k, \lambda)$	Sicherheitsabstand an der Station s_i von Linie l_k zu Linie l_j bei Fahrplan λ	15
$\delta(s_i, \lambda)$	Sicherheitsabstand an der Station s_i bei Fahrplan λ	15
$\delta(\lambda)$	Sicherheitsabstand bei Fahrplan λ	15,16
δ^*	optimaler Sicherheitsabstand	16
δ^*_Σ	optimaler Zielfunktionswert des Fahrplan-Problems	16
$\delta_\Sigma(\lambda)$	Summe der Sicherheitsabstände bei Fahrplan λ	15
$\overline{\delta(s)}$	obere Schranke für den Sicherheitsabstand an der Station s	51
$\overline{\delta}$	obere Schranke für den Sicherheitsabstand im ganzen Streckennetzwerk	51
$\overline{\delta_\Sigma}$	obere Schranke für die Summe der Sicherheitsabstände	51
$ggT(x, y)$	größte gemeinsame Teiler von x und y	
$kgV(x, y)$	kleinste gemeinsame Vielfache von x und y	
$kgV(K)$	kleinste gemeinsame Vielfache von einer endlichen Menge K von natürlichen Zahlen	

$F(s_i, s_j, l)$	Fahrzeit der Linie l von der Station s_i zu s_j	15
$\mathbb{N}_k$	$= \{0, \ldots, k-1\}$	13
ILP	ganzzahliges lineares Programm (integer linear program)	74
LK-Graph	Linienkonflikt-Graph	20
MLK-Graph	Multi-Linienkonflikt-Graph	20
M	Kantenmenge des MLK-Graphen	20,40
$M(s)$	Teilmenge von M	41
M_0	Repräsentantenmenge der Faktormenge M/R_M	41
$M_0(s)$	Teilmenge von M_0	41
C	Menge der Verbindungen	13
L	Menge der Linien	13
$L(s)$	Linien, die durch die Station s fahren	14
$L(\tilde{S})$	Linien, die durch mindestens eines der Stationen aus $\tilde{S}$ fahren ($\tilde{S} \subseteq S$)	14
$L(\lambda)$	Menge der festen Linien in Teilfahrplan λ	61
S	Menge der Stationen	13
$S(l)$	Stationen, die von der Linien l befahren werden	14
$S(\tilde{L})$	Stationen, die von mindestens einer der Linien in $\tilde{L}$ befahren wird ($\tilde{L} \subseteq L$)	14
S_Σ	Repräsentantenmenge der Faktormenge S/R_Σ	37
S_δ	maximalen Elemente von S_Σ bzgl. R_δ	38
T	Menge der Takte	13
T_l	Takt der Linie l	13
T_j	Takt der Linie l_j	13
$T(\tilde{L})$	Taktmenge der Linien in $\tilde{L} \subseteq L$	14
T_{min}	kleinste Takt aus T	74
T_{max}	grösste Takt aus T	74
T_{kgV}	kleinste gemeinsame Vielfache aller Takte	74

Ich versichere, dass ich die von mir vorgelegte Dissertation selbstständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen im Wortlaut oder dem Sinn nach entnommen sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; dass diese Dissertation noch keiner anderne Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie abgesehn von unten angegebenen Teilpublikationen noch nicht veröffentlicht worden ist sowie, dass ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Ewald Speckenmeyer betreut worden.

- Z. Genc, E. Speckenmeyer. Fahrplanoptimierung im ÖPNV. Technical Report zaik2001-426, Universität zu Köln, 2001.

Lebenslauf

Persönliche Daten:

Name, Vorname:	Genç, Zülfükar
Geburtsdatum:	15.11.1973
Geburtsort:	Hazar-Elazig/Türkei
Familienstand:	ledig
Staatsangehörigkeit:	deutsch

Bildungsgang:

1984-1986	Hauptschule in Wipperfürth
1986-1990	Realschule in Wipperfürth
1990-1993	Gymnasium in Wipperfürth Abschluss: Abitur am Engelbert von Berg Gymnasium
10/1993-7/1999	Studium der Mathematik mit Nebenfach Informatik an der Universität zu Köln
7/1999	Diplom in Mathematik
8/1999-6/2003	Promotionsstudium an der Universität zu Köln
06/06/2003	Promotion zum Dr. rer. nat. an der Universität zu Köln

Tätigkeiten:

10/1996-7/1999	Studentische Hilfskraft am Institut für Informatik
8/1999-7/2002	Stipendiat des Graduiertenkollegs "Scientific Computing" der Universität zu Köln
seit 8/2002	Wissenschaftliche Hilfskraft am Institut für Informatik