Endbericht
Projektgruppe 369
Rechnergestützte Planung von Nahverkehrsnetzen

9. November 2001

Vorwort

Die Lehrveranstaltung „Projektgruppe" ist eine Pflichtveranstaltung des Fachbereiches Informatik der Universität Dortmund. An ihr nehmen maximal 12 Studenten der Kern- oder Ingenieur-Informatik teil. Während einer Projektgruppe sollen Wissen, Methoden und Techniken aus verschiedenen Teilbereichen der Informatik auf ein konkretes Problem angewendet werden. Zusätzlich sollen die Erarbeitung, Anpassung, Erweiterung und Entwicklung problemrelevanter Methoden erarbeitet werden. Der Schwerpunkt einer Projektgruppe ist die Arbeit in Gruppen, die dazu beitragen soll, dass die Organisation, Leitung, Koordinierung, Zusammenarbeit, Darstellung und Vermittlung eigener Ideen usw. in einem großen Projekt und einer Arbeitsgruppe eingeübt werden kann.

Dieser Bericht schildert die Arbeit der Projektgruppe 369, "Rechnergestützte Planung von Nahverkehrsnetzen", die am Lehrstuhl für Quantitative Methoden, Rechnernetze und verteilte Systeme, Rechnersysteme und Leistungsbewertung der Universität Dortmund durchgeführt wird. In den Kapiteln 2 bis 4 wird das zugrunde liegende Problem allgemein beschrieben und durch eine Analyse in einen informatik-spezifischen Kontext gesetzt. Daraus wird in den Kapiteln 5 bis 8 über den Entwurf bis hin zur Realisierung der Lösungsansatz der Projektgruppe beschrieben, bevor in den Kapiteln 9 bis 11 die Ergebnisse erläutert und bewertet werden.

Für Literaturzitate wird im Folgenden [Zitat] verwendet. Fachbegriffe werden *kursiv* dargestellt.

Bei diesem Projekt wurden Daten verwendet, die uns freundlicherweise von der Straßenbahn Herne-Castrop-Rauxel GmbH (HCR), Kommunalverband Ruhrgebiet (KVR), sowie von der Verkehrsverbund Rhein-Ruhr GmbH (VRR) zur Verfügung gestellt wurden. Ein spezieller Dank richtet sich an Herrn Dietrich. Er diente uns als Ansprechpartner bei der HCR und lud uns zur Besichtigung des Betriebes und der Demonstration des dort vorgestellten Systems ein.

Dieses Dokument wurde mit Hilfe von LyX und LaTeX hergestellt. Beide Programme sind Open Source und im Internet frei verfügbar.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Problemstellung[1]

Busliniennetze leisten einen erheblichen Beitrag bei der Bewältigung von Mobilitätswünschen. Ein zunehmendes Umweltbewußtsein und steigende Benzinkosten haben bei vielen Betrieben die Fahrgastzahlen steigen lassen. In Großstädten existieren ausgeprägte Netze, die zumeist aus historischen Netzen über viele Jahre hinweg entstanden sind und nur geringfügige Änderungen erfahren haben. Diese Netze sind jedoch häufig nicht mehr in der Lage, den heutigen Mobilitätsanforderungen in ausreichendem Maße gerecht zu werden. Gleichzeitig sehen sich die Verkehrsbetriebe aufgrund geänderter finanzieller Rahmenbedingungen einem höheren Kostendruck ausgesetzt. Aus diesen Gründen müssen viele Busnetze in den nächsten Jahre reformiert werden. Neben großstädtischen Netzen werden ferner in zunehmendem Maße auch in Mittelstädten eigene Stadtbusnetze gefordert und umgesetzt.

Die Festlegung der Linienwege stützt sich im großen Umfang auf die Intuition und Erfahrung von Angebotsplanern. Eine Rechnerunterstützung der eigentlichen Planung wird nur in geringem Maße geboten und dient hauptsächlich zur Veranschaulichung der Planerideen und deren Wirkungsgefüge.

Die Komplexität des Planungsproblems hängt stark davon ab, welche Zielgrößen bei der Planung berücksichtigt werden. Ein Auswahl möglicher, teilweise konkurrierender Ziele sind:

* Kostenminimierung
* Erschließungsmaximierung
* Fahrzeugparkminimierung
* Reisezeitminimierung
* Minimierung der Umstiegshäufigkeit.

Eine manuelle Planung kann diesen Zielen nur sehr schwer gerecht werden. Aus der Literatur sind einige Verfahren zur Linienplanung bekannt, die beispielsweise eine manuelle Planung durch eine Bewertung unterstützen [Krug87] oder automatisch Linienpläne erstellen [Simonis81][Sonntag77].

Die Projektgruppe soll die Planung von Busnetzen weitestgehend automatisieren.

[1] Auszug aus dem PG-Antrag

Kapitel 2

Organisation

2.1 Projektgruppeninterne Organisation

Um alle Arbeiten zu koordinieren und den Fortschritt der Gruppe zu kontrollieren, wurden wöchentliche Sitzungen durchgeführt. Dabei wurden gerade in der Anfangszeit der PG zwei Termine pro Woche in Anspruch genommen, um auftretende Fragen schnell klären zu können. Zu Beginn der Implementierungsphase wurde dann auf die zweite Sitzung verzichtet, da der Fortschritt auch dann noch übersichtlich genug war und zudem die Zeit zum Implementieren verwendet werden konnte.

Es wurde jeweils ein Protokollführer und ein Sitzungsleiter auf den Sitzungen eingesetzt. Die Bestimmung des Protokollführers erfolgte nach dem Alphabet, während die Leitung der aktuellen Sitzung der Protokollführer der vorhergehenden Sitzung übernahm. Bei den anzufertigenden Protokollen handelte es sich um Ergebnisprotokolle. Diese wurden für alle Teilnehmer sowohl auf der PG eigenen Internetadresse [PG369-Internetadresse] als auch im gemeinsamen Verzeichnis der PG abgelegt, so dass sie jedem PG Teilnehmer zur Einsicht zur Verfügung standen. Die Aufgabe des Sitzungsleiter bestand darin, die Sitzung einzuleiten, deren Ablauf zu steuern (Diskussionen leiten, Abstimmungen durchführen und ähnliches) sowie auf die Bearbeitung der in der vorherigen Sitzung festgelegten Ziele zu achten.

Aufgrund der Vorteile einer Teilgruppenbildung wurden die Projektgruppe in 4 Gruppen zu je 3 Teilnehmern aufgespalten, welchen sich spezialisierten Aufgaben widmen konnten. Zur Koordinierung der Gruppen diente das wöchentliche Projektgruppentreffen und darüberhinaus kleine, modulübergreifende Arbeitskreise.

Die Hierarchisierung der Projektgruppe war sehr flach, so dass keine Spezialaufgaben oder Führungsposten innerhalb der Teilgruppen angewendet wurden. Durch die Gruppendynamik und den zuerst fehleingeschätzten Gruppenaufwand wurde teilweise eine Neuaufteilung notwendig.

Während der Sitzungen wurden immer wieder Kleingruppen bestimmt, die dafür verantwortlich waren, bestimmte Teilergebnisse zu erarbeiten und auf dem nächsten Treffen zu präsentieren. Zudem wurden den Teilnehmern Rollen und damit verbundene Aufgaben zugewiesen.

Aufgrund des Rollenbasierten Ansatzes nach [Gruhn98/99] ergaben sich problemorientierte Aufgabenverteilungen. So wurde die Rolle des Projektmanagers Mike Sarnowski und Reza Scheikh-ol-Eslami anvertraut, Jörg Draeger übernahm die Rolle des Testmanagers, Dokumenten- und somit auch Druckmanager wurde Stefan Magerstedt.

Der Testmanager erstellte einheitliche Testfälle, überwachte und koordinierte die bei der Bearbeitung aufgetretenen Fehler und verwaltete zudem das Testdatendokument. [Testdatendokument]

Der Dokumentenmanager mit zentralem Aufgabengebiet "Endbericht" kennzeichnete die einzelnen Teilkapitel nach Aufwand und Verantwortung, kontrollierte den Fortschritt und verteilte die Teilaufgaben nach Möglichkeit gleichmäßig und bzgl. Spezialwissen und Kompetenz unter den PG-Teilnehmern.

Zusätzlich wurden noch Vertrauenspersonen als Ansprechpartner für einzelne PG-Teilnehmer eingesetzt, welche bei Streitigkeiten als Schlichter fungieren konnten. Zu diesem Rollenbasierten Ansatz stellten sich durch einen Fragebogen Spezialisten zu diversen Themengebieten heraus, welche z.T. auch Vorträge bzw. kurze Einführungen in diese Themengebiete gaben. So wurde von Tim Schürmann zu Beginn der Projektgruppe für alle Teilnehmer ein Crashkurs in C/C++ durchgeführt, weiterhin stellten Reza Scheikh-ol-Eslami, Michael Schlüter und Michael Wintergerst die Modellierungssprache UML [Fowler98] vor, Alexander Schluck führte in LyX und Frank Domick in LaTeX ein.

2.2 Softwaretechnischer Ansatz

Da es auch zum Umfang der Projektgruppe gehört, ein vollständiges, lauffähiges Programm zu implementieren, musste einige Entscheidungen bezüglich der Vorgehensweise getroffen werden, welche unabhängig vom zu implementierenden Programm stehen.

Die PG-Teilnehmer entschieden sich für den typischen softwaretechnischen Ansatz des Spiralmodells mit Prototyping, welcher sich in der Fachwelt allgemein in die vier Teile Analyse, Design, Implementierung und Test gliedert. (Beschrieben z.B. in [Gruhn98/99]) Es gibt aber auch innerhalb der langen Zeitdistanz, in der die Projektgruppe stattfindet (ein Jahr) auch unterschiedliche Organisationsformen. So lässt sich ein Projekt wie das vorliegende auch in einer Iteration durchlaufen, die PG-Teilnehmer entschieden sich aber für einen zweifachen Durchlauf mit evolutionärem Prototyping, d.h. jeder folgende Prototyp baut auf den ersten auf.

Während der Analyse-Phase wurde ein Pflichtenheft erstellt, in welchem die Anforderungen als Muss-, Kann- und Wunsch-Kriterien einordnet wurden. In diesem Stadium wurden auch die Schnittstellen zwischen den einzelnen Modulen festgelegt, um den Datenaustausch möglichst frühzeitig zu erkennen und Probleme zu erkennen.

Im nächsten Schritt wurde das Gesamtdesign mit allen Schnittstellen zwischen den Modulen erstellt, wodurch zunächst das grobe Gerüst des Projekts realisiert wurde. Anschließend erstellten die einzelnen Gruppen das Fein-Design ihrer Module und füllten die Klassen mit Funktionen und Attributen.

Als letzte Phase folgte die eigentliche Implementierung und der Test. So erschien es sinnvoll, einen gemeinsamen, überschaubaren Testfall für das Projekt festzulegen, auf dessen Grundlage die Berechnung der einzelnen Module erfolgen sollte.

Kapitel 3

Analyse

Ziel und Aufgabe dieses Kapitel ist es eine Analyse des gestellten Problems durchzuführen. Dazu wird zuerst durch eine Kontexteinordnung das allgemeine Problem so weit beschrieben, dass es in den Kontext der Informatik gebracht wird. Darauf aufbauend wird dann in drei Unterabschnitten auf verschiedene Aspekte des Kontextes eingegangen. Im ersten wird eine Übersicht über die Fachliteratur und über bereits vorhandene Lösungsstrategien für dieses Problem gegeben. Der zweite Unterabschnitt geht dann auf spezielle Aspekte des Problems ein, für die es in der Informatik bereits allgemeine Lösungen gibt. Als dritte Kontexteinordung werden Werkzeuge und Bewertungsmöglichkeiten erwähnt. Dieser erste Abschnitt verweist hauptsächlich auf die Seminararbeiten der Teilnehmer, die neben ein paar anderen Quellen, als Grundlage für die Analyse des Problems dienten. Im letzten Abschnitt dieses Kapitels erfolgt zuerst eine allgemeine Analyse des Problems mit einer Aufteilung in identifizierte Teilprobleme und anschließend eine detailiertere Analyse und Identifikation der Probleme des jeweiligen Teilproblems.

3.1 Kontexteinordnung

Die Problematik, die sich stellt, ist im Prinzip jedem bewußt, da die Begriffe Bus, Haltestelle, Fahrplan, Abfahrtzeit, Linienverlauf und ähnliche Begriffe aus dem Kontext des ÖPNV zum Allgemeinwissen gehören. Auch scheint es auf den ersten Blick aus einer subjektiven Perspektive heraus ein einfaches Problem zu sein, einen Bus früher abfahren zu lassen oder eine Buslinie anders fahren zu lassen oder eine Haltestelle an eine andere Stelle zu stellen. Wenn man sich jedoch eingehender mit der Problematik beschäftigt, das Problem in einen größeren Zusammenhang betrachtet wird, dann merkt man schnell, dass es sich hierbei um ein sehr komplexes und stark zusammenhängendes Problem der Informatik handelt.

Da die Haltestellen über Straßen verbunden sind, ergibt sich dadurch automatisch ein Straßennetz, welches in der Informatik durch Graphen modelliert wird. Auch die zeitlichen Abhängigkeiten, die sich durch einen Fahrplan ergeben, können mit Hilfe von Graphen sehr gut modelliert werden. Kürzeste Wege, Zusammenhängende Graphen, Zyklen und ähnliche Probleme, die sich auf Graphen ergeben, werden durch die Graphtheorie behandelt.

Als nächste große Schwierigkeit ist das Behandeln der konkurrierenden Zielvorstellungen anzusehen: auf der einen Seite eine Minimierung der Reisezeit, auf der anderen Seite die Kostenminimierung oder z.B. die Spannung zwischen Minimierung der Umsteigehäufigkeit und der gleichzeitigen Fahrzeugparkminimierung. Für solche betriebswirtschaftlichen Probleme, die durch Algorithmen gelöst werden sollen, gibt es in der Informatik das Themengebiet des Operations Research, das viele Forschungsergebnisse hierfür anbietet.

In den nun folgenden drei Abschnitten wird der Kenntnisstand aufgezeigt, welcher der Projektgruppe zur Verfügung stand um eine Entscheidung für die Vorgehensweise bei dem gestellten Problem zu treffen.

3.1.1 Sichtung der Fachliteratur

Der einleitende Vortrag über die Grundbegriffe und Definitionen des öffentlichen Nahverkehrs führte in das Aufgabengebiet ein und listete die verschiedenen, konkurrierenden Ziele und Wünsche der betroffenen Gruppen auf. Desweiteren wurden Bewertungsmaßstäbe vorgestellt, um die Güte eines Nahverkehrsnetzes zu bestimmen.

Darauf aufbauend wurde in drei Seminararbeiten verschiedene Planungsverfahren für den öffentlichen Nahverkehr vorgestellt, wobei jede Arbeit unterschiedliche Schwerpunkte in den Bereichen Haltestellen, Linienplan und Fahrplan setzte.

Zum Thema Haltestellen wurden verschiedene Vorgehensweise erläutert um die individuellen Verkehrswünsche zu bündeln und so optimale Standorte für Haltestellen auf dem Straßennetz zu bestimmen. Durch das Verfahren von Klemt [Klemt78] werden zunächst die individuellen Fahrtwünsche gesammelt und zu Fahrtströmen gebündelt. Diese werden anschließend mit dem Verfahren von Maack [Maack78] an das Straßennetz angepasst, um hieraus die optimale Position der Haltestellen bestimmen zu können. Im Kontext der Haltestellenplanung wurde außerdem ein weiterer Vortrag über Evolutionäre Strategien [EA99] gehalten, der eine alternative Vorgehensweise durch einen gesteuerten Zufall propagiert.

Zum Thema Linienplanung wurden die Verfahren von Sonntag [Sonntag77], Simonis [SimonisDiss], Dienst [Dienst78] und das Verfahren von Bussieck und Zimmermann [Bussieck]vorgestellt. Der Linienplanungsprozess lässt sich in drei Stufen unterteilen: Aufbereitungsphase, in der die Verkehrswünsche auf ein optimales Liniennetzwerk verteilt werden; Verknüpfungsphase, in welcher die zu erstellenden Linien berechnet werden und abschließend die Bewertungsphase, welche ein Gütemaß des errechneten Linienplans ausgibt.

Das Hauptaugenmerk richtet sich auf die Verknüpfungsphase, wobei jedes Verfahren einen anderen Weg verfolgt:

Sonntag berechnet zunächst Basislinien (drei Knoten, die über zwei Kanten verbunden sind) und verknüpft die am stärksten Befahrenen miteinander zu Buslinien. Den Abschluss bildet ein Test auf weitere Verknüpfungsmöglichkeiten, die eine höhere Attraktivität aufweisen.

Simonis hingegen baut die Linien, ausgehend von einer stärksten Linie, sukzessiv durch Anhängen auf, indem er die am stärksten befahrene, adjazente Kante zur Linie hinzufügt.

Dienst geht einen anderen Weg, indem er den Entscheidungsbaum aller Linien aufbaut und durch entsprechende Entscheidungen über Wahl bzw. des

Ausschluss von Linien aus diesem Baum zu den optimalen Linien kommt.

Diese Vorgehensweise, alle möglichen Linien zuzulassen und dann durch ein dem Dienst ähnlichem Verfahren die optimalen Linien auszuwählen, wird auch von Bussieck und Zimmermann bevorzugt.

Für die Bewertungsphase wurden bei den jeweiligen Autoren verschiedene Kriterien vorgestellt nach denen, der entstandene Linienplan eingeschätzt werden kann. Hierbei ist besonders die Bewerungsmethode von Lewandowsky [Lewandowski]hervorzuheben, da sein eingeführtes Optimierungskriterium, das aufbauend auf einem gegebenen Haltestellen-, Linien- und Fahrplan, eine Gesamtgüte berechnet.

Zu Fahrplanerstellung wurde der Algorithmus von Weigand [Weigand81] vorgestellt, der einen vorgegebenen Linienplan in einen Transportkettengraphen umwandelt, aus welchem dann einfach ein Fahrplan bestimmt werden kann.

Die Optimierung des Fahrzeugumlaufs wurde durch den Algorithmus von Grötschel, Löber und Völker [Groetschel96] abgedeckt, der auf einen möglichst kostengünstigen Einsatzplan von Fahrzeugen und Fahrern abzielt. Desweiteren wurde kurz ein Verweis auf das Expertensystem von Krug [Krug87] gegeben, mit dem es möglich ist, einen vorgegebenen Linienplan zu analysieren, Schwachstellen zu charakterisieren und darauf aufbauend Verbesserungsvorschläge oder Alternativen anzubieten.

Im Gesamtzusammenhang der Problemstellung zur Erstellung eines Verfahrens, das eine komplette Planung von Haltestellen, Linien und Fahrplänen realisiert, wurde auch ein Vortrag von Marcus Völker (Betreuer) [Voelker01] gehalten. In dem von ihm erdachten Verfahren erfolgt eine gleichzeitige Berücksichtigung aller drei zu beachtenden Teilprobleme. Dabei werden ausgehend von einer Rendezvoushaltestelle, an der der Anschluß zwischen den Linien garantiert wird, die Hauptlinien gelegt, von denen die restlichen Linien abzweigen. Dabei ist es durch sukzessive Veränderungen möglich, dass die Linien sowohl ihre Länge als auch die Lage ihrer Haltestellen verändern können, wenn es so zu einem besseren Gesamtergebnis kommt.

3.1.2 Einordnung in den Kontext der Informatik

Wie in den vorgestellten Planungsverfahren ersichtlich ist, ergeben sich bei der Planung einige Optimierungsprobleme, für die effiziente Lösungen gesucht werden müssen. Zu diesem Zweck wurde in vier Seminarberichten über das Thema Optimierungsverfahren und Algorithmen referiert.

Der einleitende und grundlegende Vortrag zum Thema Operations Research (OR) erklärte zunächst die Begriffe, Vorgehens- und Arbeitsweise des OR. Es wurden die allgemeinen Optimierungsprobleme vorgestellt: Lineare, ganzzahlige, kombinatorische, nicht-lineare und dynamische Optimierung. Zum einen wurden hier die Charakteristika des jeweiligen Problems anhand eines Beispiels dargestellt, und zum anderen wurde auch kurz auf die (Standard-) Lösungsansätze eingegangen, die dafür bisher gefunden worden sind. Als eines der grundlegenden Werke sei hier stellvertretend Zimmermann [Zimmermann] genannt.

Bevor auf die einzelnen Optimierungsansätze eingegangen wurde, erfolgte noch ein einführender Vortrag über Graphen und Netze. Es wurden die grundlegenden Begriffe und Definitionen dieses Gebietes erklärt, auf die verschiedenen Darstellungsformen hingewiesen und die Algorithmen Dijkstra für kürzeste

Wege, Prim für minimale Spannbäume, Netzplanung mit PERT, sowie Ford-Fulkerson für maximalen Fluss vorgestellt.

Die in den ersten beiden OR-Vorträgen nur kurz umrissenen bzw. nur allgemein behandelten Probleme/Lösungsansätze wurden in den zwei folgenden Vorträgen genauer und ausführlicher beschrieben. Hier wurde insbesondere auf das Simplex-Verfahren eingegangen: Aufstellung des Tableaus und daraus folgende Phasen mit entsprechenden Berechnungsschritten. Außerdem wurde je ein Algorithmus für das Transportproblem und für das Zuordnungsproblem (Glover-Klingman) vorgestellt. Zum Thema Flüsse wurde die Berechnung von maximalen Flüssen (sowohl der Ford-Fulkerson-Algorithmus als auch der von Goldberg und Tarjan-Algorithmus), die Bestimmung der kostenminimalen Flüsse (Busacker und Gowen) betrachtet und es wurde kurz auf Mehrgüterflüsse eingegangen.

Die letzte Arbeit zum Thema Operations Research ging als erstes auf Graphen ein, mit besonderer Berücksichtigung des Algorithmus von Kruskal zur Bestimmung des Minimalgerüsts und zwei Verfahren zur Bestimmung des kürzesten Weges (label-correcting- und label-setting-Verfahren). Weiterhin wurde das Tourenplanungsproblem und seine Lösungsansätze erklärt, wobei jedoch eine bedingte Nutzbarkeit für die Linienplanung festgehalten wurde. Ebenfalls wurden Netzplantechniken dargestellt, die der optimierten Planung von Projekten dienen. Grundlage waren die Arbeiten von Wegener [Wegener] und Güting [Gue92].

3.1.3 Werkzeuge und Bewertung

Die nächste Vortragsgruppe behandelte Werkzeuge, mit deren Hilfe geographische Informationen wie Straßen, Gebäude, Flächen oder auch Haltestellen und Linien verwaltet und graphisch angezeigt werden können. Zu diesem Zweck wurde ein einführender Vortrag über Definition und Anwendungsgebieten von Geo-Informations-Systemen, kurz GIS, gehalten. Es wurde auf historische Aspekte und die Entwicklung solcher Systeme eingegangen und festgehalten, wann solche Systeme zum Einsatz kommen, wobei auch der Bezug zum ÖPNV erstellt wurde.

Nach dieser allgemeinen Darstellung von GIS wurden dann die Programme ArcInfo, ArcView und GRASS vorgestellt. Zu jedem Programm wurden die jeweiligen Konzepte und Arbeitsweisen, sowie Interna und Bedienung, als auch die Programmierung beschrieben.

Die beiden letzten Arbeiten befaßten sich mit Hilfsmitteln, die im Kontext dieser Problemstellung hätten interessant werden können. Es handelt sich dabei um das Themengebiete Simulation und Prozessketten. Zusätzlich zur Definition und Erklärung der beiden Begriffe wurden Beispiele und Anwendungen aufgezeigt. Hierbei wurde versucht, ein Bezug zu Planungssystemen herzustellen. Die Simulation bezog sich dabei auf die tatsächliche Realisierung bzw. Abbildung des Personennahverkehrs in der Bewertungsphase, während die Stärke der Prozessketten in der Modellierung und Optimierung eines Prozesses zu sehen ist, hier also des Logistikprozesses.

3.2 Identifikation der Teilprobleme

Bei der Einarbeitung in das Thema “Planung von Nahverkehrsnetzen”, ist auffällig, dass die meiste verfügbare Fachliteratur sich nie auf das gesamte Problem stützt, sondern immer nur Teilprobleme behandelt. Bei diesen Teilproblemen handelt es sich um die Haltestellenplanung, die Linienplanung und die Fahrplanerstellung. Die Planung eines Nahverkehrsnetzes wird als Abfolge dieser drei Teilprobleme betrachtet, wobei als erstes die Haltestellenplanung erfolgt. Darauf aufbauend wird die Linienplanung durchgeführt und auf dessen Ergebnis die Fahrplanerstellung angewandt.

Im folgenden werden nun diese drei Teilgebiete näher beschrieben.

3.2.1 Haltestellenplanung

Bei der Platzierung der Haltestellen liegt das Problem darin begründet, dass mehrere, teilweise konkurrierende, Parameter in die Beurteilung der Haltestellengüte mit einfließen. Erschwerend kommt die Tatsache der wechselwirkenden Beeinflussung der einzelnen Haltestellen aufeinander hinzu. Dabei lassen sich drei Hauptprobleme erkennen, welche nun genauer beschrieben werden.

Zum ersten handelt es sich um die Platzierung der Haltestellen. Hierbei sind das Auffinden der Bevölkerungsmaxima, das möglichst vollständige Versorgen der gesamten Bevölkerung und das Abdecken der gesamten Fläche — zumindest der Lokalitäten mit besonderer Attraktivität — besonders zu erwähnen. Insbesondere erschwert sich die Platzierung der Haltestellen dadurch, dass die einzelnen Haltestellen andere Haltestellen in der näheren Umgebung direkt beeinflusse, wodurch es dann zu Mehrfachversorgung kommen kann. Dies ist garantiert von Nachteil. Ein einfaches Umsetzen der umliegenden Haltestellen schlägt jedoch offensichtlich fehl, da hierdurch — ähnlich dem bekannten Schneeballsystem — weitere Haltestellen betroffen werden. Also beeinflußt das Setzen einer Haltestelle an einer Position implizit alle anderen bereits gesetzten Haltestellen.

Das nächste Problem liegt in einer sinnvollen Anpassung der platzierten Haltestellen an das Straßennetz — schließlich sollen die Haltestellen über selbiges angefahren werden können. Hierbei sind dann verkehrsplanerische Gegebenheiten zu beachten, wie z.B. dass die Lage der Haltestelle möglichst kreuzungsnah gelegen sein sollte, ohne jedoch die Kreuzung zu blockieren. Weiterhin erschwert die Topologie die Wahl der Haltestellenlage deutlich, wie z.B. Unterführungen, welche von Bussen nicht unterfahren, Straßen, welche zwar befahren werden können, jedoch keine Haltestellenplatzierung an selbiger möglich ist (z.B. Autobahnen).

Bei der Aufbereitung der Daten für das Linienplanung (siehe Kapitel 3.2.2, Seite 9) ist noch zu beachten, dass der Wegenetzgraph sinnvolle Wege enthält, also Berücksichtigung von Einbahnstraßen und Verbot des Wendens in einer Haltestelle, sofern dies nicht verkehrsplanerisch vorgesehen ist.

3.2.2 Linienplanung

Ziel der Linienplanung ist die Erstellung von Linien, die als Folgen von Haltestellen anzusehen sind. Wie auch bei der Platzierung der Haltestellen (siehe Kapitel

3.2.1) gibt es hier das Problem der teilweise konkurrierenden Eigenschaften eines Liniennetzes. Die für den Fahrgast attraktivste Linienführung wäre, wenn jede Haltestelle mit jeder anderen verbunden wäre (volle Vermaschung) und somit alle Fahrtwünsche auf direktem Wege erfüllt werden. Andererseits sind die Betreiber des Liniennetzes daran interessiert ihre Kosten möglichst gering zu halten und somit nur Linien zu befahren, die eine hohe Auslastung aufweisen und somit rentabel sind. Diese beiden Ziele lassen sich nicht gleichzeitig realisieren, somit ist es notwendig Kompromisse zu finden.

Da es den Betreibern vorgeschrieben ist eine Grundversorgung zu garantieren, um auch in dünnbesiedelten Gebieten eine ausreichende Flächenerschliessung sicherzustellen, ist bei der Linienplanung deshalb dafür zu sorgen, dass alle Haltestellen in das Liniennetz integriert werden. Um auf jeden Falle alle Fahrtwünsche erfüllen zu können muss dieses Netz dann zumindest zusammenhängend sein. Die Kosten können hierbei durch die Angabe einer Gesamtlinienlänge des Liniennetzes beeinflußt werden. Eine Erhöhung dieses Wertes kann nun dazu genutzt werden, um die Attraktivität des Netzes für den Fahrgast zu erhöhen, wie z.B. alternative Linien bei Ausfällen und ausreichende Umsteigemöglichkeiten.

Ein anderes Problem gibt es bei der Maximierung auf Direktfahrer oder der Minimierung der Reisezeit. Bei der Direktfahrermaximierung ist das Ziel so vielen Fahrgästen wie möglich ihre Fahrtwünsche mit der Benutzung nur einer Linie zu erfüllen, d.h. die für diese Kunden unattraktiven Umstiege in andere Linien werden vermeiden. Dieses führt aber dazu, dass der Wunsch auf eine geringe Gesamtreisezeit für alle Verkehrswünsche nicht optimal sein muss. So kann das Anschließen einer weiteren Haltestelle den Direktfahreranteil erhöhen, aber die Reisezeit der nun einen weiteren Weg zurücklegenden Fahrgäste steigt.

3.2.3 Fahrplanerstellung

Die Notwendigkeit der Fahrplanerstellung liegt darin begründet, dass für eine Anzahl von Linien systematisch ein ausführbarer Fahrplan erstellt werden muss. Ein Fahrplan wird dabei als eine Zuordnung von Abfahrtszeiten zu den Haltestellen der einzelnen Linien verstanden. Dabei handelt es sich bei der Fahrplanerstellung nicht um ein willkürliches Zuordnen von Abfahrtszeiten zu Haltestellen, sondern um ein komplexeres Problem, bei dem es einige Kriterien einzuhalten gilt. Diese Kriterien wirken unterschiedlich stark auf die Fahrplanerstellung ein und widersprechen sich teilweise gegenseitig. Im Folgenden werden diese Kriterien aufgezählt und beschrieben.

Ein erstes Kriterium ist die Anzahl der gleichzeitig fahrenden Fahrzeuge, damit ein Fahrplan durchgeführt werden kann. Diese Kriterium ist sicherlich aus betrieblichen Gründen von höchster Bedeutung und sollte deshalb möglichst klein gehalten werden. Ein anderes Kriterium ist die Wartezeit der einzelnen Fahrgäste. Diese besagt, wie lange ein Fahrgast zusätzlich zur normalen Fahrzeit der Fahrzeuge unterwegs sein muss, um sein Ziel zu erreichen. Dieses Kriterium ist für die Akzeptanz eines Fahrplanes wichtig, weswegen auch dieser Wert möglichst klein sein sollte. Diese beiden Kriterien widersprechen sich zum Teil, da eine kürzere Wartezeit sicherlich durch eine höhere Befahrung der Linien und einer daraus resultierenden höheren Fahrzeuganzahl erreicht werden kann. Ein weiteres Kriterium ist der Takt, mit dem es möglich ist, dass sich ein Fahrplan periodisch in der Zeit wiederholt. Dieser Takt kann wiederum für alle Linien gleich sein, oder aber variieren. In den heutigen Nahverkehrsnetzen wird meistens ein taktgetriebener Fahrplan eingesetzt, wobei die Linien dort unterschiedlich getaktet sind. Dieses Kriterium wirkt sich auch auf die ersten beiden genannten Kriterien aus. Ein hoher Takt erfordert mehrere Fahrzeuge, woraus wiederum folgt, dass die Wartezeit niedriger wird.

Da sich die Kriterien gegenseitig widersprechen, ist es notwendig diese zu gewichten. Dies erfolgt normalerweise aus betrieblichen Gründen. Wie gut die einzelnen Kriterien berücksichtigt werden können, hängt auch stark von der Wahl des Linienplanes ab. Ist ein Linienplan zum Beispiel direktfahrermaximierend (siehe Kapitel 3.2.2), wirkt sich dieses positiv auf die Wartezeit beim Fahrplan aus, da weniger Fahrgäste Zeit durch Umstiege verlieren.

3.2.4 Auswirkungen auf das Gesamtproblem

Da die einzelnen Teilprobleme stark voneinander abhängen, liegt die größte Schwierigkeit darin, eine geeignete Wahl der Teilprobleme zu finden. Wird eine falsche Wahl getroffen, können bei den einzelnen Teilproblemen unter Umständen die gestellten Kriterien nicht erfüllt werden und damit ein schlechtes Gesamtergebnis erzielt werden.

Kapitel 4

Selektion und Entwurf

Im Anschluss an die Analyse gilt es nun, eine Vorgehensweise zu beschließen. Insgesamt besteht die Aufgabe darin, ein optimales Nahverkehrsnetzes zu finden. Bei näherer Betrachtung zeigtsich jedoch, dass sich eine solche Lösung nur dann finden lassen kann, wenn bestimmte Teilprobleme optimal gelöst werden können. Dabei ist zu erst einmal überhaupt nicht klar, in welchem Zusammenhang die Teilprobleme zueinander stehen, bzw. inwiefern sich die gefundenen Teilprobleme gegenseitig beeinflussen. Daher muss geklärt werden, ob die Teilziele isoliert voneinander gelöst werden können, oder aber gleichzeitig betrachtet werden müssen. Aus der Menge der in Frage kommenden schon bestehenden Algorithmen aus der Fachliteratur und dem Kontext der Informatik müssen für jeden Teil konkrete Algorithmen ausgewählt werden, welche die identifizierten Teilprobleme lösen bzw. die Ziele realisieren sollen.

4.1 Gegenüberstellung der Ziele

Die Suche nach den Zielen wirft zuerst die Frage auf, was ein optimales Nahverkehrsnetz ist. Aus Fahrgastsicht ist es sicherlich ein Netz, in dem man nicht umsteigen und nicht warten muss und auf kürzestem Wege zur gewünschten Haltestelle fahren kann. Das hieße, dass für jede Verbindung zwischen zwei Haltestellen eine Linie verkehren müßte, die zudem noch jede Minute vorbeifährt. Der Betrieb möchte dagegen ein möglichst kostensparendes Netz aufbauen, in dem im Extremfall möglichst wenige Linien so selten wie möglich fahren. Es wird also deutlich, dass sich schon die Grundansichten von "optimal" zwischen Anbieter (Betrieb) und Nutzer (Fahrgast) eher widersprechen als sich entgegenzukommen. Um beiden Seiten gerecht zu werden, muss ein Kompromiss gefunden werden. Als zentrales Kriterium für diesen Kompromiss dient die mittlere Reisezeit, die sich aus dem Fussweg zur Haltestelle, der Fahrzeit und den Wartezeiten zusammensetzt, und die es zu minimieren gilt. Beim Setzen der Haltestellen wird versucht, jedem Einwohner einen optimalen Fussweg - in Form des minimalen Einzugsgebietes einer Haltestelle - zu ermöglichen. Falls das nicht überall gewährleistet werden kann, muss zumindest eine zumutbare Entfernung - durch das maximale Einzugsgebiet - sichergestellt werden. Die somit maximal-möglichst versorgten Bewohner werden anschließend in Fahrtwünsche zusammengefassabschließendt, die die Linienplanung zu Verkehrsströmen umrechnet. Es wird versucht, einer maximalen Anzahl von Fahrgästen eine Direktfahrt ohne Umstieg anzubieten, da sich durch jeden Umstieg die Fahrzeit erhöht, und somitdie Attraktivität des gesamten Verkehrsnetzes sinkt. Dabei muss die betriebliche Vorgabe der maximalen Streckenkilometer eingehalten werden. Im Fahrplan werden abschließend die Wartezeiten auf den errechneten Linien festgestellt. Problematisch sind dabei die netzbedingten Wartezeiten, die sich aus einer zeitlichen Abhängigkeit einer Linie von einer oder mehreren anderen ergibt. Diese müssen hier minimiert werden.

Wenn also jeder der drei Module Haltestellen, Linienplanung und Fahrplanung seinen Teil optimiert, wird die mittlere Reisezeit auch optimal sein. Hier ergibt sich allerdings das Problem, dass sich die drei Teile indirekt beeinflussen. So kann zum Beispiel das errechnete Liniennetz zwar eine maximale Anzahl an Direktfahrern liefern, aber gleichzeitig problematische Abhängigkeiten für den Fahrplan ergeben, so dass sich die netzbedingten Wartezeiten nicht bzw. schlecht minimieren lassen. Durch die Kapselung kann allerdings kein Modul Einfluss auf die Berechnung anderer Teile üben. Ebenso kann kein Modul andere Kriterien außer seinen eigenen beachten, da ansonsten beispielsweise die Linienplan eine Fahrplanung durchführen müßte, um für die eigentliche Fahrplanung gute Voraussetzungen zu liefern. Das würde aber zum einen die eigentliche Fahrplanung überflüssig machen und zum anderen der Teilung in kleinere Probleme widersprechen.

4.2 Abwägung lokale — globale Optimierung

Um der gegenseitigen Einflußnahme der Teilprobleme gerecht zu werden, hat Marcus Völker (s. [Voelker01]) ein Verfahren entwickelt, das alle drei Aspekte Haltestellensetzung, Linienplanung und Fahrplanung gleichzeitig beachtet. So kann die Linienführung zugunsten eines besseren Fahrplans eine andere Route, die eigentlich nicht optimal ist, nehmen. Dies hat den Vorteil, dass wirklich ein globales Optimum erreicht werden kann, aber den Nachteil, dass das Problem sehr viel komplexer wird, da bei jedem Vorgang alle Aspekte auf gegenseitigen Einfluss geprüft werden müssen, und dann der Vorgang geändert werden muss, was dann wieder eine Prüfung zur Folge hat.

Der Vorgang der lokalen Optimierung, in der voneinander getrennt die Berechnungen durchgeführt werden, macht das Gesamtproblem der Nahverkehrsplanung durch kleine überschaubare Teile einfacher. Zudem kann hier ein Berechnungsmodul einfach durch eine andere Variante ausgetauscht werden, was zum Beispiel ermöglicht, Haltestellen nach verschieden gewichteten Aspekten zu setzen. Jedoch kann es sein, dass ein globales Optimum nie erreicht wird, da ein Modul keine Einflußnahme auf das vorherige Modul hat, um so bessere Berechnungsgrundlagen zu erhalten.

Um trotzdem die Möglichkeit zu einem insgesamt besseren Ergebnis zu haben, wird die Möglichkeit einer Rückkopplung zwischen Linienplan und Fahrplan eingeführt. Es wird dabei versucht, die (hohen) netzbedingten Wartezeiten zu eliminieren bzw. minimieren, die durch die Linien verursacht werden. Da der Fahrplan die Linien nicht verändern kann, muss er dies dem Linienplan auftragen. Zu diesem Zweck wird zuerst eine komplette Fahrplanung durchgeführt, in der problematische Stellen identifiziert werden. Diese werden dann dem Linienplanungsmodul mitgeteilt, das anhand dieser Vorgaben die Linien ändert oder ganz neu berechnet. So kann sukzessive ein besserer Fahrplan entstehen, in dem die Wartezeiten so gering wie möglich sind. Auf diese Weise kann die modulare Struktur der getrennten Haltestellen-, Linien- und Fahrplanberechnung beibehalten werden.

4.3 Wahl der Algorithmen

In diesem Kapitel wird dargelegt welcher Algorithmen in den jeweiligen Modulen verwendet worden ist. Dabei wird auch jeweils darauf eingegangen warum diese entsprechende Entscheidung gefällt worden ist und welche Argumente dafür bzw. dagegen gesprochen haben.

4.3.1 Haltestellen

Verwendete Symbole

R Der zu betrachtende Raum.

B Bevölkerungsanzahl im zu betrachtenden Raum R.

B_{Punkt} Bevölkerungsanzahl in der Umgebung des Punktes, bestimmt durch das Einzugsgebiet. Innerhalb des minimalen Einzugsgebietes wird die gesamte Bevölkerung und innerhalb des maximalen (außerhalb des minimalen) Einzugsgebietes wird die Hälfte der Bevölkerung aufaddiert.

B_{peak} Extremwert der Bevölkerungsanzahl.

$$\max\{B_p : p \in R\}$$

Die in 3.2.1, Seite 9, aufgeführten Faktoren erhöhen den Schwierigkeitsgrad des gestellten Problems und es ist leicht offensichtlich, dass kein triviales Verfahren, wie sie z.B. in [Gue92] vorgestellt werden, das Problem optimal lösen kann. Daher werden nun eine Reihe von Lösungsstrategien vorgestellt, welche sich bemühen die Problematik sinnvoll anzugehen.

4.3.1.1 Verfahren nach Klemt und Maack

Das Verfahren von den gleichnamigen Urhebern (siehe Kapitel [Klemt78], [Maack78]), welches sich mit einem sehr ähnlichen Problem beschäftigt (Platzierung von Haltestellen, jedoch Bildung eines Wegenetzgraph für einen Schienenverkehr) kann Inspirationen liefern. Es hat jedoch einige Schwächen, welche nun vorgestellt werden.

Dieses Verfahren arbeitet mit einem Raster, welches über das zu planende Gebiet gelegt wird. Dabei ist jedes Element dieses Rasters gleich groß, kann jedoch frei verschoben werden. Der Hauptnachteil dabei ist, dass jedes Element sich nicht mit anderen überlappen darf und genau eine Haltestelle, mittels Schwerpunktfindung, impliziert. Hierdurch werden genau so viele Haltestellen gelegt, wie durch die Rasterung vorgeschrieben. Dies führt insbesondere bei ungünstiger Ausgangslage dazu, dass selbst in sehr schwach besiedelten Gebieten relativ gleichmäßig verteilt werden. In stark besiedelten Gebieten kann durch eine ungeschickte Positionierung der Elemente eine Reihe von Haltestellen entstehen, obwohl eine einzige, die gleiche Versorgung sicher gestellt hätte. Das Verfahren benötigt zudem Verkehrsstromdaten, welche in dem vorliegenden Datenbestand nicht vorhanden sind. Dies macht es schwer möglich, wenn nicht gar unmöglich, dieses Verfahren anzuwenden. Zuletzt dient dieses Verfahren, wie bereits zuvor erwähnt, der Bildung von Linien für ein Schienenverkehrsnetz. Zum einen liegt dies ausserhalb der hier gestellten Problematik (siehe hierzu vorzugsweise Kapitel 4.3.2, Seite 19 ff.) und zum anderen ist ein ein schienenbasiertes Verkehrsnetz anders geartet, als eines, welches für Busse eingesetzt werden soll (ein Busnetz hat deutlich mehr Haltestellen, als ein Schienenverkehrsnetz; Wegenetzgraph enthält keine geraden Verbindungen zwischen zwei Haltestellen).

Aufgrund der immensen Anzahl von Unbekannten und der gegenseitigen Beeinflussung der Haltestellen wurde versucht, anderen Ansätze zu finden. Eine Auswahl der angedachten Möglichkeiten folgt nun.

4.3.1.2 Stochastische Gleichverteilung

Die Anzahl der Haltestellen muss vom Benutzer vorgegeben werden, andernfalls wird eine Obergrenze von Haltestellen pro Flächeneinheit algorithmisch ermittelt (siehe Kapitel 5.3.2.5, Seite 32). Die Haltestellenkoordinaten werden durch eine gleichverteilte Zufallsfunktion innerhalb des zu betrachtenden Raumes R frei erzeugt, ohne darauf zu achten, ob sich die Haltestellen dann auch an einer Straße befinden.

Dieses Verfahren liefert schnell Ergebnisse, welche aufgrund der Vorgehensweise jedoch nicht (zumindest nur mit sehr geringer Wahrscheinlichkeit) optimale Ergebnisse liefert und wurde aufgrund des geringen Implementierungsaufwandes und der niedrigen benötigten Zeitkomplexität implementiert.

4.3.1.3 Stochastische Gleichverteilung bei Berücksichtigung von Bevölkerungsdichte

Wie 4.3.1.2, nur wird für jede potentielle Haltestelle mittels eines Zufallswertes bestimmt, ob sie gesetzt wird oder nicht. Dabei liegt der gleichverteilt ermittelte Zufallswert zwischen Null und B_{peak}. Liegt dieser unter B_{Punkt}, so wird eine Haltestelle an der entsprechenden Position erzeugt. Ansonsten wird keine Haltestelle gesetzt.
Dieser iterative Vorgang des Setzens von Haltestellen, wird so lange wiederholt, bis so viele Haltestellen vorhanden sind, wie durch den Benutzer angefordert wurden.

Dieses Verfahren liefert im Schnitt bessere Ergebnisse, als das zuvor genannte, wurde jedoch nicht implementiert, da ein anderes Verfahren weiter unten vorgestellt wird, welches die Berücksichtigung der Bevölkerungsdichte beinhaltet (siehe Kapitel 4.3.1.7, Seite 17).

4.3.1.4 Stochastische Gleichverteilung bei Berücksichtigung von bereits vorhandenen Haltestellen

Wie 4.3.1.2, nur wird für jede potentielle Haltestelle geprüft, ob im minimalen Einzugsradius eine Haltestelle vorhanden ist. Wenn dies der Fall ist, wird ein

Fehlerzähler hochgesetzt. Ist noch keine Haltestelle in der Umgebung vorhanden, so wird eine erzeugt.
Dieser Algorithmus bricht ab, wenn alle angeforderten Haltestellen gesetzt wurden, oder eine bestimmte Anzahl an Fehlversuchen hintereinander ohne Setzen einer Haltestelle erfolgt ist.

Dieses Verfahren liefert eine recht gute Flächendeckung und wurde aufgrund des geringen Implementierungsaufwandes, sowie der niedrigen benötigten Zeitkomplexität implementiert.

4.3.1.5 Flächendeckender Algorithmus

Dieser Algorithmus versucht, unabhängig von der Bevölkerungsdichte, mit einer vorgegebenen Anzahl von Haltestellen den größten Teil der gegebenen Fläche abzudecken. Dabei ist diese Lösung nicht trivial, da unterschiedliche Radien der Haltestellen erlaubt werden können. Ebenfalls kann durch den flächendeckenden Algorithmus versucht werden, natürliche Hindernis-Schwierigkeiten zu umgehen. So kann eine Haltestelle zwar direkt neben der Autobahn gesetzt werden, ihr Abdeckungsbereich wird aber nie über die Autobahn hinweg gelegt werden, die andere Seite der Autobahn wird durch diese Haltestelle nicht abgedeckt.

Dieses Verfahren erschien nicht sonderlich effizient im Hinblick auf die Optimalität. Es ist nicht implementiert und entsprechend nicht näher ausgearbeitet.

4.3.1.6 Bevölkerungsdeckender Algorithmus

Es wird zunächst ein Raster über das Untersuchungsgebiet gelegt. In jedem Rasterfeld wird die mittlere Bevölkerungsdichte ermittelt. Im folgenden werden sukzessive Haltestellen über die Gebiete gelegt, die die höchste Bevölkerungsdichte aufweisen. Die Kapazität[1] der Haltestelle wird dann von der Bevölkerungsdichte abgezogen und anschließend die nächste Haltestelle gesetzt, wiederum auf die höchste Bevölkerungsdichte zielend. Im Algorithmus kann eingebunden werden, dass ein Mindestabstand zwischen zwei Haltestellen vorhanden sein muss, ansonsten würde der Algorithmus zu häufig Haltestellen direkt nebeneinander platzieren, wenn die Bevölkerungsdichte in einem Rasterfeld sehr hoch ist.

Dieses Verfahren ist dem von Klemt und Maack (siehe Kapitel 4.3.1.1) sehr ähnlich und hat entsprechen ähnliche Schwächen. Es ist daher nicht implementiert und nicht näher ausgearbeitet.

4.3.1.7 Evolutionäre Algorithmen

> „Seit etwa 4 Milliarden Jahren gibt es Leben auf unserer Erde. Neben der erstaunlichen Artenvielfalt hat die Evolution auch viele Organismen und Formen hervorgebracht, die gut und zum Teil optimal an ihre jeweilige Umwelt angepasst sind. Warum sollte man nicht versuchen, durch die Nachahmung von grundlegenden Evolutionsprinzipien zu neuen, robusteren Optimierungsverfahren zu kommen?
>
> Anfang der sechziger Jahre stellten sich verschiedene Forscher unabhängig voneinander diese Frage. In Deutschland führte dies zu

[1] **Annahme:** An jeder Haltestelle kann mehr als nur ein Bus stehen, welche dann aufgrund deren Kapazit"at die Kapazit"at der Haltestelle vorgeben.

den Evolutionsstrategien (ES), in den U.S.A. zu den genetischen Algorithmen (GA) und zum Konzept des Evolutionary Programming (EP). Diese Verfahren, sowie das später hinzugekommene Genetic Programming (GP), das evolutionäre Suchprinzipien in den Suchraum von Programmiersprachen überträgt, werden heute unter dem Namen evolutionäre Algorithmen (EA) bzw. evolutionary computation (EC) zusammengefaßt.

Ein wichtiger Vorteil evolutionärer Verfahren ist ihre inhärente, skalierbare Parallelität. Auf diese Weise kann die Rechenleistung beliebiger Parallelrechnerarchitekturen für rechenintensive Probleme genutzt werden.

Evolutionäre Algorithmen werden im einfachsten Fall dazu verwendet, Optimierungprobleme folgender Form zu lösen:

$$\min \left\{ f_{(x)} \, : \, x \in \mathcal{M} \right\}$$

Die Zielfunktion f bildet den zulässigen Bereich $\mathcal{M}$ in die reellen Zahlen $\mathbb{R}$ ab.

Ein Individuum lässt sich als Tupel schreiben: $(x, \delta_1, ..., \delta_r)$, $r \geq 1$. Dabei sei $x \in \mathcal{M}$ eine zulässige Lösung, während die $\delta_1, ..., \delta_r$ Strategieparameter repräsentieren. Üblicherweise wird einem Individuum über die Zielfunktion ein Fitnesswert zugeordnet.

Eine Menge von Individuen bildet die Population, die sich unter Verwendung folgender (genetischer) Operatoren verändert: Die Mutation ist ein stochastischer Operator, der die genetische Information eines Individuums modifiziert, nachdem die Rekombination nur genetisches Material der Eltern neu zu einem Nachkommen zusammengesetzt hat. Die Selektion bestimmt, welche Individuen in der nächsten Generation zur Fortpflanzung zugelassen werden. Abbildung 4.1 stellt das allgemeine Iterationsschema eines evolutionären Algorithmus dar.

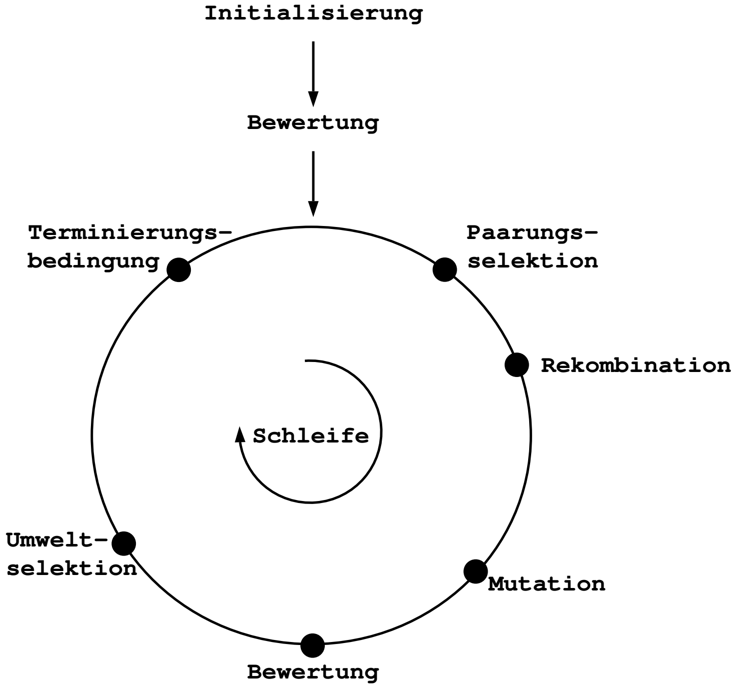
Die verschiedenen Klassen evolutionärer Algorithmen unterscheiden sich durch die Repräsentation der Individuen und demzufolge auch durch die auf ihr arbeitenden genetischen Operatoren."

Entnommen aus [EA99]. Eine weitergehende Einleitung zum Thema Evolutionäre Algorithmen findet sich in [ScHeFe94].

Evolutionäre Strategien

„Obwohl ursprünglich für die experimentelle Optimierung und diskrete Suchräume entwickelt, bildet der $\mathbb{R}^n$ seit der Realisierung auf Rechnern typischerweise den Suchraum von Evolutionsstrategien. Für diesen Fall ist die Theorie der ES am weitesten fortgeschritten. Daher besteht ein Individuum aus einem n-dimensionalen reellwertigen Objektvariablenvektor x und einer Menge von Strategieparametern, die die Mutation steuern. Die Anzahl der Strategieparameter kann je nach Einstellung zwischen 1 und $n + \frac{n(n-1)}{2}$ variieren.

Abbildung 4.1: Iterationsschema eines Standard-EA



> Die Besonderheit von ES liegt darin begründet, dass sowohl Objekt- als auch Strategieparameter rekombiniert und mutiert werden, wodurch sich bei entsprechender Selektion die Möglichkeit einer ständigen Selbstadaptation der Suchschrittweiten und gegebenenfalls auch der Vorzugsrichtungen ergibt.
>
> Wesentlich für das Gelingen der Selbstanpassung ist ein Geburtenüberschuss, der durch Selektion wieder abgebaut wird. Untersucht werden Konvergenzeigenschaften, der Einfluss verschiedener Operatoren und Erweiterungen des Grundalgorithmus auf die mehrfache Zielsetzung, dynamische Optimierung und verschiedene Parallelisierungsansätze."

Entnommen aus [EA99].

Wie die Kodierung im Detail aussieht, ist an dieser Stelle unerheblich und kann bei Bedarf unter 7.5.2, Seite 90 ff., nachgelesen werden. Dies ist der erfolgsversprechendste Ansatz und wurde implementiert. Bei geschickter Kodierung können alle Teilprobleme, wie in 3.2.1, Seite 9, definiert, für eine Optimierung herangezogen werden.

4.3.2 Linienplan

Für den Linienplan standen die Algorithmen von Dienst , Bussieck und Zimmermann, Krug[Krug87], Simonis[Simonis81] und Sonntag[Sonntag77] zur Auswahl.

Das Verfahren von Dienst bzw. Bussieck und Zimmermann ist für den schienengebundenen Fernverkehr konzipiert, wo es auf einer Strecke parallel verschiedene Angebote mit verschiedlichen Zeitaufwänden und Kosten gibt (z.B. S-Bahn vs. ICE), was im Nahverkehr nicht der Fall ist. Zudem ist ein Schienennetz nicht so engmaschig und flexibel in der Routenführung wie das Netz des nicht-schienengebundenem Personennahverkehrs. Dieses Verfahren wurde deshab nicht eingesetzt.

Bei der Arbeit von Krug handelt es sich um ein interaktives Expertensysten, das kein Implementierbares Verfahren enthält, sondern Entscheidungshilfen und Informationen für eine manuelle Planung anbietet. So ein System hätte nur in einer Bewertungsphase zum Einsatz kommen können, um Schwachstellen im Linienplan aufzuzeigen und so nachträglich den Linienplan manuell zu optimieren.

Im Bezug auf die Verknüpfungsphase geht die Arbeit von Simonis einen Weg, der sich konzeptionell und methodisch von den Verfahren der anderen Autoren unterscheidet. Hier werden die Linien sukzessiv aufgebaut, was aber dazu führt, das ein schon bestehendes Liniennetz nicht weiter optimiert werden kann.

Das Verfahren von Sonntag wurde umgesetzt, da es von den zur Auswahl stehenden Verfahren zur Planung eines Linienplan des ÖPNV am besten geeignet schien. So kommt es schneller als Simons zu einer vorläufigen, sub-optimalen Lösung, welche dann weiter optimiert wird, bis keine Verbesserung mehr möglich ist. Deswegen ist dieses Verfahren auch in der Lage andere bereits erzeugte Linienpläne zu verbessern, die z.B. mit einem beliebigen Planungsverfahren berechnet oder als bereits existierender Linienplan vorgegeben werden.

4.3.3 Fahrplan

Die Wahl des Verfahrens zur Berechnung eines Fahrplanes fiel aufgrund fehlender Alternativen auf das von [Weigand81] vorgestellte Verfahren. Trotz intensiver Literaturrecherche gelang es nicht, ein anderes als das verwendete Verfahren zur Berechnung eines Fahrplans ausfindig zu machen. Da das Verfahren eigentlich für den Schienenverkehr ausgelegt ist, gab es bei den Personen, welche die Berechnung des Fahrplans zu realisieren hatten, zuerst Bedenken bezüglich der Tauglichkeit des Verfahrens in einem Busnetz. Versuche mit kleineren Busnetzen zeigten jedoch, dass diese unbegründet waren, da lediglich die Reduktion, ein Teilschritt innerhalb der Berechnung, entfällt.

4.3.4 Rückkopplung

Ziel der Rückkopplung ist, die netzbedingten Wartezeiten bei der Fahrplanberechnung zu minimieren. Eine mögliche Rückkopplungsstrategie könnte nun darin bestehen, einen Linienplan zu berechnen,indem sich möglichst wenige Linien kreuzen. Das verwendete Verfahren zur Linienplanung geht jedoch den entgegengesetzten Weg. Viele Verknüpfungen sind hier erwünscht. Diese Strategie konnte also nicht eingesetzt werden. Aufgrund mangelnder Erfahrung in der Erstellungvon Linienplänen und Fahrplänen, fiel es schwer, weitere Alternativen zu finden, welche dazu benutzt werden könnten, eine automatische Verbesserung gefundener Lösungen mittels Rückkopplung zu verwirklichen. Die letztendlich gewählte Strategie setzt auf Wissen des Anwenders bezüglich Linien- und Fahrplanung. Der Funktion eines Expertensystems entsprechend identifiziert das Programm die Problemstelle und lässt den Anwender das weitere Vorgehen entscheiden. Dazu werden alle Linien einer Problemstelle bestimmt, welche nach dem Transportkettengraph der Fahrplanerstellung (siehe Kapitel 5.5) eine zeitliche Abhängigkeit zur Problemstelle haben. Diese werden dann absteigend sortiert, bezogen auf den Verknüpfungsgrad, ausgegeben. Der Anwender hat dann folgende Möglichkeiten:

- Haltestellen löschen
- Haltestellen versetzen, so dass sich die Fahrzeiten (und somit die zeitlichen Abhängigkeiten) verändern
- Haltestellen aufteilen, so dass verknüpfte Linien nebenläufig werden
- problematische Linien zum Teil löschen (restliche Linien werden Pflichtlinien) und Linienplan neu starten
- Pflichtlinien über die problematischen Linien legen, so dass diese nicht mehr generiert werden

Weiterhin ist es möglich, eine Verknüpfung an einer oder mehreren problematischen Stellen zu verbieten und damit das Linienplan-Modul neu zu starten. Dieses neue Ergebnis kann das Fahrplan-Modul bewerten und mit dem alten Ergebnis vergleichen. Dieses wird solange wiederholt bis das Ergebnis zufriedenstellend für den Anwender ist, oder alle Verknüpfungen verboten sind. Problematisch ist hier, dass das Vorgehen sehr viel Zeit beansprucht und eine Verbesserung nicht garantiert werden kann.

Kapitel 5

Mathematische Modelle

In den vorangegangenen Kapiteln wurde das zugrunde liegende Problem in seinen Grundaspekten analysiert und erste Designentscheidungen, mit Auswahl der jeweiligen Algorithmen, getroffen. Die bisherige Darstellung erfolgte auf einem verbalen bzw. nicht-mathematischen Niveau.

Mit Hilfe der in diesem Kapitel vorgestellten mathematischen Modelle der einzelnen Module sollen die Analyseergebnisse präzisiert werden. Dieses bedeutet, dass auf Grundlage von mathematischen Definitionen die Sachverhalte eindeutig und prägnant formuliert werden sollen.

Um die Bedeutung eines mathematischen Modells zum Ausdruck zu bringen, werden zu Beginn dieses Kapitels wesentliche Aspekte einer Modellbildung erläutert.

Daran anschließend werden die Voraussetzungen der mathematischen Modelle der einzelnen Teilmodule dargelegt. Es soll somit ein Bogen gespannt werden, der anzeigt, dass diese Modelle in Zusammenhang stehen und hinsichtlich ihrer Schnittpunkte in Einklang stehen. Weiterhin sollen Definitionen angelegt werden, auf die sich jedes einzelne Modul beziehen kann.

Folglich werden die einzelnen Modelle beschrieben und erläutert.

5.1 Sinn eines mathematischen Modells

Die Modellbildung ist ein Mittel, eine abstrakte Sichtweise auf ein zugrunde liegendes Problem einzunehmen. Es soll ein Kontext geschaffen werden, in dem ein Problem verstanden und gelöst werden kann.

Beispielsweise bildet der klassische Von-Neumann-Rechner eine einfache Abstraktionsmöglichkeit, um komplexe Rechnerstrukturen vereinfacht zu verstehen. Hiermit lässt sich das Programmiermodell des gemeinsamen Speichers verstehen und bietet die Grundlage der Definition von Programmiersprachen wie C++, die auf einem höheren Abstraktionsniveau angesiedelt sind.

Ähnlich gilt es für die Teilprobleme der Haltestellen-, der Linienplan- und Fahrplanerstellung aus den Analyeergebnissen Modelle, also Abbilder der Realität, zu schaffen, in denen sich die Probleme verstehen und lösen lassen.

Es soll die Möglichkeit eröffnet werden, prägnant das Problem zu spezifizieren, so dass in den nächsten Schritten, Lösungsmöglichkeiten gefunden werden können. Anders ausgedrückt soll ein Rahmen und Begriffsinstrumentarium geschaffen werden, in dem sich Informatiker verständigen können.

Da der Kontext hinsichtlich der Informatik zur Problemlösung im Bereich des Operations Research zu finden ist, gilt das Augenmerk dahin zu richten, eine zu optimierende Zielfunktion zu bestimmen. Die Optimierung erfolgt hierbei unter Einbehaltung bestimmter spezieller Nebenbedingungen.

Für dieses Modell gilt es in dann in den nächsten Phasen, Lösungsmöglichkeiten zu suchen, wobei erkannt werden muss, um welche Art von Optimierungsproblem es sich handelt.

5.2 Voraussetzungen der Modelle

5.2.1 Grundlegende Definitionen

Der Sinn der nun folgenden grundlegenden Definitionen ist es ein einheitliches Gerüst für die daran anschließende Darstellung der Modelle und deren Voraussetzungen zu schaffen. Es soll weiterhin bewirkt werden, dass es keine Missverständnisse bzw. Fehlinterpretationen hinsichtlich der Begrifflichkeit gibt und das keine Definitionen mehrfach durchgeführt werden.

Definition Ungerichteter Graph

Ein ungerichteter Graph $G = (V, E)$ ist definiert durch eine endliche Menge V von Knoten und eine Menge von Kanten E, wobei für beliebiges $e = \{v, w\} \in E$, $v \neq w \in V$ gilt.

Definition Gerichteter Graph

Ein gerichteter Graph $G = (V, E)$ ist definiert durch eine endliche Menge V von Knoten und eine Menge von Kanten $E \subseteq V \times V$.

Definition Ungerichteter Multigraph

Ein ungerichteter Multigraph $G = (V, E, f)$ ist ein ungerichteter Graph zusammen mit einer Funktion $f : E \to \mathbb{N}$, die die Multiplizität der jeweiligen Kante angibt.

Definition Gerichteter Multigraph

Ein gerichteter Multigraph $G = (V, E, f)$ ist ein gerichteter Graph zusammen mit einer Funktion $f : E \to \mathbb{N}$, die die Multiplizität der jeweiligen Kante angibt.

Definition Knotenbewerteter, ungerichteter Graph

Ein knotenbewerteter, ungerichteter Graph $G = (V, E, f_v, k_v)$ ist ein ungerichteter Graph zusammen mit einer Funktion $f_v : V \to \mathbb{R}^{k_v}$ für festes $k_v \in \mathbb{N}$.

Definition Kantenbewerteter, ungerichteter Graph

Ein kantenbewerteter, ungerichteter Graph $G = (V, E, f_e, k_e)$ ist ein ungerichteter Graph zusammen mit einer Funktion $f_e : E \to \mathbb{R}^{k_e}$ für festes $k_e \in \mathbb{N}$.

Definition Knoten- und kantenbewerteter, ungerichteter Graph

Ein knoten- und kantenbewerteter, ungerichteter Graph $G = (V, E, f_v, k_v, f_e, k_e)$ ist ein ungerichteter Graph, der sowohl knoten- als auch kantenbewertet ist.

Definition Knotenbewerteter, gerichteter Graph

Ein knotenbewerteter, gerichteter Graph $G = (V, E, f_v, k_v)$ ist ein gerichteter Graph zusammen mit einer Funktion $f_v : V \rightarrow \mathbb{R}^{k_v}$ für festes $k_v \in \mathbb{N}$.

Definition Kantenbewerteter, gerichteter Graph

Ein kantenbewerteter, gerichteter Graph $G = (V, E, f_e, k_e)$ ist ein gerichteter Graph zusammen mit einer Funktion $f_e : E \rightarrow \mathbb{R}^{k_e}$ für festes $k_e \in \mathbb{N}$.

Definition Knoten- und kantenbewerteter, gerichteter Graph

Ein knoten- und kantenbewerteter, gerichteter Graph $G = (V, E, f_v, k_v, f_e, k_e)$ ist ein gerichteter Graph, der sowohl knoten- als auch kantenbewertet ist.

Definition Knotenbewerteter, ungerichteter Multigraph

Ein knotenbewerteter, ungerichteter Multigraph $G = (V, E, f, g_v, m_v)$ ist ein ungerichteter Multigraph zusammen mit einer Funktion $g_v : V \rightarrow \mathbb{R}^{m_v}$ für festes $m_v \in \mathbb{N}$.

Definition Kantenbewerteter, ungerichteter Multigraph

Ein kantenbewerteter, ungerichteter Multigraph $G = (V, E, f, R_e, m_e)$ ist ein ungerichteter Multigraph zusammen mit einer Relation $R_e \subseteq E \times \mathbb{N} \times \mathbb{R}^{m_e}$, wobei für $u = (c, d, (p_1, ..., p_m)) \in R_e, c \in E$ und $1 \leq d \leq f(c)$ gilt.

Definition Knoten- und kantenbewerteter, ungerichteter Multigraph

Ein knoten- und kantenbewerteter, ungerichteter Multigraph $G = (V, E, f, g_v, m_v, R_e, m_e)$ ist ein ungerichteter Multigraph, der sowohl knoten- als auch kantenbewertet ist.

Definition Knotenbewerteter, gerichteter Multigraph

Ein knotenbewerteter, gerichteter Multigraph $G = (V, E, f, g_v, m_v)$ ist ein gerichteter Multigraph zusammen mit einer Funktion $g_v : V \rightarrow \mathbb{R}^{m_v}$ für festes $m_v \in \mathbb{N}$.

Definition Kantenbewerterter, gerichteter Multigraph

Ein kantenbewerteter, gerichteter Multigraph $G = (V, E, f, R_e, m_e)$ ist ein gerichteter Multigraph zusammen mit einer Relation $R_e \subseteq E \times \mathbb{N} \times \mathbb{R}^{m_e}$ für festes $m_e \in \mathbb{N}$, wobei für $u = (c, d, (p_1, ..., p_m)) \in R_e, c \in E$ und $1 \leq d \leq f(c)$ gilt.

Definition Knoten- und kantenbewerteter, gerichteter Multigraph

Ein knoten- und kantenbewerteter, gerichteter Multigraph $G = (V, E, f, g_v, m_v, R_e, m_e)$ ist ein gerichteter Multigraph, der sowohl knoten- als auch kantenbewertet ist.

Definition Pfad

Gegeben sei ein ungerichteter oder gerichteter Graph $G = (V, E)$. Ein Pfad ist eine endliche Folge von Knoten $(v_0, v_1, ..., v_n)$ mit $v_i \in V$, wobei im gerichteten Fall $(v_i, v_{i+1}) \in E$ und im ungerichteten Fall $\{v_i, v_{i+1}\} \in E$ für alle $0 \leq i \leq n-1$ gilt.

Definition Zusammenhängender, ungerichteter Graph

Ein ungerichteter Graph $G = (V, E)$ heißt zusammenhängend, falls zwischen zwei beliebigen Knoten $v \neq w \in V$ ein Pfad in G existiert.

Definition Teilgraph

Gegeben sei ein gerichteter (oder ungerichteter Graph) $G = (V, E)$. $G' = (V', E')$ ist Teilgraph von G genau dann, wenn $V' \subseteq V, E' \subseteq E$ und für $e = (v, w) \in E'$ (oder $e = \{v, w\} \in E$) gilt: $v, w \in V'$.

Definition Zusammenhangskomponente

Gegeben sei ein ungerichteter Graph $G = (V, E)$. Einen maximalen Teilgraphen $G' = (V', E')$ von G bezeichnet man als Zusammenhangskomponente, falls G' zusammenhängend ist.

Definition Baum

Gegeben sei ein ungerichteter Graph $G = (V, E)$. G ist ein Baum genau dann, wenn G zusammenhängend ist und $|E| = |V| - 1$ gilt.

Definition ungerichteter Zyklus

Gegeben sei ein ungerichteter Graph $G = (V, E)$ und ein Pfad $(v_0, v_1, ..., v_n)$. Dieser Pfad ist ein ungerichteter Zyklus genau dann, wenn $n \geq 2$, $v_0 = v_n$ und ansonsten $v_i \neq v_j$ für alle $i \neq j$ gilt.

Definition gerichteter Zyklus

Gegeben sei ein gerichteter Graph $G = (V, E)$ und darauf ein Pfad $(v_0, v_1, ..., v_n)$. Dieser Pfad ist ein gerichteter Zyklus genau dann, wenn $n \geq 2$, $v_0 = v_n$ und ansonsten $v_i \neq v_j$ für alle $i \neq j$ gilt.

5.2.2 Schnittstellen

Um mathematische Modelle für die einzelnen Module entwerfen zu können, die auch miteinander harmonieren, müssen Grundvoraussetzungen geschaffen werden, die über die reine Definitionsbildung hinausgehen. Dazu gehört, dass die mathematischen Modelle ineinander übergreifen, so dass die Ziele des Haltestellen-Modells die Voraussetzungen des Linienplan-Modells sind (analog hinsichtlich des Übergangs zwischen Linienplan- und Fahrplan-Modell). Damit soll die Gültigkeit der mathematischen Modelle im Gesamtzusammenhang der Problemlösung deutlich gemacht werden.

Die in der folgenden Schilderung dargelegten Voraussetzungen und Ziele der jeweiligen Modelle werden in den nachfolgenden Modellbeschreibungen näher erläutert. Es soll an dieser Stelle jedoch explizit das Zusammenwirken der Modelle zum Ausdruck gebracht werden.

Das Modell der Haltestellen geht vom Vorhandensein der folgenden Werte und Konstrukte aus:

1. Straßenverlauf
2. Bevölkerungsstruktur
3. Lage existierender Haltestellen und das Fahrtwunschaufkommen zwischen diesen
4. Geschwindigkeitsvorgaben
5. Breite der Straßen
6. Haltemöglichkeiten
7. Wendemöglichkeiten
8. Parkmöglichkeiten für Busse
9. Unterführungen mit Höhenangaben
10. Kreuzungen samt zugehörigem Manövrierradius

Ziel ist es aus diesen Daten einen Wegenetzgraphen und eine Verkehrsbedarfsmatrix zu erzeugen.

Der Wegenetzgraph ist ein ungerichteter Graph mit Knoten- und Kantenbewertungen $G_W = (H_W, U_W, f_h, 1, f_u, 1)$ (siehe Kapitel 5.2.1), wobei die Bedeutung der Parameter folgend festgelegt sind:

1. H_W : Knotenmenge symbolisiert die Menge der erzeugten Haltestellen
2. U_W : Kantenmenge stellt die Verbindungen der Haltestellen dar, die genutzt werden können
3. f_h : injektive Abbildung der Knoten bzw. Haltestellen auf eine natürliche Zahl, die den eindeutigen Bezeichner (ID) einer Haltestelle angibt
4. f_u : Abbildung der Kanten bzw. Verbindungen auf die entsprechende Entfernungsangabe in Metern

Die Verkehrsbedarfsmatrix F beinhaltet für jedes Paar von erzeugten Haltestellen den jeweiligen Verkehrsstrom, also das Verkehrsaufkommen zwischen zwei Haltestellen.

Das mathematische Modell des Linienplans setzt hierauf genau an und erzeugt das Netz der Transportverbindungen, wobei auf Konsistenz hinsichtlich der Haltestellenbezeichner geachtet wird. Das Netz der Transportverbindungen ist ein gerichteter Multigraph mit Konten- und Kantenbewertungen $G_N = (V_N, E_N, f, g_v, 1, R_e, 3)$ (siehe Kapitel 5.2.1) mit dieser Aussagekraft:

1. V_N: Knoten symbolisieren die Haltestellen
2. E_N:Kanten stellen die Verbindungen zwischen zwei Haltestellen dar, die von einer bestimmten Linie befahren wird
3. f: Angabe der Multiplizität der Kanten
4. g_v: injektive Abbildung der Knoten auf Haltestellen-Bezeichner (IDs)

5. R_e :Angabe der Liniennummer, des Verkehrsstroms und der Fahrzeit bezüglich des Tupels (u, v) mit $u \in E, 1 \leq v \leq f(u)$

Das mathematische Modell des Fahrplan behandelt diesen Graphen als ihre Eingabe und hat als Ziel, eine geordnete Folge von Elementen

$$f = (linienId, halteId, stunde, minute)$$

zu generieren. Die Bedeutung eines Elementes läßt sich verbal hierdurch erklären: eine Linie, gekennzeichnet durch ihren eindeutigen Bezeichner $linienID$, fährt an einer bestimmten Haltestelle, festgelegt durch $halteID$, um $stunde$: $minute$ Uhr ab. Dabei wird auf Konsistenz hinsichtlich der Bezeichner für Haltestellen und Linien geachtet. Die Ordnung der Folge wird im mathematischen Modell des Fahrplans erörtert.

Nachdem die Voraussetzungen und der Ziele der einzelnen Modelle kurz umrissen worden sind, werden im Folgenden detaillierte Beschreibungen dieser Abstraktionen vorgestellt.

5.3 Mathematisches Modell der Haltestellen

Es werden zunächst alle Kriterien aufgelistet und wie sie die Umgebung einer Haltestelle oder diese Haltestelle auch direkt beeinflussen.

Zum Schluss wird, darauf aufbauend, eine Zielfunktion vorgestellt. Mit selbiger kann ein Gesamtzielwert der Haltestellen ermittelt werden.

5.3.1 Verwendete Symbole

r_{min} Minimaler Einzugsradius

r_{max} Maximaler Einzugsradius

$r_{verbindung}$ Distanz innerhalb welcher zwei Haltestellen potentiell eine Verbindung im Wegenetzgraph haben.

A Gesamtfläche des zu betrachtenden Raumes

a_{xy} Maß der Flächenabdeckung an der Position x,y nach Abzug aller in der Umgebung des Punktes liegenden Haltestellen. Für jede Haltestelle, für welche der Punkt innerhalb des minimalen Einzugsgebietes r_{min} liegt, wird 1 abgezogen. Innerhalb des maximalen Einzugsgebietes r_{max} (und ausserhalb des minimalen Einzugsgebietes) $0,5$. Ausserhalb des maximalen Einzugsgebietes wird kein Betrag abgezogen.

$a_{xy}^{>}$ Wie a_{xy}, jedoch nur Werte ≥ 0, also keine Überversorgung.

$$a_{xy}^{>} := \begin{cases} a_{xy} & , wenn\, a_{xy} > 0 \\ 0 & , sonst \end{cases}$$

$a_{xy}^{<}$ Wie a_{xy}, jedoch nur Werte ≤ 0, also keine Unterversorgung.

$$a_{xy}^{<} := \begin{cases} a_{xy} & , wenn\, a_{xy} < 0 \\ 0 & , sonst \end{cases}$$

b_{xy} Anzahl der Bevölkerung an der Position x,y nach Abzug aller einwirkenden Haltestellen. Für jede Haltestelle, für welche der Punkt innerhalb des minimalen Einzugsgebietes r_{min} liegt, wird die Bevölkerung b_{xy} ohne jeglichen Einfluss abgezogen. Innerhalb des maximalen Einzugsgebietes r_{max} (und ausserhalb des minimalen Einzugsgebietes) die Hälfte der Bevölkerung subtrahiert. Ausserhalb des maximalen Einzugsgebietes wird kein Betrag abgezogen.

$b_{xy}^{>}$ Wie b_{xy}, jedoch nur Werte ≥ 0, also keine Überversorgung.

$$b_{xy}^{>} := \begin{cases} b_{xy} & , wenn\, b_{xy} > 0 \\ 0 & , sonst \end{cases}$$

$b_{xy}^{<}$ Wie b_{xy}, jedoch nur Werte ≤ 0, also keine Unterversorgung.

$$b_{xy}^{<} := \begin{cases} b_{xy} & , wenn\, b_{xy} < 0 \\ 0 & , sonst \end{cases}$$

F Fahrtwunschmatrix, wobei die Einträge die Anzahl der Bevölkerung darstellt, welche von einer Haltestelle zu einer anderen reisen wollen. Auf der Diagonalen befindet sich in jedem Eintrag der Wert -1.

F_{peak} Extremwert der Fahrtwunschmatrix.

$$\max\{F_{i,j} : i, j \in R\}$$

F_{ij} Fahrtwunsch von der Haltestelle i zur Haltestelle j. Der Betrag entspricht der Anzahl der Personen welche von i nach j reisen wollen.

F_{Punkt} Fahrtwünsche von dem $Punkt$ aus, zu allen anderen. Diese Punkte entsprechen entweder den real existierenden Haltestellen oder aber den neu berechneten, je nach Kontext.

H Menge der (berechneten) Haltestellen inklusive der Pflichthaltestellen.

H_N Anzahl der (berechneten) Haltestellen inklusive der Pflichthaltestellen.

H_{Punkt} Einzelne Haltestelle$\in Strassenverlauf$.

$H^{Pflicht}$ Menge der Pflichthaltestellen.

$H_N^{Pflicht}$ Anzahl der Pflichthaltestellen.

$H_{Punkt}^{Pflicht}$ Einzelne Pflichthaltestelle$\in Strassenverlauf$.

H_{min} Untergrenze der Haltestellenanzahl. Wird in 5.3.2.5, Seite 32 näher bestimmt.

H_{max} Obergrenze der Haltestellenanzahl. Wird in 5.3.2.5, Seite 32 näher bestimmt.

$d_{a,b}$ Entfernung von Punkt a zu Punkt b.

k_{HCR} Eine Konstante, welche als Faktor für Luftlinienabschätzungen herangezogen wird. Stammt aus Erfahrungswerten der HCR Straßenbahn Herne Castrop-Rauxel GmbH. Entspricht $0,6$.

5.3.2 Kriterien bei der Beurteilung der Güte von Haltestellenlagen

Im Folgenden wird immer von Umkreis oder Radius um eine Haltestelle gesprochen. Dies kann durch einen Kreis, der um die Haltestelle gezogen wird, veranschaulicht werden. Realistischer ist jedoch die Vorstellung eines Polygons, da durch die Berücksichtigung von Realwegen, sehr unterschiedliche Formen auftreten können.

5.3.2.1 Flächendeckung

Die Flächendeckung kann so veranschaulicht werden, dass jede Haltestelle, die Umgebung in einem bestimmten Umkreis eingraut. Werden Gebiete öfters bestrichen, so wird der Grauton dunkler. Das Gebiet ist überversorgt. Bleiben weiße Flächen zurück, so ist das Gebiet unterversorgt.

Eine optimale Flächendeckung (hellster Grauton, keine weißen Flächen) ist also erreicht bei

$$\sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} a_{xy}^{>} = 0$$

$$\sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} a_{xy}^{<} = 0$$

5.3.2.2 Bevölkerungsabdeckung

Die Bevölkerung in einem Gebiet kann anschaulich als eine Säule dargestellt werden, die in dem Gebiet entsprechend der Bevölkerungsgröße steht. Dabei ergibt sich eine Hügellandschaft, welche über das Zielgebiet verteilt ist. Jede Haltestelle reduziert diesen Hügel um einen bestimmten Betrag. Ziehen mehrere Haltestellen von der gleichen Säule einen Betrag ab, so ist die Entfernung der betroffenen Haltestellen von der Säule zu berücksichtigen und dann anteilig umzurechnen[1]. Ist noch ein Hügel vorhanden, so ist das Gebiet unterversorgt. Ist das Gebiet zu sehr abgetragen, so ist das Gebiet überversorgt. Eine anschauliche Darstellung befindet sich in Abbildung 5.1.

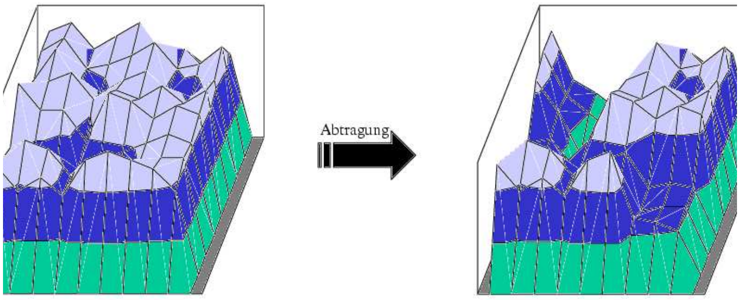
Eine optimale Bevölkerungsabdeckung (alle Gebiete bis auf nN abgetragen) ist also erreicht bei

$$\sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} b_{xy}^{>} = 0$$

$$\sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} b_{xy}^{<} = 0$$

[1]Möglich z.B. mittels Interpolation. Hier jedoch nur Unterscheidung zwischen Lage innerhalb des minimalen Einzugsgebietes r_{min}, des maximalen Einzugsgebietes r_{max} oder ausserhalb selbiger. Dabei verteilt sich die Bevölkerung nicht auf die einwirkenden Haltestellen, sondern jede dieser Haltestellen versorgt die durch das Einzugsgebiet gegebene Bevölkerungsanzahl komplett. Hierduch kommt es dann zu Fehlern, welche jedoch nicht wieter ins Gewicht fallen.

Abbildung 5.1: Bevölkerungsverteilung
Abtragung durch platzierte Haltestellen



5.3.2.3 Arbeitsplatzdeckung

Vorhandene Arbeitsplätze oder auch Freizeitstätten haben einen bestimmten Grundbedarf an Personen. Bei den Arbeitsplätzen ist klar definiert, welche Person von ihrem Wohnort zu ihrem Arbeitsplatz — und umgekehrt — fahren. Bei den Freizeitstätten handelt es sich um eine generelle Senke, welche Personen aufnimmt. Aufgrund ihrer wechselnden Eigenschaft, bleiben die Quellen unbekannt.

Das Pendeln zwischen Wohnort und Arbeitsplatz ist zu einem Großteil in den Fahrtwünschen berücksichtigt. Daher wird die Modellierung dort vorgenommen.

Die Freizeitstätten können als Bedarf an der Freizeitstätte mit einem gleichverteiltem Fahrtwunsch von allen Wohnorten angesehen werden. Dabei ist unerheblich, ob bestimmte Freizeitstätten nur an einzelnen Tagen eine Attraktivität aufweisen (z.B. saisonal bedingte Öffnungszeiten) oder generell von Besuchern aufgesucht werden. Eine Haltestelle, muss in jedem Fall in der näheren Umgebung gesetzt sein. Ein saisonaler Unterschied kann erst in der Linienplanung sinnvoll berücksichtigt werden.

5.3.2.4 Fahrtwunschberücksichtigung

Die Fahrtwünsche können ähnlich veranschaulicht werden, wie bereits die Bevölkerungsdeckung, mit dem Unterschied, dass die Säulen in diesem Fall die Fahrtwünsche repräsentieren, also Orte, von denen Personen abfahren und Orte an denen Personen ankommen. Die Höhe gibt darüber Auskunft, wie viele Personen diesen Ort aufsuchen wollen.

Die vorliegenden Fahrtwünsche geben nur punktuelle Aussagen wieder. Aus diesem Grund müssen neu berechnete Haltestellen mit Fahrtwünschen abgeschätzt werden. Dabei werden selbige mittels einer trivialen Interpolation auf die neuen Haltestellen umgelegt :

$$Punkt \in H :$$

$$Einfluss_{Punkt} := \begin{cases} 1 - \frac{d_{Haltstelle,Punkt}}{r_{max}} & , wenn\, d_{Haltestelle,Punkt} < r_{max} \\ 0 & , sonst \end{cases}$$

Bei diesem Wert handelt es sich um einen Koeffizienten, welcher den Einfluss jedes real existierenden Fahrtwunschpunktes *Punkt* auf die einzelnen (berechneten) Haltestellen $Haltestelle$ bestimmt.

5.3.2.5 Abschätzung der Grenzen der Haltestellenanzahl

Die optimale Anzahl der Haltestellen H_N bewegt sich in den Grenzen von H_{min} und H_{max}. Sind zu wenig Haltestellen verteilt, so kann die minimale Abdeckung nicht erreicht werden. Es liegt dann eine Unterversorgung vor. Sind zu viele Haltestellen vorhanden, so werden einzelne Gebiete überversorgt.

H_N lässt sich wie folgt eingrenzen:

$$H_{min} := \frac{\min(B, \sum_{i=Norden}^{Sueden} \sum_{j=Westen}^{Osten} F_{ij})}{\max(B_{peak}, F_{peak})}$$

Dieses entspricht der Annahme, dass die Bevölkerung bzw. die Fahrtwünsche komplett gebündelt, also mit maximaler Dichte, vorliegen.

$$H_{max} := \frac{A}{\pi * r_{min}^2}$$

Damit entspricht die maximale Anzahl H_{max} der maximal möglichen Bepackung über die gesamte Fläche A. Hierbei wird die Fläche, welche die einzelnen Haltestellen einnehmen, mit einer Kreisfläche abgeschätzt.

5.3.2.6 Entfernung der Haltestellen zueinander

Ein weiteres Problem entsteht, da die Entfernung einzelner oder auch aller Haltestellen zueinander so groß sein kann, dass die Haltestellen nicht innerhalb des Verbindungsradius $r_{verbindung}$ liegen. Hierdurch ist der entstehende Wegenetzgraph nicht zusammenhängend.

5.3.3 Weitere Einflüsse auf die Haltestellenlagen

5.3.3.1 Straßennetz

Das Straßennetz ist die Grundlage der Berechnung der Haltestellenlagen. Je detaillierter die Daten sind, desto besser können die Haltestellen platziert werden. Daten, welche in ein derartiges Straßennetz einfließen, sind:

1. Straßenverlauf

   Ist durch GRASS vorgegeben und wird entsprechend berücksichtigt. Dabei müssen zusätzliche Kreuzungen berechnet werden, um diese zu einem späteren Zeitpunkt als mögliche Haltestellenpositionen einzubeziehen oder auch um den Realweg abbilden zu können.

2. Geschwindigkeitsvorgaben

   Sind nicht in den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

3. Breite der Straßen

   Sind nicht in den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

4. Darf an den Straßen gehalten werden?

   Sind nicht in den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

5. Wendemöglichkeiten

   Sind nicht in den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

6. Parkmöglichkeiten für Busse

   Sind nicht unter den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

7. Unterführungen mit Höhenangaben

   Sind nicht unter den vorhandenen Daten gegeben und werden daher auch nicht berücksichtigt.

8. Kreuzungen samt zugehörigem Manövrierradius

   Kreuzungen werden, wie unter 1. beschrieben, berechnet. Manövrierradien sind nicht in den vorhandenen Daten und werden daher auch nicht berücksichtigt.

5.3.3.2 Besondere Kriterien für Örtlichkeiten

Dies sind Kriterien, welche durch besondere Umstände existieren. Gründe hierfür können sein:

1. Orte mit besonderer Attraktivität
   Dies sind zum Beispiel Schulen und Parks.

2. Orte mit definierter Ablehnung
   zum Beispiel durch Anwohner in Wohnparks.

5.3.4 Modell des Systems Haltestellen

Viele der oben aufgeführten Einflussgrößen sind nicht bekannt (Straßennetz: 2, 3, 4, 5, 6, 7, 8; Besondere Kriterien für Örtlichkeiten: 1, 2). Daher fließen sie auch nicht in die Betrachtung ein.

Andere Daten (Fahrtwünsche, Bevölkerungsaufkommen) liegen nur sehr ungenau bzw. nur für bestimmte Punkte und Regionen vor und müssen daher anderweitig abgeschätzt werden. In die Bevölkerung mussten zudem die Fahrtwünsche eingerechnet werden, um existierende Fahrtwunschpunkte erkennen zu können. Dies ist notwendig, da einige Fahrtwunschpunkte außerhalb des Bereiches liegen, in dem Bevölkerungszahlen vorhanden sind.

Alle anderen Daten (Straßenverläufe, Kreuzungen) liegen direkt vor oder können aus vorliegenden Daten berechnet werden.

Aus den gesamten oben aufgeführten Teil-Modellen lässt sich leicht ersehen, dass das anzustrebende Gesamt-Modell der Haltestelle folgenden Aufbau haben muss, um sowohl bevölkerungs-, als auch flächendeckend zu optimieren. Dabei sind

$$A^{>} := \sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} a_{xy}^{>}$$

die Summe der nicht versorgten Fläche,

$$A^{<} := \left| \sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} a_{xy}^{<} \right|$$

die Summe der mehrfach versorgten Fläche,

$$B^{>} := \sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} b_{xy}^{>}$$

die Summe der nicht versorgten Bevölkerung und

$$B^{<} := \left| \sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} b_{xy}^{<} \right|$$

die Summe der mehrfach versorgten Bevölkerung.
Als eine den Optimalitätswert berechnende Zielfunktion kann folgendes angenommen werden:

$$Z := A^{>} + A^{<} + B^{>} + B^{<},\ Z \in \mathbb{R}^{\geq 0}$$

Dabei ist das gesuchte Optimum bei

$$Z = 0$$

Um einen größeren Einfluss auf die Gewichtung der Teilziele bei dieser Optimierung zu legen, lässt sich Z auch wie folgt definieren:

$$Z := k_0 * A^{>} + k_1 * A^{<} + k_2 * B^{>} + k_3 * B^{<},\ Z \in \mathbb{R}^{\geq 0},\ n \in \{0, ..., 3\},\ k_n \in [0; 1]$$

Dabei sind die $k_0, ..., k_3$ Koeffizienten, welche die Beeinflussung der Teilziele auf den Optimalitätswert steuern. So lassen sich gezielt einzelne Teilziele als einzige Ziele herausheben. Im Extremfall, falls $k_0 = k_1 = k_2 = k_3 = 0$, ist der Optimalitätswert von 0 erreicht, aber offensichtlich nicht aussagekräftig. Im Fall von $k_0 = k_1 = k_2 = k_3$ ist die Höhe des Betrages der k_n, $n \in \{0, ...3\}$ unerheblich.

5.3.4.1 Wegenetzgraph

Zuletzt werden die Haltestellen noch in einem ungerichteten, kanten-bewerteten Graphen G_W, dem Wegenetzgraph (siehe Definition in Kapitel 5.2.2, Seite 26 f.), vermerkt. Dieser besitzt eine Auswahl an Kanten, so dass G_W eine einzige, und somit maximale, Zusammenhangskomponente ist.

5.3.4.2 Fahrtwunschmatrix

Abschließend werden die Fahrtwünsche von Haltestelle zu Haltestelle ermittelt und in der Matrix G_W (siehe Definition in Kapitel 5.2.2, Seite 26 f.) vermerkt.

5.4 Mathematisches Modell des Moduls Linienplan

Zuerst wird die Problematik der Linienplanerstellung aufgezeigt. Anschließend wird der allgemeine Ablauf beschrieben, damit ein grober Überblick über die Vorgehensweise gegeben ist. Darauf wird das mathematische Modell vorgestellt auf dessen Grundlage der Algorithmus arbeitet und dessen Bedingungen bzw. Vorgaben einzuhalten sind. Im letzten Teil dieses Kapitels wird dann in mehreren Abschnitten der eigentliche Algorithmus zur Linienplanberechnung beschrieben.

5.4.1 Problemstellung

Nachdem durch das Haltestellen-Modul eine Menge von Haltestellen erzeugt und diese im Wegenetzgraphen eingetragen wurden, gilt es einen Linienplan darauf zu erzeugen. Dieser Linienplan besteht aus einer Menge von Linien, die jeweils eine bestimmte Menge von Haltestellen in einer Folge sukzessiv miteinander zu einer Buslinie verbinden. Durch diesen Linienplan soll es ermöglicht werden, dass jeder Fahrtwunsch von einer Haltestelle zu einer anderen erfüllt wird. Dabei kann die sich aus dem Wunsch ergebende Fahrt entweder durch eine Direktfahrt ohne einen Umstieg realisiert werden oder aber es ist notwendig an bestimmten Stellen einen Linienwechsel durchzuführen, um das Ziel zu erreichen.

Das Ziel des Linienplan-Moduls ist es, solch einen Linienplan zu erzeugen, der den durchschnittlichen Gesamtreisezeitaufwand pro Person minimiert und der so möglichst viele Fahrtwünsche ohne Umsteigevorgang realisiert, also eine Direktfahrer-Maximierung erreicht.

Zu diesem Zwecke wurde das Verfahren von Sonntag mit spezifischen Abwandlungen implementiert.

5.4.2 Allgemeiner Ablauf des Linien-Planungsverfahrens

Das Verfahren von Sonntag teilt sich in drei aufeinander folgende Phasen auf: Aufbereitungs-, Verknüpfungs- und Bewertungsphase, wobei die Verknüpfungsphase noch in weitere Stufen unterteilt ist.

Am Anfang werden in der Aufbereitungsphase die vom Haltestellen-Modul übergebenen Daten ausgewertet, um so ein Modell (Netzplangraph) im Computer zu erschaffen, auf dessen Basis die Berechnungen für die Erstellung eines Linienplans erfolgen können. An diese Phase schließt sich die Verknüpfungsphase an, in der in aufeinanderfolgenden Stufen der eigentliche Linienplan-Algorithmus arbeitet und dabei versucht ein möglichst optimales Liniennetz aufzubauen. In der letzten Phase, der Bewertungsphase, wird sowohl die Güte des sich ergebenden Liniennetzwerks berechnet, als auch der vom Fahrplan-Modul benötigte Netz der Transportverbindungen aufgebaut. Das Verfahren ist so sequentiell aufgebaut, dass sich die Eingangsinformationen einer Phase jeweils aus den Ausgangsinformationen der vorangehenden Phase ergeben.

5.4.3 Mathematisches Modell

Als Modell für das zu behandelnde Straßennetz wird ein ungerichteter, knoten- und kantenbewerteter, zusammenhängender Graph $G_W = (H_W, U_W)$ verwendet. Dieser modelliert mit den Knotenmenge H_W die Haltestellen und mit der Kantenmenge U_W werden diejenigen Wegstrecken repräsentiert, die die jeweiligen Haltestellen verbinden. An jedem Knoten ist dabei die jeweilige ID der zugehörigen Haltestelle gespeichert, während an jeder Kante die Entfernung zwischen den durch diese Kante verbundenen Haltestellen als Information abgelegt ist. Dieser hier geforderte Graph entspricht dem im Schnittstellenkapitel definierten Wegenetzgraph.

Eine Untermenge des Wegenetzgraphen bildet der ungerichtete, kanten- und knotenbewertete Netzplangraph $G_P = (H_P, U_P, f_h, 2, f_u, 2)$, in dem nur noch diejenigen Haltestellen eingetragen sind, an denen Fahrgäste zwischen Linien umsteigen können oder ihre Fahrt anfangen oder enden können (Endknoten). Demnach ist also $H_P \subseteq H_W$ und $U_P \subseteq U_W$. Für diese Untermengenbildung ist eine Reduktion der Knotenmenge des Wegenetzgraphen und eine Umverteilung der Nachfrage auf die verbleibenden Knoten notwendig.

Die Funktion f_h liefert für jeden Knoten ein Tupel bestehend aus der zugehörigen Haltestellen ID und eine Liste von Werten zurück. Die Länge und die Zuordnung der Werte dieser Liste ist abhängig von den inzidenten Kanten an diesem Knoten. So speichert jeder Wert in der Liste für jede inzidente Kante ab, wie viele Fahrtwünsche an diesem Knoten über die jeweilige Kante enden.

Die Funktion f_u liefert ebenfalls ein Tupel zurück, wobei der erste Wert die Entfernung angibt und der zweite den Verkehrsstrom auf dieser Kante.

Zwischen allen Knoten i und j des Graphen G_P besteht eine Verkehrsnachfrage, die angibt wie viele Personen von der Haltestelle mit der ID i zur Haltestelle mit der ID j in dem betrachteten Zeitraum kommen möchten. Diese Information liegt in einer Verkehrsbedarfsmatrix F an der Position $f_{i,j}$ bzw. auch an der Position $f_{j,i}$ vor, da von symmetrischen Verkehrsbeziehungen ausgegangen wird und somit dieselbe Nachfragegröße besteht von i nach j zu kommen wie von j nach i. Die Position $f_{i,i}$ auf der Diagonalen der Matrix wird mit dem Wert 0 belegt, da der Binnenverkehr innerhalb eines Knotens nicht betrachtet wird.

Eine Linie $K^{(y)}_{M(y)}$ mit Liniennummer y und Länge $M(y)$ im Graphen G_P ist definiert als eine Knotenfolge, wobei für jeden Knoten gilt: $(u^{(y)}_x, u^{(y)}_{x+1}) \in U_P$ und $u^{(y)}_x \in H_P$.

Die Knotenfolge definiert also einen Pfad auf dem Graph G_P vom Anfangsknoten $u^{(y)}_1$ zum Endknoten $u^{(y)}_{M(y)}$.

$$K^{(y)}_{M(y)} = \{u^{(y)}_1, u^{(y)}_2, ..., u^{(y)}_{M(y)}\}$$

Die Länge $D^{(y)}_{M(y)}$errechnet sich über die Länge der Kanten zwischen den Knoten:

$$D^{(y)}_{M(y)} = \sum_{1}^{M(y)-1} d(u^{(y)}_x, u^{(y)}_{x+1}) = d(u^{(y)}_1, u^{(y)}_2) + ... + d(u^{(y)}_{M(y)-1}, u^{(y)}_{M(y)})$$

Die Zahl $M(x)$ gibt dabei die Gesamtanzahl der Knoten an und $d(u_x^{(y)}, u_{x+1}^{(y)})$ die Länge der Kante zwischen den betrachteten Knoten.

Ein Linienplan $L_n^{G_P}$ bestehend aus n Linien bezüglich eines Graphen G_P wird beschrieben als eine Folgenschar:

$$L_n^{G_P} = \{K_{M(1)}^{(1)}, K_{M(2)}^{(2)}, ..., K_{M(n)}^{(n)}\}$$

Für die Gesamtlänge $D_{L_n^{G_P}}$ gilt:

$$D_{L_n^G P} = \sum_{y=1}^{n} D_{M(y)}^{(y)} = D_{M(1)}^{(1)} + D_{M(2)}^{(2)} + \ldots + D_{M(n)}^{(n)}$$

Zielfunktion: Derjenige Linienplan $L_n^{G_P}$ ist zu bestimmen, für den der durchschnittliche Reiseaufwand pro Fahrtwunsch minimiert wird:

$$R_{L_n^{G_P}} = \frac{\sum_{j=1}^{n-1} \sum_{j=i+1}^{n} f_{ij} R_{L_n^{G_N}}(i,j)}{\sum_{j=1}^{n-1} \sum_{j=i+1}^{n} f_{ij}}$$

wobei $R_{L_n^{G_P}}(i,j)$ als eine Funktion anzusehen ist, die den Reisezeitaufwand (Fahrzeit +Umsteigeaufwand) berechnet, um von i nach j zu kommen, auf der Grundlage des Linienplan $L_n^{G_P}$.

Nebenbedingungen:

1. Jede Haltestelle ist durch eine Kante mit mindestens einer anderen Haltestelle verbunden, so dass es keinen isolierten Knoten gibt.

2. Jeder Knoten ist mindestens ein Mal im Linienplan enthalten. So wird sichergestellt, dass jede Haltestelle durch mindestens eine Linie bedient wird.

3. Die Länge einer Linie $D_{M(y)}^{(y)}$ darf eine vorgegebene, maximale Linienlänge nicht überschreiten.

4. Die Gesamtlänge $D_{L_n^{G_P}}$ aller Linien ist beschränkt durch eine betrieblich vorgegebene Gesamtlinienlänge, die nicht überschritten werden darf.

5. Eine Linie darf in ihrer Knotenfolge keinen Zyklus enthalten.

6. Linien, die der Menge der Pflichtlinien angehören, dürfen nicht verändert werden.

Als Ergebnis der Linienplanberechnung liegt dann zum einen der Linienplan $L_n^{G_P}$ vor und zum anderen das im Schnittstellen Kapitel unter 5.2.2 geforderte Netz der Transportverbindungen G_N. In diesem gerichteten, knoten- und kantenbewerteten Multigraphen wird der Linienplan und die Verkehrsströme auf folgende Art zusammengebracht: Jede Kante im Multigraphen speichert mit einem Wert die Liniennummer ab und repräsentiert so ein Teilstück einer Linie im Linienplan. Desweiteren wird an jeder Kante die Information abgelegt, wie viele Fahrgäste auf diesem Teilstück in diesem Bus fahren werden, wenn von dem vorgegebenen Linienplan und dem erwarteten Verkehrsaufkommen ausgegangen wird. Für jede Kante wird nun aber nicht mehr die Entfernung, sondern die Fahrzeit gespeichert, die ein Bus benötigt, um von der einen verbundenen Haltestelle zur anderen zu fahren.

Im folgenden Teil wird der eingesetzte Algorithmus von Sonntag beschrieben, der die einzelnen Linien und somit den sich ergebenden Linienplan berechnet.

5.4.3.1 Aufbereitungsphase

Diese Phase ist dafür zuständig, dass sowohl zum einen der eingehende Wegenetzgraph in den internen Netzplangraphen umgewandelt wird, als auch zum anderen die eingehenden Verkehrsbedarfswerte aus der Quelle-Ziel-Matrix auf die kürzesten Wege umgelegt werden.

Umwandeln des Wegenetzgraphen in den Netzplangraph

In diesem Schritt wird der Wegenetzgraph in den linienplaninternen Netzplangraph überführt. Dabei wird die bereits erwähnte Reduktion durchgeführt, die dafür sorgt, dass Knoten, an denen kein Umstieg stattfindet und die nicht als Endknoten gekennzeichnet sind, nicht in den Netzplangraphen übernommen werden. An dieser Stelle erfolgt keine genauere Beschreibung dieses Vorgangs, da dieses Verfahren nicht implementiert wurde (siehe Kapitel 7).

Umlegung der Nachfrage auf kürzeste Wege

Ziel dieses Umlageverfahren ist es, die Verkehrsströme eines idealisierten Liniennetzwerkes dadurch zu bestimmen, dass alle Verkehrsbedarfswerte (Nachfrage) auf den jeweils kürzesten Wege der jeweiligen Verkehrsbeziehung umgelegt werden. Auf diese Weise ergibt sich so der Nachfragestrom im Graphen, an den sich die zu berechnenden Linien in der späteren Verknüpfungsphase möglichst optimal anzupassen versuchen.

Dabei wird von der idealisierten Annahme ausgegangen, dass alle Nachfragewünsche immer auf dem direkten und kürzesten Weg bedient werden könnten. Daher wird der komplette Wert $f_{i,j}$ in einem Feld der Verkehrsbedarfsmatrix F, der dem Verkehrsbedarfswert von einer Quelle i zu einem Ziel j entspricht, jeweils allen Kanten, die auf dem kürzesten Weg zwischen den beiden Haltestellen liegen, zugewiesen. Jeder, auf einem optimalen Weg liegenden Kante im Netzplangraphen wird dann der komplette Nachfragewert der jeweiligen Haltestellenkombination zugewiesen. Außerdem wird dieser Nachfragewert als entstehender bzw. versackender Strom in der zugehörigen Kante des jeweiligen Start- und Endknoten abgespeichert. Auf diese Weise wird der entsprechende aktuelle Nachfragewert zum bisherigen Verkehrsstrom bzw. versackenden Strom hinzuaddieren. So überlagern sich die verschiedenen Nachfragewünsche der Verkehrsbedarfsmatrix, wenn sie auf die "wegkürzesten" Kanten umgelegt werden. Wenn alle Felder der Verkehrsbedarfsmatrix auf diese Art abgearbeitet worden sind, ist so das gesamte Nachfragepotential zwischen den verschiedenen Quellen und Zielen auf die Kanten des Netzplangraphen verteilt worden.

An dieser Stelle ist es auch möglich, den Nachfragewert nicht komplett auf den kürzesten Weg zu legen, sondern den jeweiligen Wert im Verhältnis zu seiner Abweichung zur optimalen Strecke auf die jeweiligen Alternativ-Wege zu verteilen. Zu diesem Zweck könnte ein Alternativrouten Verfahren verwendet werden, dass neben der kürzesten Verbindung auch noch andere Routen berechnet, die eine gewisse Abweichung vom optimalen nicht überschreiten. Abhängig von dieser Abweichung zusammen mit einer Gewichtung könnte dann der Nachfragewert dem Kirchhoff'schen Stromgesetz für parallelgeschaltete Widerstände folgend auf die einzelnen alternativen Pfade verteilt werden. Diese Möglichkeit wird an dieser Stelle hier nicht näher ausgeführt, da dieses Verfahren nicht realisiert worden ist.

Am Ende dieser Phase sind die Informationen des Wegenetzgraph und der Verkehrsbedarfsmatrix so weit in den Netzplangraphen integriert, dass die eigentliche Berechnung des Linienplans auf dem Netzplangraphen erfolgen kann.

5.4.3.2 Verknüpfungsphase

In dieser zweiten Phase erfolgt die eigentliche Erzeugung des Linienplans. Der hier vorgestellte Algorithmus von Sonntag verwendet dafür das Progressionsverfahren, bei dem, ausgehend von einem Nullnetz, also einem Netz ohne Verbindungen zwischen Haltestellen, nach und nach ein Liniennetz zusammengesetzt wird. Außerdem verfolgt das Verfahren den Ansatz der Nachfrageorientierung. Dies bedeutet, dass versucht wird, dadurch ein Optimum an Attraktivität zu erreichen, dass die Buslinien möglichst stark geforderte Strecken befahren, wobei aber gleichzeitig darauf geachtet werden muss, dass eine betrieblich vorgegebenen Grenze eingehalten wird.

Das Verfahren von Sonntag basiert auf dem Prinzip, dass aus vielen, kleinen Teilstücken, den Basislinien, durch Wahl von stark befahrenen Teilstücke, eine Auswahl von Kantenfolgen entsteht die im Linienplan enthalten sein müssen. Das Hinzufügen von Basislinien zu diese Menge wird so lange fortgeführt bis die betriebliche Vorgabe der Gesamtlänge erreicht ist. Anschließend wird in weiteren Schritten versucht, eine Verbesserung dadurch zu erreichen, indem zugelassen wird, dass Linien verknüpft werden, sich Linien überdecken, getroffene Verbindungen wieder aufgebrochen werden u.ä. Zu diesem Zweck ist das Verfahren in 5 aufeinander folgende Stufen unterteilt, die sequentiell nacheinander abgearbeitet werden und innerhalb deren immer kompliziertere und aufwendigere Verknüpfungsverfahren möglich sind. Im folgenden werden die jeweiligen Stufen einzeln für sich dargestellt.

Basislinienwahl

In dieser ersten Stufe wird der Basislinienplan erzeugt, der sich nur aus Basislinien zusammensetzt. Eine Basislinie ist dabei als eine Linie mit zwei Kanten und drei Knoten zu verstehen, also ein drei-elementiger Pfad mit einem Start-, einem End- sowie einem Zentralknoten, die jeweils durch eine Kante verbunden sind. Diese Basiseinheit stellt nach Sonntag den besten Kompromiss dar, um zum einen die wesentlichen Ströme im Netzwerk nachzubilden, aber um auch zum anderen dabei die Kombinationsmöglichkeiten einzuschränken.

Im ersten Schritt wird für den gesamte Netzplangraph für alle Basislinien ihre jeweilige Stärke berechnet: Hierzu wird der jeweilige Verkehrsstrom auf den beiden Kanten aufsummiert. Anschließend wird diese Basislinie entsprechend der berechneten Stärke in eine Rangfolgenliste einsortiert. Auf diese Weise können dann die Basislinien fallend sortiert nach der Stärke der auf ihnen enthaltenen Nachfrage abgearbeitet werden.

Für den zweiten Schritt, das Auswählen von Basislinen, wurden zwei verschiedene Algorithmen implementiert. Der Erste hält sich an die Vorgabe von Sonntag, die besagt, dass alle Kanten überdeckt werden müssen. Der Zweite hat zur Bedingung, dass alle Knoten mittels Basislinien erreichbar sind.

Erster Algorithmus: Um sicherzustellen, dass alle Kanten angeschlossen werden und somit jeder Knoten im späteren Linienplan enthalten ist (siehe Nebenbedingungen im Kapitel 5.4.3), werden in einem ersten Durchlauf Basislinien gewählt, die noch nicht angeschlossene Kanten überdecken. Dabei werden die Basislinien entsprechend der vorher aufgestellten Rangfolgenliste abgearbeitet. So lange es noch eine Kante gibt, die angeschlossen werden kann und die Basislinien-Liste noch Elemente enthält, wird dieser Vorgang wiederholt. Es werden jedoch nur die Kanten gesucht deren Überdeckungsgrad durch gewählte Basislinien dem des Durchlaufs entsprechen. Dabei hängt die Entscheidung, ob eine Kante angeschlossen werden darf oder nicht, davon ab, ob durch die Hinzunahme der betreffenden Basislinie die Länge aller ausgewählten Basislinien die Gesamtlinienlänge nicht überschritten wird (Nebenbedingung). Hierbei ist jedoch das Kriterium des Anschlusses aller Knoten stärker, als die Gesamtlinienlänge. Daher wird die Gesamtlinienlänge so lange erhöht, bis es möglich ist, alle Knoten mittels Basislinien anzuschließen.

Zweiter Algorithmus: Zur Auswahl der Basislinien wird hier der Algorithmus von Kruskal für minimale Spannbäume benutzt. Jeder Knoten befindet sich anfangs in einer eigenen Menge. Im ersten Durchlauf durch die sortierte Menge von Basislinien werden nur Kanten ausgewählt, deren drei Knoten in drei verschiedenen Mengen enthalten sind. Bei einer Auswahl der Basislinie werden die Mengen, in denen sich die Haltestellen befinden, miteinander zu einer Menge verschmolzen. Befinden sich nach dem Durchlauf nicht alle Haltestellen in einer Menge, werden in einem weitem Durchlauf Basislinien ausgewählt, in der zwei der drei Haltestellen in verschiedenen Mengen enthalten sind. In einem letztem Durchlauf werden solange Basislinien aus der sortierten Liste ausgewählt, bis die Gesamtlinienlänge erreicht ist. Auch hier gilt, wie vorhin beim ersten Algorithmus, dass die Gesamtlinienlänge erhöht wird, wenn sie nicht ausreichend ist.

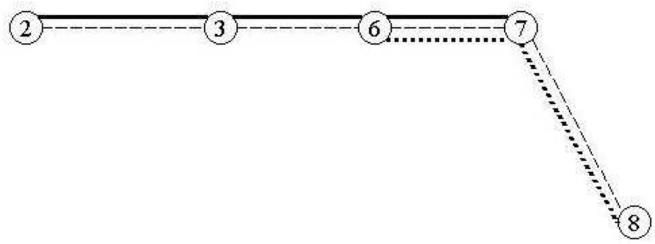
Eindeutige Linienwahl

In dieser Stufe werden die gewählten Basislinien zu Linien zusammengefügt. Dabei sind aber nur eindeutige Verknüpfungen zwischen zwei Basislinien erlaubt. Das bedeutet, dass zwei Basislinien nur dann zu einer Linie verknüpft werden dürfen, wenn sich in einer Kante nur zwei Basislinien überlagern, weil es dann nämlich keine Interpretationsmöglichkeit gibt, sondern die Wahl eindeutig ist. Es ist dabei auch darauf zu achten, dass die Endknoten der überlagerten Kante unterschiedlich sind bzw. sich keine Stichfahrten ergeben. Eine Stichfahrt würde dann entstehen, wenn ein Bus in eine Straße hineinfährt, dort wieder wendet und dieselbe Straße dann wieder zurückfährt.

Im Beispiel in der Abbildung 5.2 wird die durchgezogene Linie, welche die Knotenfolge (2,3,6,7) hat, eindeutig auf der Kante (6,7) mit der gestrichelten Basislinie verknüpft, welche die Knoten (6,7,9) verbindet.

Als Ergebnis dieser Stufe ist der Netzplangraph von einer Ansammlung von Linien durchzogen, die sich an ihren Enden gar nicht oder mehrfach mit anderen Linien überlagern. So entstehen mehrdeutige Verknüpfungsmöglichkeiten zwischen Linien, die in der nächsten Stufe bearbeitet werden.

Abbildung 5.2: Eindeutige Linienverknüpfung



Mehrdeutige Linienwahl

In dieser Stufe sind bereits alle eindeutigen Linienverknüpfungen vorher durchgeführt worden. Um nun bei den mehrdeutigen Verknüpfungsmöglichkeiten, die optimale Verknüpfung von Linien zu finden, wird versucht in jedem Durchlauf jede Linie mit jeder anderen zu verknüpfen. Wenn eine erlaubte Überlagerung der Endkanten vorliegt wird der sich aus dieser Verbindung ergebende Attraktivitätsfaktor berechnet. (Die Beschreibung dieser Attraktivitätsberechnung erfolgt im Absatz 5.4.3.2) Dabei ist jedoch zu beachten, dass eine Verknüpfung aber nur dann zustande kommen darf, wenn die nachfolgenden Nebenbedingungen eingehalten werden:

- auf dieser Verknüpfungskante ist nicht schon in einem vorherigen Schritt eine Verknüpfung durchgeführt worden
- durch diese Verknüpfung wird kein Zyklus gebildet
- die Gesamtlänge der resultierenden Linie überschreitet nicht die erlaubte maximale Länge einer Linie.

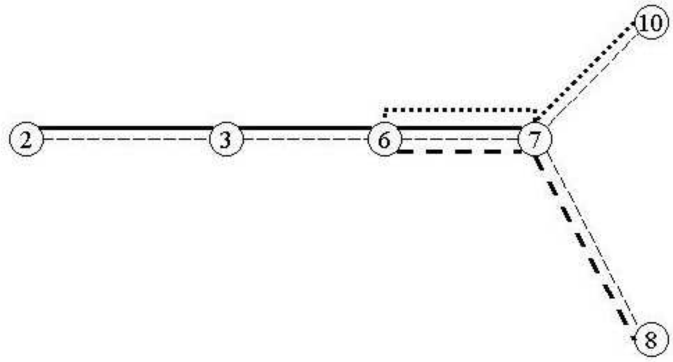
Auf diesem Wege kann der maximale Attraktivitätswert der erlaubten Verknüpfungen bestimmt werden, der sich aus dem Vergleich aller Werte der möglichen Verknüpfungen ergibt. Diese Verknüpfung wird ausgeführt und die entsprechende Kante, auf der verknüpft worden ist, wird für weitere Verknüpfungen gesperrt, so dass keine weitere Verknüpfung mehr auf dieser Kante stattfinden kann. Anschließend beginnt die Suche nach der nächsten optimalen Verknüpfung von vorne. Dies setzt sich so lange fort, bis keine Verknüpfung mehr gefunden wird, die einen Attraktivitätsgewinn erzielen würde.

In Abbildung 5.3 kann die durchgezogene Linie mit der Knotenfolge (2,3,6,7) auf der Kante zwischen Knoten 6 und 7 entweder mit der gestrichelten oder der gepunkteten Basislinie verknüpft werden.

Attraktivitätsberechnung

Diese Bewertung wird für die Bestimmung der optimalen mehrdeutigen Verknüpfung (5.4.3.2) und für die Auswahl der optimalen Aufbruchverknüpfung

Abbildung 5.3: Mehrdeutige Linienverknüpfung



(5.4.3.2) benötigt. Hierbei geht es um den Attraktivitätswert einer speziellen Verknüpfung. Um diese Berechnung durchzuführen, wird für alle durch diese Verknüpfung realisierten neuen Ströme zwischen i und j, für die nun kein Umstieg mehr nötig ist, eine Bewertung der Direktfahrerströme und der Indirektfahrerströme durchgeführt. Der sich so ergebende Faktor setzt sich zum einen aus einem Anteil von Direktfahrern, deren Anfangs- und Endhaltestelle auf der nun verknüpften Linie liegen, und zum anderen aus einem restlichen Bedarfswert, der auch noch durch diese Linie bedient wird, zusammen. Dieser restliche Wert setzt sich wiederum aus einem bestimmten Anteil Teilindirektfahrern (entweder der Anfang oder das Ziel der Fahrt liegen auf der Linie) und einem Anteil Vollindirektfahrern (sowohl Start, als auch Ende der Fahrt liegen außerhalb) zusammen. Nur der Anteil der Direktfahrer wird ungewichtet in die Attraktivität übernommen. Alle anderen Anteile, die die Linie auf sich vereinigt, werden mit dem Faktor $\frac{1}{4}$ gewichtet. Auf diese Weise wird der Tatsache Rechnung getragen, dass sich diese Anteile nicht zwingend auf diese Linie verteilen werden, sondern auch durch andere Linien realisiert werden können, die über diese Strecke führen. Am Ende der Berechnung steht ein Attraktivitätswert, der theoretisch angeben soll, wie viele Fahrten jetzt durch die Verknüpfung der beiden Linien realisiert werden konnten, ohne dass ein Umsteigen nötig ist.

Die Berechnung des Direktfahrer-Anteils D errechnet sich demnach dadurch, dass die Summe alle Direktfahrerwerte D_i^z bezüglich einer Knoten i für eine Linie z aufaddiert werden. (Wegen der symmetrischen Verkehrsbeziehung ist dies nur für eine Linie notwendig). Bei der Berechnung des Direktfahrerwerte D_i^z wird eine Haltestelle i in der Linie z mit jeder Haltestelle x der zu verknüpfenden Linie y in Bezug gebracht und dabei der entsprechende Eintrag in der Verkehrsbedarfsmatrix als realisierte Direktfahrt aufsummiert.

$$D_i^z = \sum_{x=1}^{M(y)} f_{i,x}$$

$$D = \sum_{x=1}^{M(z)} D_x^z$$

In der Abbildung 5.4 bedeutet dies, dass die Matrix entweder spalten- oder zeilenweise abgearbeitet wird. So errechnet sich der Wert $D_2^z = 0+20+10 = 30$.

Der Teilindirektfahreranteil T errechnet sich aus der Summe aller Teilindirektwerte T_i^z, dabei müssen diesmal aber beide Linie betrachtet werden, da hier keine Symmetrie mehr vorliegt wie bei der Berechnung der Direktfahrer. Für die Berechnung des Teilindirektwertes T_i^z wird der gesammelte Direktfahrerwert D_i^z vom versackenden Strom der jeweiligen Linienkante der Knoten i abgezogen und durch die Anzahl der Linien geteilt, die auf dieser Kante fahren.

$$T_i^z = \frac{(f_h(i).2 - D_i^z)}{Anzahl\ der\ Linien}$$

$$T = \sum T_i^z$$

Im Beispiel 5.4berechnet sich für den Knoten 2 der Linie z der Teilindirekfahreranteil als

$$T_2^z = \frac{170 - 30}{2} = 70$$

Der Vollindirektfahreranteil V errechnet sich aus dem Strom auf der Verknüpfungskante. Dieser wird um den berechneten Direktfahreranteil D und den Teilindirektfahreranteil T vermindert und dann durch die Anzahl der Linien auf der Verknüpfungskante geteilt.

$$V = f_u(i).2 - D - T$$

Die Summe aller drei Anteile ergeben dann den endgültigen Wert für die Attraktivität A, wobei Teilindirektfahrer- und Vollindirektfahreranteil nur zu einem Viertel in die Rechnung eingehen.

$$A = D + \frac{(T + V)}{4}$$

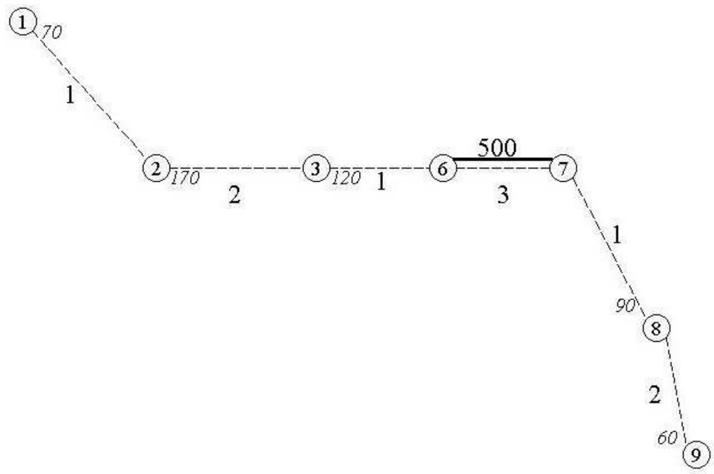
Tabelle 5.1: Verkehrsbedarfsmatrix

	8	9
1	10	0
2	30	20
3	30	10

Aufbruchverknüpfung

In dieser Stufe können Linien über andere Linien hinweg verlängert oder zusammengefügt werden, wenn dafür zur Einhaltung der Gesamtlinienlänge eine andere Linie verkürzt wird. Damit eine solche Aufbruchverknüpfung realisiert werden kann, muss der sich dadurch ergebende Gewinn an Attraktivität größer sein als der Verlust, der dadurch entsteht, dass eine andere Linie verkürzt werden muss, um die Gesamtlinienlänge einzuhalten. Die für die neue Verbindung benötigte Kapazität wird somit durch die Verkürzung einer bestehenden Linie zur Verfügung gestellt.

Abbildung 5.4: Attraktivitätsberechnung



Bevor das eigentliche Verfahren beginnt, wird deswegen eine aufsteigend sortierte Liste von potentiellen Kanten aufgestellt, die im Laufe des Verfahrens aufgebrochen werden können. Zu diesem Zweck werden diejenigen Enden aller Linien bestimmt, die aufbrechbar sind, was heißt, dass durch deren Aufbruch kein Knoten vom Liniennetz abgetrennt wird, weil mindestens zwei Linien die verbindende Kante überdecken. Ist eine solche Kante gefunden, wird ihr Attraktivitätsverlust bestimmt. Dies geschieht auf die gleiche Weise wie beim Verknüpfen zweier Linien. Um so den Verlust zu relativieren und abzudämpfen, wird dieser Wert anschließend durch die Wurzel aus der freiwerdenden Kapazität geteilt. Auf diese Weise wird eine Liste von Kanten aufgebaut, die nach steigendem Attraktivitätsverlust sortiert ist.

Um die Neuverknüpfung zu realisieren, wird jede Linie einmal als Grundlinie angesehen. Auf dieser Grundlinie wird dann jeder Knoten abgeschritten, wobei überprüft wird, ob für die zu diesem Knoten inzidente Kante eine Aufbruchverknüpfung zwischen zwei Linien verwirklicht werden kann. Nach Sonntag wird dabei zwischen 5 Fällen unterschieden:

1. Eine von der Grundlinie abzweigende Linie wird zum Anfangsknoten der Grundlinie hin verlängert.

2. Eine auf die Grundlinie einmündende Linie wird bis zum Endknoten der Grundlinie hin verlängert.

3. Ein Mittelknoten der Grundlinie wird als ein Anfangsknoten betrachtet und potentielle Linien werden entsprechend bis dorthin verlängert.

4. Ein Mittelknoten der Grundlinie wird als ein Endknoten betrachtet und anschließend wird wie in Fall 3 vorgegangen.

5. Über die Knoten der Grundlinie hinweg werden zwei auf die Grundlinie einmündende/abzweigende Linien zu einer Linie zusammengeknüpft, wobei hier alle möglichen Kombinationsmöglichkeiten von Linien ausgetestet werden müssen.

Bei jeder Verlängerung einer Linie muss der Richtungssinn der Linien beachtet werden, so dass keine Hin- und Rückfahrten auf einer Kante entstehen, sondern sich ein kontinuierlicher Linienzug ohne Zyklen bildet. Wenn für eine Kante eine potentielle Aufbruchverknüpfung (einer der fünf Fälle) erkannt worden ist, wird eine Verknüpfungsdiskussion auf dieser Kante durchgeführt, die abhängig vom betrachteten Fall ist.

Während dieser Diskussion wird jede Linie darauf geprüft, ob sie zum einen den Bedingungen des Falles entspricht (ob sie also passend verlängert bzw. zusammengefasst werden kann) und zum anderen ob sie bei einer Verknüpfung keinen Zyklus erzeugen würde bzw. ihre Linienlänge sich dadurch nicht über das erlaubte Maximum vergrößert.

Wenn dies möglich ist, wird für diese Verknüpfung der Attraktivitätsfaktor und die sich durch diese Verknüpfung neu ergebende Überlastung der Kapazität (entsprechend dem sich durch die Verknüpfung ergebenden Neubedarf an Linienstrecke) bestimmt. Um diejenigen Linien zu ermitteln, die aufgebrochen werden müssen, um Kapazität freizusetzen, benutzt man die am Anfang ermittelte Rangfolgenliste und arbeitet diese so lange ab, bis die benötigte Kapazität für diese Verknüpfung erreicht worden ist. Hierbei wird darauf geachtet, dass nicht die gerade aktuell betrachteten Linien gekürzt werden. Der sich bei jedem Aufbruch ergebende Attraktivitätsverlust wird fortlaufend aufsummiert. Dieser Attraktivitätsverlust wird dann dem Attraktivitätsgewinn gegenübergestellt. Wenn der Vergleich positiv ausfallen sollte, also ein effektiver Attraktivitätsgewinn vorliegt, wird der bisher beste Verknüpfungsfall mit diesem Verknüpfungsfall verglichen und so ein Maximalfall bestimmt.

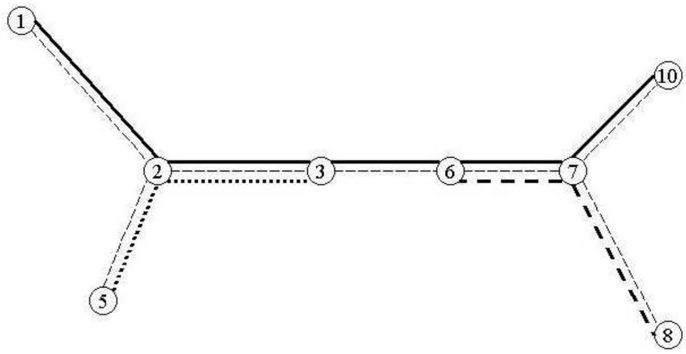
Wenn alle Linien abgearbeitet wurden, hat sich entweder keine Verbesserung einstellen können, womit diese Stufe beendet wird, oder aber es gibt unter den verschiedenen Verknüpfungsmöglichkeiten einen Fall des maximalen effektiven Attraktivitätsgewinns, der jetzt realisiert wird. Anschließend wird der nächste Maximalfall nach demselben Prinzip berechnet. Diese Prozedur wird so lange fortgesetzt, bis keine Verbesserung mehr möglich ist.

In dem Beispiel in Abbildung 5.5ist die durchgezogene Linie (1,2,3,6,7,10) als Grundlinie anzusehen und die gepunktete Linie kann über die Kante (3,6) mit der gestrichelten Linie verbunden werden. Dabei muss entweder die Kante (1,2) oder die Kante (7,10) gekürzt werden, abhängig davon welche von beiden die geringere Attraktivität aufweist und wie viel Kapazität die Kante (3,6) benötigt.

Linienendverknüpfung

In dieser letzten Stufe werden alle Kanten, die einen gemeinsamen Endknoten, aber keine gemeinsame Endkante haben, darauf hin geprüft, ob ihr Verlauf zusammengelegt werden kann. Zu diesem Zweck werden alle Linien paarweise miteinander auf dieses Kriterium hin verglichen. Ergibt sich durch die Zusammenführung zweier Linien kein Zyklus und wird durch den Zusammenschluss die maximale Linienlänge nicht überschritten, so wird der Anschlussstrom berechnet, indem der Strom auf den jeweiligen Anschlusskanten addiert wird. Am Ende ergibt sich so ein maximaler Anschlussstrom, dessen zugehörige Zusammenführung von Linien dann im Linienplan realisiert wird. Anschließend folgt ein weiterer Durchlauf dieser Verknüpfungsmethode, bis alle möglichen Kombinationen von Endknoten getestet worden sind.

Abbildung 5.5: Aufbruchverknüpfung



5.4.3.3 Bewertungsphase

In der letzten Phase wird das endgültige Resultat, also der sich aus der Verknüpfungsphase ergebende Linienplan, bewertet. Im Prinzip wird hier das gleiche durchgeführt, wie in der Aufbereitungsphase bei der Umlegung der Nachfragewerte auf das Netz. Diesmal wird aber nicht vom idealisierten Linien ausgegangen, wo es für jede Strecke eine Direktfahrtmöglichkeit auf dem kürzesten Weg gibt, sondern die Bewertung und Umlage erfolgt auf der Grundlage des während der Verknüpfungsphase berechneten Linienplans.

Dafür muss aber zuerst die interne Datenstruktur Netzplangraph in das Netz der Transportverbindungen umgewandelt werden. Als Ergebnis der Bewertungsphase werden die Gütemaße berechnet (siehe Kapitel 6.2)

Umwandeln des Netzplangraphen in ein Netz der Transportverbindungen

In dieser Funktion wird aus dem Netzplangraphen G_N und dem Linienplan $L_n^{G_N}$ ein gerichteter Graph erzeugt, der der Definition eines Netz der Transportverbindungen entspricht. Dieser gerichtete, zusammenhängende Multigraph $G_F = (H_N, U_L)$ modelliert mit der Knotenmenge H_N wieder die Haltestellen, die von Buslinien angefahren werden können. Mit der Kantenmenge U_L wird eine Buslinie repräsentiert, die die jeweils verknüpften Haltestellen verbindet. Da zwischen zwei Bushaltestellen mehrere Buslinien fahren können, kann es auch im Graphen zwischen Knoten mehrere Kanten geben, die sich aber jeweils durch die ID für die Buslinie unterscheiden. Als weitere Information wird die Fahrzeit und der Fahrgaststrom auf der Kante abgespeichert, der sich auf dieser Linie während dieses Teilstücks vereinigt.

Belastungsrechnung

In der Belastungsrechnung werden die Fahrgastströme der einzelnen Linien für das Netz der Transportverbindungen errechnet. Zu diesem Zweck wird jede Haltestelle mit jeder anderen in Verbindung gebracht und der Weg zwischen diesen beiden Haltestellen auf den berechneten Linien im Linienplan ermittelt. Dabei kann wieder wie vorher entweder ein kürzester Wege- oder ein Alternativroutenansatz verwendet werden. Anschließend wird der in der Verkehrsbedarfsmatrix angegebene Wert zum bisherigen Fahrgaststrom auf den betroffenen Kanten dazuaddiert (in diesem Fall also der entsprechenden Buslinie zugeordnet).

Zusätzlich zu den Strömen werden die Gütemaße errechnet (siehe Kapitel 6.2), wie z.B. der Direktfahreranteil.

Bewertung und Belastung entlang eines Weges

Bei diesem Ansatz führt der Weg von jeder Haltestelle zu jeder Haltestelle über einen Pfad, der im vorliegenden Liniennetz möglich ist. Es wird beim Abschreiten dieses Weges und dem Eintragen des Verkehrsstromes an die Kanten der jeweils betroffenen Linie versucht, so lange wie möglich auf den Kanten einer Linie zu bleiben, also keinen Umstieg zu erzeugen. Wenn aber ein Umsteigevorgang notwendig ist, dann wird eine definierte Umsteigezeit zur bisherigen Reisezeit als Strafe hinzuaddiert.

5.5 Mathematisches Modell des Fahrplans

5.5.1 Problemstellung

Die Aufgabe der Fahrplanberechnung besteht darin, aus einem gegebenen, gültigen Linienplan einen gültigen Fahrplan zu berechnen. Unter einem Fahrplan versteht man die räumliche und zeitliche Festlegung der Bewegungen der Transportmittel, d.h. die Festlegung der Abfahrts-, Ankunfts- und Streckenfahrtzeiten. Ein Fahrplan ist dann gültig, wenn es möglich ist, diesen Plan mit einer Anzahl X von Transportmitteln zu fahren und dabei jede Haltestelle mit einem bestimmten Takt anzufahren. Ferner darf natürlich nicht der, dem Fahrplan zugrunde liegende Linienplan verändert werden, beispielsweise durch Vertauschung der Reihenfolge der Haltestellen einer Linie. Um einen Fahrplan zu einem gegebenen Linienplan zu berechnen, ist es notwendig, die zeitlichen Abhängigkeiten des gegebenen Linienplans zu erfassen. Eine zeitliche Abhängigkeit ergibt sich innerhalb eines Linienplans immer dann, wenn es an einer Haltestelle möglich ist, die Fahrt mit einem Fahrzeug einer anderen, als der bisherigen Linie fortzusetzen. Es gilt also zunächst, eine geeignete Darstellungsmöglichkeit der zeitlichen Abhängigkeiten zu finden. Ferner gilt es, ein Bewertungskriterium für einen berechneten Fahrplan zu bestimmen und somit Fahrpläne vergleichbar zu machen. Dies wird im Folgenden der sogenannte Gesamtreisezeitaufwand sein. Dieser ergibt sich aus der Summe der Gesamtfahrzeiten aller Reisenden. In den Gesamtfahrtzeiten sind eventuelle Wartezeiten von Reisenden enthalten. Ein Fahrplan A ist besser als ein Fahrplan B, wenn der Gesamtreisezeitaufwand von B kleiner als der Gesamtreisezeitaufwand von A ist. Eine mögliche Lösung der Aufgabe liefert das Verfahren nach [Weigand81]. In diesem werden die zeitlichen Abhängigkeiten eines Linienplanes mittels eines Graphen, dem sogenannten Transportkettengraphen, dargestellt. Auf Grundlage des Transportkettengraphen können gültige Fahrpläne berechnet werden. Die für das Verfahren notwendigen mathematischen Grundlagen, sowie die genaue Vorgehensweise des Verfahrens sind in den nachfolgenden Abschnitten beschrieben.

5.5.2 Grobe Beschreibung des Verfahrens

Wie in der Problemanalyse bereits beschrieben, gilt es, die zeitlichen Abhängigkeiten eines Linienplans zu bestimmen und darzustellen. Dabei erwartet Weigands Verfahren als Eingabe zunächst das sogenannte Netz der Transportverbindungen (siehe Kapitel 5.2.2). Hierbei handelt es sich um einen gerichteten, knoten- und kantenbewerteten Multigraphen $G_N = (V_N, E_N, f, g_v, 1, R_e, 3)$ (Definition siehe Kapitel 5.2.2 und Kapitel 5.2.1). Im Folgenden wird das Netz der Transportverbindungen vereinfacht ohne Knoten- und Kantenbewertungen dargestellt als $G_N = (V_N, E_N)$. Grob dargestellt (für eine detaillierte Beschreibung siehe Kapitel 5.5.5) geht man dabei wie folgt vor: Besteht zwischen zwei Haltestellen A und B eine Verbindung, so werden diese beiden Haltestellen, einschließlich der verbindenden Kante, zu einem Knoten im Transportkettengraph zusammengefasst. Dieser hat dann die Beschriftung A, B, Liniennummer. Gibt es nun im Transportkettengraphen einen Knoten mit der Beschriftung B, X, Liniennummer,wobei X eine Variable sei, so wird zwischen diesen Knoten eine Kante eingefügt. Im Prinzip wird jede Kante des Netzder Transportverbindungen zu einem Knoten im Transportkettengraphen, wobei dieser entsprechend beschriftet wird.Es sind somit nicht mehr die einzelnen Haltestellen von Interesse, vielmehr erhalten wir einen Graphen, welcher die Umstiegsmöglichkeiten zwischen verschiedenen Haltesstellen darstellt. Damit sind dann die zeitlichen Abhängigkeiten des Linienplans erfasst. Aus einem berechnetem Transportkettengraphen lässt sich dann leicht ein gültiger Fahrplan ableiten, indem ein Gerüstdes errechneten Transportkettengraphen bestimmt wird. Wurde ein Gerüst berechnet, werden im Anschluss dierelativen Abfahrtszeiten an die Knoten des Transportkettengraphen geschrieben. Dies geschieht auf folgende Art und Weise:Es wird ein Knoten Y im Transportkettengraphen bestimmt. Dieser wird mit einer Minutenangabe, beispielsweise 15 Minuten, versehen.Ausgehend von diesem Knoten wird dann mittels eines Tiefendurchlaufs durch den Graphen die entsprechende relative Abfahrtszeit von jedem weiteren Knoten im Transportkettengraphen gesetzt werden. Existiert beispielsweise ein Kante vom Knoten Y zum Knoten Z und beträgt die Fahrtzeit zu diesem Knoten 4 Minuten, so wird an Knoten Z der Wert 19 geschrieben. Auf Grundlage der relativen Abfahrtszeiten an jedem Knoten kann ein gültiger Fahrplan für jedes beliebige Zeitintervall berechnet werden. Wir beginnen an Knoten Y und führen wiederum einen Tiefendurchlauf durch den Transportkettengraphen durch. Es soll nun ein Fahrplan für das Zeitintervall von 8.00 bis 9.00 Uhr berechnet werden. Das Transportmittel beginnt seine Fahrt an Knoten Y um 8.15 Uhr. An Knoten Z fährt um 8.19 Uhr ein Transportmittel los und so weiter. Dabeikann es passieren, dass die relativen Abfahrtszeiten an einem Knoten größer 60 sind, die Werte werden dann entsprechend umgerechnet.

5.5.3 Ergebnis

Als Ausgabe gibt das Modul Fahrplan eine sortierte Liste von Fahrplanelementen an das Programmmodul. Ein Fahrplanelement besteht aus einem 4-Tupel. Die einzelnen Werte sind zu interpretieren als Liniennummer, Haltestellennummer, Stunde und Minute. Mittels den oben bestimmten Werten ist es leicht möglich einen gültigen Fahrplan auszugeben. Dabei wird zwischen zweimöglichen Ausgabeformaten eines Fahrplans unterschieden: Zum einem Linienfahrpläne, zum anderen Haltestellenfahrpläne. Ein Linienfahrplan gibt an, zu welchem Zeitpunkt eine Transportmittel der entsprechenden Linie an welcher Haltestelle losfährt. Haltestellenfahrpläne geben hingegen an, zu welchem Zeitpunkt welche Linie die entsprechende Haltestelle bedient. Aus diesen Daten werden im Modul Programm die beschriebenen Fahrpläne erstellt. In den nachfolgenden Kapiteln wird das oben beschriebene Vorgehen präzisiert und es wird im Detail gezeigt, wie sich aus einem Transportkettengraphen ein gültiger Fahrplan berechnen lässt. Der Ablauf des Verfahrens entspricht obiger Einleitung.

5.5.4 Netzstrukturen

Unter einem Verkehrsnetz versteht man ein System aus vielen miteinander verbundenen, sich kreuzenden und voneinander abzweigenden, dem Verkehr dienenden Linien oder Strecken. Von der Mathematik her kommend wird unter einem Netz ein System aus Knoten und verbindenden Kanten, das sich als Graph darstellen lässt, verstanden.

Je nach Untersuchungszweck muss eine Netzdarstellung mehr oder weniger abstrahieren. Die Darstellung kann als Knoten alle Orte mit Quell- und Zielaufkommen enthalten oder sich beispielsweise auf große Städte beschränken. Bei einer betrieblichen Untersuchung eines Streckennetzes können beispielsweise alle Knoten aufgenommen werden, die eine betriebliche Bedeutung haben. Das Streckennetz kann ungerichtet dargestellt werden; die Fahrmöglichkeiten können aber auch durch Darstellung gerichteter Kanten aufgezeigt werden. Während diese Darstellungen eine Abstraktion der Infrastruktur und damit des zeitunabhängigen Netzes bedeuten, macht die Darstellung des Liniennetzes eine Aussage über das Betriebssystem und damit das zeitabhängige Netz.

Da das zeitabhängige Transportangebot und die sich daraus ergebenden Transportketten und Wartezeiten im Netz interessieren, werden die geographischen Orte, in unserem Fall Haltestellen, als Knoten und Transportverbindungen als gerichtete Kanten dargestellt. Da auf einer Streckenkante mehrere Linien liegen können, sind mehrere Kanten zwischen den Knoten zulässig.

Anhand der folgenden Abbildung können diese Verhältnisse nachvollzogen werden.

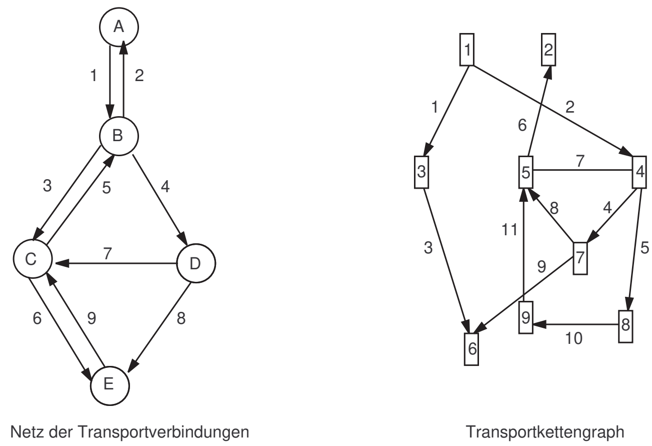
Diese Darstellungsweise wird als Netz der Transportverbindungen bezeichnet. Ein solches Netz der Transportverbindungen übergibt das Modul Linienplan an das Modul Fahrplan. Dieses wird dann zu einem Transportkettengraphen umgebaut. Auf die Konstruktion und Eigenschaften eines Transportkettengraphen wird im nächsten Unterkapitel eingegangen.

Transportkettengraph

Konstruktion des Transportkettengraph

Der Transportkettengraph $G_T = (V_T, E_T)$, ein gerichteter Graph entsprechend der Definition aus Kapitel 5.2.1) lässt sich aus dem Netz der Transportverbindungen $G_N = (V_N, E_N)$ (vereinfachte Sichtweise ohne Kanten- und Knotenbewertungen) entwickeln. Jede Kante des Netzes der Transportverbindungen E_n wird zu einem Knoten V_T des Transportkettengraph. In jedem Knoten V_N

Figure 5.6: Darstellung Netz der Transportverbindungen und Transportkettengraph



des Netzes ergeben sich zwischen allen inzidenten Kanten, nicht jedoch zwischen entgegengesetzt parallelen Kanten, Bindungen, die zu den Kanten E_T des Transportkettengraphen werden.

Formal definiert sieht der Transportkettengraph wie folgt aus:

Der Transportkettengraph $G_T = (V_T, E_T)$ des Netzes der Transportverbindungen $G_N = (V_N, E_N)$ besteht aus einer Menge von Knoten V_T, so dass jeder Kante E_{Ni} der Menge der Kanten E_N des Netzes der Transportverbindungen ein Knoten V_{Ti} zugeordnet ist und einer Menge Kanten $E_{Ti} = (V_{Ti}, V_{Tj})$, so dass Knoten V_{Ti} adjazent mit Knoten V_{Tj} ist, wenn Kante E_{Ni} im Netz der Transportverbindungen adjazent mit Kante E_{Nj} ist und die Kanten E_{Ni} und E_{Nj} nicht parallel oder entgegengesetzt parallel sind.

Die Knoten und Kanten werden durchnummeriert, dabei entsprechen die Knotennummern des Transportkettengraphen den Kantennummern des Netzes der Transportverbindungen.

Größe des Transportkettengraph

Die Größe des Transportkettengraphen ergibt sich offensichtlich aus der Netzgröße und den Knotenwertigkeiten des Netzes der Transportverbindungen. Die Knotenwertigkeit d beschreibt die Anzahl der an einem Knoten anschließenden Strecken.

Die Anzahl der Knoten des Transportkettengraphen ergibt auch offensichtlich aus der Anzahl der Kanten des Netzes der Transportverbindungen:

$$|V_T| = |E_N|$$

Die Anzahl der Kanten des Transportkettengraphen kann aus den Knotenbeziehungen des Netzes der Transportverbindungen ermittelt werden. Für ein Streckennetz, in dem auf jeder Strecke in jeder Richtung eine Transportverbindung zu berücksichtigen ist, führen in einen Knoten d anschließende Strecken. Von jeder der d ankommenden Transportverbindungen bestehen Bindungen zu allen abgehenden Transportverbindungen, nicht aber in die Gegenrichtung (siehe Definition Transportkettengraph im Kapitel 5.5.4). Also gehen von jeder ankommenden Transportverbindung $(d-1)$ Bindungen ab. Im Knoten sind dann offensichtlich $d(d-1)$ Bindungen enthalten. Faßt man dies alles zusammen, so ergibt sich für den Transportkettengraph die Anzahl der Kanten mit:

$$E_T = \sum_{i=1}^{V_N} d_i(d_i - 1)$$

Verallgemeinert man diese Überlegung, dann gilt für einen Knoten, bei dem auf den anschließenden Strecken $an_i(i = 1, \ldots, d)$ ankommende und $ab_j(j = 1, \ldots, d)$ abgehende Transportverbindungen liegen:

$$E_{Tk} = \sum_{i=1}^{d} \left(an_i \cdot (\sum_{j=1}^{d} ab_j - ab_i) \right)$$

Für den gesamten Transportkettengraph ergibt sich die Kantenzahl aus der Summe der Kanten aller Knoten des Netzes der Transportverbindungen.

Reduktion des Transportkettengraph

Hinsichtlich des Rechenzeitaufwandes, der zu bewältigen ist, wird es sinnvoll sein, den Transportkettengraph so klein wie möglich zu halten. Es ist offensichtlich, dass der Transportkettengraph um so kleiner ist, je kleiner das Streckennetz ist bzw. je einfacher es dargestellt wird.

Es ist möglich durch Zusammenfassen von Knoten des Streckennetzes, nämlich aller zweiwertigen Knoten, einen Transportkettengraphen zu reduzieren. Ferner könnte man sich vorstellen, alle Bindungen zu entfernen, welche aufgrund ihrer Lage im Netz keine, oder nur unbedeutende Verkehrsströme abdecken.

Erweiterungen des Transportkettengraph

Bisher wurden bei der Bildung des Transportkettengraphen ja lediglich die verkehrlichen Anforderungen erfasst, so dass es möglich ist, die für die Reisenden im Netz entstehenden Wartezeiten zu erfassen. Darüberhinaus ist es mittels des Transportkettengraphen möglich, betriebliche Anforderungen, wie z.B. Fahrzeugumläufe oder begrenzte Kapazitäten einer Betriebsanlage zu berücksichtigen.

5.5.5 Graphentheoretische Beschreibung des Fahrplans für ein Transportnetz

Bedeutung des Transportkettengraphen

Der Transportkettengraph stellt die zeitlichen Abhängigkeiten zwischen den Transportmitteln in einem Netz dar. Die Knoten bedeuten die Bewegung der Fahrzeuge zwischen den Netzknoten mit den Fahrzeiten, die Kanten die Wartezeiten in den Netzknoten, soweit dort fahrplantechnische Bindungen zu berücksichtigen sind.

Der Transportkettengraph kann als relativer Fahrplan aufgefasst werden. Er darf keine Widersprüche enthalten, aus ihm muss sich ein durchführbarer

Fahrplan entwickeln lassen. Was hierunter zu verstehen ist wurde bereits in Kapitel 5.5.1 erläutert. Anhand der Eigenschaften des Transportkettengraphen sollen die Gesetzmäßigkeiten des Fahrplans im Netz beschrieben werden.

Es erfolgt eine Beschränkung auf Taktfahrpläne.

Zusammenhang im Transportkettengraph

Jeder Fahrplan hat mindestens einen Freiheitsgrad. Eine zeitliche Lage eines Busses an einem Ort kann frei gewählt werden. Entsprechend hat der Transportkettengraph mindestens einen Freiheitsgrad. Die anderen zeitlichen Lagen ergeben sich aus den Fahrzeiten und den Wartezeiten.

Ein Taktfahrplan hat einen weiteren Freiheitsgrad, die Taktzeit kann frei gewählt werden.

Ist ein Transportkettengraph nicht zusammenhängend (siehe Kapitel 5.2.1), sondern zerfällt in zwei oder mehr Zusammenhangskomponenten (siehe Kapitel 5.2.1), so kann für jede dieser Zusammenhangskomponenten ein Fahrplan unabhängig von der Fahrplanzeit der übrigen Zusammenhangskomponenten erstellt werden. Die Anzahl der Zusammenhangskomponenten des Transportkettengraphen bestimmt die Zahl der Freiheitsgrade bei der Fahrplangestaltung. Beachtet werden muss hierbei, dass der gerichtete Transportkettengraph hinsichtlich der Berechnung der Zusammenhangskomponenten als ungerichtet betrachtet wird.

Durch beliebige Festlegung einer Zeitlage jeder Zusammenhangskomponente des Transportkettengraphen wird aus einem "relativen Fahrplan", bei dem nur die gegenseitige zeitliche Zuordnung festgelegt war, ein "absoluter Fahrplan", bei dem sich für jedes Fahrzeug an jedem Ort ein Zeitpunkt festlegen lässt.

Es sei an dieser Stelle darauf hingewiesen, dass alleine aus der Struktur des Streckennetzes noch nicht direkt auf die Anzahl der Freiheitsgrade des Fahrplans geschlossen werden kann. Hierzu kann die folgende Abbildung betrachtet werden.

Netze ohne Wartezeiten

Lässt sich der Transportkettengraph als gerichteter, offener Kantenzug darstellen, ergeben sich keine Schwierigkeiten für die Fahrplangestaltung. Nach Wahl einer beliebigen Abfahrts- oder Ankunftszeit können alle übrigen Zeitlagen bestimmt werden, ohne dass Wartezeiten entstehen. Offensichtlich können ausgehend von einem beliebigen Knoten immer dann zusätzliche Knoten angeschlossen werden, wenn nicht durch eine Bindung ein Knoten auf einer zweiten Kantenfolge erreicht wird. Um also netzbedingte Wartezeiten zu vermeiden, muss der Transportkettengraph eine baumförmige Struktur besitzen.

An dieser Stelle zeigt sich ein weiterer Vorteil des Transportkettengraphen. Während das vermaschte Netz der Transportverbindungen in Abbildung **5.6** auf eine eher schwierige Fahrplangestaltung hindeutet, zeigt die baumförmige Struktur des Transportkettengraphen in Abbildung 5.8 sofort, dass in diesem Netz keine netzbedingten Wartezeiten entstehen. An dieser Stelle soll kurz erläutert werden, was untereiner netzbedingten Wartezeit zu verstehen ist. Muss ein Transportmittel an einer Haltestelle warten, um somit den Anschluss an Transportmittel anderer Linien zu gewährleisten, so wird diese Zeit netzbedingte Wartezeit genannt.

Abbildung 5.7: Netze der Transportverbindungen und zugehörige Transportkettengraphen

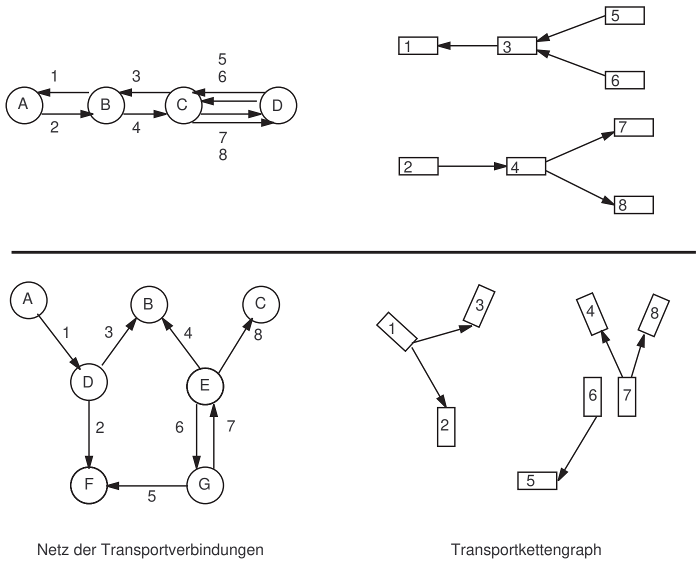
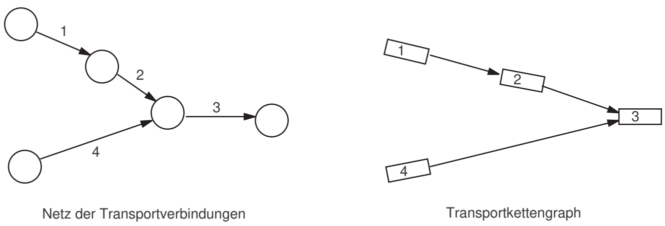


Figure 5.8: Netz der Transportverbindungen und Transportkettengraph



Kreise im Transportkettengraph

Im letzten Kapitel wurde verdeutlicht, dass in einem vermaschten Netz der Transportverbindungen genau dann keine Wartezeiten entstehen, wenn der zu diesem Netz zugehörige Transportkettengraph eine baumförmige Struktur besitzt. Enthält der Transportkettengraph hingegen Kreise, können netzbedingte Wartezeiten entstehen, wie im nachfolgenden Beispiel zu erkennen ist. Beachtet werden muss wiederum, dass die Kreise bzw. der Transportkettengraph als ungerichtet betrachtet werden.

Beispiel:

Hier ist der Knoten D von Knoten B aus über zwei Wege zu erreichen. Sind die Fahrzeiten über die beiden Wege nicht zufällig gleich oder um ein Vielfaches der Taktzeit verschieden, so muss auf einem Weg eine Wartezeit entstehen. Graphentheoretisch handelt es sich dann um eine Masche.
Auch bei diesem Beispiel entsteht eine netzbedingte Wartezeit.

Figure 5.9: Netz der Transportverbindungen und Transportkettengraph

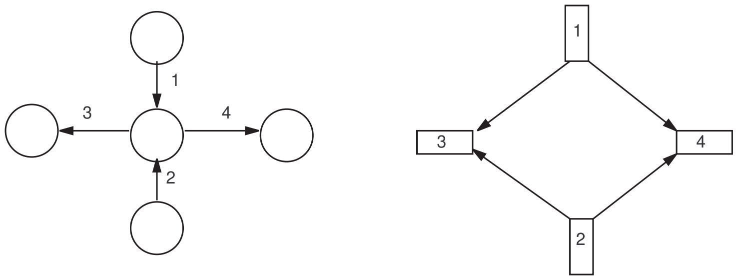
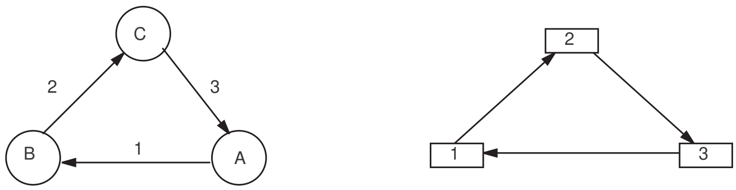


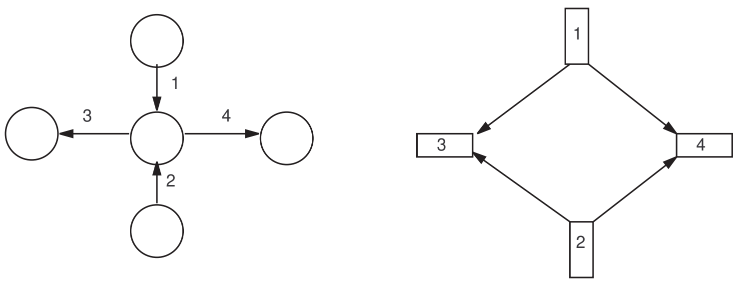
Figure 5.10: Netz der Transportverbindungen und Transportkettengraph



Denn Fahrzeug 1 hat Anschluss an Fahrzeug 2, dieser an Fahrzeug 3, und jener an eines um eine oder mehrere Taktzeiten späteres Fahrzeug 1 der Relation A-B.

In Knoten mit mehreren ankommenden oder abfahrenden Fahrzeuge entstehen Kreise im Transportkettengraphen ohne Berücksichtigung der Pfeilrichtung (siehe Beispiel).

Figure 5.11: Netz der Transportverbindungen und Transportkettengraph



Allgemein gilt, dass im Transportkettengraphen Kreise mit unterschiedlichen Längen und wechselnden Pfeilrichtungen auftreten können. In jedem Kreis im Transportkettengraph muss mindestens eine netzbedingte Wartezeit auftreten. Es können mehr als eine Wartezeit vorhanden sein, um die zeitlichen Bedingungen zu erfüllen, etwa indem die Wartezeit auf mehrere Bindungen verteilt wird. Aus Gründen der Netzstruktur genügt es, wenn eine Bindung in einem Kreis eine Wartezeit hat.

Auch wenn zufällig die netzbedingte Wartezeit t_{wn} in einzelnen Kreisen des Transportkettengraphen gleich Null ist (nämlich genau dann wenn die Fahrzeiten über die beiden Wege zufällig gleich oder um ein Vielfaches der Taktzeit verschieden sind), besteht sie doch als Zwangspunkt in der Fahrplangestaltung. Denn bei kleinsten Änderungen der Fahrzeiten oder Mindestwartezeiten wird sie wieder bemerkbar. Die netzbedingte Wartezeit muss immer kleiner als die Taktzeit T sein, da die Reisenden sonst das Fahrzeug der nächsten Taktzeit benutzen könnten. Es gilt also:

$$T > t_{wn} \geq 0$$

Zyklomatische Zahl und Gerüst

Soll ein Fahrplan für ein Netz entwickelt werden, dessen Transportkettengraph Kreise enthält, haben einige Bindungen zwangsläufig netzbedingte Wartezeiten.
Entfernt man einige Kanten aus dem Transportkettengraphen, bis die restlichen Kanten einen Graphen mit einer baumförmigen Struktur bilden, so hat man ein Gerüst des Transportkettengraphen erzeugt. Es wurde bereits gezeigt, dass für diesen Teil der Bindungen keine netzbedingten Wartezeiten entstehen. Für alle übrigen Bindungen müssen netzbedingte Wartezeiten in Kauf genommen werden.

Die Anzahl der Bindungen, die Wartezeiten enthalten müssen, entspricht der "Kreiszahl" oder "zyklomatischen Zahl" des Transportkettengraphen. Diese gibt an, wieviele linear unabhängige Kreise im Graphen vorhanden sind. Die zyklomatische Zahl μ_T für jede Zusammenhangskomponente des Transportkettengraphen ergibt sich aus der Anzahl der Knoten V_T, sowie der Anzahl der Kanten E_T:

$$\mu_T = E_T - V_T + 1$$

Die zyklomatische Zahl gibt also an, für wieviele Bindungen im Netz sich netzbedingte Wartezeiten ergeben, wieviele fahrplantechnische Bindungen also nicht beliebig gewählt werden können.

Die Größe der netzbedingten Wartezeiten

Die Größen der netzbedingten Wartezeiten t_w lassen sich aus den zeitlichen Bedingungen in den Kreisen ermitteln. Die Summe aus Fahrzeiten t_f und Wartezeiten t_w muss unter Beachtung der Richtungen in jedem Kreis Null oder ein ganzzahliges Vielfaches der Taktzeit T sein:

$$\sum t_f + \sum t_w \pm n \cdot T = 0, n = ganzzahlig$$

Bei den Wartezeiten werden netzbedingte Wartezeiten t_{wn} und Mindestwartezeiten t_{wmin} unterschieden:

$$\sum t_f + \sum t_{wmin} + \sum t_{wn} \pm n \cdot T = 0$$

Damit können die netzbedingten Wartezeiten ermittelt werden:

$$\sum t_{wn} = -\sum t_f - \sum t_{wmin} \pm n \cdot T.$$

Alternative Fahrpläne

Für ein Netz können verschiedene Fahrpläne erstellt werden. Mit der Berechnung der zyklomatischen Zahl kennt man die Anzahl der Kanten im Transportkettengraphen, auf welchen netzbedingte Wartezeiten liegen müssen. Netzbedingte Wartezeiten können jedoch auf verschiedenen Bindungen gelegt werden. Jede zulässige Zuordnung der Wartezeiten entspricht einem Gerüst im Transportkettengraphen. Somit entspricht die Anzahl der möglichen Gerüste also der Anzahl der möglichen Fahrpläne. Es gilt:

Für vollständige Graphen ergeben sich bei n Knoten n^{n-2} verschiedene Gerüste.
Die Anzahl der Gerüste eines Graphen beliebiger Struktur ist im Allgemeinen sehr groß. Nicht jedes dieser Gerüste ist im Zusammenhang mit dem Verkehrsnetz, welches der Transportkettengraph repräsentiert, jedoch sinnvoll.

5.5.6 Abhängigkeiten im Netz

Wegen der zahlreichen möglichen Gerüste im Transportkettengraph und damit unterschiedlichen Fahrplanstrukturen ist die Kenntnis der Abhängigkeiten wichtig. Insbesondere wenn man Gerüste erzeugen will, die günstige Fahrpläne repräsentieren, müssen die Zusammenhänge bekannt sein.

Für jeden Kreis im Transportkettengraphen lässt sich eine Gleichung mit den zeitlichen Bindungen entsprechend

$$\sum t_f + \sum t_{wmin} + \sum t_{wn} \pm n \cdot T = 0$$

aufstellen. Besteht der Kreis i aus der Menge der Knoten V_i und der Kanten E_i, dann gilt:

$$\sum_{j=1}^{E_T} c_{ij}(t_{wnj} + t_{wminj}) + \sum_{k=1}^{V_T} d_{ik} t_{fK} \pm n_i T = 0$$

$$c_{ij} = \begin{cases} +1, \text{falls} k \in E_{i+} \\ -1, \text{falls} k \in E_{i-} \\ 0, \text{sonst} \end{cases}$$

$$d_{ik} = \begin{cases} +1, \text{falls} k \in V_{i+} \\ -1, \text{falls} k \in V_{i-} \\ 0, \text{sonst} \end{cases}$$

n_i ganzzahlig.

Die Mindestwartezeiten und die Fahrzeiten sind konstant und bekannt und können daher für jeden Kreis i zu einer Konstanten t_{ci} zusammengefasst werden:

$$t_{ci} = \sum_{k=1}^{E_T} d_{ik} \cdot t_{fk} + \sum_{j=1}^{E_T} c_{ij} \cdot t_{wminj}$$

Damit gilt dann für jeden Kreis:

$$\sum_{j=1}^{E_T} c_{ij} \cdot t_{wnj} + t_{ci} \pm n_i T = 0$$

Die Gleichungen aller Kreise eines Graphen, die linear unabhängig voneinander sind, bilden das Fundamentalsystem des Graphen. Entwickelt man für einen Transportkettengraphen G_T mit dem Gerüst G_{T1} für jeden der μ_T Kreise des Fundamentalsystems diese Gleichung, so erhält man ein Gleichungssystem:

$$\sum_{j=1}^{E_T} c_{ij} \cdot t_{wnj} + t_{ci} \pm n_i T = 0 furalle i = 1, \ldots, \mu_T.$$

In Matrizenschreibweise ergibt sich dann:

$$C \cdot t_{wn} + \vec{t_c} \pm \vec{n} \cdot T = 0$$

Die Matrix $C = \{c_{ij}\}$ entspricht der fundamentalen Kreismatrix des Graphen. Es ist eine Matrix mit μ_T Zeilen und E_T Spalten mit den Elementen

$$c_{ij} = \begin{cases} \pm 1, \text{falls} k_j \in E_i \\ 0, \text{sonst} \end{cases}$$

Die fundamentale Kreismatrix kann in die Einheitsmatrix E und eine rechteckige Matrix $\overline{C}$ zerlegt werden. Die Vektoren der fundamentalen Kreismatrix sind linear unabhängig. Das Gleichungssystem mit den zeitlichen Bedingungen des Transportkettengraphen besteht demnach aus μ_T linear unabhängigen Gleichungen. Jeder Kreis i im Graphen enthält eine Kante j, die nicht im Gerüst enthalten ist, d.h. diese Kante ist nur in diesem Kreis enthalten. Die Koeffizienten c_{ij} dieser Kanten sind also folglich nur in der jeweiligen Spalte $j = \pm 1$, in allen anderen Spalten sind sie Null. Die Kanten, die nicht im Gerüst sind, d.h. die Bindungen, die netzbedingte Wartezeiten haben, bilden offensichtlich die Einheitsmatrix. Das Vorzeichen ist von der Kantenrichtung abhängig.

Eine Möglichkeit zur schematischen Erfassung eines Systems aus linearen Gleichungen wurde lt. Weigand im Rahmen der linearen Planungsrechnung entwickelt. Das Gleichungssystem wird dabei als Simplextableau (Definition siehe [Weigand81, S. 50 ff.]) Zyklen.

In der Basis und damit in der Kopfspalte des Tableaus stehen die Kanten, die nicht im Gerüst sind, d.h. diejenigen Bindungen, die Wartezeiten haben, in der Nichtbasis und damit in der Kopfzeile sind die Kanten zu finden, welche im Gerüst sind. Die Matrix enthält nur die Werte $+1, 0, -1$.

	Nichtbasis (Kanten im Gerüst)	rechte Seite
Basis (Kanten nicht im Gerüst)	$c_{ij} = 0$ wenn Kante j nicht im Basiskreis i $c_{ij} = \pm 1$ wenn Kante j im Basiskreis i	$0 \leq t_{wni} < T$

Die rechten Seiten des Tableaus werden mit Hilfe von Gleichungen berechnet und ergeben sich aus den Fahrzeiten und den Mindestwartezeiten unter Berücksichtigung der Taktzeit. Damit enthält die rechte Seite die netzbedingten Wartezeiten für jede Kante des Transportkettengraphen, die nicht im Gerüst enthalten ist. Die Berücksichtigung der Mindestwartezeiten entspricht der Behandlung von Untergrenzen in der linearen Planungsrechnung.

Rechnet man das Simplextableau um, tauscht man eine Variable aus der Matrix $\overline{C}$ gegen eine Variable, die bisher in der Einheitsmatrix war. Eine andere

Kante des Graphen erscheint in der Einheitsmatrix und erhält damit eine Wartezeit. Austausch von Basis- gegen Nichtbasisvariable entspricht dem Austausch einer Kante, welche nicht im Gerüst ist gegen eine Gerüstkante. Ist das entsprechende Matrixelement im Simplextableau Null, so können diese Variablen nicht vertauscht werden.

Tauscht man die Basisvariable BV_r gegen die Nichtbasisvariable NBV_s, so gelten die Rechenregeln der Simplexmethode.

Bei der Umrechnung der rechten Seite ist zu beachten, dass hier die Linearitätsbedingung nicht gegeben ist. Es handelt sich um eine Sprungfunktion, die nur im Bereich zwischen Null und der Taktzeit linear ist. Wird die Wartezeit größer als die Taktzeit, kann das nächste Fahrzeug der gleichen Transportverbindung eine Taktzeit später (oder früher) erreicht werden.

5.5.7 Der Transportkettengraph als Fahrplan

Wurde ein Gerüst im Transportkettengraph ermittelt, so ist damit gleichzeitig ein zulässiger Fahrplan ermittelt. Alle Kanten des Transportkettengraphen, die im Gerüst enthalten sind, entsprechen fahrplantechnischen Bindungen, die frei wählbar sind. Um unnötige Reisezeitverluste zu vermeiden, wird man diese im allgemeinen auf die Mindestwartezeiten beschränken. Die netzbedingten Wartezeiten werden aus den Bedingungen des Fundamentalsystems ermittelt. Da die Fahrzeiten, Mindestwartezeiten und netzbedingten Wartezeiten bekannt sind, kann ein Fahrplan erstellt werden. Dazu kann pro Zusammenhangskomponente eine Abfahrtszeit angegeben werden. Die weiteren Abfahrtszeiten ergeben sich sukzessive aus den Fahrzeiten und den Wartezeiten. Auf die genaue Schilderung des Ablaufs sei hier auf 7.7 verwiesen.

5.5.8 Kennwerte zur Beurteilung der zeitlichen Abhängigkeiten im Netz

Der Transportkettengraph eignet sich zur Beurteilung der zeitlichen Abhängigkeiten im Netz. Mit zwei Kennwerten können die wesentlichen Merkmale der Fahrplangestaltung beschrieben werden. Es gilt:

Die Anzahl der Freiheitsgrade des Fahrplans für ein Transportnetz entspricht der Anzahl der Zusammenhangskomponenten des Transportkettengraphen. Für jede Zusammenhangskomponente ist die freie Wahl einer Zeitlage möglich. Alle anderen Ankunfts- und Abfahrtszeiten liegen dann fest.

Beim Taktfahrplan besteht ein weiterer Freiheitsgrad in der freien Wahl der Taktzeit. Es gilt weiterhin:

Jedes Gerüst im Transportkettengraphen stellt einen zulässigen Fahrplan dar. Der Grad der fahrplantechnischen Überbestimmtheit entspricht der zyklomatischen Zahl im Transportkettengraphen. Ist der Transportkettengraph baumförmig, so ist das Netz fahrplanmäßig bestimmt.

5.5.9 Fahrzeugumlaufplanung

Zur Bestimmung der Mindestanzahl der Fahrzeuge, welche benötigt werden, um einen berechneten Fahrplan in der Realität zu fahren, wurde das Verfahren von Grötschel (siehe [Groetschel96]) implementiert. Dieses baut auf Grundlage der berechneten Fahrgastfahrten einen Graphen auf. Wendet man auf diesen einen minimalen Kostenflussalgorithmus (bei vorheriger Definition der jeweiligen Kosten auf den Kanten) an, entsprechen die minimalen Kosten der Mindestanzahl benötigter Fahrzeuge.

Bevor der Aufbau des Umlaufgraphen (siehe Kapitel 5.12) beschrieben wird, erfolgen zuerst zwei Begriffsdefinitionen, so dass auch in den nachfolgenden Kapiteln ein einheitliches Begriffsinstrumentarium geschaffen ist.

Bei einer Fahrgastfahrt handelt es sich um die Fahrt eines Busses auf einer Linie von seiner Anfangs- zu seiner Endhaltestelle bezüglich eines bestimmten Zeitpunktes. Die Dauer einer Fahrgastfahrt ergibt sich aus der Summe aller Fahrten zwischen allen Haltestellen der Linie zuzüglich eventueller Wartezeiten des Busses.

Unter einer Leerfahrt versteht man eine Fahrt, welche zwei Fahrgastfahrten verbindet. Auf einer Leerfahrt werden keine Fahrgäste transportiert.

Um die Fahrzeugumlaufplanung durchzuführen, wird ein gerichteter, kantenbewerteter und knotenbewerteter Graph $G_U = (V_U, E_U, g_U, 1, f_U, 3)$ entsprechend 5.2.1 aufgebaut. Die Bedeutung der Symbole lässt sich wie folgt erklären.

V_U ist die Vereinigung des Quelldepotknotens, des Senkedepotknotens, sowie der Start- und Endhaltestellen aller Fahrgastfahrten.

$E_U = E_E \cup E_F \cup E_L \cup E_A \cup E_R$. Die Kanten sind wie folgt definiert:

- E_E sind die sogenannten Einsetzfahrten. Hierbei handelt es sich um Fahrten vom Quelldepotknoten zur Starthaltestelle einer Fahrgastfahrt.

- E_F sind die Fahrgastfahrten. Diese Kanten verbindet Start- und Endhaltestellen von Fahrgastfahrten.

- E_L sind Leerfahrten. Kanten dieser Menge verbinden Endhaltestellen einer Fahrgastfahrt mit Starthaltestellen einer anderen Fahrgastfahrt. Eine derartige Verbindung kann nur in Kraft treten, falls der Anfangsknoten dieser Kanten eine Uhrzeit verkörpert, die kleiner (bzw. früher) ist als die Uhrzeit des Endknotens.

- E_A sind die sogenannten Aussetzfahrten bzw. Rückfahrten. Hierbei handelt es sich um Fahrten von Endhaltestellen einer Fahrgastfahrt zum Senkedepotknoten.

- E_R ist die Rückfluss-Kante. Diese verbindet den Senkendepotknoten mit dem Quelldepotknoten.

g_U bildet die Knotenmenge des Graphen auf entsprechende Uhrzeiten ab, wobei für den Quelldepotknoten und Senkedepotknoten diese Werte unerheblich sind. Für den Knoten, der den Beginn einer Fahrgastfahrt symbolisiert, gibt die Uhrzeit den Start dieser Fahrt an und analog gibt der Endknoten einer Fahrgastfahrt das Ende dieser an.

Um auf diesem Graphen einen minimalen Kostenfluss berechnen zu können, sind für jede Kante noch Kapazitätsschranken, sowie die Kosten der entsprechenden Kante zu setzen. Dabei sei l_e die untere Kapazitätsschranke der Kante e, u_e die obere Kapazitätsschranke der Kante e und c_E die Kosten der Kante e.

$$f_U(e) = \begin{cases} l_e = 0, u_e = 1, c_e = 1, \text{falls} e \in E_E \\ l_e = 1, u_e = 1, c_e = 0, \text{falls} e \in E_F \\ l_e = 0, u_e = 1, c_e = 0, \text{falls} e \in E_L \\ l_e = 0, u_e = 1, c_e = 1, \text{falls} e \in E_A \\ l_e = 0, u_e = \infty, c_e = 0, \text{falls} e \in E_R \end{cases}$$

Die Funktion gibt also an, dass die Fahrgastfahrten auf jeden Fall bedient werden müssen, in dem die untere und obere Kapazität auf 1 gesetzt wird. Die Kosten werden auf die Einsetzfahrten bzw. Aussetzfahrten gelegt, da bei Gebrauch dieser Fahrzeuge benötigt werden, d.h. die Summe der Flüsse auf den Kanten der Aussetzfahrten bzw. Einsetzfahrten gibt die minimale Anzahl von Fahrzeugen an, mit der der Fahrplan durchführbar ist.

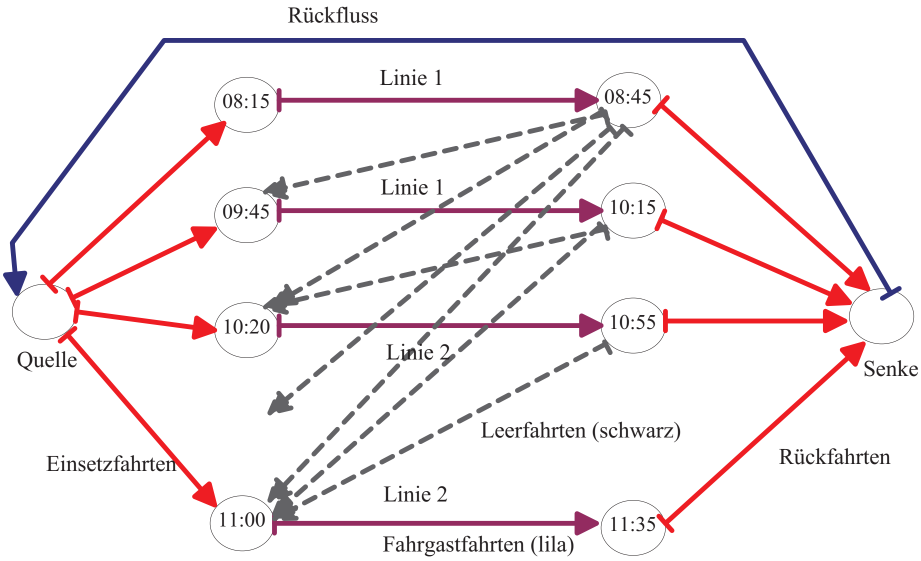
Die Rückflusskante ist nur vorhanden, um den Fluss in dem Senkedepotknoten abzuführen. Der Fluss auf dieser Kante gibt natürlich auch den minimalen Fluss wieder.

Die Leerfahrten werden mit Kosten 0 bewertet, da das Wählen dieser Vorteile dahingegend ergibt, das kein neues bzw. weiteres Fahrzeug verwendet werden muss.

Auf den so definierten Graphen wird ein minimaler Kostenflussalgorithmus angewendet, wobei der minimale Fluss die Anzahl benötigter Fahrzeuge widerspiegelt.

Diese oben geschilderten Gegegemheiten können anhand der folgenden Abbildung 5.12 nachvollzogen werden.

Figure 5.12: Graph der Fahrzeugumlaufplanung



Kapitel 6

Gütemaße

Um die Güte der vom gesamten Programm sowie den einzelnen Modulen berechneten Ergebnisse zu beurteilen, wird die Einführung bestimmter Werte notwendig. Im Folgenden werden diese Werte unter dem Begriff Gütemaße zusammengefaßt. Jedes Modul berechnet spezielle Gütemaße, die eine Bewertung und Einordnung ihres Ergebnisses zulassen. Zudem werden aus diesen modulabhängigen Werten globale Gütemaße errechnet, die eine Bewertung des Gesamtergebnisses ermöglichen.

Die angestrebte Rückkopplung, die aus den einzelnen lokalen Optima der Module ein globales Optimum errechnen soll, kann die Berechnung der Gütemaße nutzen. Hierzu sollen die Gütemaße des Fahrplans hergenommen werden und auf Schwachstellen, d.h. auf "schlechte" Werte, hin überprüft werden, um so einen Ansatz für einen erneuten Programmdurchlauf zu finden.(siehe Kapitel 4.3.4)

Die einzelnen Werte und ihre Berechnung sollen im Folgenden aufgeteilt nach Modulen näher erläutert werden.

6.1 Gütemaße der Haltestellen

Vereinbarung

H_N — Entspricht der Anzahl der berechneten Haltestellen plus der Anzahl der übernommenen Pflichthaltestellen.

$B^>$ — Bevölkerungsunterversorgung, entspricht der Summe der nicht versorgten Bevölkerung.

$B^<$ — Bevölkerungsüberversorgung, entspricht der Summe der mehrfach versorgten Bevölkerung.

$A^>$ — Flächenunterversorgung, entspricht der Summe der nicht versorgten Fläche.

$A^<$ — Flächenüberversorgung, entspricht der Summe der mehrfach versorgten Fläche.

S ist die Matrix, welche alle Entfernungen zwischen den Haltestellen auf direktem Wege (Direktfahrt) enthält. Die Einträge in der Diagonalen enthalten -1.

$S_{i,j}$ entspricht der Entfernung, ermittelt durch Direktfahrt, von Haltestelle i zu Haltestelle j.

Weitere Symbole sind vereinbart in 5.3, Seite 28 f.

Gütemaße

$s_{Fussweg}$ ist der durchschnittliche Fußweg in Metern. Wird definiert als:

$$s_{Fussweg} := \frac{\sum_{y=Norden}^{Sueden} \sum_{x=Westen}^{Osten} (\min\{d_{xy,p} : p \in H\} * B_{xy})}{H_N * B}$$

$t_{Fussweg}$ ist die Zeit in Sekunden, welche ein Fußgänger für $s_{Fussweg}$ benötigt. Dabei wird eine durchschnittliche Geschwindigkeit von $4\frac{km}{h}$ angenommen.

$B^{\%}$ Entspricht dem Anteil der versorgten Bevölkerung in Prozent.

$$B^{\%} := 100 * \frac{B}{B - B^{>}}$$

$A^{\%}$ Entspricht dem Anteil der versorgten Fläche in Prozent.

$$A^{\%} := 100 * \frac{A}{A - A^{>}}$$

6.2 Gütemaße des Linienplans

Vereinbarungen

N Anzahl der Verbindungen aller Haltestellen untereinander

$$N = H_N^2 - H_N$$

$U_{i,j}$ Anzahl der Umstiege der Verbindung von i nach j

d_{i,j_N} Entfernung auf dem kürzesten Weg von i nach j im Netzplangraph

v_L durchschnittliche Geschwindigkeit eines Busses
Sie wird benötigt, um die Entfernungen zu Fahrzeiten umzurechnen, und ist standardmäßig auf 15 $\frac{km}{h}$ gesetzt.

t_{i,j_N} Reisezeit auf dem kürzesten Weg von i nach j im Netzplangraph

$$t_{i,j_N} = \frac{d_{i,j_N}}{v_L}$$

G Anzahl aller Fahrgäste

D Anzahl der Direktfahrer

F_U Anzahl der Fahrten/Verbindungen mit Umstiegzwang

VG_m Verknüpfungsgrad einer Linie m
Der Verknüpfungsgrad gibt an, mit wievielen anderen Linien sich die Linie m kreuzt. Als optimal wird angesehen, wenn sich eine Linie mit allen anderen Linien genau einmal kreuzt.

d_{i,j_L} Entfernung auf den Linien von i nach j

t_U Strafzeit für einen Umstieg. Beträgt 3 Minuten.

t_{i,j_L} Reisezeit auf den Linien von i nach j

$$t_{i,j_L} = \frac{d_{i,j_L}}{v_L} + U_{i,j} * t_U$$

t_L durchschnittliche Reisezeit auf den Linien von i nach j

$$t_L = \sum_{i=1}^{H_N} \sum_{j=1}^{H_N} \frac{t_{i,j_L}}{N}$$

Gütemaße

U_N Anzahl der Umstiege aller Verbindungen

$$U_N := \sum_{i=1}^{H_N} \sum_{j=1}^{H_N} U_{i,j}$$

$\oslash_U$ durchschnittliche Anzahl der Umstiege pro Verbindung

$$\oslash_U = \frac{U_N}{N}$$

D_G Direktfahreranteil

$$D_G = \frac{D}{G}$$

Der Direktfahreranteil ist das zentrale Gütemaß des Linienplans, da seine Maximierung das Ziel des Algorithmus von Sonntag ist.

F_{U_N} Fahrten mit Umstieg in Prozent

$$F_{U_N} = \frac{F_U}{N}$$

l_G Gesamtstreckenkilometer

l_L Streckenkilometer der errechneten Linien

t_{MR} durchschnittliche Mindestreisezeit

$$t_{MR} = \sum_{i=1}^{H_N} \sum_{j=1}^{H_N} \frac{t_{i,j_N}}{N}$$

Die Mindestreisezeit ist die Reisezeit auf dem kürzesten Weg von i nach j.

$V_{N,L}$ Reisezeitverlängerung

$$V_{N,L} = \frac{t_{MR}}{t_L}$$

Die Reisezeitverlängerung gibt das Verhältnis von Mindestreisezeit zu der Reisezeit auf den Linien an. Da die Linien die Ströme bündeln, fahren die Gäste nicht mehr auf den kürzesten Wegen. Hier wird errechnet, wie groß der Umweg geworden ist.

VG_{N_L} mittlerer Verknüpfungsgrad

$$VG_{N_L} = \frac{\sum_{m=1}^{N_L} VG_m}{N_L}$$

Der mittlere Verknüpfungsgrad gibt an, mit wievielen anderen Linien sich eine Linie durchschnittlich kreuzt.

6.3 Gütemaße des Fahrplans

Die Gütemaße des Fahrplan teilen sich in zeitliche und betriebliche Gütemaße auf. Zu den zeitlichen Gütemaßen gehören die *netzbedingte Wartezeit*, der *Wartezeitaufwand*, der *Mindestzeitaufwand* und der *gesamte Wartezeitaufwand*. Zu den betrieblichen Gütemaßen zählen die benötigte *Anzahl fahrender Busse* und der *Fahrplanwirkungsgrad*. Im folgenden wird auf die einzelnen Gütemaße näher eingegangen.

Vereinbarungen

Gegeben sei der Transportkettengraph $G_T = (V_T, E_T)$ (siehe Kapitel 5.5.4) und Funktionen

$$f_{Netz} : E_T \rightarrow \mathbb{N}$$

$$f_{MindestHalte} : E_T \rightarrow \mathbb{N}$$

$$f_{MindestUm} : E_T \rightarrow \mathbb{N}$$

$$f_{Verkehr} : E_T \rightarrow \mathbb{N}$$

f_{Netz} gibt die netzbedingte Wartezeit für jede Kante des Netzes der Transportverbindungen an.

$f_{MindestHalte}$ symbolisiert die Abbildung der Kanten auf die Mindesthaltezeit.

$f_{MindestUm}$ stellt die Mindestumsteigezeit für jede Kante zur Verfügung.

$f_{Verkehr}$ bildet die Kanten auf ihren jeweiligen Verkehrsstrom ab.

Gütemaße

Die *netzbedingte Wartezeit* ergibt sich aus der Summe aller netzbedingter Wartezeiten im gesamten Netz und sagt somit etwas über die zusätzliche Wartezeit der Fahrgäste aus. Somit ergibt sich die *netzbedingte Wartezeit* $t_{netzGesamt}$ wie folgt:

$$t_{netGesamt} := \sum_{e \in E_T} f_{Netz}(e)$$

Der *Wartezeitaufwand* ergibt sich aus dem Produkt der *netzbedingten Wartezeit* und der *Anzahl der Fahrgäste* für alle Bindungen im Netz. Damit ist dieser Wert besser geeignet für eine Beurteilung der *netzbedingten Wartezeit*. Der *Wartezeitaufwand* $t_{Wartezeitaufwand}$ errechnet sich demzufolge als:

$$t_{Wartezeitaufwand} := \sum_{e \in E_T} (f_{Netz}(e) * f_{Verkehr}(e))$$

Der *Mindestwartezeitaufwand* $t_{Mindestzeitaufwand}$ ergibt sich aus der Zeit, welche die Fahrgäste für Umstiege benötigen und aus der Haltezeit der Busse an den einzelnen Haltestellen gewichtet mit dem Verkehrsstrom. Die Formel für die Berechnung von $t_{Mindestzeitaufwand}$ lautet:

$$t_{Mindestzeitaufwand} := \sum_{e \in E_T} ((f_{MindestHalte}(e) + f_{MindestUm}(e)) \cdot f_{Verkehr}(e))$$

Der *gesamte Wartezeitaufwand* $t_{GesWartezeitaufwand}$ berechnet sich aus der Summe des *Wartezeitaufwandes* und des *Mindestwartezeitaufwandes*:

$$t_{GesWartezeitaufwand} := t_{Wartezeitaufwand} + t_{Mindestzeitaufwand}$$

Aus diesen Gütemaßen läßt sich nur schwer etwas über einen Fahrplan aussagen. Sie dienen eher als Vergleichsmittel zwischen unterschiedlichen Fahrplanversionen.

Die betrieblichen Gütemaße sagen mehr über den Fahrplan aus. Sie geben an, wie teuer die Umsetzung des Fahrplans sein wird.

Hierzu ist zu allererst die minimale *Anzahl der Busse* n_{Busse} zu nennen, mit denen der berechnete Fahrplan gefahren werden kann. Die Erläuterung dieser Berechnung kann aus dem mathematischen Modell des Fahrplans (siehe Kapitel 5.5.9) und der entsprechenden Realisierung (siehe Kapitel 7.7) ersehen werden.

Der *Fahrplanwirkungsgrad* gibt das Verhältnis von der Dauer der Fahrgastfahrten zu der Dauer der Leerfahrten und Fahrgastfahrten an. Eine Fahrgastfahrt gibt die Fahrt eines Busses einer Linie von seiner Anfangshaltestelle bis zur Endhaltestelle an. Die Zeit einer Fahrgastfahrt stellt die gesamte Fahrtdauer, einschließlich Wartezeiten, dieser Haltestellenverbindungen dar. Die Fahrtdauer eines Busses von seiner Endhaltestelle zu dem Anfangspunkt seiner neuen Fahrgastfahrt, symbolisiert den Zeitraum einer Leerfahrt, d.h. eine Leerfahrt ist eine Busfahrt ohne Fahrgastbeförderung. Beachtet werden muss hierbei, dass auch die Standzeit des Busses bis zu dem Beginn der neuen Fahrgastfahrt in die Dauer der Leerfahrt eingerechnet wird (siehe auch Kapitel 5.5.9 und Kapitel 5.5.9).

Mathematisch gesehen ergibt sich der *Fahrplanwirkungsgrad* $f_{Wirkungsgrad}$ aus:

$$f_{Wirkungsgrad} = \frac{\sum_{fahrgastfahrten} f(fahrgastfahrten)}{\sum_{fahrgastfahrten} f(fahrgastfahrten) + \sum_{leer} g(leer)}$$

Es gilt hierbei:
f : ist eine Funktion, die die Fahrgastfahrten auf ihre Fahrtdauer abbildet
g : stellt eine Abbildung der Leerfahrten auf ihre Dauer dar

6.4 Globale Gütemaße

Die globalen Gütemaße berechnen sich aus den Gütemaßen der Module Haltestelle, Linienplan und Fahrplan.

Vereinbarungen

t_W durchschnittliche Wartezeit. Die Berechnung stellt sich wiefolgt dar:

$$t_W := \sum_{i=1}^{H_N} \sum_{j=1}^{H_N} W_{i,j}$$

Dabei ist W die Matrix der Wartezeiten. Der Wert an der Stelle i,j stellt die Wartezeit auf der Verbindung von der Haltestelle i zur Haltestelle j dar.

t_{df} die durchschnittliche Fahrtzeit der Direktfahrer von jeder Haltestelle zu jeder anderen

$$t_{df} := \frac{(\Sigma_{i=1}^{H_N} \Sigma_{j=1}^{H_N} S_{i,j})}{H_N^2 - H_N}$$

Dabei ist S die Direktfahrermatrix (siehe Kapitel 6.1)

$v_{mRz\,zu\,Df}$ das Verhältnis der mittleren Reisezeit zu den Direktfahrten mit dem Auto

$v_{MR\,zu\,Df}$ das Verhältnis der durchschnittlichen Mindestreisezeit zu den Direktfahrten mit dem Auto

n_{dBusse} die durchschnittliche Anzahl der Busse pro Linie

f die Anzahl der Stunden, für die der Fahrplan berechnet wird

$n_{Personen}$ die Gesamtzahl der zu befördernden Personen pro Tag

$$n_{Personen} := \frac{(\Sigma_{i=1}^{H_N} \Sigma_{j=1}^{H_N} F_{i,j})}{150}$$

Die Division durch 150 muss vorgenommen werden, da immerhalb der Verkehrsbedarfsmatrix F die gesamte Bevölkerung als potentielle Fahrgäste angesehen wird.

n_{PerBus} die durchschnittliche Anzahl der Personen pro Bus

Gütemaße

Der aussagekräftigste Wert ist hier die mittlere Reisezeit t_{mR}. Sie setzt sich zusammen aus der durchschnittlichen Zeit für den Fußweg zu den Haltstellen und der durchschnittlichen Wartezeit, sowie der durchschnittlichen Mindestreisezeit:

$$t_{mR} := t_{Fussweg} + t_W + t_{MR}$$

Die mittlere Reisezeit wird zudem in das Verhältnis zu den Direktfahrten gesetzt. Hiermit ist die durchschnittliche Fahrtdauer mit dem Auto von jeder Haltestellen zu jeder anderen gemeint:

$$v_{mRz\,zu\,Df} := \frac{t_{mR}}{t_{df}}$$

Durch diesen Wert wird über den öffentlichen Personennahverkehr hinaus ein Vergleich angestellt. Dies ermöglicht auch die Berechnung des Verhältnisses der durchschnittlichen Mindestreisezeit (siehe Kapitel 6.2) zu den Direktfahrten von jeder Haltestelle zu jeder anderen:

$$v_{MR\,zu\,Df} := \frac{t_{MR}}{t_{df}}$$

Schließlich werden zwei Werte berechnet, die im Hinblick auf die Kosten interessant sind. Einerseits gibt die durchschnittliche Anzahl Busse pro Linie eine grobe Auskunft über den Aufwand, andererseits wird durch den Wert der durchschnittlichen Personen pro Bus ein grober Eindruck über die Rentabilität der Fahrten vermittelt:

$$n_{dBusse} := \frac{n_{Busse}}{n_{Linien}}$$

$$n_{PerBus} := \frac{n_{Personen} * f/24}{n_{fahrgastfahrten}}$$

Die Anzahl der Personen $n_{Personen}$bezieht sich hier auf den ganzen Tag. Deshalb muss eine Normierung auf den Zeitraum, für den der Fahrplan berechnet wird, vorgenommen werden. Um n_{PerBus}zu berechnen, muss schließlich diese normierte Personenzahl durch die Anzahl der Fahrgastfahrten $n_{fahrgastfahrten}$ geteilt werden.

Als ein weiteres Gütemaß kann das Ergebnis der Routenplanung angesehen werden. Sofern die Fahrplanberechnung auf Basis der Haltestellen von Herne durchgeführt wird, kann mit Hilfe der Routenplanung eine Fahrt zwischen zwei Haltestellen berechnet werden. Die sich hieraus ergebende Fahrtzeit kann mit der Fahrtzeit verglichen werden, die im realen Busnetz von Herne existiert.

Kapitel 7

Realisierung

Nachdem in Kapitel 4 Algorithmen ausgewählt und in Kapitel 5 die mathematischen Modelle der verwendeten Verfahren vorgestellt wurden, soll dieses Kapitel aufzeigen, wie diese jeweils im Programm realisiert sind. Dazu werden zunächst in Kapitel 7.1 die in der Projektgruppe als Hilfsmittel zur Realisierung der Verfahren herangezogen Werkzeuge vorgestellt. Anschließend wird in Kapitel 7.2 die grundlegende Architektur des Programms erläutert. Sie gibt darüber Aufschluss, wie die einzelnen ausgewählten Verfahren miteinander verknüpft werden. Die dort eingesetzte Komponentenstruktur mündet in einer Aufspaltung in einzelne Module. In Kapitel 7.3 erfolgt eine Auflistung der modul- und verfahrensübergreifenden Datenstrukturen. Sie dienen dem Austausch von Daten zwischen den einzelnen Komponenten. Die abschließenden Kapitel 7.2 bis 7.7 gehen detailliert auf die Realisierung, den Ablauf und, sofern möglich, die Zeitkomplexität der so gebildeten Module ein.

7.1 Verwendete Werkzeuge

Dem Projekt liegen folgende Werzeuge und Hilfsmittel zugrunde:

- Together

  Together ist ein kommerzielles Werkzeug der Firma TogetherSoft zur Erstellung von Klassendiagrammen in der Modellierungssprache UML (siehe Kapitel G). Mit ihm wurde während der Analysephase die Softwarearchitektur, die Klassen und ihre Beziehungen untereinander festgelegt. [Fowler98]

- Geographic Ressources Analysis Support System (GRASS)

  GRASS ist ein Geo-Informationssystem, das in der Version 5.0beta8 zur Verfügung steht. Mit dem, maßgeblich von der Universität Hannover vorangetriebenen, frei verfügbaren Projekt lassen sich geografische Informationen sowohl verwalten als auch verarbeiten. [GRASS5]

- Oracle 7.3.2

  Ein kommerzielles Datenbanksystem der Oracle Corporation. Innerhalb des Projektes erfolgt der Zugriff auf die Datenbank über das Netzwerk.

- Library of Efficient Data types and Algorithms (LEDA)

  Hierbei handelt es sich um eine kommerzielle Bibliothek der Algorithmic Solution GmbH, die verschiedene, vorgefertigte Datenstrukturen und Algorithmen für, in C++ geschriebene Programme bereit hält. Dem Projekt liegt die Version 4.2 zugrunde. [LEDA4.2]

- Toolbox for Evolutionary Algorithms (TEA)

  Dies ist eine Bibliothek, welche am Lehrstuhl Systemanalyse, Fachbereich Informatik, der Universität Dortmund entwickelt wurde. Sie enthält Basis- und ergänzende Algorithmen, welche im Zusammenhang mit Evolutionären Algorithmen stehen. Sie wurde selbst in C++ entwickelt und lag dem Projekt in der Version TEA-small-2.40 vor. [TEA24]

- Concurrent Version System (CVS), grafisches Frontend: TkCVS

  Das Concurrent Version System (CVS) ist ein unter der GNU Public License (siehe Kapitel G) veröffentlichtes Dateiverwaltungsystem. Es vereinfacht die Verwaltung eines Projektes, indem es die Versionierung der Quellcode-, und Dokumentationdateien automatisiert. Darüber hinaus unterstützt es die Bearbeitung eines Projektes durch mehrere Personen. Als grafische Oberfläche dient TkCVS, ein auf Tcl/Tk basierendes, grafisches Frontend.

- make

  Make ist ein Programm, mit dessen Hilfe automatisch festgestellt werden kann, ob Quellcodedateien neu compiliert werden müssen. Liegt dieser Fall vor, wird ein entsprechender Compiler-Aufruf veranlasst. Die Abhängigkeiten werden dabei über die Textdatei "makefile" gesteuert. Veröffentlicht wurde make unter der GNU Public License.

- GNU C++ Compiler (gpp)

  Ein Übersetzer für Programme, die in der Sprache C++ verfasst wurden. Wie bereits make, wurde auch gpp unter der GNU Public License veröffentlicht.

- Tool Command Language (Tcl/Tk) mit EmbeddedTk (ET)

  Die Tool Command Language ist eine eigenständige, interpretierte Skriptsprache, mit dessen Hilfe komfortabel grafische Oberflächen erzeugt werden können. Aus diesem Grund wird zusätzlich EmbeddedTk benötigt. Hierbei handelt es sich um einen Übersetzter, der Tcl/Tk Quellcode in C, bzw. C++ Programme eingebettet.

7.2 Architektur

In diesem Kapitel wird der Aufbau und das Design des Projektes erklärt. Es wird das Zusammenspiel der einzelnen Module dargestellt und welche Aufgaben die jeweiligen Module im Gesamtzusammenhang erfüllen.

Die Aufspaltung in die drei Problembereiche Haltestellenberechnung, Linienplanung und Fahrplanerstellung spiegeln sich in der gewählten Komponentenarchitektur wieder. Halifa besteht aus den vier Komponenten Programm, Haltestelle, Linienplan und Fahrplan, die im Folgenden als Module bezeichnet werden. Die zuletzt genannten Module Haltestelle, Linienplan und Fahrplan können als Berechnungsmodule angesehen werden. Das Programm-Modul hat die Aufgabe, diese Module zu kontrollieren, ihre berechneten Daten weiterzuleiten, abzuspeichern und darzustellen. Es bildet somit eine Kontroll-Instanz über alle weiteren Module.

Damit die einzelnen Module miteinander Daten austauschen können, existiert ein allgemeines Modul Datenstruktur, welches von allen anderen Modulen gesehen und verwendet werden kann. In ihm sind alle gemeinsam verwendeten Datentypen enthalten, die über das Programmmodul ausgetauscht werden (vgl. Kapitel 7.3).

Innerhalb des Programmmoduls sind die Funktionen auf mehrere, spezialisierte Teilmodule verteilt: Innerhalb der Schnittstelle verwaltet die Kontrollklasse die Datenbank und die Schnittstellen. Darüber hinaus dient sie zur gezielten Steuerung der Berechnungsmodule. Die Schnittstelle selbst sorgt für den Datenaustausch zwischen den Berechnungsmodulen, der Datenbank und der Benutzerschnittstelle. Das GUI-Modul gewährleistet durch die Bereitstellung einer grafischen Oberfläche (GUI) die Interaktion des Programms mit dem Benutzer. Durch eine Kapselung der beiden Datenbanken GRASS und Oracle, regelt das dritte Modul die Kommunikation zwischen dem Programm und den genannten Systemen (vgl. Anhang D.2). Ein Überblick über die Komponentenarchitektur gibt Abbildung 7.1.

7.3 Datenstrukturen

In diesem Kapitel werden die Datenstrukturen aufgeführt, die modulübergreifend zum Einsatz kommen. Die in den einzelnen Modulen verwendeten Datenstrukturen sind in den jeweiligen Unterkapiteln 7.5 bis 7.7 zu finden. Ein detaillierter Aufbau der im Folgenden aufgeführten Klassen ist dem Anhang D.1 zu entnehmen.

Dem Projekt liegt die Programmierbibliothek LEDA ("Library of Efficient Data types and Algorithms" [LEDA4.2]) zugrunde. Sie definiert C++-Templates zu grundlegenden Datenstrukturen, wie z. B. Listen, Graphen oder Matrizen. Darauf aufbauend bietet sie Algorithmen aus verschiedenen Bereichen, wie z. B. die Berechnung der kürzesten Wege in einem Graphen.

Die wesentlichen Grundlagen für das Programm bilden die in LEDA enthaltene Liste (list<>), sowohl gerichtete, als auch ungerichtete Graphen (graph<>), die Matrix (matrix<>) und Arrays (array<>).

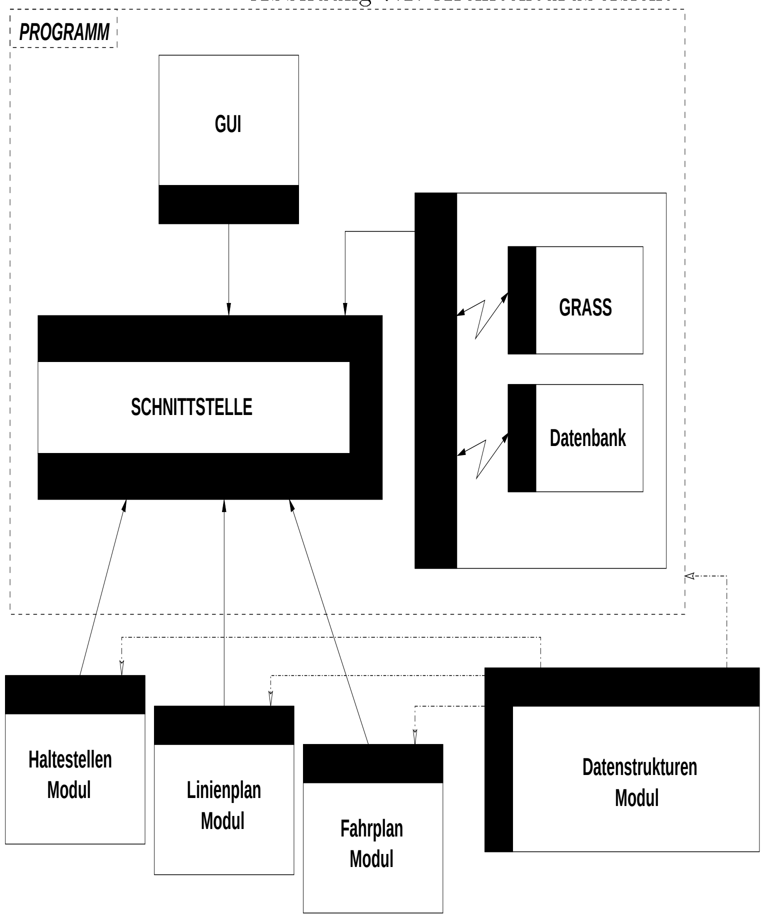
Um die Berechnungen transparenter zu gestalten, werden Haltestellen und Buslinien durch eindeutige Identifikationsnummern (ID) repräsentiert. Realisiert wurden diese Identifikationsnummern durch die Vergabe von ganzzahligen Werten in Form des C++ Datentyps int (vgl. Kapitel 7.4.1).

Die durch LEDA-Datenstrukturen realisierten, modulübergreifende Datenstrukturen besitzen folgenden Aufbau (Schreibweise folgt der C++ Syntax):

- QuelleZielMatrix

  Setzt sich zusammen aus einer Matrix, die Fließkommawerte doppelter Länge (C++-Typ double) beinhaltet (matrix<double>), als auch aus einer Liste, die Zeiger auf Objekte vom Typ CQuelleZielListeElement auf-

Abbildung 7.1: Architekturübersicht



nimmt (list<CQuelleZielListeElement*>). Diese Aufteilung ist notwendig, da ohne die zusätzlichen Daten innerhalb der Liste, eine Zuordnung der in der Matrix gespeicherten Werte zu den einzelnen Haltestellen nicht möglich wäre.

Die gespeicherten Werte innerhalb der Matrix für ein Paar (x,y) repräsentiert die Anzahl Personen, die von Haltestelle x zu Haltestelle y gelangen wollen. Die zugehörige Liste speichert die geografischen Koordinaten aller, in der QuelleZielMatrix vorhandenen Haltestellen. Das Haltestellenmodul reduziert bzw. erweitert die QuelleZielMatrix und QuelleZielListe auf die von ihm berechneten Haltestellen. Diese Daten dienen dem Linienplanmodul als ein weiteres Eingabedatum.

- Pendlermatrix

  Wie bereits die QuelleZielMatrix, setzt sich auch die Pendlermatrix zusammen aus einer Matrix, die Fließkommawerte in doppelter Länge speichert (matrix<double>), und aus einer Liste, die Zeiger auf Objekte vom Typ CPendlerListeElement aufnimmt (list<CPendlerListeElement*>).

  Die in der Matrix an der Position x,y gespeicherten Werte repräsentieren im Gegensatz zur QuelleZielMatrix die Anzahl Personen, die von einem Stadteil x in den Stadteil y gelangen möchte. Die zugehörige Pendlerliste gestattet auch in diesem Fall die Zuordnung der Matrixpositionen (x bzw. y) zu den Stadteilen. Dazu werden innerhalb der Liste die entsprechende Matrixpositionen, die geographischen Koordinaten des Stadteilmittelpunktes und der Radius, über den sich der Stadtteil erstreckt abgelegt.

- Bevölkerungsdichte

  Die Bevölkerungsdichte wird durch eine Liste repräsentiert, die Zeiger auf Objekte vom Typ CBevoelkerungsdichteElement aufnimmt (list<CBevoelkerungsdichteElement*>). Die einzelnen Objekte wiederum nehmen Daten über den geografischen Ort, sowie die dort herrschende Bevölkerungsdichte auf. Als geographische Daten dient neben einem festen Punkt eine Radiusangabe, für die die gespeicherte Bevölkerungdichte gilt.

- Wegenetzgraph

  Der Wegenetzgraph wird vom Haltestellenmodul erzeugt und dient als Ausgangsdatum für das Linienplanmodul. Darüber hinaus wird der Wegenetzgraph vom Programmmodul benötigt, um bei der Anzeige der berechneten Daten eine Zuordnung zwischen den von den Modulen verwendeten Identifikationsnummern und den realen Straßenzügen vornehmen zu können.

  Realisiert wird dieser Graph durch einen ungerichteten LEDA-Graphen, dessen Knotenbeschriftungen Zeiger auf Objekte des Typs CWegenetzKnoten und dessen Kantenbeschriftungen Zeiger auf Objekte des Typs CWegenetzKante tragen (GRAPH<CWegenetzKnoten*, CWegenetzKante*>).

- Fahrplangraph

Der Fahrplangraph wird vom Linienplanmodul errechnet und dient als Ausgangsdatum für das Fahrplanmodul. Analog zum Wegenetzgraphen wird dieser Graph durch einen gerichteten Graphen (GRAPH<CFPlanGraphKnoten*, CFPlanGraphKante*>) realisiert, dessen Knotenbeschriftungen Zeiger auf Objekte des Typs CFPlanGraphKnoten und dessen Kantenbeschriftungen Zeiger auf Objekte des Typs CFPlanGraphKante tragen.

- Buslinienliste

  Die Buslinienliste wird vom Linienplanmodul errechnet. Sie dient dem Programmmodul als Hilfe, um die visuelle Darstellung der errechneten Buslinien zu beschleunigen. Realisiert wird sie als eine Liste, die Zeiger auf Objekte vom Typ CLinie aufnimmt (list<CLinie*>).

- Pflichthaltestellen

  Hierbei handelt es sich um eine Liste von Zeigern auf Objekte des Typs CPunkt (list<CPunkt*>). Diese Liste wird vom Programmmodul erstellt und dient als ein Eingabedatum des Haltestellenmoduls. Sie enthält die geografischen Koordinaten der Haltestellen, die der Benutzer zuvor als Pflichthaltestellen in das System eingegeben hat.

- Pflichtlinien

  Analog zu den Pflichthaltestellen, handelt es sich bei den Pflichtlinien um eine Liste von Zeigern auf Objekte des Typs CLinie (list<CLinie*>). Diese Liste wird vom Programmmodul erstellt und dient als Eingabedatum des Linienplanmoduls. Sie enthält Linien (in Form einer Abfolge von Haltestellen-IDs, vgl. Anhang D.1), die der Benutzer zuvor als Pflichtlinien in das System eingegeben hat.

- Gütemaße

  Die Gütemaße werden durch ein Objekt der Klasse CGuetemasse repräsentiert. Vgl. hierzu Anhang D.2 und Kaiptel 6. Die einzelnen Gütemaße werden innerhalb der Klasse durch einfache C++ Datentypen (int, double) repräsentiert. Ausnahmen bilden die Autoreisezeiten (map2<int,int,double>) und die Reisebewertung (map2<int,int,CReisebewertung>).

- Modulparameter

  Die vom Benutzer einstellbaren Parameter, wie z. B. die Auswahl eines Algorithmus, werden jeweils einzeln, durch einfache C++ Datentypen (int, double, bool) realisiert. Eine Ausnahme bildet hier das Haltestellenmodul, dessen gesamte Eingabe über ein extra für diesen Zweck geschaffenes Objekt der Klasse CHaltestellenModulParameter erfolgt (vgl. Anhang D.1).

- Zusammenhangskomponente

  Die Zusammenhangskomponente wird als ein Ergebnis der Fahrplanberechnung an das Programmmodul zurück geliefert. Die Zusammenhangskomponenten werden durch ein Array repräsentiert, das Zeiger auf eine Liste von Zeigern auf Dreitupel aufnehmen kann (array<list<three_tuple<int,int,int>* >* >). Ein Dreiertupel beinhaltet ganzzahlige Werte (int), die wiederum zwei Haltestellen- und eine Linienidentifikationsnummer repräsentieren.

- direkter Anschluss

  Ein direkter Anschluss gibt Auskunft über eine Haltestelle, an der eine Buslinie auf eine Zweite warten soll. Diese Informationen dienen dem Fahrplanmodul als Eingabe und werden realisiert als eine Liste von Viertupeln, die wiederum ganzzahlige Werte (int) aufnimmt (list<four_tuple<int,int,int,int>* >). Der erste Wert des Viertupels gibt Aufschluss über die Zusammenhangskomponente (Identifikationsnummer), die beiden beteiligten Linien und die Haltestelle, die von dem Umsteigevorgang betroffen ist.

- Abfahrtszeiten

  Die vom Benutzer eingegeben Abfahrtszeiten werden innerhalb des Programms durch ein Array von Zeigern auf Dreitupeln realisiert (array<three_tuple<int,int,int>* >). Ein Dreitupel nimmt ganzzahlige Werte auf, welche eine Haltestellen-ID, eine Liniennummer und eine Minutenangabe repräsentieren. Die gespeicherten Daten dienen dem Fahrplanmodul als Eingabedatum.

- Mindesthaltezeit

  Die Mindesthaltezeiten für alle Haltestellen werden innerhalb eines Hash-Arrays verwaltet (h_array<int,int>) und dienen dem Fahrplanmodul als Eingabedatum. Die Integerwerte repräsentieren die Haltestellennummer und die dort geltende Haltezeit.

- Mindestumsteigezeit

  Die Mindestumsteigezeit, die dem Fahrplanmodul als Eingabe dient, gibt zu jeder Haltestellen-Buslinienkombination die dort herrschende minimale Umsteigezeit an (h_array<int,list<three_tuple<int,int,int>* >*>). Das erste Element im Hash-Array spiegelt die Haltestellen-ID wieder. Beim zweiten Element handelt es sich um eine Liste, bestehend aus erster und zweiter Liniennummer, sowie der Mindestumsteigezeit zwischen diesen Linien.

- Fahrplanintervall

  Entspricht dem Intervall, für das ein Fahrplan erstellt werden soll und dient somit dem Fahrplanmodul als Eingabe. Die Speicherung der Daten erfolgt innerhalb eines Viertupels (four_tuple<int,int,int,int>). Dabei gilt folgende Reihenfolge bei den Integer-Werten: Startzeit (Stunde, Minute), Endzeit (Stunde, Minute).

- feste Verbindungen

  Die festen Verbindungen dienen dem Fahrplanmodul als Eingabedatum. Realisiert werden sie als Liste von Zeigern auf Viertupel (list<four_tuple<int, int, int, int>* >). Die Integerwerte nehmen dabei die Zusammenhangskomponenten-ID, die IDs der ersten und zweiten Linie, sowie die Haltestellen-ID, an der der Umstieg stattfindet, auf.

- Fahrplan

  Der Fahrplan wird durch das Fahrplanmodul errechnet und in Form einer Liste, die wiederum Zeiger auf ein Objekt des Typs CFahrplanElement (list<CFahrplanElement*>) aufnimmt, an das Programmmodul zurück geliefert.

7.4 Umsetzung des Programms

Dem Programmmodul fällt in der gewählten Komponentenarchitektur die Rolle des Vermittlers zu. Desweiteren ist dieses Modul für eine Steuerung der drei Berechnungsmodule Haltestellen, Linienplan und Fahrplan zuständig. Dieses Kapitel soll zunächst darstellen, wie diese geforderten Aufgaben innerhalb des Projektes realisiert wurden. Anschließend wird in Kapitel 7.4.1.1 erläutert, wie die einzelnen Komponenten des Programmmoduls während der Ausführung sowohl untereinander als auch nach außen mit dem Benutzer interagieren.

7.4.1 Realisierung

Dem Programmmodul fallen folgende Aufgaben zu:

- Datenbeschaffung
- Datenhaltung und Versionierung
- grafische Datenaus- und Eingabe
- Steuerung des Datenflusses
- Steuerung der einzelnen Module

Die Realisierung dieser Bereiche erfolgt innerhalb des Projektes auf folgende Weise:

Als Grundlage aller Berechnungen dienen neben den geografischen Daten von Herne (in Form des Stadtplanes), auch Daten über die Lage der realen Herner Haltestellen im Jahre 1996, die Bevölkerungsdaten (Anzahl innerhalb der Stadtteile dauerthaft lebender Personen), sowie das Ergebnis einer Fahrgastzählung (Quelle-Ziel-Matrix, vgl. Anhang G).

Durch die innerhalb des Programms verwendeten Datenstrukturen (vgl. Kapitel 7.3) werden die Haltestellen und Buslinien über eindeutige Identifikationsnummern (ID, realisiert durch ganzzahlige Werte) gekennzeichnet. Die Umlage der Haltestellennamen auf diese Identifikationsnummern geschieht dabei nach folgendem Schema:

Die Haltestellen von Herne liegen nur in gedruckter Form auf einem von der HCR zur Verfügung gestellten, großen Stadtplan vor. Sofern es keine Möglichkeit gibt, diese Daten mittels Digitalisiertablett einzulesen, bleibt nur das Verfahren der manuellen Erfassung. Ohne Hilfestellung einen Punkt der HCR-Karte in GRASS wiederzufinden, gestaltet sich in der Praxis als extrem schwierig. Um diesen Misstand zu beheben, kann eine entsprechende Funktion innerhalb des Programms durch den Benutzer in Anspruch genommen werden. (Funktion "Zeige Punkt" im Anzeigenmenü; vgl. Kapitel 7.4.1.1).

Durch Einblenden der Gebäude und Hauptstraßen in GRASS, und der Anzeige der Herne-Komplettansicht konzentriert sich der Benutzer zunächst auf das Autobahnkreuz. Dieses ist sowohl in GRASS als auch auf der HCR-Karte eindeutig identifizierbar. Anschließend wird der Teil um das Autobahnkreuz vergrößert und nach einer eindeutigen Stelle gesucht. Dies kann z.B. eine Straßenkreuzung sein, an der ein Gebäude mit einer charakteristischen Form steht. Wurde diese Stelle sowohl in GRASS, als auch auf der HCR-Karte gefunden, so werden diese Koordinaten auf der Herner Karte mit einem Lineal ausgemessen. Anschließend wird die Maus in GRASS genau an dieser Stelle positioniert. Per GRASS-Kommando werden die Koordinaten abgefragt. Der Maßstab der HCR-Karte beträgt 1:20000. 1 cm auf der HCR-Karte entspricht somit 200 m. Auf diese Weise wird der Punkt 0,0 auf der HCR-Karte ermittelt (GRASS-Koordinaten - HCR-Koordinaten * 200). Zur Kontrolle wird dieser Vorgang mit einem anderen Punkt wiederholt. Der Punkt 0,0 auf der von der HCR zur Verfügung gestellten Karte wurde auf diese Weise mit [2578135,5709021] festgelegt.

Die GRASS-Koordinaten eines Punktes auf der HCR-Karte berechnen sich demnach wie folgt: Durch die Eingabe eines Punktes z.B. [10,0cm 15,0cm] im Dialogfenster von Zeige Punkt, erhält man die GRASS-Koordinaten [2580135,5712021]. Die Region wird folgendermaßen eingestellt: Nord: ywert+500, Süd: ywert-500, West: xwert-800, Ost: xwert+800.

"Anzeige aktualisieren" wird aufgerufen. Die Punktkoordinaten werden in einen temporären Sites-Layer (Layer, der zur Aufnahme von Punktdaten dient, siehe G) geschrieben, welcher anschließend angezeigt wird, so dass der Punkt in Form eines gelben Kreuzes zu sehen ist. Der gesuchte Punkt findet sich als Resultat genau in der Mitte des Bildschirms wieder. Nördlich und südlich verbleiben 500 Meter bis zum Bildschirmrand, westlich und östlich noch 800 Meter. Mit der Funktion "Haltestellen setzen" kann im Anschluss die entsprechende Haltestelle gesetzt werden. Hierbei sollte wieder auf charakteristische Kreuzungen und Gebäude geachtet werden, wobei das gelbe Kreuz als Orientierung dient. Die gesuchte Haltestelle muss sich ganz in der Nähe des Kreuzes befinden, je nachdem wie genau der Messvorgang erfolgt ist. Die Praxis hat gezeigt, dass die Abweichung meist nur wenige (<20) Meter beträgt.

Innerhalb des Programms werden die nach dem genannten Schema vorbereiteten Ausgangsdaten durch das GUI-Modul eingelesen und verarbeitet. Die dazu benötigten Dateiformate sind im Anhang A aufgeführt.

Dem Benutzer liefert die GUI eine Menüleiste und ein Anzeigefenster (siehe Diagramm 7.2 auf Seite 80). Über die Menüleiste können alle Programmfunktionen angesprochen werden. Dazu zählen die Aufrufe der einzelnen Module, das Navigieren innerhalb der Anzeige, die Auswahl der anzuzeigenden Elemente, sowie das Löschen, Hinzufügen und Verschieben einzelner Elemente. Die Menüstruktur, die Eingabefelder, Auswahlfelder und weitere GUI Objekte sind in Tcl/Tk programmiert. Der Aufbau der grafischen Oberfläche versucht die fehlenden Funktionen und Eigenschaften des von der Straßenbahn Herne-Castrop-Rauxel GmbH (HCR) verwendeten Systems, sowie des vorliegenden Geo-Informationssystems (GIS) zu kompensieren (vgl. Anhang B).

Bei GRASS handelt es sich um ein Geo-Informationssystem (GIS), welches zur Anzeige und Speicherung der geographischen Daten genutzt wird. Es bietet zwar einen graphischen Monitor (Fenster, in dem geographische Vektor-, Raster-, und Punktdaten angezeigt werden können), jedoch steht keine eigene

Benutzeroberfläche zur Verfügung.

GRASS wird im Programmmodul auf zwei Arten verwendet. Zum einen greift die GUI-Klasse direkt auf die von GRASS zur Verfügung gestellten Befehle zu, zum anderen verwendet die Klasse CGRASSSchnittstelle (vgl. Anhang D.1) die von den entsprechenden API-Bibliotheken zur Verfügung gestellten C-Funktionen. Von der API (Application Programming Interface) werden lediglich die Funktionen zum Erzeugen, Löschen, Auslesen und Speichern der Vektor- ("arcs") und Punktdaten ("sites") verwendet. Bevor diese Funktionen in Anspruch genommen werden können, müssen zunächst die eingehenden LEDA-Datenstrukturen in die von GRASS benötigten Formate transformiert werden.

Da GRASS durch eine Shell realisiert wird und die Bedienung somit über Shellkommandos erfolgt, ist eine Steuerung über ein Skript in der Sprache Tcl/Tk möglich. Das GRASS beiliegendes Tool TclTkGRASS nutzt diese Möglichkeit und bietet mittels Tcl/Tk-Skript eine graphische Benutzeroberfläche. Nachdem GRASS gestartet wurde und der Benutzer die entsprechende Geodatenbank ausgewählt hat, ruft er TclTkGRASS auf. Eine Vielzahl der in GRASS enthaltenen Funktionen können von dort per Menü aufgerufen werden. In diesem Fall dient GRASS ausschließlich der Verwaltung, der Anzeige und dem Editieren der vorliegenden, geographischen Daten. Des weiteren sollte sich der Benutzer nicht mit der Bedienung von GRASS auseinandersetzen müssen. Statt dessen soll eine intuitive Oberfläche Programm- und Anzeigefunktionen zur Verfügung stellen. GRASS soll dabei möglichst unauffällig im Hintergrund operieren.

Um dies zu erreichen, müsste zunächst GRASS und von dort aus die Tcl/Tk-Shell (wish) mit dem Skript, das zur Erzeugung der grafischen Oberfläche notwendig ist (gui.tcl), gestartet werden. Diese umständliche Vorgehensweise kann vermieden werden, da die GRASS-Shell lediglich eine Erweiterung der bestehenden, vom Betriebssystem zur Verfügung gestellten Shell darstellt. Beim Start von GRASS werden zahlreiche Umgebungsvariablen hinzugefügt und geändert. Durch das Ergänzen entsprechender Kommandos im gui.tcl Skript werden beim Programmstart diese Erweiterungen vorgenommen und somit eine GRASS-Umgebung simuliert. Anschließend besteht eine gültige GRASS-Umgebung, in der sich GRASS-Befehle ausgeführt lassen.

Um die in Tcl/Tk geschriebenen Funktionen innerhalb des C++-Codes nutzen zu können, wird Embedded Tk (ET) verwendet. ET erlaubt das Einbetten von Tcl/Tk-Code in C/C++. Nachdem ET die Tcl/Tk-Anweisungen enthaltenden Quellcodedateien in reinen C bzw. C++ Quellcode übersetzt hat, entsteht nach dem Kompilierungsvorgang durch den C++ Compiler eine ausführbare Datei.

Verschiedene Daten werden von GRASS auf verschiedenen Ebenen ("Layern") gespeichert (vgl. Anhang G). Die hier zum Einsatz kommenden Straßendaten von Herne liegen auf zwei "Layern" vor, "Hauptstraßen" und "sonstige Straßen". Ein weiterer Layer enthält die Gebäude. Für die Pflichthaltestellen, die berechneten Haltestellen, den reduzierten Straßenplan und den Linienplan, sowie für jede Einzellinie wird jeweils ein weiterer Layer angelegt. Die Anzeigefunktion von GRASS ermöglicht es, Layer einzeln anzuzeigen oder verschiedenfarbig übereinander zu legen. Ebenso kann über die Koordinatenangabe bestimmt werden, welcher Bereich angezeigt werden soll. Die Position des Mauszeigers kann abgefragt werden. Auf diese Weise kann GRASS so gesteuert werden, dass es verschiedene Ansichten der vorliegenden Daten generieren, eine Zoomfunktion und die Eingabe weiterer geographischer Objekte z.B. Haltestellen ermöglichen kann. GRASS gestattet lediglich das Speichern geographischer Objekte und deren Koordinaten. Um weitere Daten zu diesen Objekten speichern zu können, wie z.B die Umsteigezeit zwischen zwei Linien oder der berechnete Fahrplan, wird eine zusätzliche Datenbank benötigt.

Darüber hinaus fallen während der Berechnung der einzelnen Module an unterschiedlichen Stellen nicht-geografische Daten an, die entweder während der aktuellen Berechnung weiter benötigt werden, oder aber bei einem späteren Programmablauf nochmals verwendet werden sollen. So ist es beispielsweise möglich, die Berechnung nach Abschluss der Linienplanung abzubrechen, die zuvor vom Haltestellenmodul angefallenen Daten zu sichern und mit diesen gespeicherten Daten zu einem späteren Zeitpunkt erneut die Linienplanung zu starten. Eine weitere Anforderung an das Programm besteht in der Einführung einer Versionierung. Nicht nur im Hinblick auf Testläufe (vgl. Anhang E) können hiermit verschiedene Berechnungsergebnisse zwischengespeichert und durch den Benutzer miteinander verglichen werden. Da GRASS keinerlei Unterstützung für diesen Versionierungsansatz bietet, müssen auch die geografischen Daten in einer zusätzlichen, externen Datenbank zwischengespeichert werden. Somit dient GRASS der reinen Speicherung und Anzeige der geografischen Daten, sowie zur Eingabe der Pflichthaltestellen und Pflichtlinien.

Als externe Datenbank liegt diesem Projekt eine Oracle Datenbank mit der Versionsnummer 7.3.2 zugrunde. Um die Funktionalitäten der Datenbank innerhalb des Programms nutzen zu können, wird Embedded SQL eingesetzt. Mit Hilfe dieser Technik können Datenbankanfragen in der Sprache SQL direkt innerhalb von C/C++ Quellcode eingefügt werden. Ein der Datenbank beiliegender Precompiler wandelt die entsprechenden SQL-Anweisungen in C bzw. C++ Befehle um, die anschließend durch den C++ Compiler in eine ausführbare Datei übersetzt werden können.

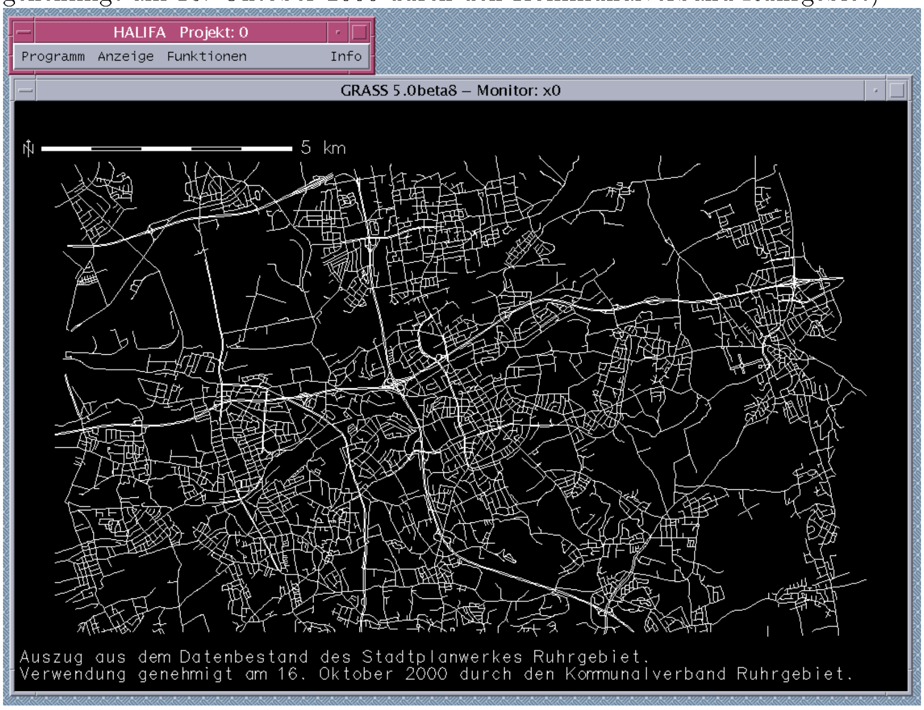
In der Architektur werden beide Datenbanken durch eine weitere, eigene Schnittstelle gekapselt. Auf diese Weise wird ein konsistenter Datenbankzugriff für alle anderen Komponenten des Programms sichergestellt. Die diese Schnittstelle repräsentierende Klasse (Datenbankklasse) ist dafür zuständig, die ankommenden Daten an eine der beiden entsprechenden Datenbanken weiter zu leiten. Analog müssen bei ankommende Anfragen aus den Schnittstellen die benötigten Daten aus den jeweiligen Datenbanken angefordert und nach den Wünschen der Schnittstellen in eine entsprechende Datenstruktur gebracht werden. Die beiden anderen Datenbankklassen bilden dabei die Programmier-Schnittstellen zu GRASS bzw. Oracle. (vergl. Kapitel C).

Die durch die Datenbanken und das GUI-Modul bereit gestellten Daten müssen durch einen geeigneten Mechanismus an die entsprechenden Stellen innerhalb des Programms weitergeleitet werden. Diese Aufgabe übernehmen die Schnittstellen. Jedes Modul verfügt dabei über seine eigene Schnittstelle. Letztere leiten Anfragen sowohl an die beiden Datenbanken, als auch die GUI weiter und führen eine Vermittlung zwischen den beteiligten Klassen durch. Auf diese Weise wird zusätzlich sicher gestellt, dass die Berechnungsmodule keinen direkten, unkontrollieren Zugriff auf die Datenbanken erhalten. Sobald eine Anfrage aus einem Modul eintrifft, leitet die Schnittstelle diese an die Datenbank oder die GUI weiter und liefert die Daten in der vom Modul gewünschten Form zurück. Dazu kennen die Schnittstellen sowohl die GUI, als auch die Datenbank.

Da alle Schnittstellenklassen den Zugriff auf die Datenbanken benötigen, kommt hier die objektorientierte Methode der Vererbung zum Einsatz. Da die Schnittstellenklassen für die einzelnen Module ähnliche Funktionalitäten übernehmen müssen, werden sie von einer entsprechenden Superklasse abgeleitet. Der GUI wird über eine eigene Schnittstelle ebenfalls ein Zugriff auf die Datenbank ermöglicht. Im Gegensatz zu den anderen Schnittstellenklassen wurde mit Hilfe einer entsprechenden Vererbungshierarchie darauf geachtet, dass diese Schnittstelle keinen direkten Zugriff auf sich selbst erhält. (Sicherheitsmaßnahme gegen Zyklen).

Die letzte Komponente im Programmmodul bildet die Kontrolle. Sie ist für die Erzeugung, Steuerung, und das Zerstören sowohl der Berechnungsmodule als auch der Datenbanken zuständig. Des weiteren hat die GUI mit Hilfe dieser Komponente die Möglichkeit, den Berechnungsvorgang der Module Haltestelle, Linienplan und Fahrplan sowohl zu starten als auch abzubrechen. Um die zuletzt genannte Funktionalität des Abbruchs zu gewährleisten, werden durch die Kontrollklasse, welche diese Komponente realisiert (CKontrolle, vgl. Anhang D.2), POSIX-Threads eingesetzt. Diese erlauben die Ausführung eines Berechnungsmoduls, während gleichzeitig die Ereigniswarteschlange der GUI nicht unterbrochen wird.

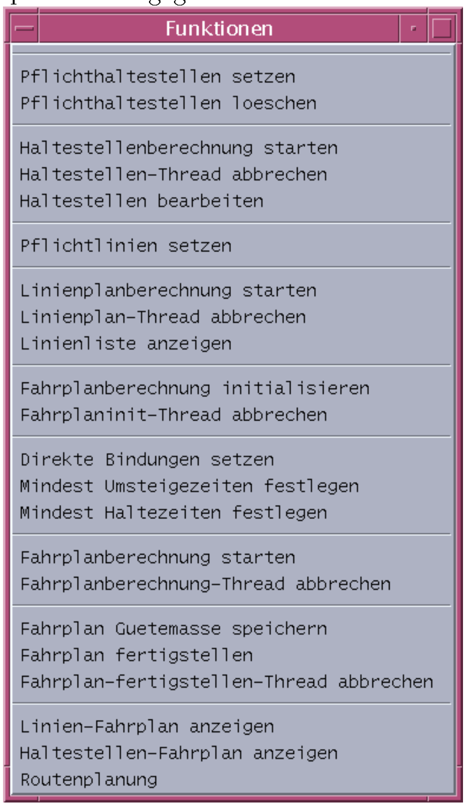
Abbildung 7.2: Der Aufbau des Bildschirms mit der Karte von Herne (Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet)



7.4.1.1 Ablauf

Um die grundlegenden Datenstrukturen und die Aufgabe der einzelnen Module aufzuzeigen, erfolgt an dieser Stelle eine kurze Beschreibung eines typischen Programmablaufs.

Abbildung 7.3: Das Funktionsmenü spiegelt innerhalb des Programms den sequentiell vorgegebenen Ablauf wieder.



Nach dem Programmstart wird eine Instanz der Klasse angelegt, welche die grafische Benutzeroberfläche (GUI) repräsentiert (CGUI, vgl. Anhang D.2). Diese prüft, ob das Programm mit einem Parameter gestartet wurde. Der Parameter "-test<Nr>" bewirkt, dass bei Programmausführung auf die Daten des Testgenerators zurückgegriffen wird. Dabei gibt <Nr> die Nummer des Testfalls an. Wird kein Parameter mitgegeben, so erfolgt ein herkömmlicher Programmaufruf, bei dem auf die realen Daten zugegriffen wird, welche in der Datenbank hinterlegt sind.

Auf eine GRASS-Datenbank kann zu einem Zeitpunkt nur ein einzelner Benutzer zugreifen. Beim Programmstart wird daher die GRASS-Datenbank aus dem gemeinsamen Verzeichnis der Projektgruppe in das temporäre Verzeichnis des entsprechenden Benutzers kopiert. Auf diesen Daten kann anschließend gearbeitet werden. Der Kopiervorgang wird nur gestartet, wenn die entsprechenden Daten noch nicht vorhanden sind, bzw. das Verzeichnis /tmp/<Benutzername>/spearfish noch nicht existiert. Ein alternativer Speicherort kann in der versteckten Datei .programm.ini im Heimatverzeichnis des

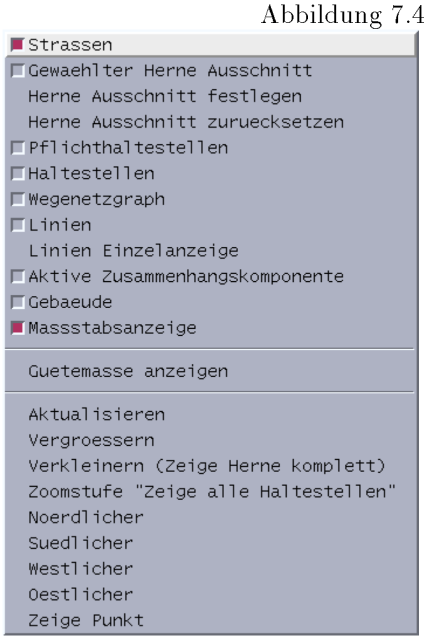
Benutzers angegeben werden.

Der Datenbankzugriff auf Oracle erfolgt unter dem Login-Namen. Somit wird sichergestellt, dass bei gleichzeitigen Testläufen verschiedener Benutzer kein ungewolltes Überschreiben der Daten stattfindet. Wird als Parameter ein Name, entweder der Benutzername oder ein anderer Name mitgegeben, so wird dieser an die Datenbank übergeben. Die Datenbankzugriffe erfolgen ab sofort unter diesem Namen.

Beispiel: Der Benutzer egon startet "./Programm" ohne Parameter. Die Datenbankzugriffe erfolgen dann unter dem Namen "egon". Ein Start durch den Benutzer egon von "./Programm hugo" bewirkt, dass die Datenbankzugriffe unter dem Namen hugo erfolgen.

Als nächstes wird die GRASS-Umgebung gesetzt. Der GRASS-Monitor wird mit einer Größe von 1000x640 Pixeln gestartet und die Menüleiste wird am linken Rand darüber platziert. Die anzuzeigenden Elemente werden gemäß den Einstellungen der Auswahlfelder im Menü Anzeige (vgl. Abbildung 7.4) im GRASS-Monitor dargestellt.

Abbildung 7.4: Das Anzeige-Menü



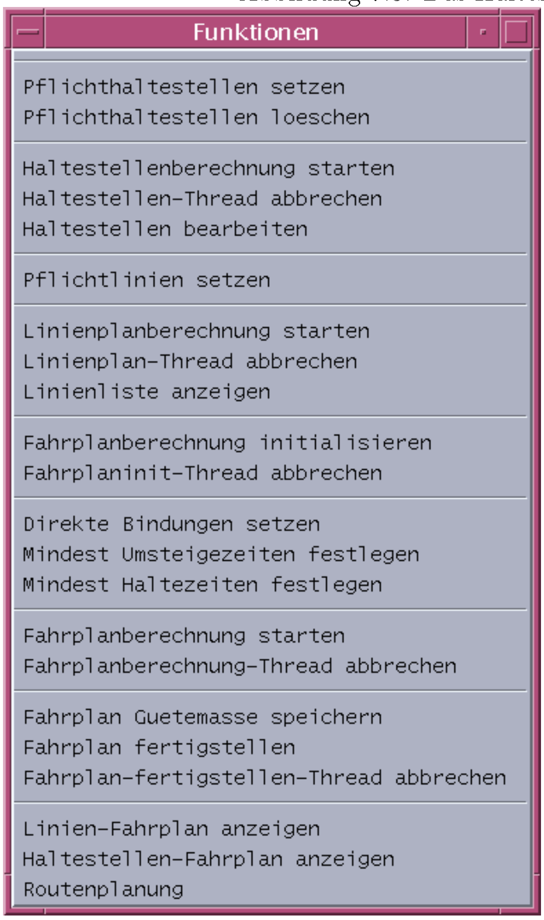
Nachdem diese vorbereitenden Maßnahmen abgeschlossen sind und die GUI ein Kontrollobjekt erzeugt hat, wird die Ereigniswarteschleife gestartet. Das Kontrollobjekt verwaltet alle restlichen Klassen des Programmmoduls, sowie die einzelnen Berechnungsmodule. Mit seiner Hilfe kann die GUI die Berechnungen in den einzelnen Modulen starten und diese auch wieder stoppen. Insbesondere kommt ihm die Rolle der Threadverwaltung zu. Die Kontrollklasse erstellt direkt nach ihrer Erzeugung die einzelnen Schnittstellen, sowie eine Instanz der Datenbank.

Da die Schnittstellenobjekte die Datenbank kennen müssen, werden bei der Instantiierung der einzelnen Schnittstellenobjekte durch die Kontrollklasse sowohl ein Zeiger auf die Datenbank als auch auf die GUI übergeben.

Der Ablauf des Programms ist durch die sequentielle Arbeit der Module und deren Abhängigkeit von den vorherigen Ergebnissen festgelegt. Das Funktionenmenü spiegelt diesen Aufbau wieder, indem es Funktionen deaktiviert (ausgraut), welche nicht genutzt werden können. Dabei ist zu beachten, dass bei einer Änderung der Voraussetzungen eines Moduls, z.B. durch Setzen einer Pflichtlinie oder durch Neustarten eines anderen Moduls, die Berechnung von dort aus wiederholt werden muss.

Der Zugriff auf die einzelnen Module erfolgt über das Funktionsmenü. Dieses Funktionsmenü lässt sich abtrennen und kann somit als Funktionsleiste auf den Desktop gelegt werden (vgl. Abbildung 7.5). Somit bekommt der Benutzer stets angezeigt, in welcher Phase er sich momentan befindet und welche Funktionen er als nächstes ausführen kann. Der Ablauf verläuft von oben nach unten. Legt

Abbildung 7.5: Das Haltestellenmenü



der Benutzer ein neues Projekt an, so stehen ihm zunächst nur die Haltestellenfunktionen zur Verfügung. Die noch nicht zur Verfügung stehenden Funktionen sind für den Benutzer nicht anwählbar und werden grau dargestellt. Der Aufruf einer Funktion bewirkt, dass die unter dem folgenden Trennstrich aufgeführten Funktionen deaktiviert werden. Während der Ausführung eines Moduls, werden mit Ausnahme der Abbruch-Funktionen, sämtliche Schritte innerhalb eines

Menüs deaktiviert.

Beispiel: Durch den erneuten Aufruf der Haltestellenberechnung wird der bestehende Linienplan gelöscht und muss neu berechnet werden. Somit wird sichergestellt, dass keine Inkonsistenzen entstehen. Zu weiteren Informationen zur Bedienung der einzelnen Funktionen siehe A.

Bevor das Haltestellen-Modul gestartet wird, können Pflichthaltestellen gesetzt und gelöscht werden. Diese Haltestellen fließen in jedem Fall unverändert in das Ergebnis ein. Sie können entweder durch die direkte Angabe der Koordinaten oder durch einen Klick mit der Maus auf die Herne-Karte gesetzt werden. Da die Anpassung an das Straßennetz im Haltestellenmodul erfolgt, müssen die Pflichthaltestellen nicht auf den Straßen liegen. Als Zusatzoption können auch die Herner Haltestellen als Pflichthaltestellen gesetzt werden. Das Löschen erfolgt analog dem Setzen durch eine direkte Angabe der Koordinaten oder eines entsprechenden Mausklicks. Zusätzlich lassen sich auch alle Haltestellen über die entsprechende Schaltfläche entfernen.

Über das Programmmenü erreicht man das Einstellungsfenster. Dort können Parameter verändert werden, die das Laufzeitverhalten der einzelnen Module steuern. Die Werte werden unter dem momentan aktiven Projekt in der Datenbank gespeichert.

Nachdem die Berechnung abgeschlossen ist, wird das Ergebnis automatisch auf der Karte angezeigt. Haltestellen werden dort durch grüne Kreuze, die Wege zwischen ihnen durch blaue Linien repräsentiert.

Nach der Berechnung der Haltestellen können Pflichtlinien für das Linienplan-Modul gesetzt werden. Im Pflichtlinienfenster lässt sich per Knopfdruck eine neue Linie hinzufügen. Nachdem eine Starthaltestelle durch einen Klick auf den gleichnamigen Schalter erzeugt oder per Mausklick aus der linken Liste selektiert wurde, wird automatisch eine Liste der von dort erreichbaren Haltestellen angelegt. Wird die Haltestelle per Mausklick direkt im GRASS-Fenster selektiert, so wird sie dort durch einen roten Rahmen markiert. Ist die Berechnung abgeschlossen, werden alle Kanten, auf denen eine Linie fährt rot selektiert. Zusätzlich lässt sich eine Linienliste anzeigen, in der zu jeder Linie aufgeführt ist, welche Haltestellen in welcher Reihenfolge befahren werden. Weiterhin existiert über das "Anzeigen"-Menü eine Einzelanzeige der Linien auf der Herne-Karte.

Die Berechnung des Fahrplans weicht in seinem Vorgehen von dem der anderen Module ab. Zunächst wird die Fahrplanberechnung initialisiert. Hier werden vorbereitende Berechnungen vorgenommen, die als Basis der Nachfolgenden dienen. Anschließend lassen sich direkte Bindungen, Mindestumsteigezeiten und Mindesthaltezeiten angeben (vergl. Anhang G). Nun kann die eigentliche Fahrplanberechnung gestartet werden. Hier besteht die Auswahl zwischen drei Lösungsansätzen: heuristisch, Entscheidungsbaumverfahren und sukzessive Lösungverbesserung. Die letzte Möglichkeit ist beim ersten Durchlauf nicht anwählbar, da mit den beiden anderen Verfahren zunächst eine Lösung ermittelt werden muss. Ist diese Berechnung abgeschlossen, besteht die Wahl zwischen der Berechnung der Gütemaße, oder der Fertigstellung des Fahrplans. Diese beiden Punkte können unabhängig voneinander gewählt werden. Soll der Fahrplan fertiggestellt werden, erscheint ein Dialog, in dem ein zu betrachtendes Zeitintervall, zusammen mit einer Abfahrtsminute, von der aus die Berechnung starten soll, angegeben werden kann (Beispiel: wird als Zeitintervall 8:00 bis 10:00 Uhr angegeben, sowie als Abfahrtsminute 7, so fängt die Berechnung bei

8:07 Uhr an). Zusätzlich muss an einer Haltestelle eine Linie angeben werden, von der die Berechnung aus gestartet wird. Nach diesem Schritt ist der Fahrplan erstellt und lässt sich als Linien-Fahrplan (welche Haltestellen wann befahren werden) oder als Haltestellen-Fahrplan (welche Linien wann an dieser Haltestelle abfahren) anzeigen.

Ein Beispiel-Ablauf anhand der Haltestellen:

1. Zunächst wird ein GUI-Objekt erzeugt. Dieses legt eine Kontrollklasse an. Anschließend wird die Ereigniswarteschlange gestartet.

2. Die Kontrollklasse erzeugt ein Datenbankobjekt und die vier benötigten Schnittstellen. Damit die Schnittstellen die Datenbank erkennen können, wird diese als Zeiger an die Schnittstellen übergeben. Damit die GUI ihre Schnittstelle ebenfalls erkennt, wird ein Zeiger auf diese Schnittstelle an das GUI-Objekt zurückgeliefert.

3. Der Benutzer startet in der GUI die Haltestellen. Die GUI teilt dies dem Kontrollobjekt mit. Dieses wiederum erzeugt einen Haltestellenthread und übergibt der Haltestelle die ihr zugeteilte Schnittstelle.

4. Die Haltestelle fragt bei der Schnittstelle ihre Eingabedaten ab.

5. Die Schnittstelle fragt bei der Datenbank nach den Haltestellenpositionen und den vom Modul benötigten Parametern.

6. Die Datenbank erkennt, dass die benötigten Daten in GRASS liegen, fordert sie von dort an und liefert sie als entsprechende LEDA-Liste an die Schnittstelle zurück. Alle weiteren, vom Haltestellenmodul benötigten Daten werden analog ermittelt und weitergeleitet.

7. Hat die Schnittstelle alle von ihr benötigten Daten, liefert sie diese an das Haltestellenmodul zurück.

8. Hat das Haltestellenmodul seine Berechnungen beendet, übergibt es der Schnittstelle die entsprechenden Daten. Letztere leitet die Daten zur Speicherung an die Datenbank und somit an Oracle weiter. Der Wegenetzgraph wird gleichzeitig der GRASS-Schnittstelle übergegeben. Dort erfolgt eine Umwandlung in die von GRASS benötigten Datenstrukturen und die gleichzeitige Aufspaltung in einzelne arcs (Liste von Punkten, vergl. Kapitel G). Nach dem Aufruf der entsprechenden API-Funktionen erzeugt GRASS einen neuen Layer, auf dem diese, zuvor umgewandelten Ergebnisdaten abgelegt werden.

9. Das Kontrollmodul meldet die Beendigung der Berechnung der GUI und der zugehörige Thread wird beendet. Die GUI stellt danach alle Ergebnisse auf dem Bildschirm dar. Hierzu veranlasst die GUI ein Neuzeichnen des GRASS-Monitors mit den, von GRASS im vorherigen Schritt erzeugten Layer. Bei Bedarf fordert die GUI weitere Ergebnisse des Berechnungsvorganges (z. B. die Gütemaße) aus der Datenbank an. Dies erfolgt über ihre eigene Schnittstelle.

Die anderen Berechnungsmodule weisen einen analogen Ablauf auf.

Sowohl die Ausgangsdaten, als auch die errechneten Ergebnisse werden grafisch im Monitor angezeigt. Diese Anzeige lässt sich über das Menü "Anzeige" in folgenden Punkten beeinflussen (vgl. Abbildung 7.4):

- Anzeige aktualisieren

  GRASS wird mitgeteilt, den Inhalt des GRASS-Monitors zu löschen. Jetzt wird überprüft, welche Auswahlfelder der Benutzer ausgewählt hat. Die entsprechenden Layer werden eingelesen und GRASS die entsprechenden Kommandos zur Anzeige geschickt.

- Vergrößern

  Es wird eine Funktion in GRASS aufgerufen, welche die momentan gewählte Region neu setzt. Mit der linken Maustaste kann die linke obere Position, mit der rechten Maustaste die untere rechte Position des erscheinenden "Gummiband-Rechtecks" festgelegt werden.

  Nach Akzeptieren der Auswahl wird die Region in GRASS neu gesetzt und "Anzeige aktualisieren" aufgerufen.

- Verkleinern (zeige Herne komplett)

  Um eine Komplettansicht von Herne zu erlangen, wird mittels eines GRASS-Kommandos die aktuell eingestellte Region auf die Randwerte des Hauptstraßen-Layers gesetzt. Da dies der Layer mit der größten Ausdehnung ist, befindet sich so das gesamte Kartenmaterial im sichtbaren Bereich. Nach Setzen der Region erfolgt "Anzeige aktualisieren"

- Zoomstufe "Zeige alle Haltestellen"

  Hier wird der Ausschnitt so gewählt, dass gerade alle berechneten Haltestellen angezeigt werden. Es wird der GRASS-Layer nach der nordwestlichsten und der südöstlichsten Haltestelle durchsucht. Anschließend wird der entsprechende Ausschnitt angezeigt.

- Nördlicher, Südlicher, Westlicher, Östlicher

  Diese Funktionen dienen zum "Schwenken" der Ansicht nach Norden, Süden, Westen und Osten (N/S/W/O). Zunächst wird per GRASS-Kommando die aktuell sichtbare Region ermittelt. Der Schwenk soll so durchgeführt werden, dass eine halbe Bildschirmseite geschwenkt wird. Die neuen Koordinaten werden demnach wie folgt berechnet:

  Bei einem Nord-Schwenk:

  $$Nord = Nord_{alt} + \frac{Nord_{alt} - Sued_{alt}}{2}$$

  $$Sued = Sued_{alt} + \frac{Nord_{alt} - Sued_{alt}}{2}$$

  Bei einem Süd-Schwenk:

  $$Nord = Nord_{alt} - \frac{Nord_{alt} - Sued_{alt}}{2}$$

$$Sued = Sued_{alt} - \frac{Nord_{alt} - Sued_{alt}}{2}$$

West- und Ost-Schwenk erfolgen analog. Die Region wird mit den berechneten Werten neu gesetzt und es wird "Anzeige aktualisieren" aufgerufen.

- Zeige Punkt

  Diese Funktion wurde als Hilfestellung für die Haltestelleneingabe entworfen (vgl. Kapitel 7.4.1).

7.5 Umsetzung der Haltestellen

Verwendete Symbole

p_{raster}	Parameter, welcher die Ausdehnung eines Sektors in Metern sowohl für die Bevölkerungs- als auch für die Flächendeckung vorgibt.

Weitere Symbole sind definiert in 5.3, Seite 28 und 6.1, Seite 61.

7.5.1 Realisierung

In diesem Abschnitt wird erläutert, in welcher Art das in 5.3, Seite 28 ff., vorgestellte mathematische Modell der Haltestellen umgesetzt wurde. Dabei werden hier nur Elemente erläutert, welche keine offensichtliche oder aber mehr als eine Möglichkeit der Realisierung haben.

Das Straßennetz ist eine Liste, welche Listen von CPunkt enthält. Hiermit werden einzelne Straßen als Folge von Punkten definiert und das gesamte Straßennetz als Liste von Straßen.

Stadteile mit den Bevölkerungszahlen werden als Liste von CBevoelkerungselement verwaltet. Objekte dieser Klasse besitzen einen Mittelpunkt, einen Radius r_s[1] und die Anzahl der in dem Gebiet wohnenden Bevölkerung b_s.

Die Pflichthaltestellen sind eine Liste von CPunkt, bei dem jede Haltestelle eine Koordinate zugewiesen bekommt.

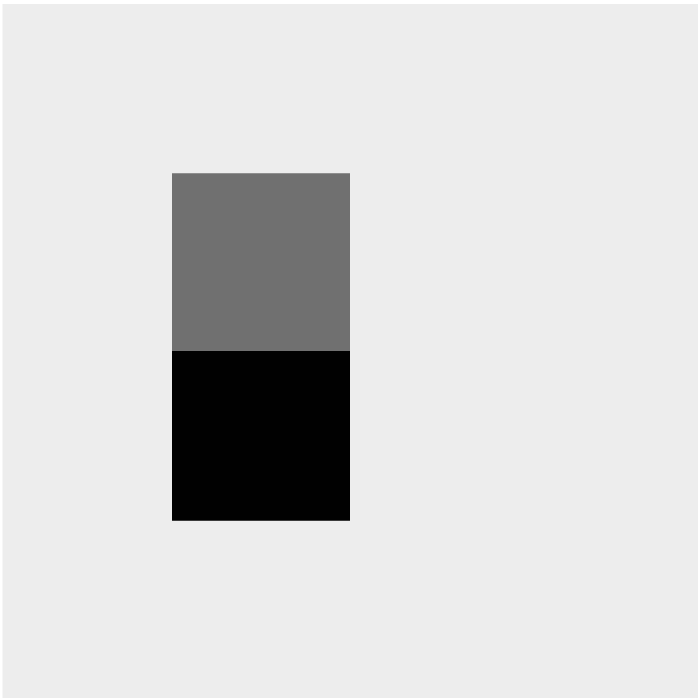
Fahrtwünsche werden in einer Matrix vermerkt. Dabei ist in dem Kreuzungspunkt aus Spalte i und Zeile j jeweils der korrespondierende Fahrtwunsch $F_{i,j}$ eingetragen. Bei $i = j$ ist der Matrixeintrag -1. Die zugehörigen Koordinaten der Fahrtwunschpunkte werden in einer Liste aus CQuelleZielListeElement verwaltet, welche den Pflichthaltestellen ähnlich ist, jedoch noch zusätzlich eine Nummer enthält, welche die Identifizierung der Spalte bzw. Zeile erlaubt.

Sowohl bei der Bevölkerungs-, als auch bei der Flächendeckung werden Summen über den zu betrachtenden Raum R gebildet. Dazu wird dieser mit einem Raster überzogen, dessen Rasterung durch den Benutzer vorgegeben wird. Die Fläche a_s, welche ein Raster einnimmt, entspricht einem Rechteck und kann durch $\frac{A}{p_{raster}^2}$ berechnet werden. Intern wird je ein Raster für Bevölkerungs- und Flächendeckung in Form einer Matrix gespeichert. Bei der Bevölkerungsabdeckung wird für jedes Matrixelement der Mittelpunkt p ermittelt und für

[1] Stadtteile werden aufgrund der fehlenden Information, wo die Stadteile exakt ihre stadtplanerischen Grenzen haben, als Kreis angenommen.

jeden Stadtteil s geprüft, ob $d_{p,s} \leq r_s$ ist. Bei Erfüllen der Gleichung wird das Matrixelement um $\frac{b_s * a_s}{\pi * r_s^2}$ erhöht. Hat ein Matrixelement mehr als einen Stadtteil, welcher den Betrag erhöht, wird durch die Anzahl der einwirkenden Stadtteile geteilt. Bei der Matrix für die Flächendeckung wird die Fläche jedes Matrixelementes a_s als Betrag eingetragen. Dieses, vom mathematischen Modell abweichende, Vorgehen wurde gewählt, um ein von der Rasterung unabhängiges Maß zu erhalten. Bei der Bewertung werden Kopien der Matrizen erstellt und für jede Haltestelle h geprüft, für welche Matrixelemente der Mittelpunkt p innerhalb der Einzugsradien liegen. Bei $d_{h,p} \leq r_{min}$ wird der Wert des Elementes aus der Original-Matrix von der kopierten Matrix abgezogen. Bei $r_{min} < d_{h,p} \leq r_{max}$ wird die Hälfte des Wertes subtrahiert. Dies gilt gleichermaßen für die Matrix der Bevökerungs-, sowie für die Matrix der Flächenversorgung. Nach Abarbeitung aller Haltestellen werden die Summen mittels der Zielfunktion Z, wie bereits im mathematischen Modell beschrieben, gebildet[2]. Beispiele mit unterschiedlicher Rasterung der Bevölkerungsdaten zeigen die Abbildungen 7.6 und 7.7. Während in Abbildung 7.6 die Stadtteile kaum zu erkennen sind, zeigt Abbildung 7.7 die kreisförmige Struktur selbiger deutlich.

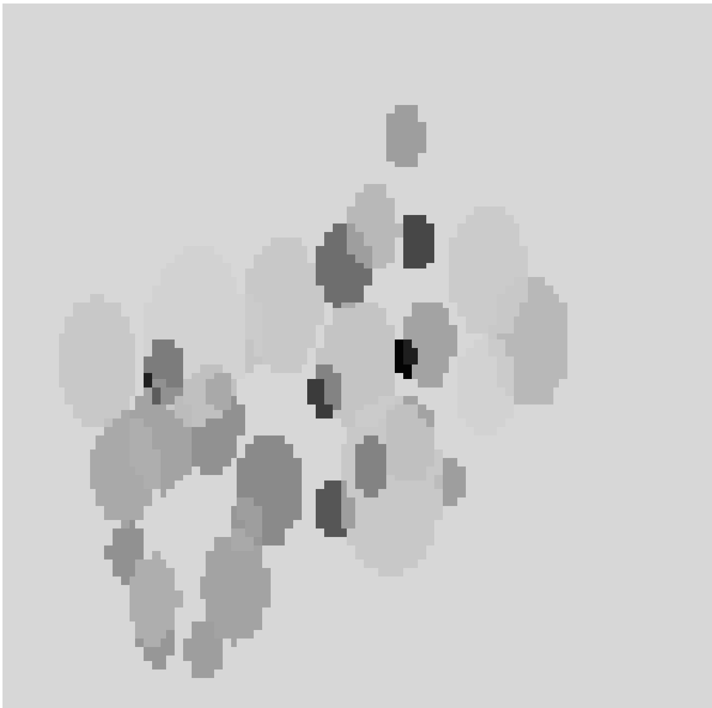
Abbildung 7.6: Rasterung der Bevölkerungsdaten mit einem Wert von 4



Die Rasterung der Daten stellt hierbei zugleich eine große Gefahr dar. Wird ein zu kleines Raster gewählt, führt dies zu einem großen Fehler bei der Bewertung. Höhere Einstellungen der Rasterung führen zu besseren Ergebnissen, aber

[2]Verwendung finden beide Varianten der Zielfunktion aus dem mathematischen Modell. Bei der Wahl der Evolutionären Strategie, stehen dem Benutzer die Koeffizienten frei, bei den anderen Algorithmen werden keine Koeffizienten einbezogen.

Abbildung 7.7: Rasterung der Bevölkerungsdaten mit einem Wert von 400 Deutlich zu sehen ist die kreisförmige Struktur der Stadtteile.



bedingen auch eine deutlich höhere Rechenzeit (vergl. Kapitel 7.5.3.2, Seite 98).

7.5.2 Ablauf

Verwendete Symbole

S_N Anzahl der Knoten im Straßennetz-Graph. Im vorliegenden Datenbestand 33882.

S_E Anzahl der Kanten im Straßennetz-Graph. Im vorliegenden Datenbestand 35610.

Die Berechnung der Haltestellen ist in drei grundlegende Teile zerlegt, welche folgend näher bestimmt werden.

7.5.2.1 Vorbereitungen

Dies sind allgemeine Arbeiten, welche die Modifikation der Eingaben aus dem Programm-Modul darstellen. Sie sind vom gewählten Algorithmus zur Legung von Haltestellen unabhängig.

1. Bestimmen des zu betrachtenden Planquadrates aus dem Straßennetz heraus.
   Hier wird über das weitergereichte Straßennetz iteriert, um so die Ausmaße zu ermitteln.

2. Aus dem Straßennetz wird ein Straßennetz-Graph erstellt.
   In ihm werden Kreuzungen, Sackgassen und „Knicke" durch Knoten, Straßen hingegen durch Kanten repräsentiert. Dazu werden paarweise alle Straßenzüge aus dem Straßennetz miteinander verglichen und die so ermittelten Kreuzungen eingetragen. Zudem werden Verbindungen zwischen diesen an die Kanten geschrieben. Sackgassen werden gleichzeitig vermerkt. Sind Knoten schon vorhanden, so wird geprüft, ob sie von einer Sackgasse zu einer Kreuzung werden. Wenn sie noch nicht vorhanden sind, so wird geprüft, ob schon vorhandene Verbindungen durch diesen Knoten verlaufen. Diese werden dann entsprechend aufgetrennt und an den Knoten angeschlossen. Intern wird dies mittels einer Methode aus der LEDA-Bibliothek durchgeführt.

3. Um sicherzustellen, dass jeder Straßenzug von überall aus erreichbar ist, werden nun, beginnend mit der kleinsten Komponente, nacheinander Komponenten entfernt, bis nur noch eine Komponente vorhanden ist.

4. Gruppierung der Straßensegmente in Sektoren
   Hierzu wird zunächst ein Raster erstellt, wobei jedes Element eine Kantenlänge von 75 Metern hat[3]. Für alle Straßensegmente S_E aus dem Straßennetz-Graph wird geprüft, welche Sektoren berührt werden. In selbige wird dann eine Liste mit den Verweisen auf diese Segmente gesetzt.

[3]Dies ist ein Erfahrungswert, welcher eine sinnvolle Abschätzung zwischen Geschwindigkeitsgewinn und Genauigkeit bei der Anpassung einer Haltestelle an das Straßennetz darstellt.

5. Einteilung der Bevölkerung in Sektoren
   Dabei wird ein Raster entsprechend den Einstellungen des Benutzers über das Gebiet gelegt. Die Anzahl der Elemente beträgt hierbei das Quadrat des Wertes. Die Raster sind hierbei alle gleich groß und gleichmäßig über R verteilt. Hierzu wird für jedes Element geprüft, ob es in einem Stadtgebiet liegt (mittels Luftlinienentfernungsbestimmung jeweils von den Zentren aus). Ist dem so, wird die Bevölkerung des Elementes auf den Wert Gesetze, welcher sich aus dem Verhältnis der Fläche des Elementes und der Fläche des Stadtgebietes ergibt. Liegt ein Element innerhalb mehrerer Stadtgebiete, wird der Mittelwert gebildet. Siehe hierzu auch 7.5.1, Seite 87.

6. Einteilung der Fläche in Sektoren.
   Dabei wird ein Raster entsprechend den Einstellungen des Benutzers über das Gebiet gelegt. Die Anzahl der Elemente beträgt hierbei dem Quadrat des Wertes. Die Raster sind hierbei alle gleich groß und gleichmäßig über R verteilt. Hierzu wird jedes Element mit dem Wert der Fläche des entsprechenden Elementes belegt. Da alle Elemente gleichgroß sind, ist dieser Wert uniform. Siehe hierzu auch 7.5.1, Seite 87.

7.5.2.2 Haltestellenberechnung

Im folgenden werden die implementierten Verfahren, welche gegeneinander ausgetauscht werden können, näher vorgestellt. Dabei kann jeder der hier genannten Algorithmen grundsätzlich sowohl mit Luftlinienentfernungen (modifiziert durch k_{HCR}) arbeiten, als auch mit den Realwegen.

Manuelle Platzierung

Der Benutzer platziert die Haltestellen manuell. Dies kann durch ein grafisches Frontend (siehe Kapitel A, Seite 137 ff.) oder aber durch Konfigurationsdateien erfolgen. Der Benutzer legt dadurch auch die Anzahl der Haltestellen implizit fest.

Diese Funktionalität wird durch das Programm-Modul zur Verfügung gestellt. Auf diese Weise werden ebenfalls die Pflichthaltestellen festgelegt.

Stochastische Gleichverteilung

Die Anzahl der Haltestellen muss vom Benutzer vorgegeben werden, andernfalls wird eine Obergrenze von Haltestellen pro Flächeneinheit algorithmisch ermittelt (siehe Kapitel 5.3.2.5, Seite 32). Die Haltestellenkoordinaten werden durch eine gleichverteilte Zufallsfunktion innerhalb des zu betrachtenden Raumes R frei erzeugt, ohne darauf zu achten, ob sich die Haltestellen dann auch an einer Straße befinden.

Stochastische Gleichverteilung bei Berücksichtigung von bereits vorhandenen Haltestellen

Wie 7.5.2.2, nur wird für jede potentielle Haltestelle geprüft, ob im minimalen Einzugsradius eine Haltestelle vorhanden ist. Wenn dies der Fall ist, wird ein Fehlerzähler hochgesetzt. Ist noch keine Haltestelle in der Umgebung vorhanden, so wird eine erzeugt.

Dieser Algorithmus bricht ab, wenn alle angeforderten Haltestellen gesetzt wurden (siehe Kapitel 5.3.2.5, Seite 32), oder eine bestimmte Anzahl an Fehlversuchen hintereinander ohne Setzen einer Haltestelle erfolgt ist. Im Programm ist dieser Wert fest auf 100 eingestellt.

Evolutionäre Strategien

Zwingende Vorgaben

- Anzahl der Eltern
  Beschreibt die Anzahl der Individuen, welche in die nächste Generation übernommen werden, wobei nur die besten herangezogen werden.
- Anzahl der Nachfahren
  Beschreibt die Anzahl der Kinder, welche aus den Eltern erzeugt werden.
- Anzahl der Generationen über die evolviert wird. Spätestens nach dieser Anzahl von Schritten endet die Strategie.
- Plus- oder Kommastrategie
  Beschreibt, ob bei der Auswahl der neuen Eltern, die bisherigen Eltern mit berücksichtigt werden (Plus) oder nicht (Komma).

Optionale Vorgaben

- Pflichthaltestellen
- Fahrtwunschmatrix F_{ij}
- Fitnesswert, bei dessen Unterschreitung abgebrochen wird
- Rekombination
  Die Rekombination bestimmt aus wievielen Eltern je ein Kind erzeugt wird und auf welche Weise. Üblich sind hier Bildung des Mittelwertes oder auch diskret, das heißt für jedes Gen wird zufällig entschieden, von welchem Elternteil es stammt.
- Constraints[4]
  Constraints beschreiben, wie der Wertebereich der Lösung eingegrenzt wird, sofern bekannt. Eine Möglichkeit ist, die Parameter immer wieder in den Wertebereich zurückzusetzen, falls sie diese verlassen. Dabei handelt es sich dann um einen Hard Constraint. Es kann jedoch auch ein Soft Constraint verwendet werden, bei dem für jeden Parameter, welcher den Wertebereich verlassen hat, ein Strafterm hinzugefügt wird. Die Höhe der Strafe hängt von der Größe der Entfernung vom Wertebereich ab. Abbildung 7.8 zeigt den Suchraum[5] ohne Einsatz von Constraints. Auffallend ist hier, dass ausserhalb des Gebietes, in welchem das Optimum liegt, ein konstanter nicht-optimaler Wert vorliegt. Dies wird als Plateau bezeichnet. In Abbildung 7.9 wurde ein linearer Soft Constraint eingesetzt. Dabei entspricht die Strafe linear proportional der Entfernung vom Wertebereich. Das Individuum wird dadurch gezwungen, immer wieder

[4] Bei der verwendeten Bibliothek TEA [TEA24] werden in der vorliegenden Version keine Soft Constraints unterstützt.

[5] Das Problem wurde zur einfacheren Veranschaulichung auf eine Dimension reduziert.

in Richtung des Wertebereiches zurückzufallen. Da sich das Individuum dadurch unter Umständen zu langsam oder bei ungünstiger Struktur der Zielfunktion(-relation) nicht oder nur sehr langsam in den gewünschten Wertebereich zurückbewegt, zeigt Abbildung 7.10 die Verwendung eines quadratischen Soft Constraints. Dies führt zu einer schnelleren Abwanderung in den gewünschten Wertebereich.

Abbildung 7.8: Keine Constraints

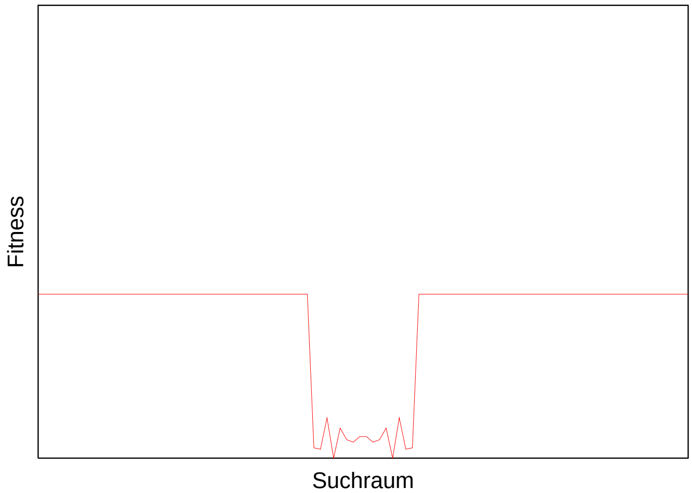
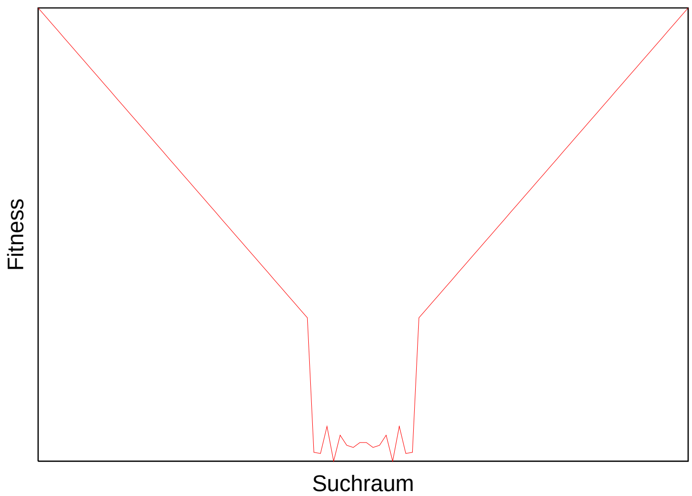
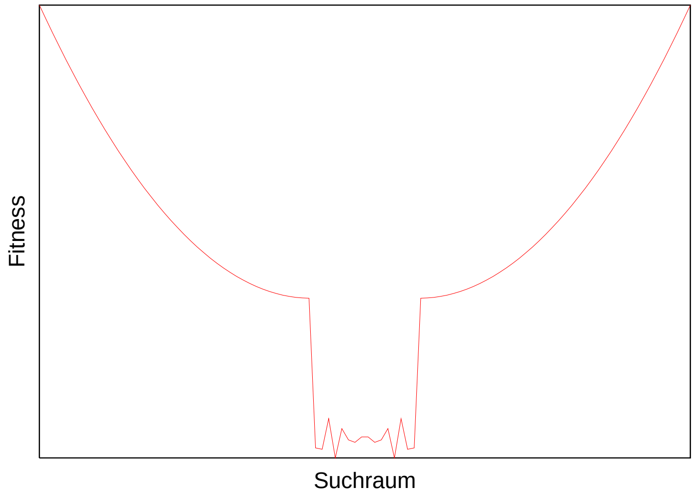


Abbildung 7.9: Lineare Soft Constraints



Ein einzelnes Individuum besteht aus $4 * H_{max}$ Genen. Die ersten $2 * H_{max}$ Gene kodieren mit je zwei nacheinanderfolgenden Genen eine Haltestelle, wobei die reelwertigen Zahlen direkt die Koordinaten bestimmen. Dabei ist die resultierende Haltestelle nicht an das Straßennetz angepasst, sondern liegt frei

Abbildung 7.10: Quadratische Soft Constraints



im Raum. Eine Anpassung erfolgt gesondert in einem späteren Schritt. Die letzten $2 * H_{max}$ Gene entsprechen den Schrittweiten σ_i, $i\epsilon[1; 2 * H_{max}]$ und korrespondieren mit den ersten $2 * H_{max}$ Genen.

Die Fitness wird durch die in 5.3, Seite 28 ff., definierte Zielfunktion Z bestimmt. Hierbei müssen die Schritte 7 bis 9 (siehe weiter unten) für jedes Individuum jeder Generation erneut durchlaufen werden.

Diese Kodierung enthält implizit eine Selbstadaption der Haltestellenanzahl, welche auf dem Straßennetz berechnet werden. Zum einen durch Verlassen des Suchraumes R einzelner Haltestellen. Zum anderen dadurch, dass eine Haltestelle in einen Sektor (siehe 4. im Kapitel 7.5.2.1) wandert, in dem kein Straßensegment liegt. Beide Varianten bedingen Plateaus[6] im Lösungsraum, wodurch selbiger nicht stetig ist. Die Suche wird hierdurch deutlich erschwert. Stehen genügend Ressourcen (Rechenkapazität) bereit, lässt sich die Größe der Sektoren weiter heraufsetzen — im Extremfall, was auch dem Idealfall entspricht, wird nur ein Sektor verwendet, was keiner Geschwindigkeitsoptimierung entspricht — und die Anzahl der Plateaus deutlich mindern.

Auch die nachträgliche Anpassung der Haltestellen an das Straßennetz bedingt Plateaus, was zudem durch die Wahl der Haltestellenlage in Kreuzungsnähe verstärkt wird. Dies liegt daran, dass eine Haltestelle einen bestimmten Spielraum[7] hat, in welcher sie aber immer auf den gleichen Punkt im Straßennetz abgebildet wird.

[6] Plateaus sind Bereiche im Lösungsraum, welche alle die gleiche (oder zumindest eine ähnliche) Fitness besitzen. Optisch kann man sich dieses als Treppen vorstellen. Je ausgedehnter diese sind, je mehr Plateaus es gibt und je unterschiedlicher der Wert zwischen den Plateaus und Nicht-Plateaus ist, desto schwieriger ist es, eine optimale Lösung zu finden. Vor allem, wenn die optimale Lösung innerhalb eines relativ hoch gelegenen und ausgedehnten Plateaus liegt.

[7] Bei Wahl von kreuzungsnaher Lage der Haltestellen ist dies eine Fläche um die Kreuzung herum. Bei einer lotnahen, oder besser nicht-kreuzungsnahen, Lage kann die Haltestelle jeden beliebigen Punkt zwischen ihrer Position im Raum und dem angepassten Punkt auf dem Straßennetz annehmen. Dies ist dann eine Linie.

Die Bestimmung der Strategie (Plus oder Komma), die Anzahl der Generationen, die Anzahl der Eltern, die Anzahl der Kinder und der Wert der Güteunterschreitung können durch den Benutzer mittels Parameter vorgegeben werden.

Constraints finden keine Verwendung.

Es sind einfache Rekombinationsregeln fest implementiert worden, welche nicht parametrisiert sind. Für genauere Informationen siehe 8.5, Seite 127.

Abschluss der Haltestellenberechnung

Zuletzt werden die bis jetzt noch im Raum R frei liegenden Haltestellen noch den Parametern entsprechend an das Straßennetz gelegt und bewertet. Bis auf die ES, welche dienun folgenden Schritte 7 bis 9 mehrfach durchläuft, werden alle Schritte nur einmal abgearbeitet.

7 Anpassen der Haltestellen an das Straßennetz

Hierbei kann nach den folgenden Modi unterschieden werden:

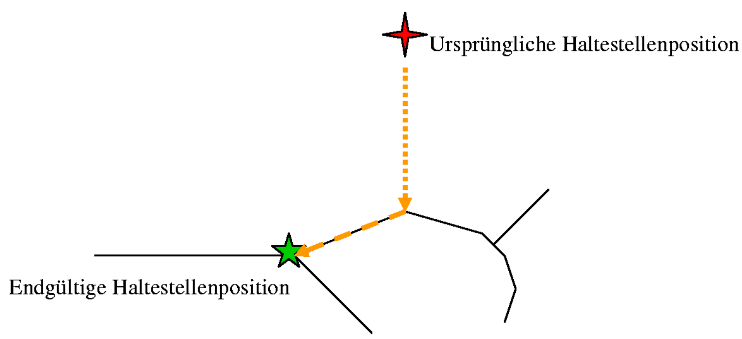
(a) Lot fällen

Es wird für jede Haltestelle jeder Straßenzug herangezogen. Aus Geschwindigkeitsgründen wird nur jedes Segment aus dem nächstgelegenen Sektor geprüft, zusätzlich der Sektoren, welche um diesen herumliegen (dies sind im gesamten dann neun Sektoren). Hat die Haltestelle einen Lotfußpunkt auf dem Teilstück eines Straßenzuges, so wird die Entfernung mit bisher ermittelten verglichen. Die jeweils Kürzeste bestimmt, welcher Lotfußpunkt zu der neuen Haltestellenposition wird. Schlägt die Suche für eine Haltestelle fehl, so wird die nächstliegende Kreuzung nach dem folgenden Verfahren herangezogen.

Ein einfaches Beispiel zeigt Abbildung 7.11, welches ein Straßennetz und eine Haltestelle darstellt. Es ist nicht möglich, von der Haltestelle aus, auf irgendeinen Straßenzug einen Lotfußpunkt zu legen.

Abbildung 7.11: Fällen des Lotes schlägt fehl

Daher wird der nächstgelegene Endpunkt der Segmente übernommen. Zudem wird die Haltestelle in Richtung der nächsten Kreuzung verschoben. Die Haltestellenlage wurde kreuzungsnah gewählt.



(b) nächste Kreuzung bestimmen

Für jede Haltestelle werden die Entfernungen zu jeder Kreuzung ermittelt. Die nächstliegende wird die neue Haltestellenposition.

(c) freie Position zwischen den beiden zuvor genannten Verfahren

Hierzu wird mittels des Parameters Haltestellenlage spezifiziert wo, anteilmäßig, die Haltestelle zwischen dem Lotfußpunkt und der nächsten Kreuzung positioniert wird. Diese Möglichkeit wurde realisiert, da sie beide oben genannten implizit enthält. Hierzu wird zunächst wie oben beschrieben das Lot gefällt und dann von diesem Punkt aus auf der Straße in beide Richtungen gelaufen, bis eine Kreuzung oder eine Sackgasse erreicht ist. Der Pfad zu den Sackgassen wird in jedem Fall verworfen. Der kürzere Weg von dem Lotfußpunkt zu den (verbleibenden) Kreuzungen der Pfade, wird als nächste Kreuzung herangezogen. Mittels des vom Benutzer einzustellenden Parameters Koeffizient Haltestellenlage wird nun anteilmäßig ermittelt, wo die Haltestelle letztendlich positioniert wird

Ein Beispiel hierzu zeigt ebenfalls Abbildung 7.11.

8 Nachbearbeiten der Haltestellen

Insbesondere durch 7. können Haltestellen zu dicht gelegt worden sein. Daher werden nacheinander die Haltestellen durchlaufen und geprüft, ob diese zu nah, also innerhalb des minimalen Einzugsgebietes r_{min}, an den vorher geprüften liegen. Wenn dies der Fall ist, werden sie gelöscht. Dabei werden Pflichhaltestellen nicht gelöscht. Dieses ist ein guter Ansatzpunkt für die Erzeugung alternativer Haltestellenpositionen, indem die zu nah liegenden Haltestellen nicht gelöscht, sondern mit einer niedrigeren Priorität versehen werden. Diese Erweiterung ist verfügbar, wird vom Linienplan-Modul jedoch nicht ausgewertet.

9 Abschließende Bewertung der Haltestellenpositionen.

Die Bewertung erfolgt nach der in 5.3, Seite 28 ff., definierten Zielfunktion Z.

7.5.2.3 Nachbearbeitungen

Bei den Nachbearbeitungen handelt es sich um Modifikationen und Vorbereitungen, welche für das nachfolgende Linienplan-Modul und auch zur Ausgabe durch das Programm-Modul bestimmt sind. Im Einzelnen sind dies:

10 Erstellen einer Verkehrsbedarfsmatrix.

Bei der Verkehrsbedarfsmatrix wird zunächst eine Art „Grundrauschen" angenommen, bei der die Bevölkerung als ausgehender gleichverteilter Fahrtwunsch von dieser Haltestelle zu allen anderen Haltestellen angenommen wird.

Hierzu wird über alle Haltestellen iteriert und die Bevölkerung aus den Sektoren, deren Entfernung zu der entsprechenden Haltestelle kleiner ist als das minimale Einzugsgebiet r_{min}, vollständig übernommen. Sektoren, deren Entfernung immerhin noch kleiner als das maximale Einzugsgebiet r_{max} ist, erhöhen um die halbe Bevölkerung. Der resultierende Wert wird zunächst durch $H_N - 1$ geteilt und dann gleichverteilt auf alle anderen Haltestellen gelegt. Nun werden die real existierenden Fahrtwünsche mittels Interpolation auf die neuen Haltestellen umgerechnet. Siehe dazu 5.3, Seite 28 ff.

11 Kontinuierliches Durchnummerieren der Haltestellen von Null beginnend.

12 Aufbauen eines Wegenetzgraph.

Knoten repräsentieren Haltestellen und Kanten repräsentieren Verbindungen zwischen den Haltestellen. Dabei werden nicht alle Verbindungen übernommen, sondern nur solche, welche nicht länger als der übergebene Verbindungsradius $r_{verbindung}$ sind.

Bei der Wahl der Verbindungart Luftlinie werden hierzu alle Haltestellen $a, b \epsilon H$ paarweise geprüft und bei erfüllen der Bedingung $d_{a,b} < r_{verbindung}$, verbunden. Die Entfernung wird zusätzlich mittels k_{HCR} abgeschätzt. Dabei wird beachtet, dass Hin- und Rückrichtung identisch sind.

Bei der Wahl von Realweg wird selbiger mittels des kürzeste-Wege-Verfahrens von Dijkstra (siehe [Gue92]) zwischen beiden Haltestellen ermittelt, welches bereits durch die LEDA-Bibliothek zur Verfügung gestellt wird. Zudem wurde berücksichtigt, dass zwei Haltestellen nicht direkt verbunden sind, wenn der Pfad zwischen ihnen über mindestens eine weitere Haltestelle führt. Der Zeitaufwand ist hierbei deutlich erhöht (siehe Kapitel 7.5.3.1, Seite 97).

Ein Problem hierbei ist, dass Graphen erzeugt werden können, welche nicht zusammenhängend sind. Im ungünstigsten Fall kann der Graph sogar vereinzelte Knoten aufweisen. Dieses wurde durch Verbinden der vereinzelten Teile mit umliegenden anderen Teilen gelöst. Diesem Vorgehen liegt die Idee eines minimalen Spannbaumes zu Grunde unter Verwendung einer modifizierten Variante von Kruskal[8] (siehe [Gue92]). Es werden nur noch die verbleibenden Pfade mit einer Länge größer als $r_{verbindung}$ für die einzelnen Komponenten geprüft, bis nur noch eine Komponente resultiert.

7.5.3 Zeitkomplexität

7.5.3.1 Erstellung des Wegenetzgraph

Bei der Verwendung von Luftlinien ist der Aufwand zur Erstellung des Wegenetzgraph unerheblich. Hier muss für jedes Paar $a, b \in H$ die Luftlinienentfernung $d_{a,b}$ ermittelt werden, wobei $d_{a,b} = d_{b,a}$ ist. Daraus resultiert ein Aufwand von $O\left(H_N^2\right)$.

Bei der Verwendung von Realwegen, muss jedes mal der kürzeste Weg von a nach b auf dem Straßennetz-Graph gesucht werden. Bei Verwendung von Dijkstra und der Variante, welche von einem Knoten a aus den kürzesten Weg zu allen anderen Knoten ermittelt, folgt eine Zeitkomplexität von $O\left(H_N * (S_N + S_E \log S_E)\right)$. Dabei tritt das zusätzliche Problem auf, dass die resultierende Datenstruktur dermaßen groß ist, dass der physikalische Speicher

[8] **Annahme:** Alle Pfade (jede Kante des „Spannbaumes") sind bekannt. Es ist auch bekannt, bis zu welcher Länge ($= r_{verbindung}$) der „Spannbaum" bereits aufgebaut wurde. Es handelt sich selbstredend innerhalb der Komponenten um keinen Spannbaum, aber die Komponenten bilden nach dem Verbinden untereinander einen solchen.

nicht ausreichte. Die resultierende Laufzeit, welche durch das Auslagern von physikalischem Speicher in den virtuellen Speicher erfolgte, ist deutlich höher.

Die Variante von Dijkstra, welche nur den kürzesten Weg von a nach b ermittelt, hat über alle Aufrufe eine Zeitkomplexität von $O\left(H_N^2 * (S_N + S_E \log S_E)\right)$. Dies ist auf den ersten Blick schlechter, jedoch nach [LEDA4.2] ist diese Variante im Schnitt deutlich schneller, wenn nur wenige, kurze Wege benötigt werden — und $H_N \ll S_N$. Des weiteren belegt die resultierende Datenstruktur deutlich weniger an physikalischem Speicher, so dass ein Auslagern im Allgemeinen nicht auftritt. Dies ist auch logisch, da bei der ersten Variante für jeden Durchlauf S_N Pfade (wenn auch sehr effizient) gespeichert werden. Bei der zweiten Variante ist für jeden Durchlauf nur ein Pfad zu speichern.

Wichtigster Fakt hierbei ist, dass bei der ersten Variante auch Wege ermittelt werden, welche zwar bei einer Haltestelle a beginnen, jedoch beliebig im Straßenetz-Graph enden. Diese Wege sind für die Betrachtung völlig unerheblich und müssen erst umständlich herausgefiltert werden. Und zum anderen, gegeben durch die immense Größe des Straßennetz-Graph, wird zur Laufzeit physikalischer Speicher durch virtuellen ergänzt. Daher wurde die zweite Variante aufgrund der besseren Laufzeit implementiert, die andere wurde nach mehreren Vergleichsläufen verworfen.

7.5.3.2 Haltestellenberechnung

Verwendete Symbole

N_{raster} Anzahl der eingestellten Rasterung für Bevölkerungs- und Flächendeckung.

N_{sektor} Ausdehnung eines Sektors in Metern. Entspricht fest eingestellten 75m.

Bei der Haltestellenberechnung sind zwei fundamentale Vorgänge, welche die Zeitkomplexität bestimmen.

Zunächst ist dies die Anpassung der Haltestellen an das Straßennetz mit einer Zeitkomplexität von $O\left(H_N * (S_E + S_E)\right)$. Dies ist nur der worst case. Das erste S_E entspricht dabei der Anzahl der Segmente, mit denen jede Haltestelle auf die nächste mögliche Position im Straßennetz geprüft wird. Bei Verwendung der Sektorgruppierung ergibt sich hier $S_E/(9 * \frac{A}{5625})$[9], was deutlich schneller ist. Das zweite S_E entspricht der Ermittlung des Pfades auf dem Straßennetz-Graph von dem Lotfußpunkt aus zur nächsten Kreuzung hin. Dies sind selten mehr als 20 Segmente, so dass sich eine realistischere Abschätzung mit $\Theta\left(H_N * (\frac{5625 * S_E}{9 * A} + 20)\right)$ ergibt.

Der zweite Vorgang ist die Haltestellenbewertung. Dabei ergibt sich eine Laufzeit von $O\left(H_N * N_{raster}^2\right)$. Hier ist deutlich zu sehen, dass die maßgebliche Beeinflussung durch die Größe der eingestellten Rasterung N_{raster} zustande kommt. Es ist jedoch zu bemerken, dass eine niedrige Einstellung zwar zu schnelleren Ergebnissen, jedoch auch zu Fehlern bei der Bewertung führt. Eine zu hohe Einstellung führt zu Genauigkeiten bei der Berechnung, welche in dem

[9]Die Zahl 5625 resultiert aus dem Wert von 75m Ausdehnung pro Sektor. Der Wert 9 stammt aus der Wahl von entsprechend vielen Sektoren, welche für einen Vergleich dere Segmente herangezogen wird. Hier wird offensichtlich, wie stark sich eine Veränderung der Ausdehnung der einzelnen Sektoren auf die Laufzeit auswirkt.

vorliegenden Datenbestand nicht vorhanden sind und somit auch zu Fehlern führen (siehe Kapitel 7.5.1, Seite 87). Bei der Bewertung ist zudem zu beachten, dass sie bei der Evolutionären Strategie mehrfach aufgerufen wird, bei den anderen Verfahren jedoch nur ein einziges Mal, wodurch die Zeitkomplexität, realistisch betrachtet, lediglich bei dieser Strategie zum Tragen kommt.

7.6 Umsetzung des Linienplans

7.6.1 Realisierung

Der Linienplan wurde in Form einer Schnittstellenklasse zum Programm, den in Sonntag ([Sonntag77]) beschriebenen drei Phasen Aufbereitung, Verknüpfung und Bewertung, einem zentralem Datenhalter, einem Graphen Netzplan und einer Liste von Linien umgesetzt.

7.6.1.1 Schnittstelle zum Programm

Die Schnittstellenklasse (CLinienplan) ist die einzige Verbindung zum Programm. Über sie wird die Linenplanberechnung gestartet und gestoppt. Alle Parameter werden von ihr abgeholt und die Ergebnisse und Gütemaße zurückgeliefert. Die eingelesenen Daten werden innerhalb des zentralen Datenhalters gespeichert. Darüber hinaus ist sie für die Steuerung der drei Phasen Aufbereitung, Verknüpfung und Bewertung zuständig. Wenn die Einstellung "Graph darstellen" ausgewählt wurde, werden der Wegenetzgraph, Netzplangraph und Fahrplangraph mittels LEDA-Methoden ausgegeben.

7.6.1.2 zentrale Datenhaltung

Der zentrale Datenhalter (CNetzplan) ist allen Klassen bekannt und hält die von allen Klassen benötigten Parameter und Datenstrukturen bereit. Im Einzelnen sind dies: maximale Linienlänge, Gesamtlinienlänge, Zeiger auf Linienliste, Zeiger auf Netzplangraph, Geschwindigkeit, sowie ein Zeiger auf CFortschritt.

7.6.1.3 Netzplangraph

Der Netzplangraph ist als ungerichteter, zusammenhängender Graph (LEDA) implementiert, der mit der Knotenmenge die Haltestellen und mit der Kantenmenge diejenigen Wegstrecken repräsentiert, die die jeweiligen Haltestellen verbinden. Die Kennwerte für die Kanten sehen wie folgt aus: Kanten-ID, Verkehrsstrom auf dieser Kante, Länge der Kante (Entfernung), sowie die Anzahl der Basislinien die diese Kante gewählt haben. Die Knotenkennwerte setzen sich zusammen aus der Konten-ID und einer Priorität.

7.6.1.4 Linie

Eine Linie wird dargestellt durch: Linien-ID, eine Liste von Haltestellen-IDs, einer Markierung ob es sich bei dieser Haltestelle um eine Pflichthaltestelle handelt. Da die Linien symmetrisch sind, gibt es keine Unterscheidung zwischen Start- und Endhaltestellen. Zudem hatte sich im Laufe der Implementierung herausgestellt, dass das Arbeiten mit Haltestellen-IDs sehr Zeitaufwendig ist, da bei der Suche nach dem dazugehörigen Knoten der gesamte Graph durchlaufen werden muss.

Die Linienliste wiederum wird durch dem parametrisiertem Datentyp Liste aus LEDA realisiert.

7.6.1.5 Basislinie

Eine Basislinie ist als eine Linie mit zwei Kanten und drei Knoten zu verstehen, also ein Weg mit einem Start-, einem End- sowie einem Zentralknoten, die durch zwei Kanten verbunden sind. Diese Basiseinheit stellt nach Sonntag den besten Kompromiss dar, um zum einen die Kombinationsmöglichkeiten einzuschränken und zum anderen die wesentlichen Ströme im Netzwerk nachzubilden. Die Kennwerte für eine Basislinie sieht wie folgt aus: Zwei Zeiger auf Kanten des Netzplangraphen, drei Zeiger auf Knoten des Netzplangraphen, sowie der Verkehrsstrom über diese Basislinie.

7.6.1.6 Aufbereitungsphase - Erzeuge Netzplan

Bei der Erzeugung des Netzplangraphen wird der Wegenetzgraph in die linienplaninterne Datenstruktur Netzplangraph überführt, so dass die zusätzlichen Informationen an den Kanten und Knoten, die während der Berechnung benötigt werden, dort abgespeichert werden können.

7.6.1.7 Aufbereitungsphase - Umlegen der Nachfrage

Ziel dieser Umlage ist es, die Verkehrsströme eines Liniennetzwerkes durch die Umlegung der Verkehrsbedarfswerte (Nachfrage) auf die kürzesten Wege einer Verkehrsbeziehung zu bestimmen. Auf diese Weise wird der ergebende Nachfrage-Strom im Netzwerk ermittelt, an den sich die zu berechnenden Linien in der späteren Verknüpfungsphase möglichst optimal anzupassen versuchen.

Dabei wird von der idealisierten Annahme ausgegangen, dass alle Nachfragewünsche immer auf dem direkten und kürzesten Weg bedient werden könnten. Daher wird der komplette Wert in einem Feld der Nachfrage-Matrix, der dem Verkehrsbedarfswert von einer Quelle i zu einem Ziel j entspricht, jeweils allen Kanten, die auf dem kürzesten Weg zwischen den beiden Haltestellen liegen, zugewiesen. Zu diesem Zweck werden für alle Haltestellen paarweise mit Hilfe des, durch die LEDA-Bibliothek bereitgestellten Dijkstra-Algorithmus die kürzesten Wege berechnet. Jeder, auf einem optimalen Weg liegenden Kante im Netzplangraphen wird dann der komplette Nachfragewert der jeweiligen Haltestellenkombination zugewiesen. Außerdem wird dieser Nachfragewert als versackender Strom in der zugehörigen Kante des jeweiligen Start- und Endknoten des kürzesten Weges abgespeichert. Auf diese Weise addieren bzw. überlagern sich die verschiedenen Nachfragewünsche, wenn sie auf die "wegkürzesten" Kanten umgelegt werden. So wird das Nachfrage-Potential zwischen den verschiedenen Quellen und Zielen auf die Kanten des Netzwerkes verteilt.

Es wäre an dieser Stelle auch möglich, die Fahrtwünsche nicht komplett auf den kürzesten Weg zu legen, sondern die jeweiligen Wünsche im Verhältnis ihrer Abweichung zur optimalen Strecke auf die jeweiligen Alternativ-Wege zu verteilen. Dieser Ansatz wurde jedoch nicht realisiert.

7.6.1.8 Aufbereitungsphase - Reduktion

Es wäre eine Reduktion der Knotenmenge und eine Umverteilung der Nachfrage auf die verbleibenden Knoten notwendig. Dies sollte in einem zweiten Prototypen realisiert werden, wurde dann aber verworfen. Hiergegen spricht insbesondere, dass nicht bekannt ist, welche Haltestelle als Endhaltestelle anzusehen ist. Dies bedeutet, dass der Linienplan keinerlei Informationen über die Art einer Haltestelle hat und somit nicht entscheiden kann, ob der Knoten reduzierbar ist und ob hier ein Bus starten bzw. enden kann. Dennoch wurde am Konzept einer eigenen Datenstruktur für die Berechnung des Linienplans festgehalten, da zum einen der Wegenetzgraph als gemeinsame Datenstruktur zwischen Haltestellen- und Linienplan-Modul dient und zum anderen der Netzplan einige zusätzliche Daten für die Berechnung speichert, die im Wegenetzgraph nicht enthalten sind.

7.6.1.9 Verknüpfungsphase

In dieser zweiten Phase erfolgt die Erzeugung des Linienplans. Aus Gründen einer möglichen Erweiterung um weitere Verknüpfungs-Algorithmen oder zum Austausch bereits vorhandener Algorithmen wurde diese in Start- und Abschlußverknüpfung geteilt.

7.6.1.10 Startverknüpfung - Pflichtlinien einfügen

Die von Benutzer eingegebenen Linien befinden sich bereits in dem entsprechendem Datentyp Linie und werden hier der Linienliste angefügt.

7.6.1.11 Startverknüpfung - Basislinenwahl

Im ersten Schritt wird das gesamte Netzwerk mit Basislinien belegt, deren jeweiliges Verkehrsaufkommen auf beiden Kanten addiert und in eine Rangfolgenliste einsortiert wird. Auf diese Weise können die Basislinien sortiert nach der Stärke der auf ihnen enthaltenen Nachfrage abgearbeitet werden.

Für den zweiten Schritt, das Auswählen von Basislinen, wurden zwei verschiedene Algorithmen implementiert. Der Erste hält sich an die Vorgabe von Sonntag. Hierbei müssen alle Kanten überdeckt werden. Der Zweite hat zur Bedingung, dass alle Knoten mittels Basislinien erreichbar sind, was zu einem geringer vermaschten Basisliniennetz führt und damit zu weniger Linien.

Erster Algorithmus: Um sicherzustellen, dass alle Kanten angeschlossen werden und somit jeder Knoten im späteren Linienplan enthalten ist (siehe Nebenbedingung im Kapitel 5.4.3), werden in einem ersten Durchlauf Basislinien gewählt, die noch nicht angeschlossene Kanten überdecken. Dabei werden die Basislinien entsprechend der vorher berechneten Rangfolgenliste abgearbeitet. So lange es noch eine Kante gibt, die angeschlossen werden kann und die Basislinien-Liste noch Elemente enthält, wird dieser Vorgang wiederholt. Es werden jedoch nur die Kanten gesucht, deren Überdeckungsgrad durch gewählte Basislinien dem des Durchlaufs entsprechen. Dabei hängt der Faktor, ob eine Kante angeschlossen werden darf oder nicht, davon ab, ob durch die Hinzunahme der betreffenden Basislinie die Länge aller ausgewählten Basislinien die Gesamtlinienlänge nicht überschreitet (Nebenbedingung).

Zweiter Algorithmus: Zur Auswahl der Basislinien wird hier der Algorithmus von Kruskal für minimale Spannbäume benutzt. Jede Knoten-ID befindet sich anfangs in einem Element eines Arrays der Länge der Anzahl aller Haltestellen. Die Zellennummern des Arrays entsprechen den Knoten und die Werte innerhalb des Arrays der Nummer einer Menge. Im ersten Durchlauf durch die sortierte Menge von Basislinien werden nur Kanten ausgewählt, deren drei Knoten in drei verschiedenen Mengen enthalten sind. Bei einer Auswahl der Basislinie werden die Mengen, in denen sich die Haltestellen befinden, miteinander zu einer Menge verschmolzen. Befinden sich nach dem Durchlauf nicht alle Haltestellen in einer Menge, werden in einem weitem Durchlauf Basislinien ausgewählt, in der zwei der drei Haltestellen in verschiedenen Mengen enthalten sind. In einem letztem Durchlauf werden solange Basislinien aus der sortierten Liste ausgewählt, bis die Gesamtlinienlänge erreicht ist.

Zum Schluss werden die ausgewählten Basislinien in die Datenstruktur Linie übertragen.

7.6.1.12 Startverknüpfung - eindeutige Linienverknüpfung

Die eindeutigen Verknüpfungen zwischen zwei Linien sucht über alle Kanten die Linien heraus, in der sich genau zwei Kanten überlagern. Da es in diesem Fall keine Interpretationsmöglichkeit gibt, fällt somit die Wahl eindeutig aus. Es ist dabei darauf zu achten, dass die Endknoten der überlagerten Kante unterschiedlich sind bzw. sich keine Stichfahrten ergeben. Das heißt, dass ein Bus nicht in eine Straße hineinfährt, dort wieder wendet und dieselbe Straße wieder zurückfährt.

7.6.1.13 Abschlußverknüpfung - mehrdeutige Linienverknüpfung

In dieser Stufe sind alle eindeutigen Linienverknüpfungen durchgeführt worden. Um eine optimale Verknüpfung von Linien zu finden, wird versucht in jedem Durchlauf jede Linie mit jeder anderen zu verknüpfen. Wenn eine erlaubte Überlagerung der Endkanten vorliegt, auf dieser Endkante nicht schon in einem vorherigen Schritt eine Verknüpfung durchgeführt worden ist, sich durch solch eine Verknüpfung kein Zyklus bildet und die Gesamtlänge der resultierenden Linie die erlaubte, maximale Gesamtlänge nicht überschreitet, wird der sich aus dieser Verbindung ergebende Attraktivitätsfaktor berechnet.

Auf diesem Wege kann der maximale Attraktivitätswert der Verknüpfungen bestimmt werden, der sich aus dem Vergleich aller Werte der möglichen Verknüpfungen ergibt. Diese Verknüpfung wird ausgeführt und die entsprechende Kante, auf der verknüpft worden ist, wird für weitere Verknüpfungen gesperrt. Auf diesem Weg wird sichergestellt, dass keine weitere Verknüpfung mehr auf dieser Kante stattfinden kann.

Anschließend beginnt die Suche nach der nächsten optimalen Verknüpfung von vorne. Dies setzt sich so lange fort, bis keine Verknüpfung mehr gefunden wird, die einen Attraktivitätsgewinn erzielen würde.

7.6.1.14 Abschlußverknüpfung - Aufbruchverknüpfung

In dieser Stufe können Linien neu zugeordnet und anders zusammengefügt werden, wenn dafür zur Einhaltung der Gesamtlinienlänge eine andere Linienverknüpfung aufgegeben wird. Damit solch eine Aufbruchverknüpfung realisiert werden kann, muss der sich dadurch ergebende Gewinn an Attraktivität größer sein als der Verlust, der dadurch entsteht, dass eine andere Verbindung aufgebrochen werden muss. Die für diese neue Verbindung disponierbare Kapazität wird somit aus der alten Verbindung freigesetzt. Bevor das eigentliche Verfahren beginnt, wird aus diesem Grund eine geordnete Liste von potentiellen Kanten aufgestellt, die während des Verfahrens aufgebrochen werden können. Zu diesem Zweck werden diejenigen Enden aller Linien bestimmt, die aufbrechbar sind, bzw. durch deren Aufbruch kein Knoten vom Liniennetz abgetrennt wird, weil mindestens zwei Linien diese Kante überdecken. Ist eine solche Kante gefunden, wird ihr Attraktivitätsverlust bestimmt. Dies geschieht auf die gleiche Weise wie beim Verknüpfen zweier Linien. Um so den Verlust zu relativieren und abzudämpfen, wird dieser Wert anschließend durch die Wurzel aus der freiwerdenden Kapazität geteilt. Auf diese Weise wird eine Liste erzeugt, die nach steigendem Attraktivitätsverlust sortiert ist.

Um die Neuverknüpfung zu realisieren, wird jede Linie einmal als Grundlinie angesehen. Auf dieser Grundlinie wird jeder Knoten abgeschritten, wobei überprüft wird, ob für dazwischen liegende Kanten eine Aufbruchverknüpfung zwischen zwei Linien vorliegt. Nach Sonntag wird dabei zwischen 5 Fällen unterschieden, wobei aber nur der fünfte Fall implementiert worden ist:

1. Eine von der Grundlinie abzweigende Linie wird zum Anfangsknoten der Grundlinie hin verlängert.
2. Eine auf die Grundlinie einmündende Linie wird bis zum Endknoten der Grundlinie hin verlängert.
3. Ein Mittelknoten der Grundlinie wird als ein Anfangsknoten betrachtet und die potentiellen Linien werden entsprechend bis dorthin verlängert.
4. Ein Mittelknoten der Grundlinie wird als ein Endknoten betrachtet und anschließend wird wie in Fall 3 vorgegangen.
5. Über die Knoten der Grundlinie hinweg werden zwei verschiedene Linien zu einer zusammengefasst, wobei hier alle möglichen Kombinationsmöglichkeiten von Linien, die auf der Grundlinie in diesem Knoten zusammentreffen können, ausgetestet werden müssen.

Bei jeder Verlängerung einer Linie muss der Richtungssinn der Linien beachtet werden, so dass keine Hin- und Rückfahrten auf einer Kante entstehen, sondern sich ein kontinuierlicher Linienzug ohne Zyklen bildet. Wenn für einen Knoten eine potentielle Aufbruchverknüpfung (einer der fünf Fälle) erkannt worden ist, wird eine Verknüpfungsdiskussion auf diesem Knoten durchgeführt, die abhängig vom betrachteten Fall ist. Während dieser Diskussion wird jede Linie darauf geprüft, ob sie zum einen den Bedingungen des Falles entspricht (ob sie also passend verlängert bzw. zusammengefasst werden kann) und zum anderen ob sie bei einer Verknüpfung keinen Zyklus erzeugen würde bzw. ihre Linienlänge sich dadurch nicht über das erlaubte Maximum vergrößern würde.

Wenn dies möglich ist, wird für diese Verknüpfung der Attraktivitätsfaktor und die sich durch diese Verknüpfung neu ergebende Überlastung der Kapazität (entsprechend dem sich durch die Verknüpfung ergebenden Neubedarf an Linienstrecke) bestimmt. Um diejenigen Linien zu ermitteln, die aufgebrochen werden müssen, benutzt man die am Anfang ermittelte Rangfolgenliste und arbeitet diese so lange ab, bis die benötigte Kapazität für diese Verknüpfung erreicht worden ist. Hierbei wird darauf geachtet, dass nicht die gerade aktuell betrachteten Linien gekürzt werden. Der sich bei jedem Aufbruch ergebende Attraktivitätsverlust wird aufsummiert. Dieser Attraktivitätsverlust wird dem Attraktivitätsgewinn gegenübergestellt. Wenn der Vergleich positiv ausfallen sollte, also ein effektiver Attraktivitätsgewinn vorliegt, wird der bisher beste Verknüpfungsfall mit diesem Verknüpfungsfall verglichen und so ein Maximalfall bestimmt. Wenn alle Linien abgearbeitet wurden, hat sich entweder keine Verbesserung einstellen können, womit diese Stufe beendet wird, oder es gibt unter den verschiedenen Verknüpfungsmöglichkeiten einen Fall des maximalen effektiven Attraktivitätsgewinns, der realisiert wird. Anschließend wird der nächste Maximalfall nach demselben Prinzip berechnet. Diese Prozedur wird so lange fortgesetzt, bis keine Verbesserung mehr möglich ist.

Es hat sich gezeigt, dass nur der Letzte bzw. der als Fünftes von Sonntag vorgestellte Aufbruchverknüpfungsfall von Interesse ist, da nur hier die Anzahl der Linien verringert und sinnvolle Verknüpfungen gefunden werden können. Aus diesem Grund wurde dieser Fall implementiert.

7.6.1.15 Abschlußverknüpfung - Linienendverknüpfung

In dieser letzten Stufe werden alle Kanten, die einen gemeinsamen Endknoten, aber keine gemeinsame Endkante habe, darauf hin geprüft, ob ihr Verlauf zusammengelegt werden kann. Zu diesem Zweck werden alle Linien paarweise miteinander auf dieses Kriterium hin verglichen. Ergibt sich durch die Zusammenführung zweier Linien kein Zyklus und wird durch den Zusammenschluss die maximale Linienlänge nicht überschritten, so wird der sich ergebende Basisstrom auf den Verknüpfungskanten berechnet. Am Ende ergibt sich so ein höchster Basisstromwert, dessen zugehörige Zusammenführung dann im Linienplan realisiert wird. Anschließend folgt ein weiterer Durchlauf dieser Verknüpfungsmethode, bis alle möglichen Kombinationen von Endknoten getestet worden sind.

7.6.1.16 Bewertungsphase - baue Fahrplangraph

In dieser Funktion wird aus dem Netzplangraphen und der Linienliste ein gerichteter Graph erzeugt, der Fahrplangraph, der der Definition eines Netz der Transportverbindungen entspricht. Dieser gerichtete, zusammenhängende Multi-Graph ([LEDA4.2]) modelliert mit der Knotenmenge wieder die Haltestellen, die von Buslinien angefahren werden können. Mit der Kantenmenge wird eine Buslinie repräsentiert, die die jeweils verknüpften Haltestellen verbindet. Da zwischen zwei Bushaltestellen mehrere Buslinien fahren können, kann es auch im Graphen zwischen Knoten mehrere Kanten geben, die sich aber jeweils durch die ID für die Buslinie unterscheiden. Als weitere Information wird die Fahrzeit, die sich aus der Entfernung und als Parameter übergebenen Geschwindigkeit berechnet, und der Fahrgaststrom auf der Kante abgespeichert, der sich auf dieser Linie während dieses Teilstücks vereinigt.

7.6.1.17 Bewertungsphase - Umlage Verkehrsbedarf

In der Umlage des Verkehrsbedarfs (Belastungsrechnung) werden die Fahrgastströme der einzelnen Linien für den Fahrplangraphen errechnet. Zu diesem

Zweck wird jede Haltestelle mit jeder anderen in Verbindung gebracht und der Weg zwischen den beiden Haltestellen bestimmt. Hierzu wird ein Bestewegeansatz eingesetzt. Ein Ansatz auf Alternativrouten wäre auch möglich, benötigt jedoch einen noch höheren Zeitaufwand und führt zu keinem direkten Vorteil. Anschließend wird der, in der Verkehrsbedarfsmatrix angegebene Wert für den Fahrgaststrom auf die Kanten gelegt (in diesem Fall also der entsprechenden Buslinie zugeordnet).

Bei dem Bestewegeansatz führt der Weg von jeder Haltestelle zu jeder Haltestelle über den kürzesten Weg, der im vorliegenden Liniennetz möglich ist. Die Berechnung des Weges wird mit einem entsprechenden Algorithmus durchgeführt (LEDA-Bibliothek Dijkstra). Es wird beim Abschreiten dieses Weges und dem Eintragen des Verkehrsstromes an die Kanten der jeweils betroffenen Linie versucht, so lange wie möglich auf dieser Linie zu bleiben, ohne umzusteigen. Steht dennoch ein Umsteigevorgang an, wird die Umsteigezeit des jeweiligen Knotens (3 Minuten) zur Reisezeit als Strafe hinzuaddiert.

7.6.2 Ablauf

1. Datenaufbereitung und Erzeugung des Netzplangraphen: Diese Phase ist dafür zuständig, dass sowohl zum einen der vom Haltestellenmodul gelieferte Wegenetzgraph in den internen Netzplan umgewandelt wird und zum anderen die Verkehrsbedarfswerte aus der Quelle-Ziel-Matrix auf die kürzesten Wege umgelegt werden.

2. Pflichtlinien einfügen: Die vom Benutzer eingegebenen Pflichtlinien werden in die Linienliste aufgenommen und im Netzplangraphen als befahren markiert.

3. Basislinienwahl: Die Ausführung der Schritte 3 bis 7 sind deaktivierbar, um ein reines Pflichtliniennetz zu bewerten und für das Modul Fahrplan vorzubereiten. Dabei wurde für Testzwecke die Möglichkeit eingeführt, die Linienverknüpfungen einzeln auszuwählen. Mittels des Netzplangraphen werden alle möglichen Basislinen erzeugt und nach der Größe der Ströme sortiert. Im Folgenden werden sie in dieser Reihenfolge betrachtet. Je nach Auswahl des Basislinienwahlalgorithmus wird sichergestellt, dass jede Kante durch eine Basislinie genutzt wird oder nur alle Haltestellen in einem zusammenhängendem Graphen von Basislinien enthalten sind. Abschließend werden die Basislinien in die Datenstruktur Linie umgewandelt.

4. Linienverknüpfungen: Die von der Basislinienwahl erzeugten Linien werden nacheinander der eindeutigen und mehrdeutigen Linienverknüpfung, Aufbruchverknüpfung und Linienendverknüpfung übergeben.

5. Bewertungsphase: Die Nachfragedaten werden auf das erzeugte Liniennetz gelegt und die sich daraus ergebenen Ströme zur Berechnung der Gütemaße herangezogen.

6. Erzeugung Fahrplangraph: Es wird aus dem Netzplangraphen und der Linienliste ein gerichteter Graph erzeugt, der im Hauptprogramm zur Speicherung der Daten und dem Fahrplan zur Weiterverarbeitung genutzt wird.

7.6.3 Zeitkomplexität

Zur Berechnung der Zeitkomplexität wird zuerst eine getrennte Berechnung von Aufbereitungsphase/Bewertungsphase, Basislinienwahl und Verknüpfungsphase durchgeführt. Anschließend werden die Ergebnisse zu der Gesamtlaufzeit zusammengerechnet. In der Verknüpfungsphase werden kontinuierlich zwei Linien zu einer verknüpft, so dass sich die Anzahl der Linien ständig verändert. Da eine Verknüpfung von der konkreten Situation abhängt, ist k nicht näher abschätzbar.

Vereinbarungen

e	Anzahl der Kanten
n	Anzahl der Knoten
k	Anzahl der Linien
w	Länge des kürzesten Weges

Aufbereitungsphase/Bewertungsphase

Da sowohl am Anfang der Linienplanung als auch am Ende eine Umlage über alle Knoten erfolgt, die in beiden Fällen die zeitaufwendigste Arbeit ist, lässt sich die Laufzeit gemeinsam abschätzen. Von jedem Knoten wird einmal der Algorithmus von Dijkstra aufgerufen mit Laufzeit $O(e + n\,logn)$ (Quelle [LEDA4.2]). Um den kürzesten Pfad zu jedem anderen Knoten zu erhalten, muss noch weitere n-mal das Ergebnis aufgerufen werden. In der Abarbeitung des kürzestes Pfades werden nur einfache Zuordnungen mit Laufzeit $O(1)$ durchgeführt. Somit beträgt die Gesamtlaufzeit der Aufbereitungsphase/Bewertungsphase $O(e * w * n^2 + w * n^3 * logn)$, gekürzt:

$$O(n^3 * logn)$$

Basislinienwahl

In der Basislinienwahl werden alle Kanten abgearbeitet und einsortiert (Laufzeit $O(e^2)$). Es werden dabei $\frac{e}{2}$ Basislinien erstellt und bearbeitet. Dies kann man mit $O(\frac{e}{2})$ abschätzen. Insgesamt weist die Basislinienwahl eine Gesamtdauer von

$$O(e^2)$$

auf.

Verknüpfungsphase

Die Berechnung der Verknüpfungsphase unterteilt sich weiter in eindeutige, mehrdeutige, End- und Abschlussverknüpfung. Da in der mehrdeutigen Verknüpfung und in der Endverknüpfung von der Laufzeit gesehen dasselbe geschieht, können sie gemeinsam berechnet werden. Wenn also im Folgenden eine von den beiden Verknüpfungen aufgeführt wird, ist die andere gedanklich hinzuzufügen.

In der eindeutigen Verknüpfung werden alle Linien mit allen anderen verglichen. Die Dauer ist somit $O(k^2)$.

Die Endverknüpfung wird solange durchgeführt bis es keine Verknüpfungsmöglichkeiten gibt. Die Anzahl der Iterationen kann nicht berechnet werden, da dies von der spezifischen Situation abhängt. Sie ist jedoch im Verhältnis zu e bzw. k eher klein und kann daher vernachlässigt werden. Bei jeder Möglichkeit werden alle Linien miteinander verglichen, so dass auch hier die Laufzeit $O(k^2)$ beträgt.

In der Aufbruchverknüpfung werden immer drei Linien miteinander verglichen und ineinander geschachtelt. Da jedoch nicht alle Möglichkeiten in Betracht gezogen werden, kann auch hier eine Komplexität von $O(k^2)$ angenommen werden.

Insgesamt ergibt sich also für die Verknüpfungsphase eine Laufzeit von:

$$O(k^2)$$

Gesamtzeitkomplexität

Im Laufe der Linienplanung, werden alle Phasen genau einmal durchlaufen. Daher können alle Einzel-Laufzeit addiert werden. Da k im Verhältnis zu n und e vernachlässigbar ist, kann $O(k^2)$ wegfallen. Somit ergibt sich eine Gesamtlaufzeit von:

$$O(n^3 * logn + e^2)$$

7.7 Umsetzung des Fahrplans

7.7.1 Realisierung

Es wird nun auf die konkrete Realisierung des Moduls Fahrplan eingegangen. Dabei wird aufgezeigt, wie das mathematische Modell konkret auf dem Rechner umgesetzt wurde. Anzumerken ist noch, dass die Reihenfolgeder Kapitel hier nicht zufällig ist, vielmehr entspricht die Reihenfolge der Implementierung.

Aufbau des Transportkettengraphen

Der Transportkettengraph ist im Programm als parametrisierter, gerichteter Graph (LEDA-Semantik) implementiert, d.h. an seinen Knoten und Kanten stehen jeweilige benutzerdefinierte Typen. In unserem Fall werden die Klassen CKnotenBeschriftung und CKantenBeschriftung bzw. Instanzen dieser Klassen an die Knoten und Kanten gebunden. Diese enthalten die jeweiligen Kennwerte der Knoten und Kanten des Transportkettengraphen. Die Kennwerte für die Knoten sehen wie folgt aus:

1. Knoten-ID
2. ID der Haltestellen, die diesen Knoten repräsentieren
3. Liniennummer hinsichtlich der Linie, die diese Verbindung befährt
4. Fahrzeit zwischen den beiden Haltestellen
5. relative Abfahrtszeit

Kantenkennwerte setzen sich hieraus zusammen:

1. Kanten-ID
2. Stärke des Verkehrsstroms
3. Netzbedingte Wartezeit
4. Mindestumsteigezeit
5. Mindesthaltezeit
6. Kennzeichnung, ob es eine Umsteigekante darstellt
7. Kennzeichnung, ob es sich um eine feste Bindung handelt
8. Kennzeichnung, ob diese Kante im Maximalgerüst enthalten ist
9. Position im Simplextableau

Die bis zu diesem Zeitpunkt noch nicht erläuterten Kennwerte werden in folgenden Kapiteln und Unterkapiteln in entsprechender Weise definiert.

7.7.1.1 Umbau des Netzes der Transportverbindungen

Der Transportkettengraph wird aus dem Netz der Transportverbindungen folgendermaßen erzeugt.

Es wird eine Iteration über alle Kanten des Netzes der Transportverbindungen durchgeführt. Für jede dieser Kanten wird ein neuer Knoten im Transportkettengraphen erzeugt. An jeden dieser Knoten wird eine entsprechende Instanz von CKnotenBeschriftung gebunden, wobei diese mit den Kennwerten KnotenID (fortlaufende interne ID), den beiden Haltestellen-IDs (identisch hinsichtlich des Netzes der Transportverbindungen) und der Fahrzeit (siehe Netz der Transportverbindungen) initialisiert werden. Die anderen Kennwerte werden jeweils mit Standard-Werten belegt (0 oder false). Damit eine effiziente Erstellung der Knotenverbindungen möglich ist, wird ein Hash-Array aufgebaut, welches von den Kanten des Netzes der Transportverbindungen (Key) auf die entsprechenden Knoten des Transportkettengraph (Value) verweist.

Nachdem dieses abgeschlossen ist, werden die Kanten im Transportkettengraph erzeugt. Eine Kante zwischen zwei Knoten stellt einen Umsteigepunkt zwischen zwei Linien im Netz der Transportverbindungen dar. Folglich wird über alle Knoten des Netzes der Transportverbindungen iteriert und für jeden dieser Knoten werden seine Ingradkanten betrachtet. Für jede Ingradkante werden die Outgradkanten des aktuellen Knoten analysiert und bei Vorhandensein einer Umsteigekante wird eine Kante in den Transportkettengraph eingefügt (hinsichtlich der Ingradkante und der Outgradkante, denen zwei Knoten im Transportkettengraphen gegenüberstehen). Dabei wird das oben erwähnte Hilfs-Hash-Array verwendet.

An dieser Stelle werden auch die Kantenkennwerte gesetzt. Der Verkehrsstrom ergibt sich hierbei aus der Summe der Verkehrsstrom-Werte an den Kanten des Netzes der Transportverbindungen.

7.7.1.2 Interaktive Festlegung Parameter Mindesthaltezeiten

Seitens des Benutzers besteht die Möglichkeit, die Mindesthaltezeiten bezüglich der einzelnen Haltestellen interaktiv zu setzen bzw. zu verändern. Dabei gibt die Mindesthaltezeit an, wie lange ein Bus einer bestimmten Haltestelle halten muss, um die Fahrgäste aussteigen bzw. einsteigen zu lassen.
Seitens der Programmgruppe wird daher über die Schnittstelle ein Hash-Array an das Modul Fahrplan übergeben. Dieses Hash-Array beinhaltet für die entsprechend vom Benutzer spezifizierten Haltestellen die Mindesthaltezeiten.

Dieses Hash-Array wird an alle Zusammenhangskomponenten weitergereicht, da eine Haltestelle in verschiedenen Zusammenhangskomponenten liegen kann. In diesen Komponenten wird dann über alle Knoten des Graphen der Zusammenhangskomponente iteriert. Für jeden, der so erhaltenen Knoten, werden die einzelnen ausgehenden Kanten betrachtet. Folglich werden an diesen Kanten bzw. an der Kantenbeschriftung der jeweiligen Kante die Mindesthaltezeit, falls spezifiziert, vermerkt.

Es sollte dabei beachtet werden, dass eine Kante im Transportkettengraph (bzw. in den jeweiligen Zusammenhangskomponenten), grob gesagt, die Verbindungshaltestelle zwischen zwei Haltestellen darstellt. Daher muss aus dem Hash-Array der Eintrag für die zweite Haltestelle hinsichtlich des aktuell betrachteten Knoten ausgelesen werden.Interaktive Festlegung der Parameter Mindestumsteigezeit

Weiterhin ist es seitens des Benutzers möglich, die Mindestumsteigezeiten an den jeweiligen Haltestellen zwischen zwei dort fahrenden Linien zu setzen bzw. zu ändern. Die Mindestumsteigezeit besagt hierbei also, wie lange es dauert, von Linie A nach Linie B an der Haltestelle C umzusteigen.

Über die Schnittstelle wird dementsprechend ein Hash-Array übergeben, das ein Mapping von einer Haltestelle (ID) auf eine Liste von Tripeln darstellt. Die einzelnen Tripeln der Liste sagen das oben erwähnte aus, nämlich die Umsteigezeit (in Minuten) zwischen zwei spezifizierten Linien (als IDs) an der jeweiligen Haltestelle.

Daraufhin wird ähnlich der obigen Schilderung der Festlegung der Mindesthaltezeiten verfahren. Das erhaltene Hash wird an die einzelnen Zusammenhangskomponenten weitergegeben. Dort wird wiederum über alle Knoten iteriert und falls für die Umsteigekante (zweite Haltestellen-ID der Knotenbeschriftung) Werte bzw. eine Liste spezifiziert sind bzw. ist, werden die ausgehenden Kanten betrachtet. Falls diese Kante eine Umsteigekante ist, d.h. es findet ein Linienwechsel statt (Knoten haben unterschiedliche Liniennummern in der Knotenbeschriftung), wird für diese Kante mittels einer linearen Suche ein entsprechender Eintrag in der Liste der Umsteigezeiten gesucht und falls gefunden an die jeweilige Kantenbeschriftung geschrieben.

7.7.1.3 Veränderung der Festlegungen von Bindungen

Angenommen der Benutzer des Endproduktes hat sich einen Linienplan erstellen lassen, betrachtet ihn nun aufmerksam und entdeckt eine Haltestelle, an welcher sich zwei Linien treffen. Aufgrund seiner Erfahrung bei der Erstellung von Verkehrsnetzen weiß er nun, dass es gut wäre, wenn der Fahrgast ohne unnötige Wartezeit von Linie A zu Linie B wechseln kann.
Im Kontext des mathematischen Modells würde das bedeuten, dass diese Ver-

bindung im Transportkettengraphen keine netzbedingten Wartezeiten enthalten soll, also folglich eine Kante im Gerüst werden muss. Diesem Umstand wird mittels der festen Verbindungen Rechnung. Ist eine Kante im Transportkettengraphen mit dem Flag feste Verbindung versehen, so wird sie auf jeden Fall in das Maximalgerüst aufgenommen, sofern dies möglich ist.

7.7.1.4 Mögliche Fehler bei der Wahl fester Bindungen

Es ist dem Benutzer nicht erlaubt, jede mögliche Kombination und Anzahl von festen Bindungen zu wählen, da ansonsten Kreise entstehen. Die Anzahl der vom Benutzer in einer Zusammenhangskomponente wählbaren Kanten entspricht der zyklomatischen Zahl. Ist die Anzahl der vom Benutzer eingegebenen Kanten größer als die zyklomatische Zahl, so ist diese Fehleingabe zu erkennen und der Benutzer auf seine Fehleingabe hinzuweisen. Weiterhin darf sich die Fehleingabe nicht auf den Programmablauf auswirken, es darf nicht zu einem Absturz kommen.

Setzen fester Bindungen (Ablauf des Algorithmus)

Das Modul Programm übergibt an das Modul Fahrplan ein 4-Tupel. Die einzelnen Werte des 4-Tupels sind wie folgt zu interpretieren:

1. ID der Zusammenhangskomponente, in welcher sich die Verbindung befindet.
2. ID der Linie 1.
3. ID der Linie 2.
4. ID der Haltestelle, welche die beiden Linien verbindet.

Mittels dieser Informationen ist es sehr leicht möglich, die entsprechende Kante im Transportkettengraphen zu finden. Zu diesem Zweck wird über die erhaltene Liste iteriert und jeder Eintrag auf die nachfolgend beschriebene Art behandelt. Um das Vorgehen besser nachvollziehen zu können, wurde folgender Beispieleintrag generiert:$E = (0, 2, 1, 6)$.
Beim Durchlauf durch die Liste sei E das aktuell gewählte Listenelement. Die Kante, welche im Transportkettengraph mit dem Flag feste Verbindung versehen werden soll, befindet sich also in Zusammenhangskomponente 0. An die Methode *setzenFesteVerbindung* in der nullten Zusammenhangskomponente des Transportkettengraphen werden nun die Informationen $(2, 1, 6)$ übergeben, die Nummer der Zusammenhangskomponente wird nicht mehr benötigt. Die Methode *setzenFesteVerbindung*macht nun folgendes:

1. Finden der Kante.
2. Überprüfen, ob Wahl der Kante legal ist und wenn ja, setzen des Flags feste Verbindung, andernfalls ausgeben einer Fehlermeldung.

Dieses wurde wie folgt realisiert:

1. Die Methode *setzenFesteVerbindung* übergibt die Werte $(2, 1, 6)$ an die Methode *suchenUmsteigekante* in der gleichen Zusammenhangskomponente. Diese iteriert nun über alle Kanten dieser Zusammenhangskomponente und vergleicht mittels der Informationen des Source- und Targetknotens der jeweils betrachteten Kante, ob es sich um die gesuchte Verbindung handelt. Es handelt sich an dieser Stelle um nichts anderes als den

Vergleich von Integer-Werten. Wurde die Kante gefunden, so wird sie an die Methode *setzenFesteVerbindung* übergeben. Falls sie nicht gefunden werden sollte, was eigentlich nicht möglich ist, wird eine entsprechende Fehlermeldung geworfen.

2. Wurde die entsprechende Kante gefunden und so wird sie an die Methode *setzenUmsteigekante*, welche wiederum in der gleichen Zusammenhangskomponente zu finden ist, übergeben. Dieser Methode obliegt es nun, zu prüfen, ob der Benutzer eine Fehleingabe im Sinne des vorangegangene Kapitels vorgenommen hat, oder nicht.

   Zu diesem Zweck wird wie folgt vorgegangen. Sobald zum ersten Mal eine Kante in einer Zusammenhangskomponente mit dem Flag feste Verbindung versehen werden soll, wird eine Knotenpartition Pinitialisiert. P besteht aus allen Knoten, welche sich in der entsprechenden Zusammenhangskomponente befinden. Jeder dieser Knoten wird zu Beginn in eine eigene Partition geschoben.

   Von der Methode *setzenFesteVerbindung* wurde die Kante mit den Werten $(2, 1, 6)$ übergeben. Es ist nun noch zu prüfen, ob die Benutzerauswahl legitim ist. Legitim bedeutet in diesem Zusammenhang, dass kein Zyklus durch die Wahl dieser Kante als Kante mit fester Verbindung entstehen darf. Dies ist aber offensichtlich nur genau dann der Fall, wenn die beiden Knoten, welche die Kante verbindet, bereits in einer Partition sind. Es wird mittels LEDA-Methoden überprüft, ob sich die beiden Knoten, welche die Kante $(2, 1, 6)$ verbindet, in einer Partition befinden. Nun können zwei Fälle auftreten:

   - Die Knoten befinden sich nicht in einer Partition. Die Kante wird mit dem Flag feste Verbindung versehen werden und die beiden Partitionen der Knoten der Kanten werden vereinigt.
   - Die Knoten befinden sich in einer Partition. Die Wahl dieser Kante als Kante mit fester Verbindung ist nicht möglich. Es wird eine Exception geworfen. Weder an der Knotenpartition, noch am Flag feste Verbindung wird irgendetwas verändert.

Dieses Vorgehen wird auf alle Eintrage der Liste angewendet.

7.7.1.5 Erzeugung der Ausganglösung

Mittels einer Heuristik wird für jede Zusammenhangskomponente eine Ausgangslösung ermittelt. Wie bereits in der Beschreibung des mathematischen Modells dargestellt,ist die Anzahl der möglichen Gerüste für einen gegebenen Graphen sehr groß. Jedes dieser Gerüste entspricht jedoch einem gültigen Fahrplan. Um **den** optimalen Fahrplan zu bekommen, müsste man alle möglichen Gerüste berechnen und diese dann nach bestimmten Kriterien miteinander vergleichen. Dieses ist im endgültigen Programm möglich, allerdings ist der Rechenaufwand bereits bei kleinen Beispielen hoch.

7.7.1.6 Beschreibung der verwendeten Heuristik

Am Netz der Transportverbindungen sind neben einigen weiteren Informationen die Verkehrströme auf den Verbindungen vermerkt. Dieser Wert gibt an, wieviele Leute im Durchschnitt auf einer Verbindung verkehren. Die Heuristik besteht nun darin, diejenigen Strecken, welche von vielen Leuten benutzt werden zu bevorzugen, also auf diesen Verbindungen keine netzbedingten Wartezeiten entstehen zu lassen. In dem Moment, indem das Gerüst mittels des Algorithmus von Kruskal berechnet wird, wählt man bevorzugt diejenigen Kanten aus, auf welchen ein hoher Verkehrsstrom liegt. Kanten mit fester Verbindung dürfen hier nicht vergessen werden.

7.7.1.7 Die Vergleichsklasse

Zur Sortierung der Kanten wird ein von LEDA bereitgestellter Quicksort benutzt. Bei diesem besteht die Möglichkeit eine Methode selbst zu definieren, welche zur Sortierung der Kanten verwendet wird. Diese wurde wie folgt realisiert (die Reihenfolge entspricht der Reihenfolge in der Vergleichsklasse, als Eingabe erhält die Klasse stets zwei Kanten, Rückgabewert ist ein Integer. Dieser ist -1, wenn die erste Kante die größere ist, 1 falls die zweite Kante größer ist und 0 falls beide Kanten gleich sind.):

1. Fall: Beide Kanten sind mit dem Flag feste Bindung versehen: Es wird -1 oder 1, je nachdem welche Kanten den größeren Verkehrsstrom hat, zurückgegeben. Haben beide Kanten gleiche Verkehrsströme, wird 0 zurückgegeben.

2. Fall: Eine der beiden Kanten ist mit dem Flag feste Verbindung versehen: Je nachdem ob es sich um die erste oder zweite Kante handelt, wird -1 oder 1 zurückgegeben. Eine Null kann in diesem Fall offensichtlich nicht der Rückgabewert sein.

3. Fall: Keine der beiden Kanten ist mit dem Flag feste Verbindung versehen: Es wird ein Vergleich analog zum ersten Fall durchgeführt.

Nachdem alle Kanten mittels Quicksort zuzüglich der Vergleichsklasse sortiert sind, beginnt der Aufbau des Gerüstes.

7.7.1.8 Berechnung des Gerüstes (Ablauf des Algorithmus)

Der Algorithmus von Kruskal wurde wiederum der LEDA-Bibliothek entnommen. Als Rückgabe liefert dieser eine Liste der Gerüstkanten. Ablauf Kruskal:

1. Sortierung der Kanten mittels Quicksort.

2. Einbettung der Knoten des Graphen in jeweils eine einelementige Partition.

3. Falls es mehr als eine Knotenpartition gibt: Wahl der ersten Kante (also derjenigen, welche im Anschluss an die Sortierung und zum jeweiligem Zeitpunkt die „größte" Kante ist). Falls es nur eine Knotenpartition gibt, terminiert der Algorithmus.

4. Test, ob die Knoten, welche diese Kante verbindet, bereits in einer Partition liegen.

    - 1. Fall: Die Knoten liegen in einer Partition: Die Kante wird verworfen, weiter bei 3.

- 2. Fall: Die Knoten liegen nicht in einer Partition: Die gerade betrachtete Kante wird in die Gerüstkantenliste eingefügt, die beiden betrachteten Knotenpartitionen werden gemerget, weiter bei 3.

7.7.1.9 Erzeugung des Simplextableau

Der Aufbau des Simplextableaus (für jede einzelne Zusammenhangskomponente) erfolgt zur Berechnung der netzbedingten Wartezeiten. Voraussetzung für diese Berechnung ist die Erstellung eines maximalen Gerüstes. Die Aufstellung dieses Tableaus ist auch sinnvoll im Hinblick auf die Verbesserung der einfach errechneten Heuristik mittels der Methode der Lokalen Suche.

Wie sich in der Testphase auf den kompletten Herne-Daten zeigte, war der Speicherplatzbedarf der ursprünglichen Implementierung des Simplextableaus sehr hoch. Durch eine Implementierung des Simplextableau als spärlich besetzte Matrix, konnte dieses Problem umgangen werden.

7.7.1.10 Aufbau des Simplextableaus

Prinzipiell gesehen stellt das Simplextableau nichts anderes als eine Matrix dar. Die Zeilen stehen für die Basisvariablen, d.h. für jede Kante des Transportkettengraphen, die sich nicht im maximalen Gerüst befindet bzw. die eine netzbedingte Wartezeit enthält, existiert eine Zeile. Dementsprechend ist für jede Nicht-Basisvariable, d.h. die Kanten die im maximalen Gerüst sind bzw. die keine netzbedingten Wartezeiten haben, eine Spalte vorhanden. Die Einträge der Matrix beinhalten die Werte -1, 0 oder 1. Diese Einträge bedeuten folgendes:

Falls die zugrunde liegende Maximal-Gerüstkante (Spalte) auf dem ungerichteten Kreis liegt, der durch Hinzunahme der Nicht-Maximal-Gerüstkante (Zeile) in das Maximalgerüst entsteht, wird der Matrixeintrag entweder auf -1 oder 1 gesetzt. Falls die Nicht-Basisvariable bzw. Maximalgerüst-Kante dieselbe Richtung auf dem Kreis hat wie die Nicht-Maximalgerüst-Kante wird eine 1 vermerkt, ansonsten eine -1. Falls die zugrunde liegende Maximalgerüst-Kante nicht auf dem Kreis liegt, ist der entsprechende Matrixeintrag 0.

Folglich ist das Simplextableau eine einfache Matrix (dünnbesetzt), die aus soviel Zeilen besteht, wie es Nichtgerüstkanten für die betrachtete Zusammenhangskomponente gibt, und genauso viele Spalten, wie Gerüstkanten vorhanden sind. Hierbei ist jedoch zu bemerken, dass nur solche Spalteneinträge abgespeichert werden, die einen Wert ungleich Null aufweisen. Um einen Zusammenhang zwischen Spalten- bzw. Zeilenindizes und Gerüst- und Nicht-Gerüstkanten herzustellen, wird in der entsprechenden Kantenbeschriftung die Position im Simplextableau vermerkt. Über die boolesche Variable istImMaximalGerüst hinsichtlich der Kantenbeschriftung kann entschieden werden, ob dieser Index ein Spalten- oder Zeilenindex ist. Gleichzeitig wird zur effizienten Implementierung ein Hash aufgebaut, das die umgekehrte Relation beinhaltet, d.h. für einen Spalten- bzw. Zeilenindex wird die entsprechende Kante vermerkt (also Key: Index; Value: Kante).

Zur Abspeicherung der netzbedingten Wartezeiten, die errechnet werden, dient ein normaler Vektor, der soviel Einträge hat, wie Zeilen im Simplextableau vorhanden sind. Die dort errechneten Einträge werden an die entsprechenden Nicht-Gerüstkanten geschrieben.

7.7.1.11 Berechnung des Simplextableaus

Die Klasse CSimplextableau erhält einen Graphen bezüglich der Zusammenhangskomponente, für die das Simplextableau erzeugt werden soll. Nun wird die oben erwähnte Verknüpfung von Kanten und Zeilen bzw. Spalten vorgenommen. Dieses bedeutet, dass eine Iteration über alle Kanten des Graphen der Zusammenhangskomponente erfolgt und für jede dieser Kanten überprüft wird, ob diese Kante im Maximalgerüst ist oder nicht (durch Betrachtung der Kantenbeschriftung). Ist eine Maximalgerüstkante vorhanden, wird der aktuelle Spaltenindex inkrementiert und in der jeweiligen Kantenbeschriftung vermerkt. Umgekehrt wird an dieser Position mittels des Spaltenhashes die Kante als Value eingetragen. Analog wird mit den Nicht-Gerüst-Kanten verfahren.

Nachdem diese Initialisierung vorgenommen ist, erfolgt eine Füllung des Simplextableau mit Werten. Dazu werden für jede Nichtgerüstkante (jede Zeile) die Gerüstkanten des Kreises berechnet, der bei Hinzufügen der Nichtgerüstkante in das Maximalgerüst entsteht. Für diese Berechnung wird eine Tiefensuche von dem Targetknoten der jeweiligen Nichtgerüstkante durchgeführt. Dabei werden nur solche Kanten berücksichtigt, die eine Gerüstkante darstellen. Falls eine Nichtgerüstkante gefunden wird, wird die dortige Tiefensuche beendet. Diese Tiefensuche endet bei Erreichen des Source-Knotens der Nicht-Gerüstkante. Bei dieser Suche wird der Graph als ungerichtet interpretiert und die gefundenen Kreiskanten auf einen Stack zwischengespeichert.

Nachdem die Zykluskanten berechnet worden sind, erfolgt die eigentliche Füllung der aktuellen Zeile des Simplextableaus. Dazu werden die Kanten des Zyklusstacks nacheinander entfernt, und die Richtung zur Nicht-Gerüstkante bestimmt. Dieses kann einfach dadurch entschieden werden, indem die Richtung der Vorgängerkante und die jeweilige Knoten-ID des Endknotens (hinsichtlich des Kreisumlaufs) der Vorgängerkante gespeichert wird. Nämlich durch das Abspeichern auf einen Stack kann der Kreisumlauf entsprechend mit Beginn der Nicht-Gerüstkante nachvollzogen werden. Als beispielhafter Ablauf sei dieser erwähnt: Als erste Kante wird die Nicht-Gerüstkante betrachtet, wobei als aktuelle Richtung eine eins vermerkt wird und als Vorgänger-ID die Knoten-ID des Source-Knotens der Nicht-Gerüstkante. Dann wird die erste Kante des Stacks betrachtet. Hat der Targetknoten dieselbe ID wie die vermerkte Vorgänger-ID, braucht die aktuelle Richtung nicht verändert werden (bleibt also eins). Die Vorgänger-ID bekommt den ID-Wert des Source-Knotens der aktuellen Kante. Entsprechend wird fortgefahren. Hat jedoch nun der aktuelle Targetknoten eine unterschiedliche ID, ändert sich die Richtung und als neuer Vorgänger-Knoten-ID wird die Targetknoten-ID vermerkt. Hinsichtlich der nächsten Kante muss nun als Vergleichs-ID, die des Sourceknotens genommen werden. Dieses wird fortgesetzt, bis der Stack leer ist.

7.7.1.12 Berechnung der netzbedingten Wartezeiten

Nachdem das Simplextableau berechnet worden ist, können nun die netzbedingten Wartezeiten berechnet werden. Dazu wird entsprechend der geschilderten mathematischen Verhältnisse vorgegangen. Dieses bedeutet, dass für die jeweiligen Zeilen des Simplextableaus über die Spalteneinträge iteriert wird. Bei Auffinden eines Nicht-Null-Eintrags werden für die der Spalte entsprechenden Kante die Mindestumsteigezeit und Mindesthaltezeit summiert, mit der der invertierten Richtung multipliziert und zur momentanen netzbedingten Wartezeit addiert. Zusätzlich wird noch die Fahrzeit (multipliziert mit der invertierten Richtung) des Source-Knotens dieser Kante zu der netzbedingten Wartezeit summiert.

Somit sind die netzbedingten Wartezeiten berechnet. Jedoch muss jetzt noch die Beachtung des Taktes erfolgen, d.h. die netzbedingte Wartezeit wird modulo Taktzeit gerechnet, so dass die netzbedingte Wartezeit größer Null und kleiner als der Takt ist.

Diese Werte werden an den entsprechenden Kanten in der Kantenbeschriftung vermerkt, womit die Berechnung des Simplextableaus abgeschlossen ist.

7.7.1.13 Sukzessive Lösungsverbesserung

Die sukzessive Lösungsverbesserung ist in der theoretischen Informatik im Bereich der Lokalen Suche anzusiedeln. Dieses bedeutet, dass ausgehend von einer Basislösung, der oben genannten Ausgangslösung, eine lokale Umgebung definiert wird, in der sich weitere mögliche Lösungen befinden und in der damit potentielle Lösungsverbesserungen vorhanden sein können. Die in dieser lokalen Umgebung befindlichen Lösungen ergeben sich aus der Ausgangslösung, indem diese an geeigneten Stellen abgewandelt und so versucht wird, eine Verbesserung der Lösung zu erzielen.

Die lokale Umgebung in diesem speziellen Fall ist wie folgt definiert. Das Problem der Ausgangslösung ist, dass große netzbedingte Wartezeiten an Kanten entstehen können, die einen großen Verkehrsstrom beinhalten. Dieses bedeutet, dass dort der Reiseaufwand als Produkt der netzbedingten Wartezeit und der Reisendenzahl entsprechend groß ist. Dieses wird nun versucht, zu verhindern, indem diese Kante, welche eine Nichtgerüstkante ist, als eine Gerüstkante ausgezeichnet wird und eine Kante, die sich in dem Zyklus befindet, zu einer Nichtgerüstkante mutiert. Es findet also ein Wechsel oder Tausch von Nichtgerüstkante und Gerüstkante statt. Damit entsteht ein neues Gerüst, also eine neue Lösung. Diese Lösung wird nun hinsichtlich einer Lösungsverbesserung untersucht, wobei als Kriterium der Lösungsverbesserung der gesamte Reiseaufwand, der sich aus der Summe der Reiseaufwände der Nichtgerüstkanten ergibt, dient. Sollte sich eine Lösungsverbesserung feststellen, wird diese Lösung als neue Ausgangslösung ausgezeichnet und seitens dieser eine neue Lokale Suche gestartet. Wenn der negative Fall eintritt, dass sich keine Lösungsverbesserung findet, wird versucht, einen anderen Tausch von Gerüstkante gegen Nichtgerüstkante vorzunehmen.

Tritt in einer lokalen Umgebung keine Lösungsverbesserung mehr auf, wird diese Lösung als lokal optimal bezeichnet. Damit ist die Berechnung abgeschlossen.

Der genaue Ablauf der sukzessiven Lösungsverbesserung lässt sich wie folgt darstellen. Ausgehend von der Ausgangslösung werden die Nichtgerüstkanten entsprechend ihres Reiseaufwandes absteigend sortiert. Diese entstehende Liste wird absteigend abgearbeitet. Der aktuelle Listeneintrag korrespondiert mit einer Nichtgerüstkante, die sich in einem Zyklus des Transportkettengraph befindet. Diese Zykluskanten sind mit ihrer jeweiligen Orientierung im Simplextableau, d.h. in der dünn besetzten Matrix, vorhanden. Diese Gerüstkanten werden der Reihe nach (beliebig) bearbeitet. Dabei wird der oben geschilderte Ablauf des Tauschens von Gerüstkante gegen Nichtgerüstkante durchgeführt.

Dabei wird das Simplextableau entsprechend umgerechnet. Die Rechenregeln für dieses Vorhaben sind dem Gebiet des Operations Research zu entnehmen. Das Stichwort hierfür ist der Tausch einer Basisvariable gegen eine Nichtbasisvariable. Auch [Weigand81] schildert in seiner Dissertation auf den Seiten 51ff. diese Rechenoperationen.

Nach dem Umrechnen des Simplextableaus wird geprüft, ob sich eine Lösungsverbesserung hinsichtlich des gesamten Reiseaufwandes ergeben hat. Sollte dieses der Fall sein, wird die berechnete Lösung als neue Ausgangslösung angesehen und das Verfahren der lokalen Suche beginnt von neuen. Wenn keine Lösungsverbesserung vorhanden sein sollte, wird die Umrechnung des Simplextableaus rückgängig gemacht und in der Liste der Gerüstkanten wird die nächste betrachtet. Sollte das Ende dieser gefunden werden, wird in der Liste der Nichtgerüstkante die nachfolgende untersucht und somit in einem neuen Zyklus versucht, Verbesserungen zu finden.

Sollte das Ende der Liste der Nichtgerüstkanten erreicht werden, ist die aktuell gefundene Lösung lokal optimal und somit ist die Berechnung der sukzessiven Lösungsverbesserung abgeschlossen.

7.7.1.14 Entscheidungsbaumverfahren

Das Entscheidungsbaumverfahren hat die Eigenschaft eine ideale Lösung für den Algorithmus nach Weigand zu berechnen. Der Nachteil dabei ist, das dieser eine extrem große Laufzeit hat. Zur Berechnung der idealen Lösung wird ein Binärer-Baum aufgebaut, der in jeder Ebene eine Kante des Transportkettengraphen hat. In jeder Ebene wird dann gewählt, ob die Kante der Ebene in das Maximalgerüst aufgenommen werden muss, oder nicht aufgenommen werden darf.

Der Ablauf des Entscheidungsbaumverfahrens ist wie folgt. Es wird beim Rootknoten des Baumes begonnen. Als erstes wird die Kante der ersten Ebene in das Gerüst aufgenommen. Mit dieser Information wird nun eine Ausgangslösung berechnet, indem die Kante der ersten Ebene ins Gerüst kommt. Die so berechnete Ausgangslösung wird mit dem zugehörigen Wert des Wartezeitaufwandes als derzeit beste Lösung gespeichert. In den nachfolgenden Schritten werden nun nach und nach die nachfolgenden Ebenen des Baumes abgearbeitet. Dabei wird solange eine Kante ins Maximalgerüst aufgenommen, bis das Maximalgerüst voll ist. Danach wird der aktuelle Ast abgebrochen und eine Ebene zurückgesprungen. In dieser Ebene wird explizit festgelegt, dass die Kante nicht im Maximalgerüst enthalten sein darf. Die nachfolgenden Kanten werden dann erst wieder ins Maximalgerüst aufgenommen und später verboten. Immer wenn im Baum eine Entscheidung getroffen wurde, wird für das erzeugte Maximalgerüst eine Ausgangslösung berechnet. Die fehlenden Kanten werden dabei nach der ursprünglichen Methode der Ausgangslösung gewählt. Nach einer Berechnung wird der Wartezeitaufwand der berechnete Ausgangslösung mit der derzeit besten verglichen. Wenn die neue Lösung besser ist (kleinerer Wartezeitaufwand), wird die aktuelle Ausgangslösung zur besten. Nachdem der Baum abgearbeitet wurde, ist die beste Lösung für den gegebenen Transportkettengraphen bekannt.

7.7.1.15 Ermittlung des Fahrplans

Nachdem nun alle benötigten Werte berechnet wurden, kann der Fahrplan abgelesen werden. Dazu ist es notwendig den Fahrplan jeder einzelnen Zusammenhangskomponente zu ermitteln und diese dann zu einem Gesamtfahrplan zusammenzufügen.

Zur Ermittelung eines Fahrplans einer Zusammenhangskomponente ist es notwendig, dass neben den schon gegeben Werten für die Wartezeit, die Fahrzeit und den Takt noch ein Wert für das Fahrplanintervall (der Zeitraum für den der Fahrplan berechnet werden soll) und einen Wert für eine Abfahrtszeit pro Zusammenhangskomponente angegeben werden. Die Abfahrtszeit bezieht sich dabei auf eine bestimmte Haltestellen-Linien-Kombination und wird demnach im Zusammenhangskomponentengraphen an einen bestimmten Knoten geschrieben. Dieser Knoten soll im folgenden Anfangsknoten heißen. Alle Abfahrtszeiten die in den Knoten der Zusammenhangskomponente stehen, sind in Minuten angegeben. Dies reicht aus, da der Fahrplan sich in Taktabständen wiederholt.

Um nun die Abfahrtszeiten an den restlichen Knoten zu setzen, wird ausgehend von dem Anfangsknoten eine Breitensuche durchgeführt. Wird ein Knoten gefunden, an dem noch keine Abfahrtszeit gesetzt wurde, so muss, da es sich um einen gerichteten Graphen handelt, unterschieden werden, ob es sich um einen Vorgänger- oder Nachfolgerknoten handelt. Handelt es sich um einen Vorgängerknoten, so berechnet sich die Abfahrtszeit des aktuellen Knotens aus der folgenden Formel:

$$Abfahrtszeit_{Ausgangsknoten} - Fahrzeit_{AktuellerKnoten} - Wartezeit_{Kante}$$

Die Wartezeit an der Kante zwischen den beiden Knoten besteht dabei aus der Mindestwartezeit, der Mindestumsteigezeit und der netzbedingten Wartezeit. Handelt es sich bei dem gefundenen Knoten dagegen um einen Nachfolgerknoten, so berechnet sich die Abfahrtszeit durch die folgende Formel:

$$Abfahrtszeit_{Ausgangsknoten} - Fahrzeit_{Ausgangsknoten} - Wartezeit_{Kante}$$

Wenn nun alle Knoten eine Abfahrtszeit haben, kann es je nach Reihenfolge der Breitensuche zu Zeitverschiebungen innerhalb einer Linie kommen. So kann es zum Beispiel vorkommen, dass bei zwei aufeinanderfolgenden Knoten innerhalb einer Linie der zweite Knoten eine frühere Abfahrtszeit hat als der erste Knoten. Deshalb ist es notwendig jede Linie nocheinmal durchzurechnen. Dabei wird bei der Starthaltestelle der Linie begonnen und dort als erstes die Abfahrtzeit angepasst. Und zwar sollte die Abfahrtszeit an der ersten Haltestelle im Bereich $[0, takt[$ liegen. Stimmt die Zeit am ersten Knoten, so wird der iterativ der Folgeknoten gewählt und dort die Zeit neu berechnet. Dies erfolgt durch die schon oben angegebene Formel:

$$Abfahrtszeit_{Ausgangsknoten} - Fahrzeit_{Ausgangsknoten} - Wartezeit_{Kante}$$

Wurde nun jede Linie berechnet, kann der eigentliche Fahrplan für die Zusammenhangskomponente berechnet werden. Dazu wird einfach über alle Knoten iteriert und bei jedem Knoten die Abfahrtszeiten in Stunden und Minuten in eine Liste gespeichert. Dafür wird die Anzahl der zu berechneden Abfahrten benötigt. Dieser Wert wird aus dem Fahrplanintervall und dem Takt ermittelt. Die einzelnen Abfahrtszeiten werden dann ersteinmal in Minuten berechnet durch den Beginn des Fahrplanintervalls + Takt * Anzahl schon berechneter Anfahrten + Abfahrtszeit des Knoten. Dieser Wert lässt sich dann einfach in Stunden und Minuten umrechnen.

Sind nun alle Fahrpläne der einzelnen Zusammenhangskomponenten berechnet, so werden die einzelnen Listen einfach aneinander gehängt.

7.7.1.16 Ermittlung der Fahrzeuganzahl

Um die berechnete Lösung entsprechend bewerten zu können bzw. dem Benutzer mitzuteilen, ob die von ihm vorgegebene Fahrzeuganzahl eingehalten werden kann, wird die benötigte Anzahl von Fahrzeugen für die bisher errechnete Lösung bestimmt. Zur Ermittlung der benötigten Fahrzeuganzahl wurde das Verfahren von [Groetschel96] implementiert. Dieses Verfahren modelliert das Problem der Fahrzeugumlaufplanung als ein Flussproblem. Zu diesem Zweck muss der zugrunde liegende Fahrplan offensichtlich in einen Graphen, im folgenden Umlaufgraph genannt, umgewandelt werden. Dabei wird wie folgt vorgegangen: Zu Beginn werden in den noch leeren Umlaufgraphen zwei Knoten eingefügt. Bei diesen Knoten handelt es sich um die Depotknoten. Sie repräsentieren das Busdepot der HCR. Der Vollständigkeit halber soll an dieser Stelle noch erwähnt werden, dass die Berücksichtigung mehrerer Depots im verwendeten Verfahren sehr wohl möglich ist. Das Depot der HCR wird in zwei Knoten aufgeteilt, einen Quelldepotknoten, sowie einen Senkendepotknoten. Im Anschluss daran gilt es, die Fahrgastfahrten in den Graphen einzufügen. Eine Fahrgastfahrt ist in diesem Zusammenhang die Fahrt eines Busses, beginnend an der Starthaltestelle einer Linie und endend an der Endhaltestelle derselben Linie. Diese Fahrgastfahrten lassen sich dem Fahrplan entnehmen. Fahrgastfahrt werden wie folgt in den Umlaufgraph eingefügt: Pro Fahrgastfahrt werden zwei Knoten in den Umlaufgraph eingefügt. Einer dieser Knoten repräsentiert die Starthaltestelle der Fahrgastfahrt, der zweite Knoten die Endhaltestelle. Durch entsprechende Werte an den Knoten, zur Darstellung des Umlaufgraphen fand ein parametrisierten LEDA-Graph Verwendung, kann stets unterschieden werden, um was für einen Knoten es sich handelt (also: Start-, End- oder einer der beiden Depotknoten). Es wird pro Fahrgastfahrt eine Kante zwischen dem Startknoten und dem Endknoten der Fahrgastfahrt, dem Startknoten und dem Quelldepotknoten, sowie zwischen Endknoten und Senkendepotknoten eingefügt.
Wurde dies für alle Fahrgastfahrten gemacht, so werden Endhaltestellen von Fahrgastfahrten mit Starthaltestellen anderer Fahrgastfahrten verbunden, sofern dies zeitlich möglich ist. Dieses Vorgehen bedeutet in der Realität, dass ein Bus nach Beendigung seiner aktuellen Fahrgastfahrt eine neue Fahrtgastfahrt beginnt, ohne vorher wieder in das Depot zu fahren. Theoretisch ist es möglich, Fahrgastfahrten beliebiger Linien miteinander zu koppeln. Im fertigen Produkt werden jedoch lediglich Fahrgastfahrten einer Linie miteinander gekoppelt. Nähere Ausführungen für den Grund dieser Beschränkung finden sich im Kapitel Erfahrungen des Moduls Fahrplan. Zuletzt wird eine Kante zwischen dem Senkendepotknoten, sowie dem Quelldepotknoten eingefügt. Diese ist nötig, um die Kirchhoff'sche Knotenregel im Senkendepotknoten überhaupt erfüllen zu können.

Nachdem alle Kanten eingefügt wurden, gilt es durch Kapazitätswerte, sowie Kosten an den Kanten den Umlaufgraphen zu komplettieren. Dabei werden die

Kapazitätsschranken wie folgt gesetzt:

1. Kanten zwischen Quelldepotknoten und Startknoten einer Fahrgastfahrt erhalten als untere Kapazitätsschranke den Wert 0, die obere Kapazitätsschranke erhält den Wert 1. Dies bedeutet, dass auf dieser Kante ein Fluss von 1 oder 0 zulässig ist, also in der Realität ein Bus diese Fahrgastfahrt vom Depot aus bedienen kann, aber nicht muss. Kosten dieser Kanten sind 0.

2. Kanten zwischen Start-, sowie Endknoten einer Fahrgastfahrt erhalten als untere und obere Kapazitätsschranke den Wert 1. Eine Fahrgastfahrt muss also von einem Bus bedient werden. Kosten dieser Kanten sind 0.

3. Kanten zwischen Endknoten einer Fahrgastfahrt und Startknoten einer anderen Fahrgastfahrt erhalten als untere Kapazitätsschranke den Wert 0, als obere Kapazitätsschranke den Wert 1. Kosten dieser Kanten sind 0.

4. Kanten zwischen Endknoten einer Fahrgastfahrt und Senkendepotknoten erhalten als untere Kapazitätsschranke den Wert 0, als obere Kapazitätsschranke den Wert 1. Kosten dieser Kanten sind 1.

5. Die Kante zwischen Senkendepotknoten und Quelldepotknoten erhält die untere Kapazitätsschranke den Wert 0, als obere unbegrenzt. Kosten dieser Kanten ist 0.

Bevor erläutert wird, wie mit dem nun fertigen Umlaufgraph weiter verfahren wird, soll noch kurz erläutert werden, wann die Verknüpfung zweier Fahrgastfahrten mittels einer Kante vom Typ 3 möglich sind. Werden zwei Fahrgastfahrten miteinander verbunden, so bedeutet dies, dass ein Bus, nachdem er eine Fahrt von Start- zu Endhaltestelle einer Linie beendet hat, eine neue Fahrt beginnt. Im Kontext der Implementation kann er also zurück zur Starthaltestelle dieser Linie fahren, oder aber die Linie in Gegenrichtung bedienen. Es ist bekannt, wann ein Bus an der Endhaltestelle einer Fahrgastfahrt ankommt. Es gilt zwei Fälle zu unterscheiden:

1. Kante zwischen Endknoten einer Fahrgastfahrt und Startknoten einer anderen Fahrgastfahrt in gleicher Richtung.

2. Kante zwischen Endknoten einer Fahrgastfahrt und Startknoten einer anderen Fahrgastfahrt in Gegenrichtung.

Im ersten Fall ist noch die Zeit, die gebraucht wird, um von der Endhaltestelle zur Starthaltestelle zurückzukehren, zu berücksichtigen. Diese wird auf den Zeitpunkt der Ankunft des Busses an der Endhaltestelle addiert. Nun ist bekannt, ab welchem Zeitpunkt der Bus die nächste Fahrgastfahrt dieser Linie in gleicher Richtung beginnen kann. Mittels Vergleichsoperationen im Umlaufgraph werden diese Kanten eingefügt.

Der zweite Fall verläuft analog zum ersten mit dem Unterschied, dass hier keine zusätzliche Zeit auf den Ankunftszeitpunkt am Endknoten addiert werden muss (dieser Wert ist auf Null initialisiert, in der Realität ist zum Wenden der Busse Zeit hinzuzurechnen).

Es sind nun alle Kanten eingefügt und die Kapazitätsschranken, sowie Kosten an diesen gesetzt. Der fertige Umlaufgraphen wird als Eingabe für einen, in LEDA implementierten Min-Cost-Flow Algorithmus benutzt. Kosten entstehen ja lediglich auf Kanten vom Typ 4. Dies entspricht allen Bussen welche wieder im Depot ankommen. [10] Die minimale Kosten entsprechen dann also der Anzahl der mindestens benötigten Fahrzeuge für diesen Fahrplan.

7.7.1.17 Ermittlung der Gesamtwartezeiten

Für die Beurteilung des Fahrplans und zum Vergleich weiterer berechneter Fahrpläne werden die folgenden vier Kennwerte ermittelt:

1. Summe der netzbedingten Wartezeiten
2. Wartezeit-Aufwand
3. Mindestwartezeit-Aufwand
4. Gesamtwartezeit-Aufwand

Die Summe der netzbedingten Wartezeiten ergibt sich, wie der Name schon besagt, aus der Summe der netzbedingten Wartezeiten in allen Zusammenhangskomponenten und die wiederum aus der Summe der netzbedingten Wartezeiten an ihren Kanten.

Der Wartezeit-Aufwand beinhaltet die Summe der Produkte aus dem Verkehrsstrom und der jeweiligen netzbedingten Wartezeit über alle Kanten und Zusammenhangskomponenten.

Der Mindestwartezeit-Aufwand wiederum beinhaltet die Summe der Produkte aus dem Verkehrsstrom und der Summe der Mindesthaltezeiten und Mindestumsteigezeiten hinsichtlich aller Kanten und Zusammenhangskomponenten.

Der gesamte Warteaufwand kann sich als Summe aus Mindestwartezeit-Aufwand und Wartezeit-Aufwand.

Diese Werte können dadurch ermittelt werden, indem in jeder Zusammenhangskomponente über alle Kanten iteriert wird und die jeweiligen Werte aus der Kantenbeschriftung gelesen und multipliziert bzw. aufsummiert werden.

7.7.2 Ablauf

1. Datenaufbereitung und Erzeugung Transportkettengraph:

   Das Modul Fahrplan erhält vom Programm das Netz der Transportverbindungen, für welches ein gültiger Fahrplan berechnet werden soll. Dieses wird in einen Transportkettengraphen umgebaut. Zu diesem Zeitpunkt finden alle Berechnungen innerhalb des Moduls Fahrplan ohne irgendwelche Benutzereingaben statt, der Benutzer hat folglich keine Möglichkeit, in den Umbau des Netzes der Transportverbindung in den Transportkettengraphen einzugreifen.

   Ist der Umbau abgeschlossen, so werden innerhalb der graphischen Oberfläche des Programmes Menüpunkte anwählbar. Der Benutzer hat nun die Möglichkeit, die Fahrplanberechnung zu steuern. Die Eingriffsmöglichkeiten des Benutzers in die Fahrplanberechnung werden in den Punkten 2

[10] Man hätte natürlich auch Kanten vom Typ 1, anstelle von Kanten vom Typ 4, mit Kosten 1 versehen können, dies würde dann allen Bussen entsprechen, welche das Depot verlassen.

bis 4 dargestellt. Bis zu dem Zeitpunkt zu dem der Benutzer einen Fahrplan berechnen lässt (hier Punkt 5), finden keine Berechnung innerhalb des Moduls Fahrplan statt.

2. Interaktive Festlegung der Parameter Mindesthaltezeiten:

   Mittels des Transportkettengraphen sind die zeitlichen Abhängigkeiten des Netzes der Transportverbindungen erfasst. Eventuell fehlen jedoch noch die Mindestwartezeiten. Diese sind im Netz der Transportverbindungen nicht enthalten. Unter Mindestwartezeiten ist Folgendes zu verstehen: Es kann vorkommen, dass ein Bus aus irgendwelchen Gründen an einer Haltestelle stets fünf Minuten warten soll. Diesen Umstand berücksichtigt man mittels einer Mindestwartezeit an dieser Bushaltestelle. In der GUI ist es dem Benutzer jetzt möglich, diese für alle Haltestellen einzutragen, die GUI reicht diese Werte an das Modul Fahrplan weiter.

3. Interaktive Festlegung der Parameter Mindestumsteigezeiten:

   Mittels des Parameters Mindestumsteigezeit kann der Benutzer entscheiden, wieviel Zeit seitens eines Reisenden an einer bestimmten Haltestelle benötigt wird, um von Linie A nach Linie B zu wechseln. Diesen

4. Veränderung der Festlegungen von Bindungen:

   Der Benutzer hat die Möglichkeit, zu verhindern, dass einzelne Verbindungen netzbedingte Wartezeiten erhalten. Um dies zu erreichen, müssen diese Verbindungen im Maximalgerüst enthalten sein. Der Benutzer nimmt hier seine Auswahl vor. Die Auswahl wird dann vom Modul Fahrplan auf ihre Gültigkeit hin überprüft. Dies geschieht allerdings erst, nachdem die Berechnung eines Fahrplans angestoßen worden ist.

5. Erzeugung der Ausgangslösung:

   Bei einer Ausgangslösung handelt es sich um einen gültigen Fahrplan. Es wird ein Gerüst des erzeugten Transportkettengraphen berechnet. Der Begriff Ausgangslösung soll hier darauf hindeuten, dass die in diesem Schritt von berechnete Lösung nur eine von vielen möglichen, und nicht zwangsläufig die optimale Lösung ist.

6. Erzeugung des Simplextableaus:

   Wurde ein Maximalgerüst bestimmt, kann nun das Simplextableau jeder Zusammenhangskomponente berechnet werden. Dazu wird das Tableau entsprechend dem Maximalgerüst aus Gerüstkanten (in den Spalte) und Nicht-Gerüstkanten (Zeilen) ausgebaut. Die Einträge innerhalb der Matrix werden, wie in den einleitenden Kapiteln geschildert, gesetzt und somit stehen am Ende dieser Phase die netzbedingten Wartezeiten zur Verfügung, so dass folglich die zeitlichen Abhängigkeiten im Fahrplan bestimmt worden sind.

7. Interaktive Fahrplanverbesserung:

   Nachdem eine gültige Ausgangslösung erzeugt wurde, steht es dem Benutzer frei, diese mittels zweier Verfahren zu verbessern. Eine Lösung ist in diesem Zusammenhang dann besser als eine vorher berechnete Lösung, wenn die Summe der Wartezeitaufwände kleiner ist, als die Summe der Wartezeitaufwände der vorher berechneten Lösung. Um eine solche Verbesserung zu erreichen, werden Gerüst- und Nichtgerüstkanten vertauscht. Zwei Verfahren sind implementiert worden. Zum einen die sukzessive Lösungsverbesserung, welche sich darauf beschränkt, ein lokales Optimum innerhalb einer definierten Umgebung zu finden, zum anderen die Entscheidungsbaummethode, welche alle möglichen Gerüste berechnet. Hierbei ist es möglich, bestimmte Abbruchkriterien festzulegen. Aus Gründen des Rechenaufwands sollte die sukzessiven Lösungsverbesserung jedoch bevorzugt werden.

8. Ermittlung des Fahrplans:

   Da im vorherigen Schritt das Simplextableau und damit die netzbedingten Wartezeiten berechnet worden sind, kann nun der Fahrplan ausgelesen werden. Dazu kann seitens des Benutzers eine Abfahrtszeit pro Zusammenhangskomponente (Freiheitsgrad) festgelegt werden, d.h. die Abfahrtszeit einer Linie an einer bestimmten Haltstelle kann so festgelegt werden. Die folgenden Abfahrtszeiten ergeben sich sukzessive aus den Fahrzeiten (an den Knoten des Transportkettengraphen) und Wartezeiten (an den Kanten des Transportkettengraphen).

9. Ermittlung der Fahrzeuganzahl:

   Der Benutzer kann sich durch Anwahl des entsprechenden Menüpunktes die für den Fahrplan benötigte Fahrzeuganzahl anzeigen lassen. Zu diesem Zweck wurde das Verfahren von Grötschel [Groetschel96].

10. Ermittlung der Gesamtwartezeiten:

    Weiterhin können zur Beurteilung der Güte eines Fahrplans seitens des Benutzers und auch hinsichtlich späterer möglicher Lösungsverbesserungen die verschiedenen Wartezeiten im Fahrplan angezeigt werden.

7.7.3 Zeitkomplexität

Es folgt eine grobe Zeitabschätzung der Gesamtlaufzeit des Moduls Fahrplan.Die Berechnung der einzelnen Zusammenhangskomponenten ist hinsichtlich der O-Notation in der Summe aus Anzahl Knoten und Kanten des Transportkettengraphen beschränkt.

Das maximale Gerüst lässt sich hinsichtlich der O-Notation wie folgt abschätzen: Anzahl Kanten der Zusammenhangskomponenten multipliziert mit dem Logarithmus der Kantenanzahl (Sortieren der Kanten entsprechend ihrer Kosten).

Die Laufzeit bezüglich des Simplextableaus lässt sich nur schwer abschätzen, daher soll auf eine Angabe an dieser Stelle verzichtet werden. Die Gesamtlaufzeit des Moduls Fahrplan lässt sich wie folgt abschätzen: Sei n gleich der Anzahl der Knoten und e gleich der Anzahl der Kanten im Transportkettengraph. Dann ergibt sich eine Gesamtkomplexität von:

$$O(e \log e + ne)$$

Schätzt man das ganze bezüglich der Eingabe ab und sei e gleich der Anzahl Kanten im Netz der Transportverbindungen, so erhält man:

$$O(e^2 \log e + e^3) \in O(e^3)$$

Kapitel 8

Ergebnisse

Die in dieser Projektgruppe umgesetze Realisierung, der in Kapitel 5.5.1 beschriebenen Problemstellung kann nur bewertet werden, indem Ergebnisse gesammelt und zu einem späterem Zeitpunkt evaluiert werden. In den folgenden Kapiteln 8.1 bis 8.7 soll diese Bewertung vorgenommen werden. Anhand der durchgeführten Testläufe soll dort das Testen und die Sammlung der Ergebnisse beschrieben werden. Zunächst wird in Kapitel 8.1 die allgemeine Vorgehensweise beim Testen vorgestellt, um dann die Ergebnisse und Einflüsse der Parameter anhand von Testläufen des gesamten Programmes zu veranschaulichen. Das zentrale Ergebnis wurde während der Realisierung deutlich: Die angestrebte Rückkopplung konnte nicht verwirklicht werden.

8.1 Allgemeine Vorgehensweise

Aufgrund der großen Datenmengen der kompletten Eingangsdaten für das gesamte Programm, sind Testläufe mit Eingaben dieser Größenordnung nicht möglich. Durch sie würde die Fehlersuche erschwert und die Laufzeit für kleine Tests unnötig ausgedehnt. Aus diesem Grund wurden in den frühen Versionen des Programmes spezielle Testfälle (siehe Anhang E.4) konstruiert. Damit ein Test für alle Module gewährleistet blieb, mussten diese Daten alle Typen der Eingangsdaten abdecken. Dies bedeutete insbesondere, dass neben einem Straßenplan für das Haltestellen-Modul auch die Bevölkerungsdaten zur Verfügung gestellt werden mussten. Um das Testen der einzelnen Module unabhängig von den Ergebnissen und damit dem korrekten Durchlaufen der in den Arbeitsschritten vorgelagerten Programmteile zu ermöglichen, mussten auch für die Module Linienplan und Fahrplan die entsprechenden Eingabedaten erzeugt werden. Aus diesem Grund wurden diese Daten fest implementiert und die einzelnen Testfälle im Testdatendokument festgehalten (siehe Anhang E.7).

Der nächste Schritt bestand in der Konstruktion größerer Testfälle, die eine erhöhte Datenmenge beinhalteten. Hiermit sollte eine erste Überprüfung der berechneten Ergebnisse ermöglicht werden. Um die Größe der Daten variabel selektieren zu können, wurde eine entsprechende Auswahlfunktion in das Programm integriert. Mit Hilfe dieser Funktion wurde es möglich, einen Bereich von Straßen - und damit die Größe der Eingangsdaten - zu bestimmen, für den ein Berechnungsvorgang gestartet werden sollte.

Sobald eine endgültige Programmversion vorlag, wurden durch den Tester Testszenarien festgelegt. Als Berechnungsgrundlage diente das gesamte Stadtgebiet. Dabei wurden die Parameter der einzelnen Module variiert, um so die Beeinflussung durch diese Werte herauszufinden. Mittels der Haltestellen-Parameter wurde die Haltestellenanzahl und -lage unter den hier einsetzbaren Algorithmen überprüft. Im Linienplan-Modul wurde die Differenzen zwischen dem ersten und zweiten Algorithmus und der Einfluss der maximalen Linienlänge kontrolliert. Schließlich wurde beim Fahrplan besonderer Wert auf die Gütemaße und die Konsistenz des Fahrplans gelegt. Das allgemeine Vorgehen bestand darin, dass jeder Teilnehmer der Projektgruppe zur Berechnung mindestens ein Testszenario zugewiesen bekam. Die Ergebnisse der Durchläufe wurden schließlich durch den Tester gesichtet und dokumentiert.

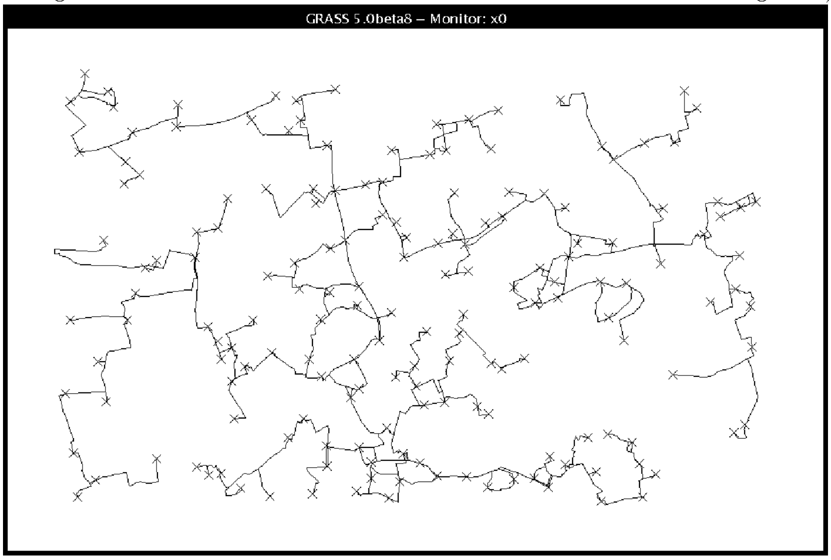
8.2 Beschreibung eines Testlaufes

Im Folgenden wird exemplarisch für alle Testläufe ein kompletter Durchlauf auf dem gesamten Stadtgebiet aufgezeigt. Dieser Lauf entspricht dem Testszenario 10 (siehe Anhang E.7). Zu Anfang legt der Tester die Parameter des Testszenarios fest. Generell wird ein Testziel festgelegt, das den Zweck des Durchlaufes definieren soll. Im Testszenario 10 liegt das Ziel darin, einen kompletten Durchlauf auf dem Stadtgebiet von Herne durchzuführen und somit adäquate Daten für die Präsentation des Programmes zu erhalten. Aufgrund der fehlenden Versionierung ist die verwendete Programmversion datumsabhängig aufgeführt. Im hier betrachteten Testszenario handelt es sich um die Programmversion vom 11. Juli 2001, 18:00 Uhr. Die Haltestellen-Parameter sind so gewählt, dass voraussichtlich ca. 180 Haltestellen gesetzt werden. Dies kann durch die Wahl der Parameter *minimales* und *maximales Einzugsgebiet*, sowie dem *Verbindungsradius* beeinflusst werden. Als Algorithmus kommt der stochastische Algorithmus zum Einsatz. Die *Haltestellenlage* wird auf den Wert 1,0 eingestellt, um so eine kreuzungsnahe Lage der Abfahrpunkte zu bekommen. Die Haltestellen sollen untereinander auf den Realwegen, d.h. nicht durch Luftlinienverbindungen, sondern über real existierenden Straßen, verbunden werden. Nach Berechnung der Haltestellen wird der Wegenetzgraph visualisiert (Abbildung 8.1) und daraufhin überprüft, dass der Graph zusammenhängend ist. Im Testszenario 10 besteht er aus 171 Haltestellen.

Bei den Einstellungen des Linienplan-Moduls besteht das Ziel darin, möglichst wenig Linien zu erzeugen. Dazu wird der Algorithmus 2 mit einer *maximalen Gesamtlinienlänge* von 120.000 Metern und einer *maximalen Linienlänge* von 30.000 Metern voreingestellt. Desweiteren werden alle Linien-Verknüpfungsmöglichkeiten aktiviert. Das Ergebnis der Linienplan-Berechnungen liefert 68 Linien. Nach Kontrolle der einzelnen Linien wird ersichtlich, dass ein Großteil nur über drei oder vier Haltestellen führt.

Der Fahrplan wird mit einem *Mindesttakt* von 15 Minuten und einer *Mindestumsteigezeit* von 2 Minuten gestartet. Da ein Nachvollziehen des kompletten Fahrplans ein extremer Aufwand wäre, wird der Fahrplan lediglich stichprobenartig verifiziert. Das Augenmerk liegt hier verstärkt auf den Gütemaßen. Es müssen 233 Busse bei den zugrunde liegenden Linien eingesetzt werden. Die netzbedingte Wartezeit ist 5214 Minuten und der Fahrplanwirkungsgrad liegt bei 77,71%.

Abbildung 8.1: Der Wegenetzgraph des Testszenarios 10
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet)



Um einen Eindruck über das Gesamtergebnis zu erhalten, wird eine Kontrolle der globalen Gütemaße vorgenommen (Kapitel 6.4). Die *mittlere Reisezeit* liegt bei 66 Minuten. Die *durchschnittliche Anzahl der Personen pro Bus* beträgt 2,67 und die *durchschnittliche der Busse pro Linien* ist 3,51.

Abschließend kann dieser Testfall als ein durchschnittlicher Testlauf des Programmes angesehen werden. Die Anzahl der Haltestellen ist auf das gesamte Stadtgebiet in keinem Fall zu hoch. Dagegen spiegelt die hier berechnete Linienanzahl den Trend des Programms, viele kurze Linien zu erzeugen, wieder. Hierdurch bedingt ist die hohe Anzahl der Busse. Der Fahrplanwirkungsgrad liegt mit 77,71% im allgemeinen Durchschnitt der Berechnungen des Fahrplan-Moduls.

8.3 Vergleich verschiedener Testszenarien

Während der Programmläufe der einzelnen Testszenarien wurden die Auswirkungen der einzelnen Parameter auf die Berechnungen der einzelnen Module deutlich. Die Wahl der Algorithmen zur Haltestellenberechnung hat deutlichen Einfluss auf die Anzahl der Haltestellen. So liefert der stochastische Algorithmus mehr Abfahrpunkte, als der stochastische Algorithmus, der zusätzlich die Umgebung prüft. Dieser wiederum setzt mehr Haltestellen als das evolutionäre Verfahren. Diese Aussage hängt im erheblichen Maße von den Werten für das minimale und maximale Einzugsgebiet einer Haltestelle ab. So beträgt die Anzahl der Haltestellen bei der Berechnung mit dem stochastischen Algorithmus bei minimalem Einzugsgebiet von 250 Metern und einem maximalen Einzugsgebiet von 650 Metern im Testszenario 10 (siehe Anhang E.7) 171 Haltestellen. Vergrößert man diese Werte nun auf 300 bzw. 800 Meter und behält den Algorithmus bei, so verringert sich die Anzahl der Haltestellen auf 88 für das gesamte Stadtgebiet (Testszenario 19). Die Anzahl der Abfahrpunkte reagiert demzufolge sehr sensibel auf die Werte dieser Parameter.

Die Ergebnisse des Linienplan-Moduls hängen sehr stark von der Wahl des Algorithmus ab. So erzeugt der erste Algorithmus mehr Linien, als der Zweite. Der zweite entscheidende Parameter des Linienplans ist die maximale Linienlänge, die erheblichen Einfluss auf die Buslinienlänge ausübt. Je größer dieser Wert gewählt wird, desto länger werden die berechneten Linien, wodurch auch die Gesamtanzahl der Linien reduziert wird.

Die Berechnung des Fahrplans ist deterministisch. Aus diesem Grund kann das Ergebnis dieses Moduls reproduziert werden. Durch die starke Interaktion mit dem Benutzer und der fehlenden Garantie, einen stabilen Zustands in einer Berechnung zu gewährleisten, werden die Ergebnisse des Fahrplans nicht in der Datenbank abgelegt. So muss die Fahrplanberechnung nach Einlesen eines Projektes und Beenden des Programmes stets neu gestartet werden. Die Berechnung des Fahrplans mittels des heuristischen Lösungsverfahrens liefert im Hinblick auf die Gütemaße ein akzeptables Ergebnis (siehe Anhang E.7). Das Verfahren der sukzessiven Lösungsverbesserung erreicht zudem eine weitere Verbesserung des Ergebnisses. Dies spiegelt sich vor allem in der geringeren netzbedingten Wartezeit wider.

8.4 Rückkopplung

Die Rückkopplung wurde Aufgrund konzeptioneller Probleme nicht realisiert. Zusätzlich zum Widerspruch bezüglich des Verknüpfungsgrades (s.4.3.4) ist die gezielte Veränderung der Linien zu Gunsten des Fahrplans schwierig. Problematische Stellen im Fahrplan können leicht durch die netzbedingte Wartezeit einer Verbindung identifiziert werden. Durch die gegenseitigen Abhängigkeiten im Transportkettengraph kann der Ursprung dieser Wartezeit jedoch an einer anderen Stelle liegen. Wird demnach die identifizierte Stelle verändert, bleiben die Folgen nicht abschätzbar. Die netzbedingten Wartezeiten könnte immer noch vorhanden sein oder an einer anderen Stelle erscheinen. Als Konsequenz müssten demnach die Ursprünge der netzbedingten Wartezeiten ermittelt werden. Diese lassen sich aufgrund von Erfahrungswerten und den genannten Problemen nicht ermitteln. Somit ergibt sich kein Ansatzpunkt für eine gezielte Verbesserung.

Die in der Algorithmenwahl (s. 4.3.4) angedachte Vorgehensweise, die identifizierten Stellen zu verbieten und den Linienplan neu zu berechnen, erweist sich demnach als ungeeignet. Die Rechenzeit wäre enorm hoch, da sukzessive in jedem Durchlauf die problematischen Verbindungen identifiziert und verboten werden müssten. Ohne Absehbarkeit auf Verbesserung ist diese Rechenzeit nicht zu rechtfertigen.

Die erste Idee, die Erfahrung des Anwenders zu nutzen, ist somit der einzige, mögliche Ansatz. Dieser konnte jedoch aus Zeitmangel nicht mehr realisiert werden. Nachdem die Lauffähigkeit auf dem gesamten Herner Gebiet gewährleistet werden konnte, hat jede Gruppe neue Funktionalitäten in ihre Module eingebaut, die aber wiederum zu einer Einschränkung der Lauffähigkeit führten. Die Beseitigung der Fehler nahm soviel Zeit in Anspruch, dass keine ausführlichen Erfahrungen gesammelt werden konnten.

8.5 Erzielte Resultate des Haltestellen-Moduls

Es folgen nun kurze Auswertungen und Bemerkungen zu den implementierten Lösungsstrategien.

Stochastische Gleichverteilung

Bei diesem Verfahren ist offensichtlich, dass keine Optimierung vorgenommen wird. Durch Verwendung einer gleichverteilten Stochastik sind die resultierenden Haltestellen oft ungünstig und auch sehr ungleichmäßig verteilt. Entsprechend ist die resultierende Güte im Allgemeinen schlecht.

Stochastische Gleichverteilung mit Prüfen der Umgebung

Auch dieses Verfahren optimiert nicht und versucht willkürlich in R Haltestellen zu erzeugen. Durch die Überprüfung der Umgebung jeder potentiellen Haltestelle, wird jedoch im Allgemeinen eine recht gute Flächendeckung erzielt. Die resultierende Güte unter Einbeziehung der Bevölkerungsversorgung ist aber weiterhin schlecht, wenn auch nicht so, wie bei dem ersten Verfahren.

Evolutionäre Strategie

Bei diesem Verfahren hat sich in den Testläufen[1] gezeigt, dass nach circa der Hälfte des ursprünglichen Fitnesswertes eine drastische Einbuße der Optimierung auftrat. Zudem wurde dieser Wert in den meisten Fällen in relativ wenigen Generationen erreicht. Beispiele zeigen Abbildung 8.2, mit einer (10,100)-Strategie, und Abbildung 8.3, mit einer (10+20)-Strategie. Beide wurden als Vertreter aus der Menge der Testläufe herausgewählt, da sie das typische Verhalten der ES aufweisen.

Offensichtlich hat die ES Potential, um das Problem zu lösen, aber mit den bis zum Abschluss dieses Berichtes ermittelten Einstellungen, konnte das Optimum nicht genügend angenähert werden.

Im weiteren Vorgehen empfiehlt sich eine genauere Analyse der Auswirkungen der Parameter (sowohl einzeln, als auch in Kombination). Sinnvollerweise sollten hier Verfahren aus der Statistik angewandt werden, um eine sinnvolle Parameterauswahl zu finden. Da hierzu jedoch etliche Testläufe nötig sind, ist dies mit den begrenzten Mitteln, wie sie innerhalb der Projektgruppe vorliegen, nicht möglich. Ebenso sollte die Sektorgruppierung auf einen größeren Wert und damit auf weniger Sektoren eingestellt werden, um weniger, am besten keine, Haltestellen bei der Anpassung an das Straßennetz, zu verlieren. Dies ist auch entscheidend für die Reduzierung von Plateaus, welche den Lösungsraum erschweren. Zudem sollte die Rasterung der Bevölkerungs- und Flächenversorgung deutlich erhöht werden, damit genauere Werte ermittelt werden. Der eingestellte Wert von 40 könnte ein weiterer Grund für den jetzigen Verlauf der Fitnesskurve sein.

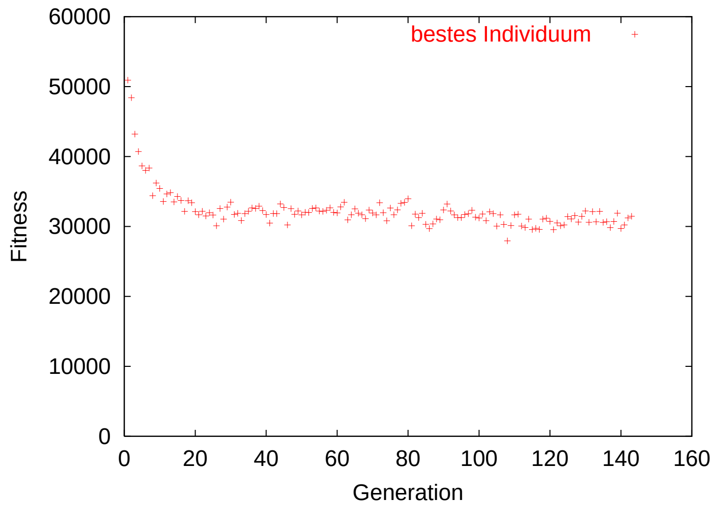
Im Gesamten können die ermittelten Ergebnis der ES nur Hinweise darauf geben, dass Optimierungspotential vorliegt, welches vermutlich durch eine ES

[1]Es wurden keine Constraints verwendet. Rekombination wurden, bei einem Elter, mit einem Individuum, ansonsten mit je zwei oder drei Individuen ausprobiert. Auch die Art der Rekombination (intermediat, diskret) wurde in verschiedenen Läufen variiert. Als Raster wurde für die Bevölkerungs- und Flächenversorgung ein Wert von 40 eingestellt.

genutzt werden kann. Es lassen sich aber zu dem jetzigen Zeitpunkt keine Aussagen darüber treffen, wie gut sich dem theoretischen Optimum von 0, oder aber dem realen Optimum > 0 genähert wird. Erst weitergehende Untersuchungen erlauben ernsthafte Aussagen.

Abbildung 8.2: (10,100)-Strategie

(10,100)-Strategie über 143 Generationen und Rekombination aus drei Individuen, wobei der Mittelwert gebildet wurde.



8.6 Erzielte Resultate des Linienplan-Moduls

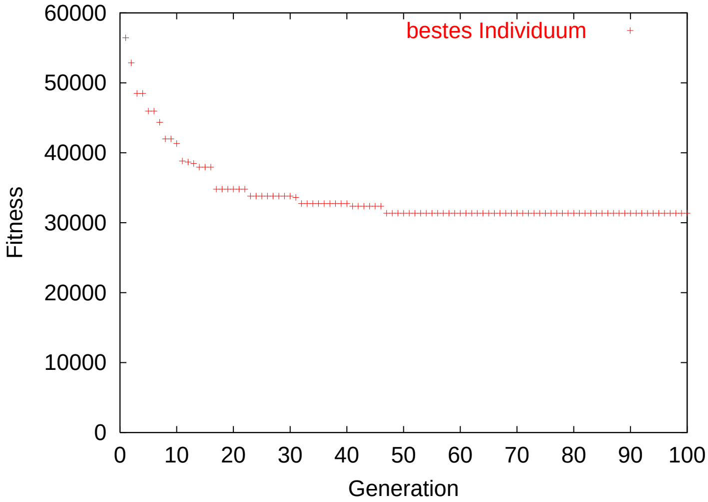
Für den Linienplan hat sich die Realisierung des Algorithmus von Sonntag ([Sonntag77]) als problematisch erwiesen. Zwar war das grobe Verständnis des Ablaufes noch leicht verständlich, allerdings stellten sich die Details als tückisch heraus.

Die Dissertation begann mit einer theoretischen Abhandlung und endete mit dem Quellcode des Algorithmus. In einigen Teilen wichen allerdings die beiden Teile voneinander ab, so dass unklar war, welches die richtige Version war.

Erschwerend kam hinzu, dass der Quellcode in einer uns unbekannten Programmiersprache abgedruckt war und die Variablen nicht-intuitive Buchstabenkürzel besaßen. Teilweise blieben noch am Ende Details verborgen, was dazu führte, dass der Algorithmus in Teilen nach eigenem Verständnis programmiert wurde.

Ein anderes großes Problem waren die erzeugten Linien. Nach ersten Tests wurde festgestellt, dass sie eine schlechte Basis für das Fahrplanmodul waren. Dies lag, wie sich später herausstellte, an zu vielen Verknüpfungen, die sich aber konzeptbegründet nicht verringern ließen (siehe Kapitel 8.4). Die Güte der Linien hing auch von einer realistischen Festlegung der Parameter Gesamtlinienlänge und maximale Linienlänge ab. Sowohl eine zu große als auch eine zu kleine Einstellung konnte das Gesamtbild des Linienplans extrem verschlechtern. Da allerdings Teile des Algorithmus sehr spät realisiert wurden und so die Ergebnisse auch mit gleichen Einstellungen sich ständig veränderten, konnte nicht

Abbildung 8.3: (10+20)-Strategie
(10+20)-Strategie über 100 Generationen und Rekombination aus drei Individuen, wobei der Mittelwert gebildet wurde.



genug Erfahrung gesammelt werden, um eine gute bzw. optimale Einstellung feststellen zu können.

8.7 Erzielte Resultate des Fahrplan-Moduls

Seitens des Fahrplan-Moduls konnte das von Weigand ([Weigand81]) vorgestellte Verfahren im Hinblick auf die Implementierung umgesetzt werden. Es wurden keine neuen Ideen bzw. Verfahren entwickelt, sondern die von Weigand vorgestellten Lösungsansätze wurden verfolgt und in das Programmpaket integriert.

Das Ergebnis dieses Ansatzes lässt sich nur in Grenzen bewerten. Der Grund hierfür ist derjenige, dass aufgrund des modularen Aufbaus das Fahrplan-Modul auf dem Modul des Linienplans aufsetzt. Die erzeugten Linienpläne haben jedoch einen sehr hohen Verknüpfungsgrad. Folge hiervon ist es, dass eine Vielzahl von Kreisen in dem Transportkettengraphen (siehe Kapitel 5.5.4) entstehen. Somit wird eine große Anzahl zeitlicher Abhängigkeiten aufgebaut und die Anzahl der möglichen Gerüste (bzw. die Anzahl der Lösungsmöglichkeiten) ist immens. Dementsprechend ist die Wahrscheinlichkeit als gering einzustufen, dass mit der einfachen Ausgangsheuristik (siehe Kapitel 7.7.1.6) ein optimaler Fahrplan hinsichtlich der Wartezeiten entsteht.

Bei generischen erzeugten Linienplänen, bei denen der Verknüpfungsgrad entsprechend gering ist, erweist sich das Verfahren als durchaus sinnvoll, da sich die Anzahl zeitlicher Abhängigkeiten in Grenzen erhält.

Problematisch erweist sich die Tatsache, dass die netzbedingten Wartezeiten so groß wie der Takt (bzw. Takt minus 1) sein können und auch teilweise werden. Dieses ist bei großen Taktzeiten hinreichend schlecht, so dass hierbei Verbesserungsmöglichkeiten anzustreben sind.

Als weiterer Kritikpunkt des Verfahrens muss die Beschränkung genannt werden, dass alle Linien den gleichen Takt fahren müssen. Dieses ist natürlich in der Realität nicht immer wünschenswert.

Die seitens der Bearbeiter des Moduls durchgeführten Lösungsverbesserungen können teilweise erhebliche Verbesserungen des Fahrplans erzielen. Das Entscheidungsbaumverfahren (siehe Kapitel 7.7.1.14), das eine optimale Lösung berechnet, ist jedoch praktisch nicht durchführbar (hinsichtlich der Zeitkomplexität), da die Anzahl der Lösungsmöglichkeiten zu hoch ist. Wird die Baumtiefe eingeschränkt, lassen sich bei größeren Problemen keine nennenswerten Verbesserungen erzielen. Bei kleineren Problemfällen, bei denen die Anzahl der Lösungen hinreichend beschränkt ist, ist eine Verbesserung anzutreffen, aber auch hier ist der Gewinn gering. Bei der sukzessiven Lösungsverbesserung (siehe Kapitel 7.7.1.13) fällt der Gewinn bei weitem höher aus. Selbst bei großen Problemfällen lassen sich in akzeptabler Zeit deutliche Fortschritte hinsichtlich der Lösung des Fahrplanproblems erzielen. Wird das Verfahren jedoch komplett durchlaufen, ist auch hier ein beträchtlicher Zeitfaktor in Kauf zu nehmen, so dass seitens des Benutzers das Verfahren abgebrochen werden kann. Abbruch bedeutet jedoch nicht Misserfolg bzw. kein Fortschritt, sondern die bis zu dem Zeitpunkt des Abbruchs erzielte Verbesserung wird registriert.

Bezüglich der Fahrzeugumlaufplanung wurde das Verfahren von Grötschel ([Groetschel96]) mit einem Busdepot implementiert. Das dort erzielte Resultat lässt sich positiv bewerten, da der berechnete Fahrplanwirkungsgrad (siehe Kapitel 6.3) durchschnittlich im Bereich von 75% liegt.

Hinsichtlich der Real-Gesamtlaufzeit der Implementierung sollte bemerkt werden, dass selbst bei großen Problemfällen die Laufzeit der einzelnen Teilpunkte des Moduls gering ist. Weiterhin ist der Speicherplatzbedarf moderat.

Insgesamt gesehen lassen sich die erzielten Resultate als durchweg positiv bewerten, jedoch sind Möglichkeiten der Verbesserung und Erweiterung des Verfahrens gegeben.

Kapitel 9

Erfahrungen

Eine der ersten Erfahrungen war die immer wiederkehrende Erkenntnis, dass auch eine noch so gründliche theoretische Analyse nicht vor unvorhergesehenen Problemen in der Praxis schützen kann. So hatte sich gezeigt, dass die Rückkopplung zwischen den einzelnen Modulen trotz der Analyse durch eine Kleingruppe letztlich nicht umzusetzen war, was auch unter anderem auf die komplexe Materie zurückzuführen ist, die Monate währende Einarbeitung gefordert hat, und so grundsätzliche Probleme erst sehr spät entdeckt wurden. Diese waren aufgrund des vorliegenden Materials nicht vorherzusehen und haben sich nur durch die Praxis offenbart. Als ein weiterer hemmender Faktor kam hinzu, dass die vorliegende Literatur zum Teil sehr alt war und nicht den heutigen Qualitätsanforderungen hinsichtlich der Analyse der beschriebenen Verfahren genügt, insbesondere was deren Grenzen angeht.

Als sehr sinnvoll hat sich die Einrichtung eines Testdatengenerators herausgestellt, der es bereits zu einem frühen Zeitpunkt ermöglicht hat, erste Tests durchzuführen, was die Implementierung besonders am Anfang beschleunigt hat. Besonders sinnvoll war, dass der Generator ohne größeren Aufwand um weitere Testfälle erweitert werden konnte.

Im Bezug auf die Seminarberichte kann gesagt werden, dass die dort vorgestellten Themen zielorientierter bzw. praktischer und nicht so theoretisch angelegt sein sollten. So sollte schon nach dem ersten Treffen dafür gesorgt werden, dass sich die Teilnehmer möglichst schnell hinsichtlich der Kenntnisse und Fähigkeiten annähern. Dies schließt sowohl die Kenntnisse über die Vor- und Herangehensweise bei einem Softwareprojekt, als auch die programmiertechnischen Fähigkeiten, die gefordert werden, mit ein. Es hat sich im Verlauf der PG gezeigt, dass dieses Kenntnis- und Erfahrungsniveau von jedem einzelnen erst mühsam erarbeitet werden musste, was gerade am Anfang einen beträchtlichen Teil der Ressourcen gebunden hat. Wenn man früher eine Umfrage bezüglich der Kenntnisse, Fähigkeiten und Interessen der Teilnehmer vorgenommen hätte, wären die Teilnehmer sicher effizienter auf die Module zu verteilen gewesen.

9.1 Struktur der PG

Als wichtigster Punkt im Bezug auf die Organisation ist der Aspekt der Aufgaben- und Rollenverteilung zu nennen. Der rollenbasierte Ansatz war im ersten Semester der Projektgruppe nicht ausgeprägt und nur oberflächlich vorhanden. So waren die Dokumente für den Zwischenbericht eine Gemeinschaftsproduktion, ohne dass jemand auf die Lesbarkeit, das Layout o. Ä. geachtet hätte. Hier bestand für den Endbericht die Notwendigkeit, einen Dokumentenmanager einzusetzen, der die Funktion eines Redakteurs übernehmen und so die Dokumente zusammenfügen und koordinieren würde. Ähnlich verhielt es sich mit dem Testen. Fehleraufdeckendes Testen wurde während des ersten Semesters nur oberflächlich durchgeführt und die Integrationstests [Gruhn98/99] jeweils nur von den Entwicklern selbst durchgeführt. Auch beim Schnittstellenentwurf haben die Gruppen ohne einen strikten vorgebenen Leitfaden gearbeitet, was zu jeweils eigenen Stilen und zu einem stark hybriden Datenfluss führte. Deshalb wurde beschlossen, in der zweiten Hälfte der PG stärker auf die konsequente Umsetzung der Rollenkonzeptes zu achten.

Zu Beginn der zweiten Phase der Projektgruppe (Beginn des Sommersemesters 2001) wurden somit notwendige Rollen bestimmt und die Plätze entsprechend besetzt. Die Situation entspannte sich merklich, da nun die Verantwortung zugeordnet und ein definierter Ansprechpartner vorhanden war. Dennoch traten weiterhin Probleme auf. So waren zwar nun die zu erstellenden Texte und Testberichte als Skelette rechtzeitig fertig, es zeigte sich jedoch, dass sich die übrigen Teilnehmer der Projektgruppe noch sehr viel Zeit ließen, diese mit Inhalt zu füllen. Eine Möglichkeit, dieses Problem zu lösen, wären sicherlich Sanktionen gewesen, welche vom entsprechenden Manager gegen einzelne Personen oder Teilgruppen verhängt werden. Diese Möglichkeit wurde allerdings nicht ernsthaft in Erwägung gezogen, vielmehr wurde stattdessen das Gespräch zu den Verantwortlichen gesucht.

Der Testmanager stand während der ganzen Zeit vor dem Problem, dass es zu keinem Zeitpunkt einen stabilen Code gab, der durch einen richtigen Codefreeze realisiert worden wäre. Die Ursache ist darin zu sehen, dass von den Betreuern und der Rückkopplungsgruppe immer wieder neue Features bzw. Bewertungsfunktionen gefordert wurden, die dann wieder neue Fehler und andere Seiteneffekte erzeugten und so das Testen und die Beseitigung existierender Fehler erschwerten. Durch eine striktere Versionierung, verbunden mit kurzfristigen Quellcode-Sicherungen oder markierten Releases, hätte das Testen wesentlich besser und organisierterter ablaufen können. Auch hätte die frühere Einführung und tatsächliche, aktive Nutzung des Fehler-Dokumentes wesentlich dazu beigetragen, dass auftretende Fehler koordiniert und organisiert identifiziert, klassifiziert und offiziell bekannt gemacht worden wären, ohne dass dafür ein informaler Weg hätte beschritten werden müssen.

Abschließend kann zum Thema Rollenvergabe gesagt werden, dass es empfehlenswert ist, Verantwortliche für wichtige Teilbereiche einzusetzen. Ob eine merkliche Verbesserung der Situation eintritt, ist dann jedoch von der Einsicht eines jeden Teilnehmers der Projektgruppe, sowie der Durchsetzungsfähigkeit des entsprechenden Managers abhängig.

Abschließend sollte noch erwähnt werden, dass sich die gesamte Gruppe nicht zu abhängig von einer einzelnen Person machen sollte, sondern dass jeder für einen bestimmten Zeitraum ersetzt werden kann bzw. die Gruppe trotzdem weiterarbeiten kann, wenn eine Person ausfällt. Dies sollte vor allem dadurch erreicht werden, dass die jeweilige Arbeit des Einzelnen so weit kommentiert, dokumentiert und abgesprochen bzw. erklärt worden ist, dass ein anderes Mitglied sie im Notfall auch übernehmen kann. Zudem sollte es tunlichst vermieden werden, dass eine einzelne Person als alleiniger Experte für einen wichtigen Teil des Programms gilt. Es kann schließlich nicht garantiert werden, dass diese Person für den gesamten Zeitraum des Projekts ständig zur Verfügung steht, so dass dann der eventuelle Ausfall (oder nur die Abwesenheit beim Auftreten eines Problems) solch einer Person dann die gesamte Gruppe aufhalten würde.

9.2 Zeitplanung

Die Einhaltung des Zeitrahmens ist in dieser Projektgruppe ziemlich gut gelungen. Es wurde ziemlich früh gemeinsam ein Zeitplan aufgestellt, bei dem auch versucht wurde, ihn strikt einzuhalten. Auch das Definieren von Meilensteinen bzw. fertigzustellenden Aufgabenpaketen hat viel dazu beigetragen, dass klar zu erkennen war, was noch bis wann zu erledigen war. Problematisch in diesem Zusammenhang war, dass sich während des zweiten Semesters die Anforderungen von Seiten der Betreuer, aber auch von der Rückkopplungsgruppe, geändert hatten, so dass neue Features gefordert wurden, die in der Form nicht eingeplant gewesen waren. Dadurch wurde die letzte Phase bzw. die letzten Wochen sehr arbeitsintensiv, weil gleichzeitig auch der Endbericht fertigzustellen war, was dazu führte, dass die eingeplante Pufferzeit in Anspruch genommen werden musste. Das zeigte deutlich den Sinn einer solchen Pufferzeit.

Als besonders positiv hat es sich erwiesen, auf eine strikte Einhaltung der Zeitrahmen zu achten, da so eine selbststabilisierende Gruppendynamik erzeugt wurde, welche letztlich verhindert hat, dass sich die Arbeit bis weit in die Semesterferien erstreckte.

9.3 Erfahrungen bei der praktischen Arbeit

Es trat zu Tage, besonders deutlich gegen Ende der Implementierung, dass die verwendete Datenbank Oracle für Halifa überdimensioniert ist. Oracle ist eine professionelle Datenbank, die für gigantische Projekte entworfen wurde, und eine dementsprechend gute Rechnerausstattung des Datenbankservers erfordert. Ein solcher stand nicht zur Verfügung, so dass ein Rechner des Pools zweckentfremdet werden musste, was eine niedrige Performanz zur Folge hatte. Dies hätte man vermeiden können, indem man eine alternative, schlankere Datenbank wie mySQL verwendet. Die Verwendung wäre problemlos möglich gewesen, da keine Oracle-Spezifika verwendet wurden.

Die Programmiersprache C++ wurde zwar im Bezug auf die Zeiger und der damit verbundenen Speicherverwaltung durch den Programmierer als negativ empfunden, jedoch wurde dies durch die Rechengeschwindigkeit aufgewogen. Es zeichnete sich ein deutlicher Lernerfolg unter den PG-Teilnehmern ab.

Damit die PG sich gänzlich auf die Implementierung der Algorithmen konzentrieren konnte, anstatt noch Standard-Datenstrukturen zu implementieren, entschied man sich für den Einsatz der LEDA-Bibliothek ([LEDA4.2]). Hier zeigten sich schnell die Vorteile bei der Anwendung einer Bibliothek (kein Design- und Implementierungsaufwand für Datenstrukturen), aber auch Nachteile (Nicht ausreichende Dokumentation, Verhalten einiger Algorithmen unterschiedlich zu dem dokumentierten Verhalten). Ganzheitlich betrachtet ist es jedoch sehr vorteilhaft gewesen, eine Bibliothek wie LEDA zu verwenden. Es sollte aber vor der Verwendung einer Bibliothek geprüft werden, wie gut die beigefügte Dokumentation und wie groß der Funktionsumfang ist. Fehlende Algorithmen müssen sonst umständlich an die bereitgestellten Datenstrukturen angepasst werden, was einen zusätzlichen Aufwand darstellt.

Das Geoinformationssystem GRASS erwies sich entgegen den Erwartungen aus der Analyse als benutzungsfreundlich. Im Laufe der Erweiterungen an den verarbeiteten Daten aber wurde schließlich klar, dass es effizienter ist, GRASS ausschließlich zur Anzeige zu verwenden und sämtliche Daten in Oracle zu speichern. Damit wurde GRASS letztlich redundant, da die Anzeige der verwendeten Graphen auch mit LEDA effizient möglich ist.

Die Verwendung von Tcl/Tk zur Implementierung der GUI erwies sich als vorteilhaft, insbesondere im Hinblick auf die Ansteuerung von GRASS. Der gemeinsame Shell-Charakter von GRASS und Tcl/Tk kommen dem zu Gute. Es traten jedoch unerwartete Schwächen in der Ereignisbehandlung in Verbindung mit Threads zu Tage, welche auf die Kombination von Tcl/Tk mit CDE zurückzuführen waren.

Bei der Erstellung der zum Projekt gehörenden Dokumente wurde sehr schnell deutlich, dass sich der Einsatz des sog. "what you see is what you mean" Editors LyXals eher arbeitshinderlich erwies. Neben den CVS-Problemen im Mehrbenutzerbetrieb wurde die Arbeit durch eine eigenwillige Oberfläche erschwert, die sich zwar an bekannte Programme anlehnte, aber verwirrenderweise anders funktionierte. Zudem kam es zu Inkompatibilitäten zwischen verschiedenen LyX-Versionen, eine Integration von LaTeXwar nicht ohne weiteres möglich und das Einbinden von Bildern zog häufig weitere Formatierungsprobleme nach sich. Durch einen Umstieg auf reine LaTeX-Dokumente hätten einige dieser Probleme vermieden werden können.

Kapitel 10

Fazit

An dieser Stelle erfolgt eine abschließende Gesamtbeurteilung des Projekts. So werden hier die Einsichten aus dem Kapitel 8, Seite 123, und dem Kapitel9, Seite 131, zusammengefasst und so ein Gesamtfazit gezogen. Desweiteren wird im zweiten Teil ein Ausblick darüber gegeben, wie insgesamt weiter vorgegangen werden könnte und welche Verbesserungen in den jeweiligen Teilmodulen möglich sind.

10.1 Erzielte Resultate

Als globales Fazit der Projektgruppe kann gesagt werden, dass innerhalb der geforderten Zeit von zwei Semestern ein lauffähiges Programm erzeugt worden ist, dass in vertretbarer Zeit das Problem bearbeitet und dabei eine Lösung generiert. Diese Lösung ist zwar bei weitem nicht realisierbar und sie ist auch weit entfernt davon optimal zu sein, aber sie kann als Anhaltspunkt gesehen werden, als Grundlage mit der ein Experte weiterarbeiten bzw. auf der jemand mit viel Erfahrung auf diesem Gebiet gut aufbauen kann. Die Software ist in der Lage solch einen Fachmann sowohl durch eine gute graphische Visualisierung der Daten, also auch durch eine gute Datenhaltung mittels einer Versionierung auf einer Datenbank zu unterstützen. Weiterhin gibt es Möglichkeiten die vorhandenen Daten zu manipulieren und anzupassen durch Pflichthaltestellen bzw. Pflichtlinien und durch die benutzerabhängigen Eingaben bei der Fahrplanerstellung.

Aus diesem Grund sollte die von der Projektgruppe erstellte Software als eine Art Framework angesehen werden, welches als Grundlage dienen kann, um durch den Austausch der Module entweder verschiedene Algorithmen für das jeweilige Teilprobleme oder ein Verfahren für das Gesamtproblem zu testen. Durch den modularen Aufbau und das Konzept der Kapselung durch Schnittstellen ist so eine gute Testumgebung geschaffen worden.

Es kann zusammenfassend gesagt werden, dass die gestellte Aufgabe zu komplex und zu groß gewesen war und dass die zugrundeliegenden Arbeiten/Dissertationen/Papers in der Analysephase veraltet gewesen sind. Dadurch konnte nicht genügend Arbeitsaufwand, vor allem in der Analysephase, auf die eigentlichen Algorithmen konzentriert werden, sondern es mußte viel Zeit und Aufwand in Aspekte der Programmierung bzw. Realisierung von Teilen gesteckt werden, die nichts mit der eigentlichen Problemstellung zu tun hatten. So wurde nicht rechtzeitig erkannt, dass die verwendeten Algorithmen von Sonntag zur Linienplanberechnung und von Weigand zur Fahrplanerstellung zwar von den Vorraussetzungen her (Direktfahrermaximierung) kompatibel waren, aber die Art der Berechnung und die Bildung der Linien nach Sonntag nicht gut zusammenpaßte mit dem was für die Fahrplanermittlung optimal wäre. Aus diesem Grund mußte auch der geplante Versuch einer Rückkopplung scheitern, da durch die Wahl der Algorithmen sich die Problematik implizit ergibt und diese nicht durch lokale Veränderungen behoben werden kann.

10.2 Ausblick

Wie schon im erzielten Resultat ein wenig vorweggenommen, wäre es nun vielleicht sinnvoll eine weitere Projektgruppe für diese Aufgabe zu bilden, die sich nun auf die eigentliche algorithmische Problematik konzentrieren könnte, da sie auf der Grundlage des geschafften Frameworks aufbauen könnte. Hier wäre es dann möglich andere Kombinationen von bekannten Algorithmen miteinander zu testen oder Varianten von neuen Lösungsansätzen zu implementieren oder auch ganz neue Verfahren zu entwickeln.

Anhang A

Handbuch

Dieses findet sich in dem externen Dokument [PG369-Handbuch].

Anhang B

Analysedokument

Während der Analysephase wurden von jeder Modul-Gruppe Kriterien aufgestellt. Diese Kriterien lassen sich in drei Kategorien unterteilen. Die Musskriterien beschreiben die Eigenschaften des Moduls, die zwingend im Endprodukt enthalten sein müssen. Die Wunschkriterien sind jene Eigenschaften, die nicht unbedingt im Programm enthalten sein müssen, bei denen es aber wünschenswert wäre und schließlich sind die Abgrenzungskriterien die Programmeigenschaften, die in keinem Fall vom jeweiligen Modul erfüllt werden.

Die Kriterien werden nun im Folgenden beschrieben. Jedes Kriterium ist mit einer Priorität versehen, wodurch eine Gewichtung vorgenommen wird. Zudem wird angegeben, wann Kriterium erfüllt wurde. Analog werden für das Haltestellen-, das Linienplan- und das Fahrplanmodul die Schnittstellen und die Programmfunktionen beschrieben.

B.1 Haltestelle

B.1.1 Musskriterien

/**AH10**

Beschreibung

Die Haltestellen werden gemäß der Fahrtwünschen gesetzt.

Priorität

10

Im Endprodukt enthalten: *Nein.*

Für die Berechnung der Haltestellen kann ein stochastischer Algorithmus, ein stochastischer Algorithmus, der die Umgebung prüft, oder ein evolutionärer Algorithmus ausgewählt werden.

/**AH20**

Beschreibung

Die Pflichthaltestellen werden gemäß der vom Modul Programm übergebenen Vorgaben gesetzt.

Priorität

10

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen*

/AH30

Beschreibung

Es werden Prioritäten für die einzelnen Haltestellen ermittelt und zugewiesen.

Priorität

10

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen*

/AH40

Beschreibung

Alle Haltestellen werden an das Straßennetz angepasst.

Priorität

9

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen*

/AH50

Beschreibung

Abhängig von der Besiedlung wird die Haltestellenvergabehäufigkeit parametrisiert.

Priorität

7

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen*

/AH60

Beschreibung

Es wird ein Straßenplan berechnet, in dem die kürzesten Wege zwischen zwei Haltestellen in einem durch den Benutzer angegebenem Radius liegen.

Priorität

8

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen.*

/AH70

Beschreibung

Aus dem Straßennetz wird ein Wegenetzgraph erzeugt.

Priorität

7

Im Endprodukt enthalten: *Ja. Seit dem ersten Prototypen.*

B.1.2 Wunschkriterien

/AH80

Beschreibung

Der Fußweg überschreitet nicht eine durch den Benutzer vorgegebene Länge

Priorität

4

Im Endprodukt enthalten: *Ja.*

/AH90

Beschreibung

Die Haltestellen werden möglichst kreuzungsnah gesetzt.

Priorität

5

Im Endprodukt enthalten: *Ja. Im Endprodukt kann gewählt werden, ob die Haltestellen kreuzungsnah oder lotnah gesetzt werden sollen*

/AH100

Beschreibung

Implementierung eines Algorithmus, der die Pflichthaltestellen automatisch setzt

Priorität

2

Im Endprodukt enthalten: *Nein.*

/AH110

Beschreibung

Die Flächen- /Einwohnererschließung ist ausreichend.

Priorität

5

Im Endprodukt enthalten: *Nein.*

B.1.3 Abgrenzungskriterien

/**AH120**

Beschreibung

Es wird weder ein Liniennetz, noch ein Fahrplan erstellt.

/**AH130**

Beschreibung

Es findet keine Fahrtzeitenermittlung für den Wegenetzgraphen statt.

/**AH140**

Beschreibung

Es findet keine Beschriftung der Knoten des Wegenetzgraphen mit Bedarfswerten statt.

/**AH150**

Beschreibung

Der Wegenetzgraph enthält bei Abschluss des Moduls Haltestelle nur die Werte:
* Eindeutige Bezeichnung einer Haltestelle (Knoten)
* x,y-Position (Knoten)
* Priorität (Knoten)
* Entfernung in km (Kanten)
* Liste der benutzten Straßenzüge (Kanten)

B.1.4 Funktionalität

Das Modul Haltestellen soll dazu dienen, aus einem vorgegebenen Straßennetz eine (möglichst) optimale Besetzung von Haltestellen zu ermitteln.

Hierzu berechnet das Modul aus dem übergebenen Straßennetz die Kreuzungen. Daraus resultierend werden die Entfernungen zwischen ihnen ermittelt um daraus einen Straßennetzgraphen aufzubauen.

Nach dem Setzen der Pflichthaltestellen, die die höchste Priorität erhalten, werden weitere Haltestellen gesetzt. Das Verfahren, mit dem die Haltestellen vergeben werden ist abhängig vom gewählten Algorithmus.

Abschließend werden der Wegenetzgraph und die Verkehrsbedarfsmatrix erzeugt. Zur Bewertung des Ergebnisses werden dann die Gütemaße der Haltestellen berechnet.

B.1.5 Schnittstellen

/**SH10**

Init
Initialisieren des Moduls, Vorbereitungen

/**SH20**

Start
Starten der Berechnung

/**SH30**

Stopp
Anhalten des Moduls

/**SH40**

Werteübergabe
Übergabe der erforderlichen Werte

/**SH50**

Ergebnisrückgabe
Rückgabe des Straßenplans und des Wegenetzgraphen an die Schnittstelle

B.1.6 Funktionen

/**FH10**

Kreuzungen bestimmen
Da das übergebene Straßennetz keine Kreuzungen enthält, müssen sie von zunächst errechnet werden. Hierzu wird der Straßennetzgraph aufgebaut.

/**FH20**

Entfernungen berechnen
Es werden die Straßenkilometer der Strecken zwischen den ermittelten Kreuzungen berechnet und an die Kanten geschrieben.

/**FH30**

Anpassen einer Haltestelle an das Straßennetz
Da nicht gewährleistet werden kann, dass die Haltestellenkoordinaten exakt den Koordinaten(-liste) einer Straße entsprechen, wird die Haltestellenkoordinate bei Bedarf angepasst.

/**FH40**

Setzen der Pflichthaltestellen
Die vom Benutzer bestimmten Pflichthaltestellen werden gesetzt, bei Bedarf an das Straßennetz angepasst und erhalten die Priorität 10. Beim Anpassen der Haltestellen wird /FH30 benutzt.

/FH50

Setzen der restlichen Haltestellen

In Abhängigkeit vom gewählten Algorithmus werden die Haltestellen gesetzt. Anschließend werden sie mit Hilfe von /FH30 an das Straßennetz angepasst. Die Höhe der Fahrtwünsche bestimmt die zu vergebende Priorität. Ein von der Schnittstelle gegebener maximaler und minimaler Radius der Erschließung kann die Haltestellensetzung beeinflussen. Jede Haltestelle erhält eine eindeutige Nummerierung (ID)

/FH60

Berechnen des Wegenetzgraph

Der für das Modul Linienplan erforderliche Wegenetzgraph wird erstellt. Ein Knoten repräsentiert eine Haltestelle in dem folgende zusätzliche Daten abgespeichert sind: eindeutige Nummer (ID), Koordinaten, Priorität und Bedarf. Zwei Haltestellen erhalten eine verbindende Kante, wenn sie sich innerhalb einer manuell bestimmten Entfernung zueinander befinden. An der Kante werden folgende Daten gespeichert: Verkehrsfluss, Liste der realen Straßen, Luftlinienentfernung und Straßenentfernung.

/FH70

Bestimmen der Verkehrsbedarfsmatrix

Zu den berechneten Haltestellen und den zu verwendenden Pflichthaltestellen wird eine Fahrtwunschmatrix aufgestellt. Die einbezogenen Kenngrößen sind der Modellbeschreibung (siehe Kapitel ??, Seite ?? ff.) zu entnehmen.

/FH80

Bewertung

Aus den an das Straßennetz angepassten Haltestellen wird nach dem im Modell beschriebenen Zielwert eine Bewertung ermittelt (siehe Kapitel ??, Seite ?? ff.).

B.2 Linienplan

B.2.1 Musskriterien

/AL10

Beschreibung Der Linienplan muss gemäß den Fahrtwünschen generiert werden unter Berücksichtigung der Kriterien

- Direktfahrermaximierung
- Max. Linienlänge
- Max. Gesamtlinienlänge

Priorität 10

Im Endprodukt enthalten: Ja.

/AL20

Beschreibung Linienplan muss einen Fahrplan entsprechend der Linienliste und der eingestellten Geschwindigkeit erstellen

Priorität 10

Im Endprodukt enthalten: Ja.

/**AL30**

Beschreibung Die Pflichtlinien müssen bei der Erstellung des Linienplans berücksichtigt werden.

Priorität 8

Im Endprodukt enthalten: Ja.

/**AL40**

Beschreibung Der Linienplan soll auf Güte hin bewertet werden können

Priorität 6

Im Endprodukt enthalten: Ja.

/**AL50**

Beschreibung Die übergebenen Linien müssen zusammenhängend sein, d.h. die Reihenfolge der Haltestellen in der Linienliste muss einem existierenden Pfad im Netzplangraph entsprechen.

Priorität 10

Im Endprodukt enthalten: Ja.

B.2.2 Wunschkriterien

/**AL60**

Beschreibung Es wird zur Generierung des Linienplans mindestens einer der folgenden Algorithmen zusätzlich verwendet:

- Simonis
- Das Verfahren nach Marcus Völker
- Durch die Projektgruppe durchgeführte Verbesserungen an Sonntag

Priorität 5

Im Endprodukt enthalten: Ja.

/**AL70**

Beschreibung Statt Dijkstra wird der k-kürzeste Wege-Algorithmus von Weigand verwendet

Priorität 3

Im Endprodukt enthalten: Nein, da Komplexität zu hoch.

/**AL80**

Beschreibung Rückkopplungsalgorithmus

Priorität 4

Im Endprodukt enthalten: Nein, siehe Kapitel ??.

/**AL90**

Beschreibung Knoten, an denen nicht umgestiegen werden kann, werden für die Verknüpfungsphase aus dem Netzplangraphen eliminiert

Priorität 3

Im Endprodukt enthalten: Nein, da Aufwand größer als Nutzen.

B.2.3 Abgrenzungskriterien

Keine gleichzeitige Berücksichtigung von Fahrplan oder Haltestellen

B.2.4 Funktionalität

Das Modul erzeugt aus einem vorgegebenen Wegenetz mit vorgegebenen Verkehrswünschen und einschränkenden Parametern (Liniengesamtlänge, Linienlänge) nach dem Verfahren von Sonntag einen Linienplan. Dabei kann auf einen vorgegebenen Linienplan zurückgegriffen werden. Zu diesem Zweck wird der Wegenetzgraph, der vom Haltestellen-Modul erzeugt wird, in den Netzplangraphen überführt. Anschließend erfolgt die Liniengenerierung, gefolgt von einer Bewertung der Linien durch die errechneten Gütemaße. Abschließend wird der Fahrplangraph erstellt, der als Berechnungsgrundlage des Fahrplanmoduls dient.

B.2.5 Schnittstellen

/**SF10**

Initialisierung Initialisiert das Modul Linienplan über den Konstruktor.

Eingabe Schnittstelle für Datenrückgabe

Ausgabe NULL

Im Endprodukt enthalten: Ja.

/**SF20**

Start Startet das Modul Linienplan.

Eingabe Wegenetzgraph, Verkehrsbedarfsmatrix, optional eine Linienliste (für vorgegebene Linien) und die Parameter:

- maximale Gesamtlinienlänge
- maximale Linienlänge
- Geschwindigkeit eines Busses
- Angabe, welche Untermodule des Linienplans verwendet werden sollen
- Angabe, welcher Algorithmus bei der Basislinienwahl benutzt werden soll

- Angabe, ob die erstellten Graphen angezeigt werden sollen

Ausgabe Keine Ausgabe, Methodenaufruf in der Schnittstelle zur Übergabe der Daten.

Im Endprodukt enthalten: Ja.

/**SF30**

Stopp Stoppt das Modul Linienplan.

Eingabe nicht vorhanden

Ausgabe nicht vorhanden

Im Endprodukt enthalten: Ja.

/**SF40**

Rückkopplung Spezielle Startmethode, um eine abgeschlossene Linienplanung erneut anzustoßen.

Eingabe Liniengesamtlänge, Linienlänge und optional eine Linienliste (für vorgegebene Linien)

Ausgabe nicht vorhanden

Im Endprodukt enthalten: Nein. Siehe Kapitel 6

B.2.6 Funktionen

/**FL10**

Initialisierung

Im Endprodukt enthalten: Ja.

/**FL40**

Kontrolle Diese Funktion koordiniert die komplette Berechnung. (Hauptfunktion des Algorithmus von Sonntags)

Preconditions Graph liegt vor, allgemeine Preconditions wurden beachtet (z.B. liegen Daten über die Liniengesamtlänge vor)

Postconditions Die Berechnung wurde durchgeführt

Im Endprodukt enthalten: Ja.

/**FL50**

Kontrolle-Überarbeitung Diese Funktion koordiniert die Berechnung, sofern ein Linienplan schon vorliegt bzw. erweitert werden soll.

Preconditions Graph liegt vor, vorgegebene Linien sind bestimmt

Postconditions Die Berechnung wurde durchgeführt

Im Endprodukt enthalten: Ja.

/**FL60**

Netzplan Ist von allen internen Klassen aus erreichbar. Verwaltet den internen Graphen, die Linienliste und die für die Linienerstellung nötigen Parameter (Liniengesamtlänge, Linienlänge und Geschwindigkeit). Enthält zudem Methoden, die von mehreren Klassen gebraucht werden.

Preconditions Wegenetzgraph liegt vor

Postconditions Netzplan ist aufgebaut, d.h. die drei Parameter entsprechenden sind richtig gesetzt und der Wegenetzgraph ist in einen Netzplangraph konvertiert worden.

Im Endprodukt enthalten: Ja.

/**FL80**

Umlage Legt die Nachfrage auf den Netzplan um, in dem sie auf dem kürzesten Weg zwischen allen Haltestellen den jeweiligen Fahrtwunsch hinzuaddiert.

Preconditions Netzplan liegt vor.

Postconditions Die entsprechenden Bewertungen liegen vor.

Im Endprodukt enthalten: Ja.

/**FL100**

Basislinienbestimmung Diese Funktion berechnet alle möglichen Basislinien und sortiert sie in einer Liste nach der Größe des Basisstroms.

Preconditions angepasster Netzplan liegt vor.

Postconditions Liste der Basislinien

Im Endprodukt enthalten: Ja.

/**FL110**

Basislinienwahl Entlang der Liste werden die Basislinien ausgewählt und anschließend entsprechend der Restriktionen (Gesamtlinienlänge, Haltestellenanschluss) angepasst.

Preconditions Liste der Basislinien und angepasster Netzplan liegen vor.

Postconditions Eine Liste von Linien wird zurückgegeben.

Im Endprodukt enthalten: Ja.

/**FL120**

EindeutigeLinienverknüpfung Endkanten von Linien werden verknüpft, wenn sie eindeutig und erlaubt sind.

Preconditions Auswahl von Linien sind gegeben.

Postconditions Eine Menge von Linien liegen vor.

Im Endprodukt enthalten: Ja.

/**FL130**

MehrdeutigeLinienverknüpfung Die optimalen Verknüpfungen von mehrdeutigen Linienüberdeckungen (an den Endkanten) werden gewählt.

Preconditions Eine Menge von Linien liegen vor.

Postconditions Liste von Linien.

Im Endprodukt enthalten: Ja.

/**FL140**

Attraktivitätsberechnung Der Attraktivitätsgewinn bei der Verknüpfung zweier Linien auf einer Kante wird berechnet.

Preconditions Zwei sich überlagernde Linien werden übergeben.

Postconditions Der Wert wurde berechnet.

Im Endprodukt enthalten: Ja.

/**FL150**

Aufbruchverknüpfung Die optimalen Aufbruchverknüpfungen (Definition siehe Sonntag) werden gefunden und durchgeführt.

Preconditions Es liegen Linien vor.

Postconditions Es liegen immer noch Linien vor.

Im Endprodukt enthalten: Ja.

/**FL160**

Bewertung Legt die Nachfrage auf den Fahrplangraph um, wobei aber nur die Kanten des Linienplans benutzt werden dürfen. Berechnet die Gütemaße:

- Anzahl der Linien
- mittlerer Verknüpfungsgrad
- mittlere Mindestreisezeit
- gesamte Streckenkilometer des Linienplans
- Kilometer der betrachteten Wege
- Anzahl der Umstiege
- Anzahl der Verbindungen mit Umstiegzwang
- Prozentsatz der Verbindungen mit Umstiegzwang
- Direktfahreranteil
- Reisezeitverlängerung

Preconditions Wegenetz und Linienplan liegen vor.

Postconditions Der Fahrplangraph ist erstellt und die entsprechenden Bewertungen sind berechnet und eingetragen.

Im Endprodukt enthalten: Ja.

B.3 Fahrplan

B.3.1 Musskriterien

/**AF10**

Fahrplanerzeugung

Beschreibung: Erzeugung eines Fahrplans durch die Angabe der räumlichen und zeitlichen Festlegung von Fahrten.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF20**

Zeitabhängige Taktänderung

Beschreibung: Der Fahrplan muss für unterschiedliche Zeitgruppen erstellt werden können. Dies bedeutet, dass der Fahrplan in den Morgenstunden anders sein kann, als der Fahrplan in den Abendstunden.

Priorität: 7

Im Endprodukt enthalten: Ja.

/**AF30**

Maximale Fahrzeuganzahl

Beschreibung: Einhaltung einer maximalen Fahrzeuganzahl. Das ist die Anzahl von gleichzeitig fahrenden Fahrzeugen im Fahrplan.

Priorität: 7

Im Endprodukt enthalten: Nein.

/**AF40**

Allgemeiner Mindesttakt

Beschreibung: Ein für alle Linien gleicher Takt (allgemeiner Mindesttakt) muss berücksichtigt werden.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF50**

Auswahl von Linienpaaren

Beschreibung: Es müssen Linienpaare berücksichtigt werden, die direkten Anschluss aneinander haben müssen.

Priorität 10

Im Endprodukt enthalten: Ja.

/**AF60**

Umsteigezeiten

Beschreibung: Es müssen Umsteigezeiten in den Haltestellen berücksichtigt werden.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF70**

Verschiedene Takte auf verschiedenen Linien

Beschreibung: Manche Linien dürfen einen größeren Takt haben als den allgemeinen Mindesttakt. Dies muss berücksichtigt werden.

Priorität: 6

Im Endprodukt enthalten: Nein.

/**AF80**

Interaktive Fahrplanoptimierung

Beschreibung: Der Benutzer muss immer die Möglichkeit haben, in den Planungsprozess manuell eingreifen zu können. Dies muss insbesondere an wichtigen Stellen geschehen.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF90**

Ausgabe eines beliebigen Haltestellenplans

Beschreibung: Es muss möglich sein, den Fahrplan für eine beliebige Haltestelle auszugeben, wobei die Abfahrtszeiten der einzelnen Linien an der Haltestelle angezeigt werden müssen.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF100**

Ausgabe eines beliebigen Linienfahrplans

Beschreibung: Es muss möglich sein, den Fahrplan für eine beliebige Linie auszugeben, wobei die Abfahrtszeiten an den einzelnen Haltestellen ausgegeben werden müssen.

Priorität: 10

Im Endprodukt enthalten: Ja.

/**AF110**

Bewertung des Fahrplans

Beschreibung: Der Fahrplan muss bewertet werden können. Die Bewertung muss anhand der Reisezeit, Wartezeit und Umsteigeaktivitäten erfolgen.

Priorität: 10

Im Endprodukt enthalten: Ja, seit dem ersten Prototypen.

/AF120

Bewertung von Routen

Beschreibung: Es muss möglich sein, eine einzelne Route eines Fahrplans bewerten zu können. Dabei gelten die gleichen Kriterien wie bei /AF110.

Priorität: 7

Im Endprodukt enthalten: Nein.

B.3.2 Wunschkriterien

/AF130

Automatische Berechnung des Mindesttaktes

Beschreibung: Es kann eine automatische Berechnung des allgemeinen Mindesttaktes anhand des Verkehrsstromes erfolgen.

Priorität: 5

Im ersten Endprodukt enthalten: Nein.

/AF140

Feste An- und Abfahrtszeiten

Beschreibung: Es können feste Abfahrts- bzw. Ankunftszeiten an bestimmten Haltestellen berücksichtigt werden.

Priorität: 3

Im Endprodukt enthalten: Teilweise.

/AF150

Ausgabe Reiseroute

Beschreibung: Ausgabe einer Reiseroute von einer Haltestelle a zu einer Haltestelle b kann erfolgen.

Priorität: 4

Im Endprodukt enthalten: Nein.

B.3.3 Funktionalität

Das Ziel bzw. die Aufgabe des Moduls ist die Minimierung des Reisezeitaufwandes. Der Reisezeitaufwand ergibt sich hierbei aus der Summe der Produkte aus Anzahl Reisender und Reisezeit.

Da die Fahrzeiten zwischen den Haltestellen, sowie die Mindesthaltezeiten und Mindestumsteigezeiten vorgegeben sind, kann mittels des Fahrplans nur die netzbedingte Wartezeit (Wartezeit, die über die Mindestwartezeit und Mindestumsteigezeit hinausgeht) beeinträchtigt werden.

Ziel des Fahrplans ist die Minimierung der netzbedingten Wartezeit unter Beachtung des Verkehrsstroms.

Die Erreichung dieses Ziels soll durch Darstellung der zeitlichen Abhängigkeiten des vorgegebenen Liniennetzes bzw. Netz der Transportverbindungen mittels des Transportkettengraphen (TKG) erfolgen.

Der TKG lässt sich aus dem Netz der Transportverbindungen dadurch ableiten, dass für jede Kante ein Knoten erzeugt wird, der somit die Fahrtzeit zwischen zwei Haltestellen symbolisiert. Knoten werden dann durch eine gerichtete Kante verbunden, wenn eine Verbindung hinsichtlich des gegebenen Liniennetzes möglich ist. Die Bewertung der sich ergebenen Kanten spiegelt die netzbedingte Wartezeit (zusätzlich multipliziert mit dem entsprechenden Verkehrsstrom) wieder. Dieses bedeutet, dass die Bindungen den Warteaufwand angeben.

B.3.4 Schnittstellen

/SF10

Initialisierung Initialisiert das Modul Fahrplan.

Parameter Wegenetzgraph mit Linien.

Rückgabetyp Boolean: *True* wenn Initialisierung erfolgreich abgeschlossen wurde. Sonst *False*

Im Endprodukt enthalten? Ja.

/SF20

Start Startet das Modul Fahrplan.

Parameter NULL

Rückgabetyp Fahrplan.

Im Endprodukt enthalten? Ja.

/SF30

Stopp Stoppt das Modul Fahrplan.

Parameter NULL

Rückgabetyp Boolean: *True* wenn das Modul erfolgreich gestoppt wurde.

Im Endprodukt enthalten? Ja.

/SF40

Heuristische Fahrplanerstellung Erstellt eine Ausgangslösung mit heuristischen Mitteln. Es wird dabei kein Optimum garantiert.

Parameter NULL

Rückgabetyp Boolean: *True*, wenn die Rechnung beginnt.

Im Endprodukt enthalten? Ja.

/SF50

Sukzessive Lösungsverbesserung Verbessert die Lösung der Heuristischen Fahrplanerstellung

Parameter NULL

Rückgabetyp Boolean: *True*, wenn die Rechnung beginnt.

Im Endprodukt enthalten? Ja.

/**SF60**

Entscheidungsbaum Größenbeschränkt Es wird ein Optimaler Fahrplan erstellt.

Parameter Anzahl der Knoten die eine Zusammenhangskomponente höchstens haben darf damit auf dieser Zusammenhangskomponente das Entscheidungsbaumverfahren angewandt werden soll.

Rückgabetyp Boolean: *True*, wenn die Rechnung beginnt.

Im Endprodukt enthalten? Ja.

/**SF70**

Entscheidungsbaum Stopkriterium Wie Schnittstelle /SF70 außer das nach endlich vielen Schritten gestoppt wird.

Parameter Iterationsschritte nach denen das Entscheidungsbaumverfahren stoppen soll.

Rückgabetyp Boolean: *True*, wenn die Rechnung beginnt.

Im Endprodukt enthalten? Ja.

/**SF80**

Fahrplan Ausgeben Gibt den Fahrplan aus. Nur möglich nach /SF40 oder /SF60 oder /SF70

Parameter NULL

Rückgabetyp Gibt den erzeugten Fahrplan an die Schnittstelle.

Im Endprodukt enthalten? Ja.

B.3.5 Funktionen

/**FF10**

Einlesen eines Graphen Diese Funktion dient dazu, den von der Schnittstelle des Programmes gelieferten Graphen einzulesen und entsprechend der, im Modul Fahrplan vereinbarten Datenstruktur aufzubereiten.

Anforderung /AF10

Preconditions Graph gemäß Schnittstellenvereinbarung

Postconditions Bereitstellung eines Graphen gemäß interner Schnittstellenvereinbarung (Netz der Transportverbindungen).

Im Endprodukt enthalten? Ja.

/**FF20**

Transportkettengraph Diese Funktion dient dazu, den Aufbau eines Transportkettengraphen (TKG) aus dem, von der Schnittstelle erhaltenen Graphen zu steuern. Dabei werden auch Verkehrswünsche usw. entsprechend interner Vereinbarung umgelegt.

Anforderung /AF10; /AF60

Preconditions Der eingelesene Graph (Linienplan) wurde von /FF10 entsprechend aufbereitet.

Postconditions Ein TKG gemäß Vereinbarung wurde erstellt.

Im Endprodukt enthalten? Ja.

/**FF30**

Anforderung des Mindestverkehrsstroms Erfragt bei dem Modul Programm den zu berücksichtigenden Mindestverkehrsstrom auf einer Linie.

Anforderung /AF10

Preconditions Keine.

Postconditions Der Mindestverkehrsstrom ist bekannt.

Im Endprodukt enthalten? Ja.

/**FF40**

Reduktion des TKG Diese Funktion reduziert den, in /FF20 erstellten TKG, indem sie beispielsweise alle zweiwertigen Knoten entfernt.

Anforderung /AF10

Preconditions Ein TKG ist vorhanden.

Postconditions Die TKG wurde korrekt reduziert und kann weiter bearbeitet werden.

Im Endprodukt enthalten? Nein.

/**FF50**

Zusammenhangskomponenten Diese Funktion stößt die Berechnung der Zusammenhangskomponenten eines vorhandenen TKG an.

Anforderung /AF10

Preconditions TKG ist vorhanden.

Postconditions Die Zusammenhangskomponenten des betrachteten TKG wurden berechnet

Im Endprodukt enthalten? Ja.

/**FF60**

Zyklomatische Zahl Diese Funktion stößt die Berechnung der zyklomatischen Zahl eines vorhandenen TKG an.

Anforderung /AF10

Preconditions TKG ist vorhanden.

Postconditions Die zyklomatische Zahl des betrachteten Transportkettengraphen wurde berechnet.

Im Endprodukt enthalten? Ja.

/**FF70**

Ausgangslösung Diese Funktion stößt die Berechnung einer Ausgangslösung, sowie des zugehörigen Simplextableaus mittels Maximalgerüsten an, dabei können einzelne Zusammenhangskomponenten einzeln betrachtet werden.

Anforderung /AF10

Preconditions TKG vorhanden, zyklomatische Zahl bekannt, Zusammenhangskomponenten bekannt.

Postconditions Für betrachteten Graphen wurde zulässige Ausgangslösung, sowie Simplextableau berechnet.

Im Endprodukt enthalten? Ja.

/**FF80**

Ausgabe Kennwerte der Lösung Diese Funktion berechnet die folgende Kennwerte einer vorhandenen Lösung und gibt sie an die Schnittstelle weiter, diese lässt sie auf dem Monitor ausgeben:

1. Die netzbedingte Wartezeit in den Knoten.
2. Der Wartezeitaufwand, d.h. das Produkt aus netzbedingten Wartezeiten und der Anzahl der Reisenden für alle Bindungen.
3. Der netzbedingte Wartezeitaufwand im Netz.
4. Der Mindestwarteaufwand im Netz.
5. Der gesamte Wartezeitaufwand im Netz.

Anforderung /AF80; /A110

Preconditions Eine Lösung wurde berechnet.

Postconditions Die aufgezählten Werte wurden berechnet und an die Schnittstelle weitergegeben.

Im Endprodukt enthalten? Ja.

/**FF90**

Interaktive Lösungverbesserung Diese Funktion, am Besten aufrufbar mittels eines Menüs auf GUI-Ebene, ermöglicht dem Benutzer im Anschluss an /FF70 und /FF80 (vorher entsprechendes Menü, oder einzelne Menüpunkte auf GUI-Ebene ausgegraut) die erhaltene Ausgangslösung wahlweise und abwechselnd wie folgt zu verbessern:

1. Die Lösung interaktiv und schrittweise verbessern (Auswahl von Bindungen ohne netzbedingte Wartezeiten). Aufruf /FF120.

2. Heuristisches Verfahren der sukzessiven Lösungsverbesserung (Tausch von Knotenbindungen, siehe Weigand Seite 65).
3. Das exakte Entscheidungsbaumproblem bei kleinen Problemen (Test, ob dies möglich übernimmt nach Auswahl des entsprechenden Menüpunktes im zweiten Prototyp /FF170).
4. Entscheidungsbaumverfahren mit heuristischen Abbruchkriterien. Aufruf /FF170

Im Anschluss an die Verbesserung wird /FF80 mit der neu erhaltenen Lösung aufgerufen. Der Benutzer kann die interaktiv berechnete Lösung behalten, oder verwerfen, bzw. als alternative Lösung zwischenspeichern. Zu diesem Zweck wird Funktion /FF210 verwendet.

Anforderung /AF80

Preconditions Eine Ausgangslösung ist vorhanden, sowie deren Kennwerte sind dem Benutzer bekannt.

Postconditions Eine Lösungsverbesserung wurde gemäß Benutzerwunsch berechnet, die Kennwerte wurden ausgegeben. Mit erhaltener Lösung wurde entsprechend Benutzerwunsch verfahren.

Im Endprodukt enthalten? Ja.

/**FF100**

Einlesen Mindesthalte- und Umsteigezeit Diese Funktion gibt die Anforderung der Mindesthalte- und Umsteigezeit — einzugeben durch den Benutzer — an die Schnittstelle und nimmt von ihr schließlich den angeforderten Wert entgegen.

Anforderung /AF60

Preconditions Werte für Mindesthalte- und Umsteigezeit werden benötigt.

Postconditions Die Werte für Mindesthalte- und Umsteigezeit sind im Modul verfügbar.

Im Endprodukt enthalten? Ja.

/**FF110**

Einlesen Taktzeit Diese Funktion gibt die Anforderung der Taktzeit, einzugeben durch den Benutzer, an die Schnittstelle weiter, und nimmt schließlich den angeforderten Wert von ihr entgegen.

Anforderung /AF40; /AF70; /AF130

Preconditions Die Taktzeit wird benötigt.

Postconditions Die Taktzeit ist im Modul verfügbar.

Im Endprodukt enthalten? Ja.

/**FF120**

Einlesen von Bindungen ohne netzbedingte Wartezeit Diese Funktion gibt die Anforderung an die Schnittstelle, den Benutzer zu fragen, ob er oder sie Bindungen ohne netzbedingte Wartezeiten auswählen möchte und falls ja, welche dies seien sollen. Für den Fall, dass der Benutzer Bindungen gewählt hat, stößt diese Funktion die Funktion /FF190 an. Sollten gewählten Bindungen nicht plausibel sein, wird eine erklärende Fehlermeldung ausgegeben. Neuwahl dann möglich.

Anforderung /AF50

Preconditions Keine.

Postconditions Wurden Bindungen ohne netzbedingte Wartezeit gewählt, so stehen diese bereit. Durch die Kontrollfunktion ist die Auswahl korrekt.

Im Endprodukt enthalten? Ja.

/**FF130**

Fahrplanaufbereitung Diese Funktion stößt die Aufarbeitung des bisher berechneten Fahrplans an.

Anforderung /AF90; /AF100; /AF150

Preconditions Ein gültiger Fahrplan wurde berechnet.

Postconditions Die Daten wurden, entsprechend aufbereitet, an die GUI weitergegeben.

Im Endprodukt enthalten? Ja.

/**FF140**

Qualitätssicherung Diese Funktion übernimmt die Qualitätskontrolle eines berechneten Fahrplans. Hierunter fällt unter anderem die Überprüfung, ob die maximale Fahrzeuganzahl eingehalten werden kann und die Bewertung des Fahrplans. Weitere Kriterien sind denkbar.

Anforderung /AF30; /AF110; /AF120

Preconditions Ein gültiger Fahrplan ist vorhanden.

Postconditions Die Funktion liefert "Qualitätsverletzungen" zurück.

Im Endprodukt enthalten? Ja.

/**FF160**

Simplexberechnungen Diese Funktion führt die im Verfahren von Weigand notwendigen Simplex Berechnungen durch. Dabei können bestimmte Parameter gesetzt werden.

Anforderung /AF10

Preconditions Zumindest ein TKG ist vorhanden.

Postconditions Die Simplexberechnungen wurden korrekt durchgeführt und die Ergebnisse zurückgegeben.

Im Endprodukt enthalten? Ja.

/**FF170**

Entscheidungsbaum Diese Funktion führt die im Verfahren von Weigand beschriebenen Entscheidungsbaum Berechnungen durch. Dabei hat diese Funktion die Fähigkeit unter bestimmten Kriterien abzubrechen, bzw. sie kann vom Benutzer angehalten werden.

Anforderung /AF10

Preconditions Ein TKG existiert.

Postconditions Die Berechnungen wurden korrekt durchgeführt und die Ergebnisse zurückgegeben.

Im Endprodukt enthalten? Ja.

/**FF180**

Kruskal Diese Funktion dient der Berechnung eines Maximalgerüstes, dabei können vom Benutzer Verbindungen vorgegeben werden.

Anforderung /AF10; /AF50

Preconditions TKG, sowie evtl. Benutzereingaben sind vorhanden.

Postconditions Das Maximalgerüst wurde entsprechend der Preconditions berechnet.

Im Endprodukt enthalten? Ja.

/**FF190**

Plausibilität Da das Verfahren von Weigand interaktiv arbeitet, an mehreren Stellen also Benutzereingaben vorgenommen werden können, welche direkten Einfluss auf die Berechnung haben, werden diese mittels dieser Funktion daraufhin untersucht, ob sie zulässig sind.

Anforderung

Preconditions Eine Benutzereingabe wurde gemacht.

Postconditions Antwort, ob Eingabe erlaubt.

Im Endprodukt enthalten? Ja.

/**FF200**

Datenbeschaffung Diese Funktion ist in der Lage, indirekte Anfragen, d. h. Anfragen über eine Schnittstelle an die, im Modul Programm gelagerte Datenbank zu entrichten. Die Anfragen können dabei aus beliebigen anderen Funktionen innerhalb des Moduls Fahrplan stammen.

Anforderung /AF30; /AF40; /AF50; /AF60; /AF70; /AF140

Preconditions Daten werden benötigt.

Postconditions Daten wurden beschafft und, entsprechend aufbereitet, richtig weitergeleitet.

Im Endprodukt enthalten? Ja.

/**FF210**

Datenspeicherung Diese Funktion ist in der Lage, zu speichernde Daten an die Schnittstelle weiterzugeben, diese legt sie in der Datenbank unter einem entsprechenden Schlüssel ab.

Anforderung /AF90; /AF100; /AF150

Preconditions Es sind zu speichernde Daten vorhanden.

Postconditions Die Daten wurden gespeichert.

Im Endprodukt enthalten? Ja.

/**FF220**

Statusmeldung Diese Funktion teilt der Schnittstelle zu bestimmten Zeiten (müssen noch definiert werden) den Status im Berechnungsprozeß mit.

Anforderung

Preconditions Fahrplanerstellung ist an einer bestimmten Stelle angekommen

Postconditions Keine.

Im Endprodukt enthalten? Ja.

B.4 Programm

B.4.1 Musskriterien

Dies sind die in der Analysephase erarbeiteten Anforderungen an den Teil der Programmgruppe. Dieser Teil wurde im Hinblick auf die Vor- und Nachteile des bei der Straßenbahn Herne-Castrop-Rauxel GmbH (HCR) verwendeten Systems, sowie des Vorliegenden Geo-Informations-Systems (GIS) geschrieben. Funktionalitäten, welche beim HCR System, sowie im GIS vermisst werden, sollen hier besonders berücksichtigt werden.

/**AP10** ANZEIGE_GEOGR

Beschreibung: Dem Benutzer muss es auf einfachem Weg ermöglicht werden, die geografischen Daten der zu betrachtenden Stadt in einem Fenster anzeigen zu lassen.

Rationale: Im bisher verwendeten System sind zu viele Schritte notwendig, um eine grafische Ausgabe zu erhalten. Dies verringert nicht nur die Benutzerakzeptanz, sondern erhöht neben dem Zeitaufwand bei der Entwicklung der anderen Module auch die Zeit der Einarbeitung der anderen, am Projekt beteiligten Personen.

Im Endprodukt enthalten: Ja.

/**AP20** ANZEIGE_DATEN

Beschreibung: In einem entsprechenden Fenster müssen sich die, vorwiegend in der Datenbank anfallenden, zusätzlichen Daten übersichtlich und für den Benutzer verständlich einblenden lassen.

Rationale: Im bisherigen System ist eine Anzeige von nicht-geografischen Daten nicht möglich. Eine Kontrolle der gespeicherten Daten ist somit nicht möglich.

Im Endprodukt enthalten: Ja: Nicht geografische Daten wie die zu übergebenden Parameter der einzelnen Module, die Gütemaße und die Ergebnisse der Fahrplanberechnung können angezeigt werden.

/**AP30** ANZEIGE_ERGEBNISSE

Beschreibung: In einem entsprechenden Fenster müssen sich die, von den anderen Modulen errechneten Ergebnisse, visuell anzeigen lassen.

Rationale: Für die Ergebnisse steht bislang keine Ausgabemöglichkeit zur Verfügung. Durch eine entsprechende, visuelle Anzeigemöglichkeit kann der Benutzer eine wesentlich bessere Bewertung der Ergebnisse vornehmen.

Im Endprodukt enthalten: Ja.

/**AP40** MANIPULATION_MENÜ

Beschreibung: Dem Benutzer muss zur Bedienung des Systems ein Menü zur Seite gestellt werden, in dem er a) die Berechnungsprozesse der einzelnen Module starten, b) die verschiedenen, anzuzeigenden Daten auswählen und c) weitere notwendige Einstellungen vornehmen kann.

Rationale: Im bisherigen System fehlt ein einheitliches Menü, von dem aus der gesamte Programmablauf steuerbar ist. Ein solches Menü führt zu einer einfacheren Bedienung.

Im Endprodukt enthalten: Ja.

/**AP50** MANIPULATION_GEOGR_LAGE

Beschreibung: Dem Benutzer muss es in einem entsprechenden Fenster ermöglicht werden, alle verwendeten Objekte (hierzu zählen: Knotenpunkte, Haltestellen, Linien und Straßen) markieren und in ihrer Position verändern zu können. Die relative Lage der betroffenen, bzw. angrenzenden oder verbundenen Objekte zum veränderten Objekt muss dabei jederzeit berücksichtigt bzw. gewährleistet bleiben.

Rationale: Eine Veränderungsmöglichkeit von geografischen Daten ist im bisherigen System nicht möglich. Insbesondere die Anforderungen der berechnenden Module machen derartige Manipulationsmöglichkeiten durch den Benutzer notwendig und tragen gleichzeitig zu einer Steigerung der Benutzerakzeptanz bei.

Im Endprodukt enthalten: Nur für Pflichthaltestellen und Haltestellen. Linien und Straßen können nicht verschoben werden. Das Straßennetz zählt zu den statischen Eingabedaten. Es wird aus einem GRASS-Layer bezogen. Wird ein anderes Straßennetz, als das von Herne gewünscht, so ist dieses als GRASS-Layer zur Verfügung zu stellen. Es können Pflichtlinien gesetzt und berechnete Linien gelöscht werden. Eine Verschiebung einer berechneten Linie ist aufgrund des im Haltestellenmodul berechneten Wegenetzgraphen nicht möglich.

/**AP60** MANIPULATION_GEOGR_HINZUFÜGEN

Beschreibung: Dem Benutzer muss es in einem entsprechenden Fenster ermöglicht werden, neue Objekte (hierzu zählen: Knotenpunkte, Haltestellen, Linien und Straßen) erstellen und somit auch in das bestehende Umfeld eingliedern zu können.

Rationale: Eine Eingabemöglichkeit ist im bisherigen System nur über den Dateiimport möglich. Insbesondere die Anforderungen der berechnenden Module machen derartige Manipulationsmöglichkeiten notwendig und tragen gleichzeitig zu einer Steigerung der Benutzerakzeptanz bei.

Im Endprodukt enthalten: Haltestellen und Linien können hinzugefügt werden. Straßen nicht. Siehe auch AP50. Um Straßen hinzuzufügen, ist der Straßenlayer mit einem geeigneten GIS System zu erweitern.

/**AP70** MANIPULATION_GEOGR_RESTRIKTIONEN

Beschreibung: Bei der Veränderung oder Erzeugung eines Objektes (hierzu zählen: Haltestellen, Knotenpunkte, Linien und Straßen) durch den Benutzer oder eines der Module, muss sichergestellt werden, dass die gewünschten Änderungen erlaubt sind und nicht gegen geografische Restriktionen verstoßen. Beispielsweise darf eine Haltestelle nicht außerhalb einer Straße liegen.

Rationale: Würde eine Überprüfung der Eingaben auf ihre Korrektheit nicht stattfinden, so würden diese Fehleingaben die Ergebnisse der einzelnen Module negativ beeinflussen. Insbesondere bei der Haltestellen- und Linienplanung könnten die generierten Daten durch Häuser oder andere Hindernisse hindurch verlegt werden.

Im Endprodukt enthalten: Ja: Eine manuell gesetzte Haltestelle wird vom Haltestellenmodul auf die nächstgelegene Straße gesetzt. Bei Eingabe der Pflichtlinien wird zusätzlich überprüft, ob sie sich über den Wegenetzgraphen bewegen. Die Eingabe von Straßen ist nicht möglich. (Vgl. /AP60)

/**AP80** MANIPULATION_DATEN

Beschreibung: Der Benutzer muss möglichst einfach und komfortabel, zu einem Objekt zusätzliche Daten eingeben, verändern und dauerhaft speichern können. Dies umfasst insbesondere die Daten, die von den einzelnen, verbundenen Modulen angefordert werden.

Rationale: Die angeschlossenen Module benötigen neben den geometrischen auch erweiterte Daten, wie z.B. den Verkehrsfluss oder die Reichweite einer Haltestelle. Hätte der Benutzer eine solche Eingabemöglichkeit nicht, könnten vorhandene Daten nicht ergänzt und korrigiert werden. Dies hätte Auswirkungen auf die Ergebnisse der einzelnen Module.

Im Endprodukt enthalten: Ja. (z.B.: Mindesthaltezeiten, Mindestumsteigezeiten, direkte Bindungen...)

/**AP90** MANIPULATION_DATEN_IMPORT

Beschreibung: Dem Benutzer muss es ermöglicht werden, bereits als Datei vorliegende, zusätzliche Daten zu einem oder mehreren Objekten, vom System einlesen zu lassen ("Importfunktion").

Rationale: Da der Benutzer nicht jedes notwendige Datum zu einem Objekt selbst eingeben muss, stellt eine solche Importfunktion eine enorme Erleichterung in der Bedienung dar.

Im Endprodukt enthalten: Ja: Die Pendlermatrix, Quelle-Ziel-Matrix und die Bevölkerungsdichte werden über Dateien eingelesen.

/**AP100** ANSICHT_FAHRPLAN

Beschreibung: Der vom System generierte Fahrplan muss für den Benutzer visuell und leicht verständlich einsehbar sein.

Rationale: Würde diese Ausgabe nicht geschehen, könnte der Benutzer ein Hauptziel des Gesamtsystems, sowie weitere Ausgaben des Fahrplanmoduls nicht kontrollieren.

Im Endprodukt enthalten: Ja: Anzeige als Haltestellenfahrplan (Ähnlich wie er an einer Haltestelle ausgehängt wird) und als Linienfahrplan. (Ähnlich dem Fahrplanbuch)

/**AP110** MANIPULATION_MODULANFRAGE

Beschreibung: Sofern ein Modul eine Eingabe durch den Benutzer erfordert, muss der benötigte Wert durch ein entsprechendes Fenster beim Benutzer abgefragt und an das anfragende Modul zurückgeliefert werden.

Rationale: Da die einzelnen Module keine Möglichkeit zur direkten Kommunikation mit dem Benutzer haben, würden Ausgabemeldungen diesen nicht erreichen. Dies gilt insbesondere für Fehlermeldungen.

Im Endprodukt enthalten: Ja.

/**AP120** MANIPULATION_GEOGR_AUSWAHL

Beschreibung: Es muss dem Benutzer ermöglicht werden, geografische Objekte (dazu zählen: Punkte, Haltestellen, Linien, Straßen) als "ausgewählt" markieren zu können.

Rationale: Die einzelnen Module verlangen nach einer Möglichkeit, Objekte als "ausgewählt" zu markieren. Würde diese Möglichkeit nicht zur Verfügung stehen, könnten die Module nicht optimal ausgeführt werden.

Im Endprodukt enthalten: Ja: Haltestellen können in verschiedenen Funktionen markiert werden. (Haltestellen hinzufügen, löschen, Pflichtlinien setzen) Linien bei der Einzelanzeige und Straßen bei festlegen eines Berechnungsauschnittes.

/**AP130** DATENBANK_DATEN_SPEICHERUNG

Beschreibung: Zu den einzelnen, geografischen Objekten (dazu zählen: Punkte, Haltestellen, Linien, Straßen) müssen zusätzliche Daten dauerhaft speicherbar und jederzeit abrufbar sein.

Rationale: Die an das Programmmodul angebundenen Module benötigen über die geografischen Daten hinaus, zwingend zusätzliche Daten, ohne die sie ihre Arbeit nicht korrekt ausführen könnten.

Im Endprodukt enthalten: Ja.

/**AP140** DATENBANK_FAHRPLAN_SPEICHERUNG

Beschreibung: Es müssen die vom Fahrplanmodul erzeugten Fahrpläne dauerhaft speicherbar und jederzeit abrufbar sein.

Rationale: Das Modul zur Fahrplanerzeugung würde ohne eine Speicherung seine Ergebnisse verlieren. Darüber hinaus können die Daten auf diese Weise von anderen Modulen genutzt bzw. vom Benutzer abgefragt werden.

Im Endprodukt enthalten: Ja. Speicherung der Fahrpläne erfolgt jedoch ausschließlich als Textdatei.

/**AP150** DATENBANK_VERSIONIERUNG

Beschreibung: Es muss eine Versionierung der erzeugten Daten möglich sein.

Rationale: Können bereits in einem früheren Lauf erzeugte Daten, durch eine Versionierung nicht verwaltet werden, so kann eine Rückkopplung nicht einwandfrei durchgeführt werden.

Im Endprodukt enthalten: Ja.

/**AP160** INTERFACE_DATEN_LIEFERUNG

Beschreibung: Die von einem Modul benötigten Daten müssen in der verlangten Form geliefert werden.

Rationale: Ein Modul kann ohne die Daten von Außen seine Arbeit nicht aufnehmen.

Im Endprodukt enthalten: Ja.

/**AP170** INTERFACE_DATEN_WEITERLEITUNG

Beschreibung: Daten, die von einem Modul geliefert werden, müssen an die entsprechenden Stellen (dies sind die Datenbank, die Benutzeroberfläche oder ein anderes Modul) weitergeleitet werden.

Rationale: Würden die Daten nicht weitergeleitet, wären diese verloren und somit die Arbeitsweise des gesamten Systems beeinträchtigt bzw. gefährdet.

Im Endprodukt enthalten: Ja.

/**AP180** INTERFACE_MODUL_STEUERUNG

Beschreibung: Die im System enthaltenen Module müssen verbunden und verwaltet werden. Dies beinhaltet auch die Steuerung der einzelnen Module.

Rationale: Würden die Module nicht verwaltet, wäre ein kontrolliertes Starten der Berechnungen in den angebundenen Modulen nicht möglich. Hierdurch würde das System seine Hauptziele nicht erreichen.

Im Endprodukt enthalten: Ja.

/**AP190** INTERFACE_MODUL_ANFRAGE

Beschreibung: Benötigt ein Modul eine Eingabe vom Benutzer, so muss dies an die Benutzeroberfläche weitergeleitet und die von dort eingelesenen Daten wieder an das Modul zurückgeliefert werden.

Rationale: Ein manuelles Eingreifen des Benutzers wäre nicht möglich, wodurch wiederum Optimierungen in den Modulen nicht durchgeführt werden könnten (insbesondere im Hinblick auf die Rückkopplungen zwischen den Modulen).

Im Endprodukt enthalten: In dieser Form von keinem Modul benötigt. Daten werden vor dem Anstoßen der Berechnungen eingegeben.

/**AP200** INTERFACE_MODUL_MELDUNG

Beschreibung: Möchte ein Modul eine Meldung an den Benutzer senden, so muss dieser Wunsch an die Benutzeroberfläche weitergeleitet werden.

Rationale: Würden Meldungen an den Benutzer nicht weitergeleitet, würden diese vor dem Benutzer verborgen bleiben. Dies gilt insbesondere für Fehlermeldungen.

Im Endprodukt enthalten: Ja.

/**AP210** INTERFACE_NETZGENERATOR

Beschreibung: Es muss eine Möglichkeit geschaffen werden, die Ausgangsdaten der einzelnen Module aus einer vorher definierten Testdatenmenge zu gewinnen. Diese Möglichkeit soll optional sein.

Rationale: Wäre es nicht möglich, auf einen genau definierten Satz von Testdaten mit bekannten Berechnungsergebnissen zurückzugreifen, würde dies den Test erheblich erschweren.

Im Endprodukt enthalten: Ja.

B.4.2 Wunschkriterien

/**AP300** WUNSCH_NAVIGATION

Beschreibung: Die Navigation innerhalb der Fenster und Menüs muss so einfach, intuitiv und einheitlich wie möglich gestaltet werden.

Rationale: Im bisher verwendeten System ist die Benutzeroberfläche und die Bedienung durch die Mischung von textuellen und grafischen Bedienelementen nicht konsistent und daher nur schwer erlernbar.

Im Endprodukt enthalten: Ja.

/**AP310** WUNSCH_ABSCHOTTUNG

Beschreibung: Der Benutzer soll so gut wie möglich, von dem verwendeten GIS, der verwendeten Datenbank und den darunter liegenden Modulen abgeschottet werden.

Rationale: Da das GIS primär der Datenspeicherung dient, braucht der Benutzer nicht zusätzlich die Bedienung des GIS zu erlernen.

Im Endprodukt enthalten: Ja.

/**AP320** WUNSCH_LÖSCHEN

Beschreibung: Dem Benutzer muss es in einem entsprechenden Fenster ermöglicht werden, bereits existierende Objekte (hierzu zählen: Knotenpunkte, Haltestellen, Linien und Straßen) zu entfernen. Die relative Lage der betroffenen, bzw. angrenzenden oder verbundenen Objekte zum entfernten Objekt muss dabei jederzeit berücksichtigt bleiben.

Rationale: Der Benutzer konnte im bisherigen GIS einmal gemachte Änderungen nicht wieder Rückgängig machen. Gerade im Hinblick auf die Schaffung einer Einfügemöglichkeit von Objekten durch den Benutzer, trägt diese Forderung zu einer Erhöhung des Bedienkomforts bei. Darüber hinaus kann das Entfernen von Objekten (insbesondere von Haltestellen und Linien) im Hinblick auf die Rückkopplung der einzelnen Module notwendig werden.

Im Endprodukt enthalten: Ja: Pflichthaltestellen, Pflichtlinien, Haltestellen und Linien können gelöscht werden. (Straßen nicht, siehe AP 60)

/**AP330** WUNSCH_FAHRPLAN_ÄNDERUNG

Beschreibung: Der vom System generierte Fahrplan muss durch den Benutzer änderbar sein.

Rationale: Aufgrund der Rückkopplung der Module könnte eine Änderung des Fahrplans durch den Benutzer das Gesamtergebnis optimieren.

Im Endprodukt enthalten: Ja: Durch Ändern der direkten Bindungen, Mindestumsteigezeiten, Mindesthaltezeiten, der Abfahrtszeit und des Taktes, kann der Fahrplan manipuliert werden.

/**AP340** WUNSCH_ZOOM

Beschreibung: Das Fenster, das für die Anzeige der geografischen Daten zuständig ist, muss seine Daten in mehreren Stufen sowohl vergrößert, als auch verkleinert anzeigen können.

Rationale: Könnte der Benutzer den angezeigten Stadtplan nicht vergrößern, wären bestimmte Teilgebiete der Karte nicht mehr einsehbar.

Im Endprodukt enthalten: Ja.

B.4.3 Abgrenzungskriterien

Das Modul "Programm" ist nicht zuständig für:

- die Berechnung der Haltestellen
- die Berechnung der Buslinien
- die Berechnung der Fahrpläne

Anhang C

Datenbankstruktur

C.1 Problemstellung

Während der Berechnung der einzelnen Module fallen an unterschiedlichen Stellen Daten an, die entweder während der aktuellen Berechnung weiter benötigt werden, oder aber bei einem späteren Programmablauf nochmals verwendet werden sollen. So ist es beispielsweise möglich, die Berechnung nach Abschluss der Linienplanung abzubrechen, die angefallenen Daten zu sichern und mit diesen gespeicherten Daten zu einem späteren Zeitpunkt mit der Fahrplanberechnung fortzufahren.

Um diese Aufgabe erfüllen zu können, müssen die benötigten Daten persistent gehalten werden. Zu diesem Zweck wird in diesem Projekt eine Oracle Datenbank mit der Versionsnummer 7.3.2 eingesetzt. Im Folgenden werden die einzelnen Relationenschemata, danach das Datenbankschema und im Anschluss das Relationenkalkül dargestellt.

C.2 Relationenschemata

In diesem Abschnitt werden die einzelnen Relationenschemata beschrieben. Dazu gehört die formale und die umgangssprachliche Beschreibung. Die formale Beschreibung erfolgt durch die nachstehende Schreibweise:

$<$RELATIONENSYMBOL / *Attribute* / *semantische-Bedingungen*$>$.

- **Speicherstand**

  $<$SPEICHERSTAND /

  {id,Bezeichnung,Zustand} /

  {id→Bezeichnung,Zustand}$>$

  Diese Relation dient der Speicherung der unterschiedlichen Speicherstände. Mit ihrer Hilfe ist es möglich unterschiedliche Fahrplanversionen zu berechnen. Die Relation hat die folgenden Attribute :

  - id, zur eindeutigen Identifizierung.
  - Bezeichnung, zur Namensgebung.

- Zustand, zur Fortschrittsbestimmung bei der Fahrplanberechnung.

Das Attribut id bildet den Schlüssel der Relation.

- **Knoten**

  <KNOTEN /
  {SpeicherID,HaltestellenID,Name,XKoordinate,YKoordinate,
  Priorität,Pflicht} /
  {SpeicherID,HaltestellenID→Name,XKoordinate,
  YKoordinate,Priorität,Pflicht}>

  Diese Relation dient der Speicherung von Haltestellen. Die Attribute sind:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - HaltestellenID, zur eindeutigen Identifikation der Haltestelle.
  - Name, zur Bezeichnung der Haltestelle.
  - XKoordinate, zur Auffindung der Haltestelle in GRASS.
  - YKoordinate, zur Auffindung der Haltestelle in GRASS.
  - Priorität, zur Kennzeichnung der Wichtigkeit einer Haltestelle.
  - Pflicht, zur Kennzeichnung ob die Haltestelle berücksichtigt werden muss.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und HaltestellenID.

- **KnotenWerte**

  <KNOTENWERTE /
  {SpeicherID,HaltestellenID,UmsteigezeitMinuten,
  MindesthaltezeitMinuten} /
  {SpeicherID,HaltestellenID→UmsteigezeitMinuten,
  MindesthaltezeitMinuten}>

  Über diese Relation werden weitere Informationen über die Haltestellen in der Datenbank gespeichert. Die Attribute sind :

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - HaltestellenID, zur eindeutigen Identifikation der Haltestelle.
  - UmsteigezeitMinuten, zur Angabe der Zeit, die zum Umsteigen an dieser Haltestelle benötigt wird. Die Zeit wird in Minuten angegeben.
  - MindesthaltezeitMinuten, zur Angabe der Zeit, die zum Halten an dieser Haltestelle von jedem Bus benötigt wird. Die Zeit wird in Minuten angegeben.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und HaltestellenID.

- **KantenStrassen**

  <KANTENSTRASSEN /

  {SpeicherID,StrassenID,lfdNr,XKoordinaten,YKoordinaten} /

  {SpeicherID,StrassenID,LfdNr→XKoordinaten,YKoordinaten}>

  Diese Relation dient dazu, Informationen über Straßen zu speichern, die in GRASS verwendet werden. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – StrassenID, zur eindeutigen Identifikation einer Straße.
  – LfdNr, zur Festlegung des Straßenverlaufes.
  – XKoordinaten, zur Auffindung der Straße im GRASS.
  – YKoordinaten, zur Auffindung der Straße im GRASS.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, StrassenID und LfdNr.

- **Kanten**

  <KANTEN /

  {SpeicherID,KantenID,StartHaltestelle,EndHaltestelle,ReisezeitMinuten,

  Länge,ReiseFluss,StrassenID} /

  {SpeicherID, KantenID → StartHaltestelle, EndHaltestelle, ReisezeitMinuten, Länge, ReiseFluss, StrassenID}>

  Diese Relation dient dazu, Informationen über die Verbindungen zweier benachbarter Haltestellen zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – KantenID, zur eindeutigen Identifizierung der Verbindung.
  – StartHaltestelle, die ID der Haltestelle, an der die Verbindung beginnt.
  – EndHaltestelle, die ID der Haltestelle, an der die Verbindung endet.
  – ReisezeitMinuten, zur Angabe der Zeit, die ein Bus benötigt um von der ersten Haltestelle zur zweiten Haltestelle über die Verbindung zu gelangen.
  – Länge, zur Angabe der Länge der Verbindung.
  – ReiseFluss, zur Angabe der Reiseaufkommens an dieser Verbindung.
  – StrassenID, zur Angabe der in GRASS gespeicherten Koordinaten der Verbindung.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und KantenID.

- **LinienPlan**

  <LinienPlan /

  {SpeicherID,PlanID,Beschreibung,LinienID} /

  {SpeicherID,PlanID,LinienID→Beschreibung}>

  Diese Relation dient dazu, unterschiedliche Linienpläne zu verwalten. Ein Linienplan definiert sich dabei über die einzelnen Linien. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – PlanID, zur Identifizierung des zugehörigen Linienplans
  – Beschreibung, zur Namensgebung des Linienplans
  – LinienID, zur Identifizierung der Linie.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, PlanID und LinienID.

- **LinienWeg**

  <LinienWeg /

  {SpeicherID,PlanID,LinienID,LfdNr,HaltestellenID} /

  {SpeicherID,PlanID,LinienID,LfdNr→HaltestellenID}>

  Diese Relation dient dazu, die Wege der einzelnen Linien zu speichern. Ein Weg wird dabei aus einer Folge von Haltestellen gebildet. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – PlanID, zur Identifizierung des zugehörigen Linienplans
  – LinienID, zur Identifizierung der Linie.
  – LfdNr, zur Festlegung der Haltestellenreihenfolge.
  – HaltestellenID, zur Angabe der einzelnen Haltestellen.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, PlanID, LinienID und LfdNr.

- **LinienWerte**

  <LinienWerte /

  {SpeicherID,LfdNr,PlanID,LinienID,VonHaltestelle,

  NachHaltestelle,Reisezeit,Verkehrsstrom} /

  {SpeicherID,PlanID,LinienID,LfdNr→VonHaltestelle,

  NachHaltestelle,Reisezeit,Verkehrsstrom}>

  Diese Relation dient dazu, Informationen über Linien zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.

- LfdNr, zur internen Nummerierung.
- PlanID, zur Identifizierung des Linienplans
- LinienID, zur Identifizierung der Linie.
- VonHaltestelle, Anfangshaltestelle der Linie
- NachHaltestelle, Endhaltestelle der Linie
- Reisezeit, zur Angabe der Reisezeit in Minuten.
- Verkehrsstrom, zur Angabe der Anzahl Reisender über diese Linie

Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, PlanID, LfdNr und LinienID.

- **Anschluss**

  <ANSCHLUSS /

  {SpeicherID,LinieVon,LinieNach,Haltestelle} /

  {∅}>

  Diese Relation dient dazu, Verbindungen zu speichern, die einen direkten Anschluss haben müssen. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - LinieVon, zur Identifizierung der ankommenden Linie.
  - LinieNach, zur Identifizierung der abgehenden Linie.
  - Haltestelle, zur Identifizierung der Haltestelle, an der die beiden Linie aufeinandertreffen.

  Der Schlüssel dieser Relation liegt auf allen Attributen.

- **Fahrplan**

  <FAHRPLAN /

  {SpeicherID,HaltestellenID,LinienID,AbfahrtszeitStd,AbfahrtszeitMin} /

  {SpeicherID,HaltestellenID,LinienID $\rightarrow$ *AbfahrtszeitStd, AbfahrtszeitMin}*>

  Diese Relation dient dazu, Fahrpläne zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - HaltestellenID, zur Identifizierung der Haltestelle, für die die Abfahrtszeit gilt.
  - LinienID, zur Identifizierung der Linie, für die die Abfahrtszeit gilt.
  - AbfahrtszeitStd, zur Stundenangabe der Abfahrtszeit.
  - AbfahrtszeitMin, zur Minutenangabe der Abfahrtszeit.

  Der Schlüssel liegt auf den Attributen SpeicherID, HaltestellenID und LinienID.

- **QuelleZielMatrix**

  <QuelleZielMatrix /

  {SpeicherID, VonHaltestelle, NachHaltestelle, Reiseaufkommen} /

  {SpeicherID, VonHaltestelle, NachHaltestelle → Reiseaufkommen}>

  Diese Relation dient dazu, die Quelle-Ziel-Matrix zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - VonHaltestelle, zur Identifizierung der Ausgangshaltestelle.
  - NachHaltestelle, zur Identifizierung der Ankunftshaltestelle.
  - Reiseaufkommen, zur Angabe des Aufkommens, das von der ersten zur zweiten Haltestelle befördert werden muss.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, VonHaltestelle und NachHaltestelle.

- **QuelleZielListe**

  <QuelleZielListe /

  {SpeicherID, xKoordinate, yKoordinate, ElementID} /

  {SpeicherID, xKoordinate, yKoordinate → ElementID}>

  Diese Relation dient dazu, die Quelle-Ziel-Liste zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - xKoordinate, zur Identifizierung in GRASS.
  - yKoordinate, zur Identifizierung in GRASS.
  - ElementID, zur Identifizierung der HaltestellenID der Quellezielmatrix.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, xKoordinate und yKoordinate.

- **Pendlermatrix**

  <Pendlermatrix /

  {SpeicherID, StadtteilVon, StadtteilNach, Dichte} /

  {SpeicherID, StadtteilVon, StadtteilNach → Dichte}>

  Diese Relation dient dazu, die Pendlermatrix zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - StadtteilVon, zur Angabe des Ausgangs-Stadtteils.
  - StadtteilNach, zur Angabe des Ankunft-Stadtteils.

– Dichte, zur Angabe des Aufkommens, das vom ersten zum zweiten Stadtteil befördert werden muss.

Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, StadtteilVon und StadtteilNach.

- **Pendlerliste**

  <PENDLERLISTE /

  {SpeicherID,xKoordinate,yKoordinate,ElementID,Radius} /

  {SpeicherID,xKoordinate,yKoordinate→ElementID,Radius}>

  Diese Relation dient dazu, die Pendlermatrix zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – xKoordinate, zur Identifizierung in GRASS.
  – yKoordinate, zur Identifizierung in GRASS.
  – ElementID, zur Identifizierung des Stadtteils in der Pendlerliste.
  – Radius, zur Angabe der Größe.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, xKoordinate und yKoordinate.

- **Zusammenhangskomponente**

  <ZUSAMMENHANGSKOMPONENTE /

  {SpeicherID,KomponentenID,LfdNummer,HaltestellenIDVon,

  HaltestellenIDNach,LinienID} /

  {SpeicherID,KomponentenID,LfdNummer→HaltestellenIDVon,

  HaltestellenIDNach,LinienID}>

  Diese Relation dient dazu, die berechneten Zusammenhangskomponenten zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – KomponentenID, zur eindeutigen Identifizierung der Zusammenhangskomponente.
  – LfdNummer, zur Nummerierung der Einträge.
  – HaltestellenIDVon, zur Identifizierung der Starthaltestelle einer Verbindung.
  – HaltestellenIDNach, zur Identifizierung der Endhaltestelle einer Verbindung.
  – LinienID, zur Identifizierung der Linie, die die beiden Haltestellen verbindet.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, KomponentenID und LfdNummer.

- **Ausgangslösung**

  $<$AUSGANGSLÖSUNG /

  {SpeicherID,LösungsID,Beschreibung,WartezeitaufwandMin,

  MindestwartezeitMin,GesamtwartezeitMin} /

  {SpeicherID,LösungsID$\rightarrow$*Beschreibung,WartezeitaufwandMin,*

  MindestwartezeitMin,GesamtwartezeitMin}$>$

  Diese Relation dient dazu, die berechnete Ausgangslösung zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - LösungsID, zur eindeutigen Identifizierung der Ausgangslösung.
  - Beschreibung, zur Angabe eines Namens für die Ausgangslösung.
  - WartezeitaufwandMin, zur Minutenangabe des Wartezeitaufwandes (siehe Kapitel 6.3).
  - MindestwartezeitMin, zur Minutenangabe der Mindestwartezeit (siehe Kapitel 6.3).
  - GesamtwartezeitMin, zur Minutenangabe der Gesamtwartezeit (siehe Kapitel 6.3).

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und LösungsID.

- **AusgangslösungHaltestelle**

  $<$AUSGANGSLÖSUNGHALTESTELLE /

  {SpeicherID,LösungsID,Haltestelle,NetzbedingteWartezeit,

  Wartezeitaufwand} /

  {SpeicherID,LösungsID,Haltestelle$\rightarrow$*NetzbedingteWartezeit,*

  Wartezeitaufwand}$>$

  Diese Relation dient dazu, weitere Informationen über die Ausgangslösung zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - LösungsID, zur Identifizierung der Ausgangslösung.
  - Haltestelle, zur Identifizierung der Haltestelle.
  - NetzbedingteWartezeit, zur Angabe der netzbedingten Wartezeit an der Haltestelle (siehe Kapitel 6.3).
  - Wartezeitaufwand, zur Angabe des Wartezeitaufwandes an der Haltestelle (siehe Kapitel 6.3).

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, LösungsID und Haltestelle.

- **LinienplanBewertung**

  <LINIENPLANBEWERTUNG /
  {SpeicherID,LinienplanID,MittlereReisezeit,Direktfahreranteil} /
  {SpeicherID,LinienplanID→MittlereReisezeit,Direktfahreranteil}>

  Diese Relation dient dazu, eine Bewertung der Linienpläne zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – LinienplanID, zur Identifizierung des Linienplans.
  – MittlereReisezeit, zur Angabe der mittleren Reisezeit des Linienplans (siehe Kapitel 6.2).
  – Direktfahreranteil, zur Angabe des direktfahrer Anteils des Linienplans (siehe Kapitel 6.2).

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und LinienplanID.

- **HaltestellenModulParameter**

  <HALTESTELLENMODULPARAMETER /
  {SpeicherID,Aufruf,Algorithmus,ZufiInitialisierung,Radiustyp,
  HaltestellenLage,Toleranzgrenze,BevoelkerungsRaster,MaxRadius,
  MinRadius,VerbindungsRadiusStadt,VerbindungsRadiusLand,
  Bevoelkerungsueberversorgung,Bevoelkerungsunterversorgung,
  Flaechenueberversorgung,Flaechenunterversorgung,
  Fahrtwunscheinbeziehung,Strassenanpassung} /
  {SpeicherID,Aufruf→Algorithmus,ZufiInitialisierung,Radiustyp,
  HaltestellenLage,Toleranzgrenze,BevoelkerungsRaster,MaxRadius,
  MinRadius,VerbindungsRadiusStadt,VerbindungsRadiusLand,
  Bevoelkerungsueberversorgung,Bevoelkerungsunterversorgung,
  Flaechenueberversorgung,Flaechenunterversorgung,
  Fahrtwunscheinbeziehung,Strassenanpassung}>

  Diese Relation dient dazu, die Parameter zu speichern, mit denen das Haltestellenmodul aufgerufen wurde. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – Aufruf, zur eindeutigen Identifizierung des Aufrufs.
  – Algorithmus, siehe Haltestellenmodul.
  – ZufiInitialisierung, siehe Haltestellenmodul.
  – Radiustyp, siehe Haltestellenmodul.
  – HaltestellenLage, siehe Haltestellenmodul.
  – Toleranzgrenze, siehe Haltestellenmodul.

- BevoelkerungsRaster, siehe Haltestellenmodul.
- MaxRadius, siehe Haltestellenmodul.
- MinRadius, siehe Haltestellenmodul.
- VerbindungsRadiusStadt, siehe Haltestellenmodul.
- VerbindungsRadiusLand, siehe Haltestellenmodul.
- Bevoelkerungsueberversorgung, siehe Haltestellenmodul.
- Bevoelkerungsunterversorgung, siehe Haltestellenmodul.
- Flaechenueberversorgung, siehe Haltestellenmodul.
- Flaechenunterversorgung, siehe Haltestellenmodul.
- Fahrtwunscheinbeziehung, siehe Haltestellenmodul.
- Strassenanpassung, siehe Haltestellenmodul.

Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und Aufruf.

- **LinienplanModulParameter**

  <LINIENPLANMODULPARAMETER /
  {SpeicherID,Aufruf,Gesamtlänge,MinLinienAnzahl,MaxLinienAnzahl,
  LinienplanID,Geschwindigkeit,Algorithmus,AnzeigeBerechnung,
  EindeutigeLinienverkn,MehrdeutigeLinienverkn,
  Aufbruchsverknuepfung,Linienendverknuepfung,
  Basislininewahl} /
  {SpeicherID,Aufruf→Gesamtlänge,MinLinienAnzahl,MaxLinienAnzahl,
  LinienplanID,Geschwindigkeit,Algorithmus,AnzeigeBerechnung,
  EindeutigeLinienverkn,MehrdeutigeLinienverkn,
  Aufbruchsverknuepfung,Linienendverknuepfung,
  Basislininewahl}>

  Diese Relation dient dazu, die Parameter zu speichern, mit denen das Linienplanmodul aufgerufen wurde. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - Aufruf, zur eindeutigen Identifizierung des Aufrufs.
  - Gesamtlänge, siehe Linienmodul.
  - MinLinienAnzahl, siehe Linienmodul.
  - MaxLinienAnzahl, siehe Linienmodul.
  - LinienplanID, siehe Linienmodul.
  - Geschwindigkeit, siehe Linienmodul.
  - Algorithmus, siehe Linienmodul.
  - AnzeigeBerechnung, siehe Linienmodul.
  - EindeutigeLinienverkn, siehe Linienmodul.

- MehrdeutigeLinienverkn, siehe Linienmodul.
- Aufbruchsverknuepfung, siehe Linienmodul.
- Linienendverknuepfung, siehe Linienmodul.
- Basislininewahl, siehe Linienmodul.

Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und Aufruf.

- **FahrplanModulParameter**

  <FAHRPLANMODULPARAMETER /
  {SpeicherID,Aufruf,MaxFahrzeuganzahl,AllgemeinerTakt,
  Mindesthaltezeit,Mindestumsteigezeit} /
  {SpeicherID,Aufruf→MaxFahrzeuganzahl,AllgemeinerTakt,
  Mindesthaltezeit,Mindestumsteigezeit}>

  Diese Relation dient dazu, die Parameter zu speichern, mit denen das Fahrplanmodul aufgerufen wurde. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - Aufruf, zur eindeutigen Identifizierung des Aufrufs.
  - MaxFahrzeuganzahl, siehe Fahrplanmodul.
  - AllgemeinerTakt, siehe Fahrplanmodul.
  - Mindesthaltezeit, siehe Fahrplanmodul.
  - Mindestumsteigezeit, siehe Fahrplanmodul.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und Aufruf.

- **Bevölkerungsdichte**

  <BEVÖLKERUNGSDICHTE /
  {SpeicherID,XKoordinate,YKoordinate,Radius,Dichte} /
  {SpeicherID,XKoordinate,YKoordinate→Radius,Dichte}>

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - XKoordinate, zur Angabe der X-Koordinate im GRASS.
  - YKoordinate, zur Angabe der Y-Koordinate im GRASS.
  - Radius, zur Angabe des Radius für den die Dichte gilt.
  - Dichte, zur Angabe der eigentlichen Bevölkerungsdichte.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, XKoordinate und YKoordinate.

- **Mindestumsteigezeit**

  $<$MINDESTUMSTEIGEZEIT /
  {SpeicherID,HaltestellenID,LinieA,LinieB,Umsteigezeit} /
  {SpeicherID,HaltestellenID→LinieA,LinieB,Umsteigezeit}$>$

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – HaltestellenID, bezeichnet die Haltestelle an der umgestiegen wird.
  – LinieA, bezeichnet die ankommende Linie.
  – LinieB, bezeichnet die abgehende Linie.
  – Umsteigezeit, bezeichnet die Zeit, die der Umstieg dauert.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und HaltestellenID.

- **Mindesthaltezeit**

  $<$MINDESTHALTEZEIT /
  {SpeicherID,HaltestellenID,Haltezeit} /
  {SpeicherID,HaltestellenID→Name,XKoordinate,
  Haltezeit}$>$

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – HaltestellenID, bezeichent die Haltestelle, für die eine Haltezeit existiert.
  – Haltezeit, bezeichnet die Zeit, die ein Bus an der gegebenen Haltestelle warten muss.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID und HaltestellenID.

- **Pflichthaltestellen**

  $<$PFLICHTHALTESTELLEN /
  {SpeicherID,XKoordinate,YKoordinate} /
  MindesthaltezeitMinuten} /

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – XKoordinate, zur Angabe der Haltestellen X-Koordinate im GRASS.
  – YKoordinate, zur Angabe der Haltestellen Y-Koordinate im GRASS.

Der Schlüssel dieser Relation liegt auf allen Attributen.

- **Ausschnitt**

  <AUSSCHNITT /
  {SpeicherID,Links,Oben,Rechts,Unten} /
  {SpeicherID→Links,Oben,Rechts,Unten}>

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – Links, die X-Koordinate der linken Seite.
  – Oben, die Y-Koordinate des oberen Randes.
  – Rechts, die X-Koordinate des rechten Randes.
  – Unten, die Y-Koordinate des unteren Randes.

  Der Schlüssel dieser Relation liegt auf dem Attribut SpeicherID.

- **UrsprungsQZMatrix**

  <URSPRUNGSQZMATRIX /
  {SpeicherID,VonHaltestelle,NachHaltestelle,Reiseaufkommen} /
  {SpeicherID,VonHaltestelle,NachHaltestelle→Reiseaufkommen}>

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – VonHaltestelle, zur Identifikation der Ausgangshaltestelle.
  – NachHaltestelle, zur Identifikation der Ankunftshaltestelle.
  – Reiseaufkommen, die Anzahl der Reisenden zwischen den beiden Haltestellen.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, VonHaltestelle und NachHaltestelle.

- **UrsprungsQZListe**

  <URSPRUNGSQZLISTE /
  {SpeicherID,XKoordinate,YKoordinate,ElementID} /
  {SpeicherID,XKoordinate,YKoordinate→ElementID}>

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  – XKoordinate, zur Angabe der X-Koordinate im GRASS.
  – YKoordinate, zur Angabe der Y-Koordinate im GRASS.

  - ElementID, zur Identifizierung der HaltestellenID aus der Ursprungs-QZMatrix.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, XKoordinate und YKoordinate.

- **Guetemasse**

  <GUETEMASSE /

  {SpeicherID, HSAnzahlHaltestellen, HSBevUeberVersorgung, HSBevUnterVersorgung, HSFlUeberVersorgung, HSFlUnterVersorgung, HSBevVersorgungProzent, HSFlVersorgungProzent, HSFussweg, HSDirekteReisezeitAuto, LPAnzahlUmstiege, LPVerbindungenUmstiegszwang, LPGesamtfahrer, LPDirektfahrer, LPMittlereReisezeit, LPDirektfahreranteil, LPStreckenkilometer, LPGesamteStreckenkilometer, LPMittlereMindestReisezeit, LPReisezeitverlaengerung, LPAnzahlLinien, LPMittlererVerknuepfungsgrad, FPAnzahlBusse, FPNetzbedingteWartezeit, FPWartezeitAufwandGesamt, FPMindestwartezeitAufwand, FPGesamtzeitAufwand, FPFahrplanwirkungsgrad, GBDurchschnittAnzahlBusse, GBMittlereReisezeit, GBAnzahlPersonen, GBReisezeitZuDirektfahrer, GBMindestreisezeitZuDirekt, LPProzentVerbkeineDir} /

  {SpeicherID → HSAnzahlHaltestellen, HSBevUeberVersorgung, HSBevUnterVersorgung, HSFlUeberVersorgung, HSFlUnterVersorgung, HSBevVersorgungProzent, HSFlVersorgungProzent, HSFussweg, HSDirekteReisezeitAuto, LPAnzahlUmstiege, LPVerbindungenUmstiegszwang, LPGesamtfahrer, LPDirektfahrer, LPMittlereReisezeit, LPDirektfahreranteil, LPStreckenkilometer, LPGesamteStreckenkilometer, LPMittlereMindestReisezeit, LPReisezeitverlaengerung, LPAnzahlLinien, LPMittlererVerknuepfungsgrad, FPAnzahlBusse, FPNetzbedingteWartezeit, FPWartezeitAufwandGesamt, FPMindestwartezeitAufwand, FPGesamtzeitAufwand, FPFahrplanwirkungsgrad, GBDurchschnittAnzahlBusse, GBMittlereReisezeit, GBAnzahlPersonen, GBReisezeitZuDirektfahrer, GBMindestreisezeitZuDirekt, LPProzentVerbkeineDir}>

  Diese Relation dient dazu, die Bewertungen der einzenlnen Module zu sichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - HSAnzahlHaltestellen, siehe Haltestellenmodul.
  - HSBevUeberVersorgung, siehe Haltestellenmodul.
  - HSBevUnterVersorgung, siehe Haltestellenmodul.
  - HSFlUeberVersorgung, siehe Haltestellenmodul.
  - HSFlUnterVersorgung, siehe Haltestellenmodul.
  - HSBevVersorgngProzent, siehe Haltestellenmodul.
  - HSFlVersorgungProzent, siehe Haltestellenmodul.
  - HSFussweg, siehe Haltestellenmodul.
  - HSDirekteReisezeitAuto, siehe Haltestellenmodul.

- LPAnzahlUmstiege, siehe Linienplanmodul.
- LPVerbindungenUmstiegszwang, siehe Linienplanmodul.
- LPGesamtfahrer, siehe Linienplanmodul.
- LPDiektfahrer, siehe Linienplanmodul.
- LPMittlereReisezeit, siehe Linienplanmodul.
- LPDirektfahreranteil, siehe Linienplanmodul.
- LPStreckenkilometer, siehe Linienplanmodul.
- LPGesamteStreckenkilometer, siehe Linienplanmodul.
- LPMittlereMindestReisezeit, siehe Linienplanmodul.
- LPReisezeitverlengerung, siehe Linienplanmodul.
- LPAnzahlLinien, siehe Linienplanmodul.
- LPMittlererVerknuepfungsgrad, siehe Linienplanmodul.
- FPAnzahlBusse, siehe Fahrplanmodul.
- FPNetzbedingteWartezeit, siehe Fahrplanmodul.
- FPWartezeitAufwandGesamt, siehe Fahrplanmodul.
- FPMindestwartezitAufwand, siehe Fahrplanmodul.
- FPGesamtzeitAufwand, siehe Fahrplanmodul.
- FPFahrplanwirkungsgrad, siehe Fahrplanmodul.
- GBDurchschnittAnzahlBusse, siehe Gesamtmodell.
- GBMittlereReisezeit, siehe Gesamtmodell.
- GBAnzahlPersonen, siehe Gesamtmodell.
- GBReisezeitZuDirektfahrer, siehe Gesamtmodell.
- GBMindestreisezeitZuDirekt, siehe Gesamtmodell.
- LPProzentVerbkeineDir, siehe Linienplanmodul.

Der Schlüssel dieser Relation liegt auf dem Attribut SpeicherID.

- **GuetemasseReisezeitAuto**

  <GUETEMASSEREISEZEITAUTO /
  {SpeicherID,HaltestelleA,HaltestelleB,Reisezeit} /
  {SpeicherID,HaltestelleA,HaltestelleB→Reisezeit}>

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  - SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.
  - HaltestelleA, identifiziert die Ausgangshaltestelle.
  - HaltestelleB, identifiziert die Ankunftshaltestelle.
  - Reisezeit, identifiziert die Reisezeit zwischen den Haltestellen.

  Der Schlüssel dieser Relation liegt auf den Attributen SpeicherID, HaltestelleA und HaltestelleB

- **Pflichtlinien**

  $<$Pflichtlinien /

  {SpeicherID,LinienID,HaltestellenID} /

  $\{\emptyset\}>$

  Diese Relation dient dazu, Informationen über die Bevölkerungsdichte zu speichern. Die Attribute lauten:

  – SpeicherID, zur Unterscheidung unterschiedlicher Datenbestände.

  – LinienID, bezeichnet eine Pflichtlinie.

  – HaltestellenID, Haltestellenfolge der Linie.

  Der Schlüssel dieser Relation liegt auf allen Attributen.

C.2.1 Datenbankschema

Mit den oben beschriebene Relationenschemata kann das Datenbankschema RS aufgestellt werden.

RS$=<$

$<$Speicherstand /
{id,Bezeichnung,Zustand} /
{id→Bezeichnung,Zustand}$>$,

$<$Knoten /
{SpeicherID,HaltestellenID,Name,XKoordinate,YKoordinate, Priorität,Pflicht} /
{SpeicherID,HaltestellenID→Name,XKoordinate, YKoordinate,Priorität,Pflicht}$>$,

$<$KnotenWerte /
{SpeicherID,HaltestellenID,UmsteigezeitMinuten, MindesthaltezeitMinuten} /
{SpeicherID,HaltestellenID→UmsteigezeitMinuten, MindesthaltezeitMinuten}$>$,

$<$KantenStrassen /
{SpeicherID, StrassenID, LfdNr, XKoordinaten, YKoordinaten} /
{SpeicherID, StrassenID, LfdNr → XKoordinaten, YKoordinaten}$>$,

$<$Kanten /
{SpeicherID, KantenID, StartHaltestelle, EndHaltestelle, ReisezeitMinuten, Länge, ReiseFluss, StrassenID} /
{SpeicherID, KantenID → StartHaltestelle, EndHaltestelle, ReisezeitMinuten, Länge, ReiseFluss, StrassenID}$>$,

$<$LinienPlan /
{SpeicherID,PlanID,Beschreibung,LinienID} /
{SpeicherID,PlanID,LinienID$\rightarrow$*Beschreibung}*$>$,

$<$LinienWeg /
{SpeicherID,PlanID,LinienID,LfdNr,HaltestellenID} /
{SpeicherID,PlanID,LinienID,LfdNr$\rightarrow$*HaltestellenID}*$>$,

$<$LinienWerte /
{SpeicherID,LfdNr,PlanID,LinienID, VonHaltestelle, NachHaltestelle,Reisezeit, Verkehrsstrom} /
{SpeicherID,PlanID,LinienID,LfdNr$\rightarrow$ *VonHaltestelle, NachHaltestelle,Reisezeit, Verkehrsstrom}*$>$,

$<$Anschluss /
{SpeicherID,LinieVon,LinieNach,Haltestelle} /
{$\emptyset$*}*$>$,

$<$Fahrplan /
{SpeicherID, HaltestellenID, LinienID, AbfahrtszeitStd, Abfahrtszeit Min} /
{SpeicherID, HaltestellenID, LinienID $\rightarrow$ *AbfahrtszeitStd, AbfahrtszeitMin}*$>$,

$<$QuelleZielMatrix /
{SpeicherID, VonHaltestelle, NachHaltestelle, Reiseaufkommen} /
{SpeicherID, VonHaltestelle, NachHaltestelle $\rightarrow$ *Reiseaufkommen}*$>$,

$<$QuelleZielListe /
{SpeicherID,xKoordinate,yKoordinate,ElementID} /
{SpeicherID,xKoordinate,yKoordinate$\rightarrow$*ElementID}*$>$,

$<$Pendlermatrix /
{SpeicherID,StadtteilVon,StadtteilNach,Dichte} /
{SpeicherID,StadtteilVon,StadtteilNach$\rightarrow$*Dichte}*$>$,

$<$Pendlerliste /
{SpeicherID, xKoordinate, yKoordinate, ElementID, Radius} /
{SpeicherID, xKoordinate, yKoordinate $\rightarrow$ *ElementID, Radius}*$>$,

$<$Zusammenhangskomponente /
{SpeicherID, KomponentenID, LfdNummer, HaltestellenIDVon, HaltestellenIDNach, LinienID} /
{SpeicherID, KomponentenID, LfdNummer $\rightarrow$ *HaltestellenIDVon, HaltestellenIDNach, LinienID}*$>$,

$<$Ausgangslösung /
{SpeicherID, LösungsID, Beschreibung, WartezeitaufwandMin, MindestwartezeitMin, GesamtwartezeitMin} /
{SpeicherID, LösungsID $\rightarrow$ *Beschreibung, WartezeitaufwandMin, MindestwartezeitMin, GesamtwartezeitMin}*$>$,

<AusgangslösungHaltestelle /

{SpeicherID, LösungsID, Haltestelle, Netzbedingte Wartezeit, Wartezeitaufwand} /

{SpeicherID, LösungsID, Haltestelle→Netzbedingte Wartezeit, Wartezeitaufwand}>,

<LinienplanBewertung /

{SpeicherID, LinienplanID, MittlereReisezeit, Direktfahreranteil} /

{SpeicherID, LinienplanID → MittlereReisezeit, Direktfahreranteil}>,

<HaltestellenModulParameter /

{SpeicherID, Aufruf, Algorithmus, ZufiInitialisierung, Radiustyp, HaltestellenLage, Toleranzgrenze, BevoelkerungsRaster, MaxRadius, MinRadius, VerbindungsRadiusStadt, VerbindungsRadiusLand, Bevoelkerungsueberversorgung, Bevoelkerungsunterversorgung, Flaechenueberversorgung, Flaechenunterversorgung, Fahrtwunscheinbeziehung, Strassenanpassung} /

{SpeicherID, Aufruf → Algorithmus, ZufiInitialisierung, Radiustyp, HaltestellenLage, Toleranzgrenze, BevoelkerungsRaster, MaxRadius, MinRadius, VerbindungsRadiusStadt, VerbindungsRadiusLand, Bevoelkerungsueberversorgung, Bevoelkerungsunterversorgung, Flaechenueberversorgung, Flaechenunterversorgung, Fahrtwunscheinbeziehung, Strassenanpassung}>,

<LinienplanModulParameter /

{SpeicherID, Aufruf, Gesamtlänge, MinLinienAnzahl, MaxLinienAnzahl, LinienplanID, Geschwindigkeit, Algorithmus, AnzeigeBerechnung, EindeutigeLinienverkn, MehrdeutigeLinienverkn, Aufbruchsverknuepfung, Linienendverknuepfung, Basislininewahl} /

{SpeicherID, Aufruf → Gesamtlänge, MinLinienAnzahl, MaxLinienAnzahl, LinienplanID, Geschwindigkeit, Algorithmus, AnzeigeBerechnung, EindeutigeLinienverkn, MehrdeutigeLinienverkn, Aufbruchsverknuepfung, Linienendverknuepfung, Basislininewahl}>,

<FahrplanModulParameter /

{SpeicherID, Aufruf, MaxFahrzeuganzahl, AllgemeinerTakt, Mindesthaltezeit, Mindestumsteigezeit} /

{SpeicherID, Aufruf→MaxFahrzeuganzahl, AllgemeinerTakt, Mindesthaltezeit, Mindestumsteigezeit}>,

<Bevölkerungsdichte /

{SpeicherID, XKoordinate, YKoordinate, Radius, Dichte} /

{SpeicherID, XKoordinate, YKoordinate→Radius, Dichte}>,

<Mindestumsteigezeit /
{SpeicherID, HaltestellenID, LinieA, LinieB, Umsteigezeit} /
{SpeicherID, HaltestellenID→LinieA, LinieB, Umsteigezeit}>,

<Mindesthaltezeit /
{SpeicherID, HaltestellenID, Haltezeit} /
{SpeicherID, HaltestellenID→Haltezeit}>,

<Pflichthaltestellen /
{SpeicherID, XKoordinate, YKoordinate} /
{∅}>,

<Ausschnitt /
{SpeicherID, Links, Oben, Rechts, Unten} /
{SpeicherID→Links, Oben, Rechts, Unten}>,

<UrsprungsQZMatrix /
{SpeicherID, VonHaltestelle, NachHaltestelle, Reiseaufkommen} /
{SpeicherID, VonHaltestelle, NachHaltestelle → Reiseaufkommen}>,

<UrsprungsQZListe /
{SpeicherID, XKoordinate, YKoordinate, ElementID} /
{SpeicherID, XKoordinate, YKoordinate→ElementID}>,

<Guetemasse /
{SpeicherID, HSAnzahlHaltestellen, HSBevUeberVersorgung,
HSBevUnterVersorgung, HSFlUeberVersorgung, HSFlUnterVersorgung,
HSBevVersorgungProzent, HSFlVersorgungProzent, HSFussweg,
HSDirekteReisezeitAuto, LPAnzahlUmstiege, LPVerbindungenUmstiegszwang,
LPGesamtfahrer, LPDirektfahrer, LPMittlereReisezeit,
LPDirektfahreranteil, LPStreckenkilometer, LPGesamteStreckenkilometer,
LPMittlereMindestReisezeit, LPReisezeitverlaengerung, LPAnzahlLinien,
LPMittlererVerknuepfungsgrad, FPAnzahlBusse, FPNetzbedingteWartezeit,
FPWartezeitAufwandGesamt, FPMindestwartezeitAufwand, FPGesamtzeitAufwand,
FPFahrplanwirkungsgrad, GBDurchschnittAnzahlBusse, GBMittlereReisezeit,
GBAnzahlPersonen, GBReisezeitZuDirektfahrer, GBMindestreisezeitZuDirekt,
LPProzentVerbkeineDir} /

{SpeicherID→HSAnzahlHaltestellen, HSBevUeberVersorgung,
HSBevUnterVersorgung, HSFlUeberVersorgung, HSFlUnterVersorgung,
HSBevVersorgungProzent, HSFlVersorgungProzent, HSFussweg,
HSDirekteReisezeitAuto, LPAnzahlUmstiege, LPVerbindungenUmstiegszwang,
LPGesamtfahrer, LPDirektfahrer, LPMittlereReisezeit,
LPDirektfahreranteil, LPStreckenkilometer, LPGesamteStreckenkilometer,
LPMittlereMindestReisezeit, LPReisezeitverlaengerung, LPAnzahlLinien,
LPMittlererVerknuepfungsgrad, FPAnzahlBusse, FPNetzbedingteWartezeit,
FPWartezeitAufwandGesamt, FPMindestwartezeitAufwand, FPGesamtzeitAufwand,
FPFahrplanwirkungsgrad, GBDurchschnittAnzahlBusse, GBMittlereReisezeit,
GBAnzahlPersonen, GBReisezeitZuDirektfahrer, GBMindestreisezeitZuDirekt,
LPProzentVerbkeineDir}$>$,

$<$ GUETEMASSEREISEZEITAUTO /
{SpeicherID, HaltestelleA, HaltestelleB, Reisezeit} /
{SpeicherID, HaltestelleA, HaltestelleB→Reisezeit}$>$,

$<$ PFLICHTLINIEN /
{SpeicherID, LinienID, HaltestellenID} /
$\{\emptyset\}>$

|

$\{\ \pi_{SpeicherID}(Knoten) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(KnotenWerte) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(KantenStrassen) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Kanten) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(LinienWeg) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(LinienWerte) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Anschluss) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(LinienPlan) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Fahrplan) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(QuelleZielMatrix) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(QuelleZielListe) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Pendlermatrix) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Pendlerliste) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Zusammenhangskomponente) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(Ausgangsloesung) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(AusgangsloesungHaltestelle) \subset \pi_{id}(Speicherstand),$
$\pi_{SpeicherID}(LinienplanBewertung) \subset \pi_{id}(Speicherstand),$

$\pi_{SpeicherID}(HaltestellenModulParameter) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(LinienplanModulParameter) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(FahrplanModulParameter) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Bevoelkerungsdichte) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Mindestumsteigezeit) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Mindesthaltezeit) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Pflichthaltestellen) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Ausschnitt) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(UrsprungsQZMatrix) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(UrsprungsQZListe) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Guetemasse) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(GuetemasseReisezeitAuto) \subset \pi_{id}(Speicherstand)$,
$\pi_{SpeicherID}(Pflichtlinien) \subset \pi_{id}(Speicherstand)$,
$\pi_{HaltestellenID}(KnotenWerte) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{StartHaltestelle}(Kanten) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{EndHaltestelle}(Kanten) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenID}(LinienWeg) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{NachHaltestelle}(LinienWerte) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{VonHaltestelle}(LinienWerte) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{Haltestelle}(Anschluss) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenID}(Fahrplan) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{VonHaltestelle}(QuelleZielMatrix) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{NachHaltestelle}(QuelleZielMatrix) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenIDVon}(Zusammenhangskomponente) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenIDNach}(Zusammenhangskomponente) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{Haltestelle}(AusgangsloesungHaltestelle) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenID}(Mindestumsteigezeit) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenID}(Mindesthaltezeit) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{VonHaltestelle}(UrsprungsQZMatrix) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{NachHaltestelle}(UrsprungsQZMatrix) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestelleA}(GuetemasseReisezeitAuto) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestelleB}(GuetemasseReisezeitAuto) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{HaltestellenID}(Pflichtlinien) \subset \pi_{HaltestellenID}(Knoten)$,
$\pi_{StrassenID}(Kanten) \subset \pi_{StrassenID}(KantenStrassen)$,
$\pi_{LinienID}(LinienWeg) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinieVon}(Anschluss) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinieNach}(Anschluss) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinienID}(LinienWerte) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinienID}(Fahrplan) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinienID}(Zusammenhangskomponente) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinieA}(Mindestumsteigezeit) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinieB}(Mindestumsteigezeit) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LinienID}(Pflichtlinien) \subset \pi_{LinienID}(LinienPlan)$,
$\pi_{LoesungsID}(AusgangsloesungHaltestelle) \subset \pi_{LoesungsID}(Ausgangsloesung)$,
$\pi_{PlanID}(LinienWeg) \subset \pi_{PlanID}(LinienPlan)$,
$\pi_{PlanID}(LinienWerte) \subset \pi_{PlanID}(LinienPlan)$,
$\pi_{LinienplanID}(LinienplanBewertung) \subset \pi_{PlanID}(LinienPlan)$,

$\pi_{LinienplanID}(LinienplanModulParameter) \subset \pi_{PlanID}(LinienPlan)$ }
|
$a(SpeicherID) := a(id) := a(Aufruf) := a(HaltestellenID) := integer,$
$a(Mindestumsteigezeit) := a(Mindesthaltezeit) := integer,$
$a(StrassenID) := a(LfdNr) := a(KantenID) := integer,$
$a(StartHaltestelle) := a(EndHaltestelle) := integer,$
$a(ReisezeitMinuten) := a(LinienID) := a(Zustand) := integer,$
$a(LinieVon) := a(LinieNach) := a(Haltestelle) := integer,$
$a(StrassenID) := a(PlanID) := a(AbfahrtszeitStd) := integer,$
$a(Reisezeit) := a(AbfahrtszeitMin) := integer,$
$a(VonHaltestelle) := a(NachHaltestelle) := integer,$
$a(HaltestelleA) := a(HaltestelleB) := integer,$
$a(KomponentenID) := a(LfdNummer) := integer,$
$a(HaltestellenIDVon) := a(HaltestellenIDNach) := integer,$
$a(LoesungsID) := a(WartezeitaufwandMin) := integer,$
$a(MindestwartezeitMin) := a(GesamtwartezeitMin) := integer,$
$a(NetzbedingteWartezeit) := a(Wartezeitaufwand) := integer,$
$a(LinienplanID) := a(MittlereReisezeit) := integer,$
$a(Direktfahreranteil) := a(Gesamtlaenge) := integer,$
$a(MaxLinienLaenge) := a(MaxLinienAnzahl) := integer,$
$a(MaxFahrzeuganzahl) := a(AllgemeinerTakt) := integer,$
$a(StadtteilVon) := a(StadtteilNach) := integer,$
$a(LinieA) := a(LinieB) := a(Umsteigezeit) := integer,$
$a(Haltezeit) := a(HSAnzahlHaltestellen) := integer,$
$a(Aufruf) := a(Algorithmus) := integer,$
$a(Radiustyp) := a(Geschwindigkeit) := integer,$
$a(LPAnzahlUmstiege) := a(LPAnzahlLinien) := integer,$
$a(FPAnzahlBusse) := integer,$
$a(Prioritaet) := prioritaet,$
$a(Pflicht) := a(PflichtLinie) := bool,$
$a(AnzeigeBerechnung) := a(Basislinienwahl) := bool,$
$a(Aufbruchsverknuepfung) := a(Linienendverknuepfung) := bool,$
$a(EindeutigeLinienverkn) := a(MehrdeutigeLinienverkn) := bool,$
$a(XKoordinate) := a(YKoordinate) := float,$
$a(xKoordinate) := a(yKoordinate) := float,$
$a(XKoordinaten) := a(YKoordinaten) := float,$
$a(Laenge) := a(ReiseFluss) := a(ReiseZeit) := float,$
$a(Links) := a(Oben) := a(Rechts) := a(Unten) := float,$
$a(Radius) := a(Dichte) := a(Reiseaufkommen) := float,$
$a(MaxRadius) := a(MinRadius) := a(Toleranzgrenze) := float,$
$a(VerbindungsRadiusStadt) := a(VerbindungsRadiusLand) := float,$
$a(Verkehrsstrom) := a(ZufiInitialisierung) := float,$
$a(HaltestellenLage) := a(Bevoelkerungsraster) := float,$
$a(Bevoelkerungsueberversorgung) := a(Bevoelkerungsunterversorgung) := float,$
$a(Flaechenueberversorgung) := a(Flaechenunterversorgung) := float,$
$a(Fahrtwunscheinbeziehung) := a(Strassenanpassung) := float,$
$a(HSBevUeberVersorgung) := a(HSBevUnterVersorgung) := float,$
$a(HSFlUeberVersorgung) := a(HSFlUnterVersorgung) := float,$
$a(HSBevVersorgungProzent) := a(HSFlVersorgungProzent) := float,$

$a(HSFussweg) := a(HSDirekteReisezeitAuto) := float,$
$a(LPVerbindungenUmstiegszwang) := a(LPGesamtfahrer) := float,$
$a(LPDirektfahrer) := a(LPMittlereReisezeit) := float,$
$a(LPDirektfahreranteil) := a(LPStreckenkilometer) := float,$
$a(LPGesamteStreckenkilometer) := a(LPMittlereMindestreisezeit) := float,$
$a(LPReisezeitverlaengerung) := a(LPMittlererVerknuepfungsgrad) := float,$
$a(FPNetzbedingteWartezeit) := a(FPWartezeitAufwandGesamt) := float,$
$a(FPMindestwartezeitAufwand) := a(FPGesamtzeitAufwand) := float,$
$a(FPFahrplanwirkungsgrad) := a(GBDurchscnittlicheAnzahlBusse) := float,$
$a(GBMittlereReisezeit) := a(GBAnzahlPersonen) := float,$
$a(GBReisezeitZuDirektfahrer) := a(GBMindestreisezeitZuDirekt) := float,$
$a(LPProzentVerbkeineDir) := float,$
$a(Bezeichnung) := a(Name) := a(Beschreibung) := alpha,$
$a(ElementID) :=?$
$>$

Die semantischen Bereichsnamen können mit den folgenden Domänen zugeordnet werden:
$b(integer) := n\varepsilon\aleph > 0,$
$b(prioritaet) := e\varepsilon\{0..9\}$
$b(float) := r\varepsilon\Re,$
$b(bool) := b\varepsilon\{0,1\}$ (0=false,1=true),
$b(alpha) := \sum^* \subset \{(alphanumerischeZeichen)\}$

C.2.2 Relationenkalkül

Im Programm reichen überwiegend einfache Anfragen aus, die den gesamten Inhalt der einzelnen Relationen für eine bestimmte SpeicherID zurückliefern. Diese SpeicherID wird beim Programmstart vergeben und soll im folgenden Speichernummer heißen. Als einzige Ausnahme ist der Fahrplan zu nennen, da dort die Ausgabe für zwei unterschiedliche Sortierungen erfolgen muss. Im folgenden werden nun die einzelnen Anfragen beschrieben.

- "Bestimme die beiden Haltestellen und das Reiseaufkommen der Quelle-Ziel-Matrix, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $QuelleZielMatrix(SpeicherID : v_{SpeicherID}, VonHaltestelle : v_{VonHaltestelle},$

  $NachHaltestelle : v_{NachHaltestelle}, Reiseaufkommen : v_{Reiseaufkommen}) \wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die beiden Koordinaten und das Element der Quelle-Ziel-Liste, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $QuelleZielListe(SpeicherID : v_{SpeicherID}, xKoordinate : v_{xKoordinate},$

  $yKoordinate : v_{yKoordinate}, ElementID : v_{ElementID})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die beiden Haltestellen und das Reiseaufkommen der Ursprungs-Quelle-Ziel-Matrix, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $UrsprungsQZMatrix(SpeicherID : v_{SpeicherID}, VonHaltestelle : v_{VonHaltestelle},$

  $NachHaltestelle : v_{NachHaltestelle}, Reiseaufkommen : v_{Reiseaufkommen})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die beiden Koordinaten und das Element der Ursprungs-Quelle-Ziel-Liste, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $UrsprungsQZListe(SpeicherID : v_{SpeicherID}, xKoordinate : v_{xKoordinate},$

  $yKoordinate : v_{yKoordinate}, ElementID : v_{ElementID})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die beiden Stadtteile und die Dichte der Pendlermatrix, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $Pendlermatrix(SpeicherID : v_{SpeicherID}, StadtteilVon : v_{StadtteilVon},$

  $StadtteilNach : v_{StadtteilNach}, Dichte : v_{Dichte})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die beiden Koordinaten, das Element und den Radius der Pendlerliste, bei denen die SpeicherID gleich der gegebenen *Speichernummer* ist"

  $Pendlerliste(SpeicherID : v_{SpeicherID}, xKoordinate : v_{xKoordinate},$

  $yKoordinate : v_{yKoordinate}, ElementID : v_{ElementID},$

  $Radius : v_{Radius})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Knoten enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Knoten(SpeicherID : v_{SpeicherID}, HaltestellenID : v_{VonHaltestellenID},$

  $Name : v_{Name}, XKoordinate : v_{XKoordinate},$

  $YKoordinate : v_{YKoordinate}, Prioritaet : v_{Prioritaet},$

  $Pflicht : v_{Pflicht})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Kanten enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Kanten(SpeicherID : v_{SpeicherID}, KantenID : v_{KantenID},$

  $StartHaltestelle : v_{StartHaltestelle}, EndHaltestelle : v_{EndHaltestelle},$

  $ReisezeitMinuten : v_{ReisezeitMinuten}, Laenge : v_{Laenge},$

  $ReiseFluss : v_{ReiseFluss}, StrassenID : v_{StrassenID}) \wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Koordinaten einer bestimmten Straße x, für die gilt, dass die SpeicherID gleich der *Speichernummer* ist"

  $KantenStrassen(SpeicherID : v_{SpeicherID}, StrassenID : v_{StrassenID},$

  $LfdNr : v_{LfdNr}, XKoordinate : v_{XKoordinate},$

  $YKoordinate : v_{YKoordinate}) \wedge$

  $(v_{SpeicherID} = Speichernummer) \wedge$

  $(v_{StrassenID} = x)$

- "Bestimme alle Linien eines Linienplans x, für die gilt, dass die SpeicherID gleich der *Speichernummer* ist"

  $LinienPlan(SpeicherID : v_{SpeicherID}, PlanID : v_{PlanID},$

  $Beschreibung : v_{Beschreibung}, LinienID : v_{LinienID}) \wedge$

  $LinienWeg(SpeicherID : v_{SpeicherID}, PlanID : v_{PlanID},$

  $LinienID : v_{LinienID},$

  $LfdNr : v_{LfdNr}, HaltestellenID : v_{HaltestellenID}) \wedge$

  $(v_{SpeicherID} = Speichernummer) \wedge$

  $(v_{PlanID} = x)$

- "Bestimme die Werte, die in der Tabelle HaltestellenModulParameter enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $HaltestellenModulParameter(SpeicherID : v_{SpeicherID}, Aufruf : v_{Aufruf},$

  $Toleranzgrenze : v_{Toleranzgrenze}, MaxRadius : v_{MaxRadius},$

  $MinRadius : v_{MinRadius}, VerbindungsRadiusStadt : v_{VerbindungsRadiusStadt},$

  $ZufiInitialisierung : v_{ZufiInitialisierung}, Radiustyp : v_{Radiustyp},$

  $HaltestellenLage : v_{HaltestellenLage}, BevoelkerungsRaster : v_{BevoelkerungsRaster},$

  $Bevoelkerungsueberversorgung : v_{Bevoelkerungsueberversorgung},$

  $\pi_{LinienplanID}(LinienplanModulParameter) \subset \pi_{PlanID}(LinienPlan) Bevoelkerungsunterversorgung : v_{Bevoelkerungsunterversorgung},$

  $Flaechenueberversorgung : v_{Flaechenueberversorgung},,$

  $Flaechenunterversorgung : v_{Flaechenunterversorgung},$

$Fahrtwunscheinbeziehung : v_{Fahrtwunscheinbeziehung}, Strassenanpassung : v_{Strassenanpassung},$

$VerbindungsRadiusLand : v_{VerbindungsRadiusLand}, Algorithmus : v_{Algorithmus})\wedge$

$(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Bevölkerungsdichte enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Bevoelkerungsdichte(SpeicherID : v_{SpeicherID}, XKoordinate : v_{XKoordinate},$

  $YKoordinate : v_{YKoordinate}, Radius : v_{Radius},$

  $Dichte : v_{Dichte})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle LinienplanModulParameter enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $LinienplanModulParameter,$

  $(SpeicherID : v_{SpeicherID}, Aufruf : v_{Aufruf},$

  $Gesamtlaenge : v_{Gesamtlaenge}, MaxLinienLaenge : v_{MaxLinienLaenge},$

  $Geschwindigkeit : v_{Geschwindigkeit}, Algorithmus : v_{Algorithmus},$

  $AnzeigeBerechnung : v_{AnzeigeBerechnung}, EindeutigeLinienverkn : v_{EindeutigeLinienverkn},$

  $Aufbruchsverknuepfung : v_{Aufbruchsverknuepfung}, MehrdeutigeLinienverkn : v_{MehrdeutigeLinienverkn},$

  $Linienendverknuepfung : v_{Linienendverknuepfung}, Basislinienwahl : v_{Basislinienwahl},$

  $MaxLinienAnzahl : v_{MaxLinienAnzahl}, LinienplanID : v_{LinienplanID})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme für eine Linie x den Linienweg. Das Attribut SpeicherID muß dabei gleich *Speichernummer* sein"

  $LinienWeg(SpeicherID : v_{SpeicherID}, PlanID : v_{PlanID},$

  $LinienID : v_{LinienID},$

  $LfdNr : v_{LfdNr}, HaltestellenID : v_{HaltestellenID})\wedge$

  $(v_{SpeicherID} = Speichernummer)\wedge$

  $(v_{LinienID} = x)$

- "Bestimme die Werte, die in der Tabelle LinienPlan enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $LinienPlan(SpeicherID : v_{SpeicherID}, PlanID : v_{PlanID},$

  $Beschreibung : v_{Beschreibung}, LinienID : v_{LinienID})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle KnotenWerte enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $KnotenWerte(SpeicherID : v_{SpeicherID}, HaltestellenID : v_{HaltestellenID},$
  $UmsteigezeitMinuten : v_{UmsteigezeitMinuten}, MindesthaltezeitMinuten : v_{MindesthaltezeitMinuten})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle LinienWerte enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $LinienWerte(SpeicherID : v_{SpeicherID}, LinienID : v_{LinienID},$
  $Reisezeit : v_{Reisezeit}, PlanID : v_{PlanID},$
  $VonHaltestelle : v_{VonHaltestelle}, NachHaltestelle : v_{NachHaltestelle},$
  $LfdNr : v_{LfdNr}, Verkehrsstrom : v_{Verkehrsstrom})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Anschluss enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Anschluss$
  $(SpeicherID : v_{SpeicherID}, LinieVon : v_{LinieVon},$
  $LinieNach : v_{LinieNach}, Haltestelle : v_{Haltestelle})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Fahrplan enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist. Die Sortierung soll auf den Attributen HaltestellenID und LinienID liegen"

  $Fahrplan(SpeicherID : v_{SpeicherID}, HaltestellenID : v_{HaltestellenID},$
  $LinienID : v_{LinienID}, AbfahrtszeitStd : v_{AbfahrtszeitStd},$
  $AbfahrtszeitMin : v_{AbfahrtszeitMin})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Fahrplan enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist. Die Sortierung soll auf den Attributen LinienID und HaltestellenID liegen"

  $Fahrplan(SpeicherID : v_{SpeicherID}, LinienID : v_{LinienID},$
  $HaltestellenID : v_{HaltestellenID}, AbfahrtszeitStd : v_{AbfahrtszeitStd},$
  $AbfahrtszeitMin : v_{AbfahrtszeitMin})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Zusammenhangskomponente enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Zusammenhangskomponente(SpeicherID : v_{SpeicherID}, KomponentenID : v_{KomponentenID},$
  $HaltestellenIDVon : v_{HaltestellenIDVon}, HaltestellenIDNach : v_{HaltestellenIDNach},$

$LinienID : v_{LinienID}, LfdNummer, v :_{LfdNummer})\wedge$
$(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle Ausgangslösung enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $Ausgangsloesung(SpeicherID : v_{SpeicherID}, LoesungsID : v_{LoesungsID},$
  $Beschreibung : v_{Beschreibung}, WartezeitaufwandMin : v_{WartezeitaufwandMin},\wedge$
  $MindestwartezeitMin : v_{MindestwartezeitMin}, GesamtwartezeitMin : v_{GesamtwartezeitMin})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle AusgangslösungHaltestelle enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $AusgangsloesungHaltestelle(SpeicherID : v_{SpeicherID},$
  $Haltestelle : v_{Haltestelle}, NetzbedingteWartezeit : v_{NetzbedingteWartezeit},$
  $Wartezeitaufwand : v_{Wartezeitaufwand}, LoesungsID : v_{LoesungsID})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Linienplanbewertung für einen bestimmten Linienplan x, wobei die SpeicherID gleich der *Speichernummer* ist"

  $LinienplanBewertung(SpeicherID : v_{SpeicherID},$
  $LinienplanID : v_{LinienplanID}, MittlereReisezeit : v_{MittlereReisezeit},$
  $Direktfahreranteil : v_{Direktfahreranteil})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte, die in der Tabelle FahrplanModulParameter enthalten sind und für die gilt, dass SpeicherID gleich *Speichernummer* ist"

  $FahrplanModulParameter(SpeicherID : v_{SpeicherID}, Aufruf : v_{Aufruf},$
  $Mindesthaltezeit : v_{Mindesthaltezeit}, Mindestumsteigezeit : v_{Mindestumsteigezeit},$
  $MaxFahrzeuganzahl : v_{MaxFahrzeuganzahl}, AllgemeinerTakt : v_{AllgemeinerTakt})\wedge$
  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die id des Speicherstandes mit der Bezeichnung x"

  $(\exists v_{Bezeichnung})$
  $[Speicherstand(id : v_{id}, Bezeichnung : v_{Bezeichnung}, Zustand : v_{Zustand})\wedge$
  $(v_{Bezeichnung} = x)]$

- "Bestimme die Mindestumsteigezeit für alle Haltestellen, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Mindestumsteigezeit(SpeicherID : v_{SpeicherID},$

$HaltestellenID : v_{HaltestellenID}, LinieA : v_{LinieA},$

$LinieB : v_{LinieB}, Umsteigezeit : v_{Umsteigezeit})\wedge$

$(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Mindesthaltezeit für alle Haltestellen, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Mindesthaltezeit(SpeicherID : v_{SpeicherID},$

  $HaltestellenID : v_{HaltestellenID}, Haltezeit : v_{Haltezeit})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme alle Pflichthaltestellen, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Pflichthaltestellen$

  $(SpeicherID : v_{SpeicherID},$

  $XKoordinate : v_{XKoordinate}, YKoordinate : v_{YKoordinate})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Grenzen eines Ausschittes, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Ausschnitt(SpeicherID : v_{SpeicherID}, Links : v_{Links}$

  $Rechts : v_{Rechts}, Oben : v_{Oben}, Unten : v_{Unten})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme alle Pflichtlinien, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Pflichthaltestellen$

  $(SpeicherID : v_{SpeicherID},$

  $LinienID : v_{LinienID}, HaltestellenID : v_{HaltestellenID})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte der GuetemasseReisezeitAuto, wobei die SpeicherID gleich der *Speichernummer* ist"

  $GuetemasseReisezeitAuto(SpeicherID : v_{SpeicherID},$

  $HaltestelleA : v_{HaltestelleA}, HaltestelleB : v_{HaltestelleB},$

  $Reisezeit : v_{Reisezeit})\wedge$

  $(v_{SpeicherID} = Speichernummer)$

- "Bestimme die Werte der Gütemaße, wobei die SpeicherID gleich der *Speichernummer* ist"

  $Guetemasse(SpeicherID : v_{SpeicherID},$

  $HSAnzahlHaltestellen : v_{HSAnzahlHaltestellen}, HSBevUeberVersorgung : v_{HSBevUeberVersorgung},$

  $HSBevUnterVersorgung : v_{HSBevUnterVersorgung}, HSFlUeberVersorgung : v_{HSFlUeberVersorgung},$

$HSFlUnterVersorgung : v_{HSFlUnterVersorgung}, HSBevVersorgungProzent : v_{HSBevVersorgungProzent}, HSFlVersorgungProzent : v_{HSFlVersorgungProzent}, HSFussweg : v_{HSFussweg}, HSDirekteReisezeitAuto : v_{HSDirekteReisezeitAuto}, LPAnzahlUmstiege : v_{LPAnzahlUmstiege}, LPVerbindungenUmstiegszwang : v_{LPVerbindungenUmstiegszwang}, LPGesamtfahrer : v_{LPGesamtfahrer}, LPDirektfahrer : v_{LPDirektfahrer}, LPMittlereReisezeit : v_{LPMittlereReisezeit}, LPDirektfahrerAnteil : v_{LPDirektfahrerAnteil}, LPStreckenkilometer : v_{LPStreckenkilometer}, LPGesamteStreckenkilometer : v_{LPGesamteStreckenkilometer}, LPMittlereMindestReisezeit : v_{LPMittlereMindestReisezeit}, LPReisezeitverlaengerung : v_{LPReisezeitverlaengerung}, LPAnzahlLinien : v_{LPAnzahlLinien},$

$LPMittlererVerknuepfungsgrad : v_{LPMittlererVerknuepfungsgrad}, FPAnzahlBusse : v_{FPAnzahlBusse}, FPNetzbedingteWartezeit : v_{FPNetzbedingteWartezeit}, FPWartezeitAufwandGesamt : v_{FPWartezeitAufwandGesamt}, FPMindestwartezeitAufwand : v_{FPMindestwartezeitAufwand}, FPGesamtzeitAufwand : v_{FPGesamtzeitAufwand}, FPFahrplanwirkungsgrad : v_{FPFahrplanwirkungsgrad}, GBDurchschnittAnzahlBusse : v_{GBDurchschnittAnzahlBusse}, GBMittlereReisezeit : v_{GBMittlereReisezeit}, GBAnzahlPersonen : v_{GBAnzahlPersonen}, GBReisezeitZuDirektfahrer : v_{GBReisezeitZuDirektfahrer}, GBMindestreisezeitZuDirekt : v_{MindestreisezeitZuDirekt}, LPProzentVerbkeineDir : v_{LPProzentVerbkeineDir})\wedge$
$(v_{SpeicherID} = Speichernummer)$

Anhang D

Verwendete Klassen

D.1 Liste aller Klassen

Die Liste aller Klassen wurde in eine Anlage (siehe [PG369-Klassendokumentation]) ausgelagert.

D.2 Klassendiagramme in UML-Notation

Abbildung D.1: Das Klassendiagramm der gemeinsam verwendeten Datenstrukturen

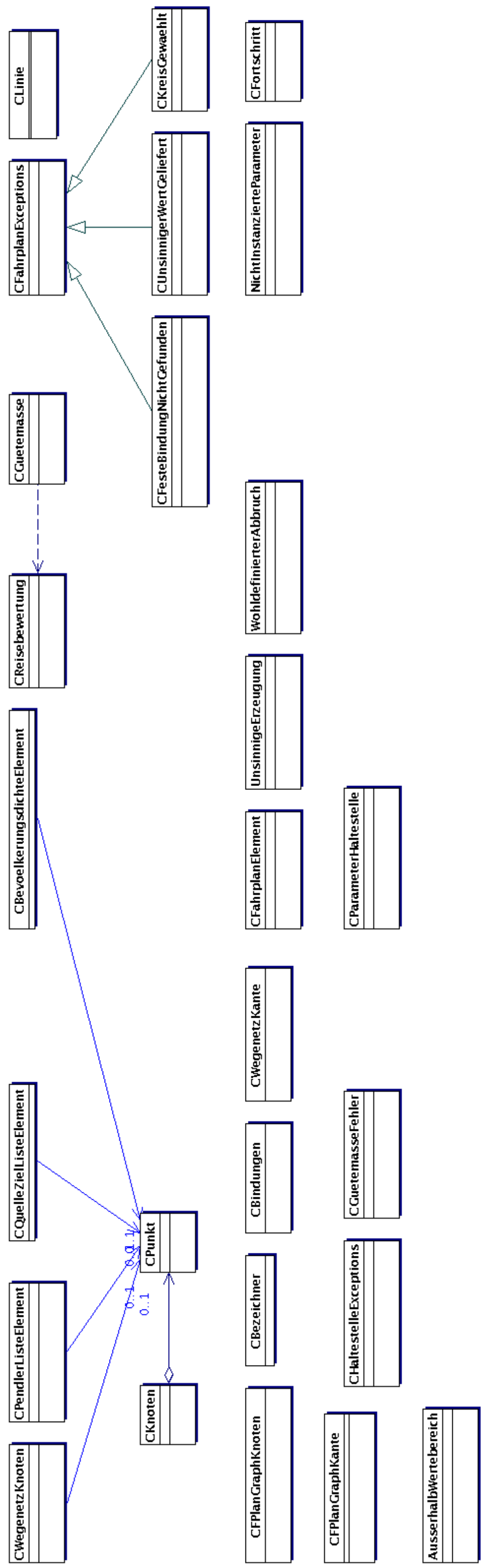


Abbildung D.2: Das Klassendiagramm des Haltestellenmoduls

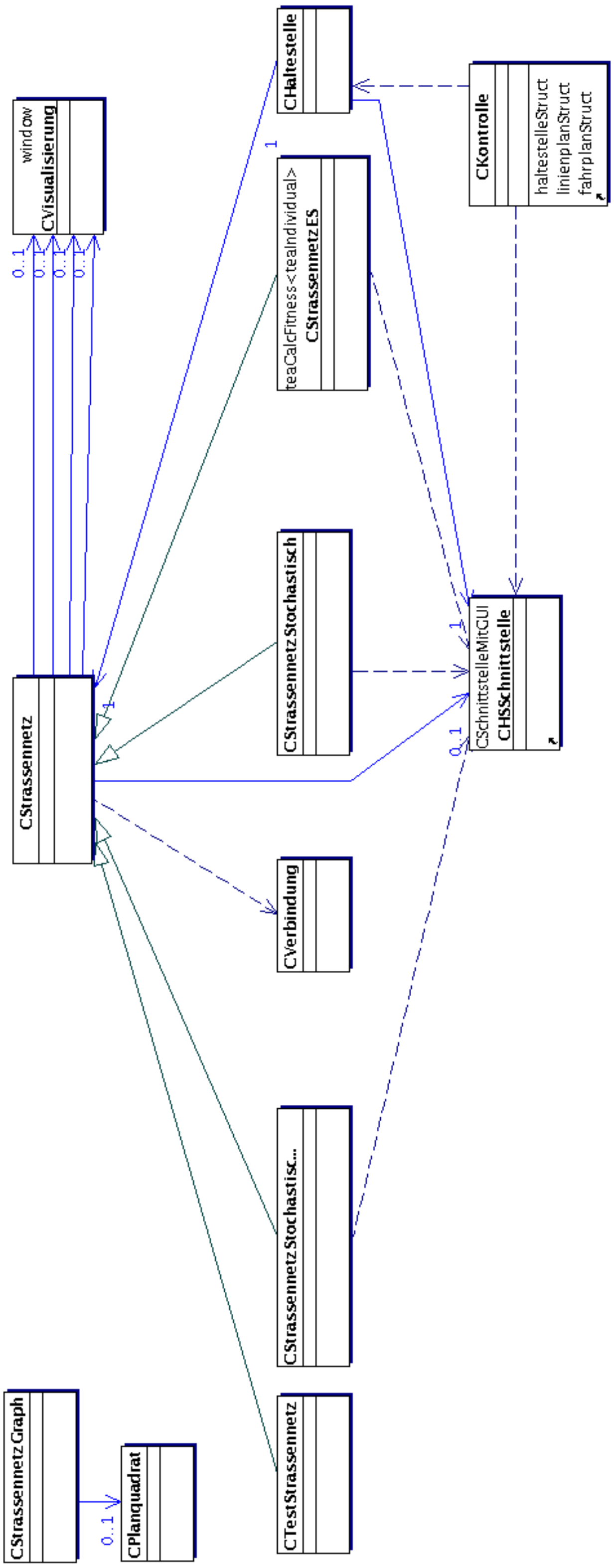


Abbildung D.3: Das Klassendiagramm des Linienplanmoduls

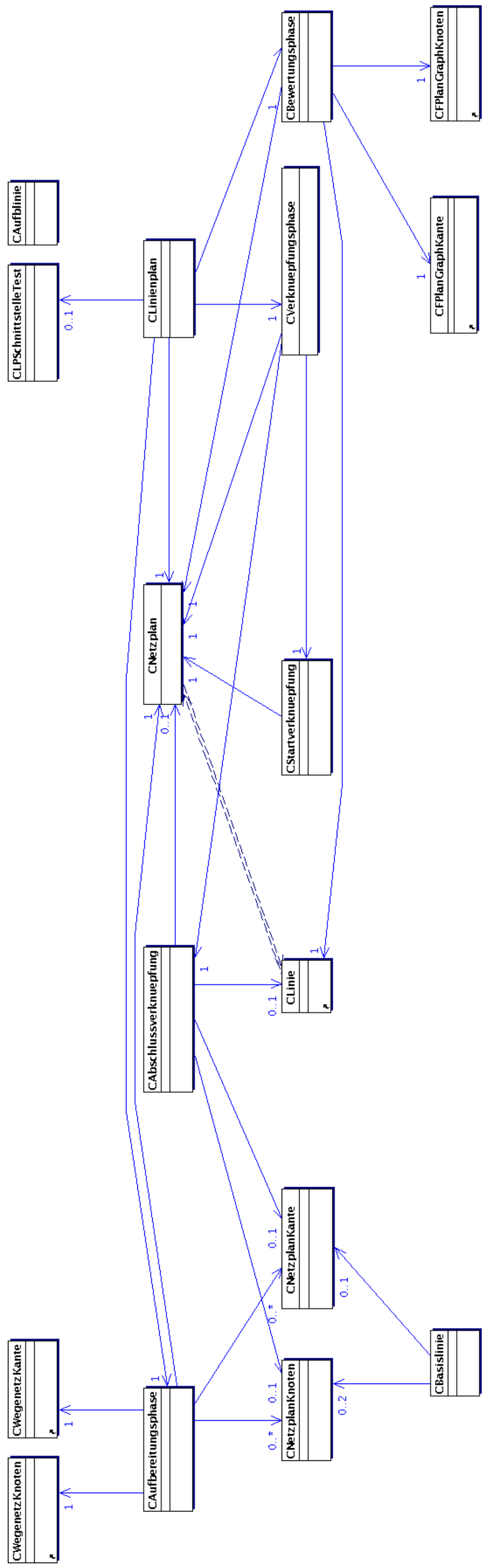


Abbildung D.4: Das Klassendiagramm des Fahrplanmoduls

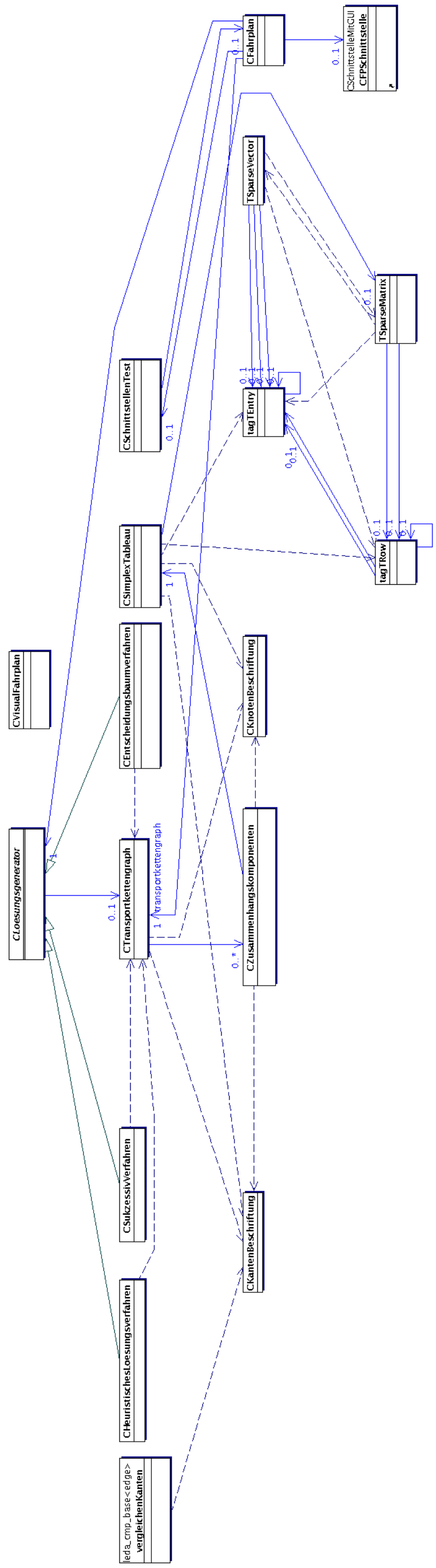
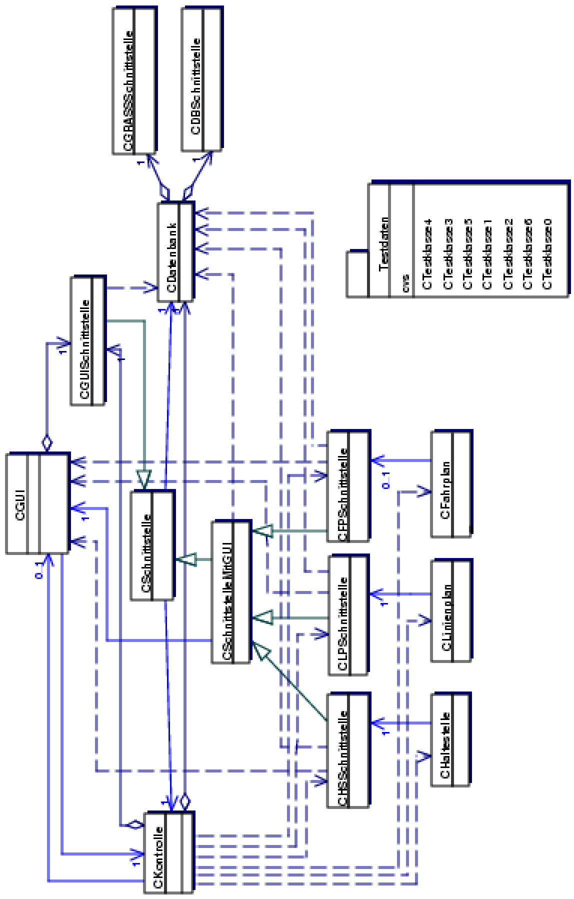


Abbildung D.5: Das Klassendiagramm des Programmmoduls



Anhang E

Testdokument

E.1 Einleitung

Dieser Abschnitt soll alle Testdokumente zusammenfassen und das fiktive Testszenario beschreiben. Dabei richete es sich grob nach der von der IEEE aufgestellen Testdokumentation [IEEE-829]. Es soll hier nur der Systemtest für den ersten Prototypen festgehalten werden. Die Klassen- und Modultest wurden von den Entwicklern selber durchgeführt und die Integrationstests innerhalb der Modul-Gruppen vorgenommen.

Bei dem Systemtest handelt es sich lediglich um einen Test des Datenflusses zur Kontrolle des korrekten Zusammenspiels der einzelnen Module und der Kommunikation über die Schnittstelle. Mit diesem Testszenario wird auch die Funktion der Datenbank überprüft. Fehleraufdeckende Test, sein es Fehler in den Algorithmen oder Test hinsichtlich Robustheit und Zuverlässigkeit, bleiben dem Entwickler überlassen und werden durch den Systemtest nicht berücksichtigt.

In den folgenden Kapiteln werden einige Dokumente, wie sie in der Software-Entwicklung eingesetzt werden, aufgeführt. Diese Dokumente beschreiben das Testszenario und die Testfälle, die darin zum Einsatz kommen.

In Kapitel E.7 werden die Testläufe beschrieben. Dazu werden alle Einstellungen mit den gewählten Werten aufgeführt und die jeweiligen Ergebnisse dokumentiert.

E.2 Testplan

E.2.1 Ziel des Systemtests

Das Ziel des Systemtests ist, den korrekten Programmablauf zu überprüfen. Testgegenstand ist in diesem Fall also das gesamte System. Zu diesem Zweck wurde ein Testszenario erdacht, das all die Eingabe- und Systemdaten, die zum Programmablauf benötigt werden, bereitstellt. Damit ist es möglich einen kompletten Programmdurchlauf anzusterben und die Fehler in der Kommunikation der Modul mit der Schnittstelle und der Schnittstelle mit der Datenbank aufzudecken und nach Möglichkeit zu beheben.

E.2.2 Herangehensweise

Das Szenario beinhaltet einen kleinen Straßenplan, bestehend aus fünf Straßen. Diese Stadt wurde dann in vier Stadtgebiete unterteil, den Pendlerbewegungen zugeordnet wurden. Auf dem Straßenplan wurden willkürlich die schon in der "Stadt" vorhandene Haltestellen platziert und ihnen Fahrtwünsche in der Form "von diese Haltestellen wollen x Fahrgäste zur Haltestelle y" für alle anderen Haltestellen beigefügt. Desweiteren wurden Pflichthaltestellen, die später im Programm vom Benutzer gesetzt werde können, eingefügt. Auch die anderen Eingaben, die später durch den Benutzer eingegeben werden, wurden festgelegt. Damit der Test aller Module gleichzeitig möglich ist, wurden die Ausgabedaten der Module Haltestelle, Linienplan und Fahrplan ebenfalls im Vorfeld festgesetzt.

E.2.3 Umfang des Tests

Wie oben schon angesprochen, soll der Systemtest das Zusammenspiel des gesamten Programmes testen. Dabei geht es in erster Linie um den Datenfluß, der korrekt durch die Module laufen muß. Sollten dabei logische Fehler in den Alogrithmen aufgedeckt werden, müssen diese natürlich korrigiert werden.

E.2.4 Verantwortlichkeiten

Jede Gruppe, die für ein Modul verantwortlich ist, ist auch für das Testen der Module und den Test des Gesamtsystems verantwortlich. Fehler werden allerdings nur in den eigenen Modulen und Klassen berichtigt.

E.3 Testentwurfsspezifikation

Testgegenstände des Systemtests sind die einzelnen Module, als da wären Haltestelle, Linienplan, Fahrplan und das Programm-Modul.

E.3.1 Testgegenstand Haltestelle

Im Modul Haltestelle sollen die Haltestellen berechnet werden. Zudem muß der Wegenetzgraph für das Modul Linienplan aufgebaut werden. Um die Berechnungen durchführen zu können, benötigt das Modul einen Straßenplan, wie er später aus dem Geoinformationssystem GRASS geliefert wird. Dieser wurde mit Erstellung des Testszenarios aufgebaut und ist das zentrale Element darin, da sich alle Eingabedaten, die nicht vom Benutzer stammen, auf diesen Straßenplan beziehen. Der Straßenplan muß in jedem Fall Straßen enthalten, die sich kreuzen. Zudem sollten die Straßen nicht alle gerade verlaufen, sondern auch Kurvenverläufe, wie sie später in GRASS vorkommen, enthalten. Diese Eingabe, die nicht vom Benutzer beeinflußbar sind, lauten:

- Pendlermatrix
- Pendlerliste
- Fahrtwunschmatrix
- Fahrtwunschliste
- Bevölkerungsdichteliste

Diese Daten wurden zufällig ausgewählt und festgehalten. Die Eingabe, die durch den Benutzer vorgenommen wurden, wurden ebenfalls erdacht und festgehalten. Eine Besonderheit bei den Benutzereingaben sind die Pflichthaltestellen, die später einmal in GRASS manuell gesetzt werden sollen können. Für diese wurden ebenfalls Koordianten festgelegt, an denen sich die Pflichthaltestellen befinden sollen. Die sonstigen Benutzereingaben sind:

- max. Radius Einzugsgebiet einer Haltestelle
- min. Radius Einzugsgebiet einer Haltestelle
- Verbindungsradius
- Toleranzgrenze

E.3.2 Testgegenstand Linienplan

Das Linienplan-Modul ist für die Berechnung eines korrekten Linienplans verantwortlich und ist auf die vom Haltestellen-Modul vergebenen Haltestellen angewiesen. Diese werden dem Linienplan in Form eines Wegenetzgraphen übergeben, in dessen Knoten die Haltestellen stehen, und deren Kanten mit der Entfernung der einzelnen Haltestellen beschriftet sind. Diese Graph wurde von den Gruppen, die die beiden Module realisieren abgesprochen und festgelegt. Er wird dem Linienplan vom Programm bereitgestellt, nachdem dieses denb Graphen von den Haltestellen erhalten hat. Als Benutzereingabe bekommt der Linienplan folgende Daten übergeben.

- Liniengesamtlänge
- max. Linienanzahl
- max. Linienfahrtzeit
- Verkehrsbedarfsmatrix
- vorgegebene Linien

Diese Daten wurden zwischen Programm- und Linienplangruppe abgestimmt und festgelegt.

E.3.3 Testgegenstand Fahrplan

Der Fahrplan in der gesamten Berechnung eines Busliniennetzes wird von dem Modul der Fahrplan-Gruppe erstellt. Als zentrale Eingabe bekommt dieses Modul den sogenannten Fahrplangraphen übergeben. Verkehrsstrom, Fahrtzeit, Linien-ID sind die Kanteninformationen, wohingegen in den Knoten lediglich die Haltestellen-ID steht. Auch dieses Modul bietet die Möglichkeit einer flexiblen Berechnung des Fahrplan durch Eingaben, die der Benutzer festlegen kann. Dabei handelt es sich um:

- Mindesthaltezeit
- Mindestumsteigezeit
- Feste Verbindungen
- Taktzeit
- Abfahrtszeit
- Fahrplanintervall
- Zusammenhangskomponenten - Nummer

E.3.4 Testgegenstand Programm

Das Programm stellt die Kommunikation mit dem Benutzer als auch die Kommunikation der Module untereinander her. Zudem ist es für die Darstellung der Ergebnisse und der Speicherung der Ergebnisse und sonstige Daten zuständig. Die Gesamten Benutzereingaben werden somit vom Programm an die einzelnen Module weitergeleitet. Im Testfall ist das Programm also der Datenlieferant und stellt alle Eingaben des Benutzers zur Verfügung. Darüberhinaus verwaltet das Programm die Oracle-Datenbank und liefert somit die anderen Daten, die nicht vom Benutzer eingegeben werden an die Module. Aus diesen Gründen sind alle unter E.3.3 bis E.3.2 aufgeführten Daten für den Testgegenstand Programm relevant.

E.4 Testfallspezifikation

Im Folgenden sollen die Werte des Testszenarions beschrieben werden.

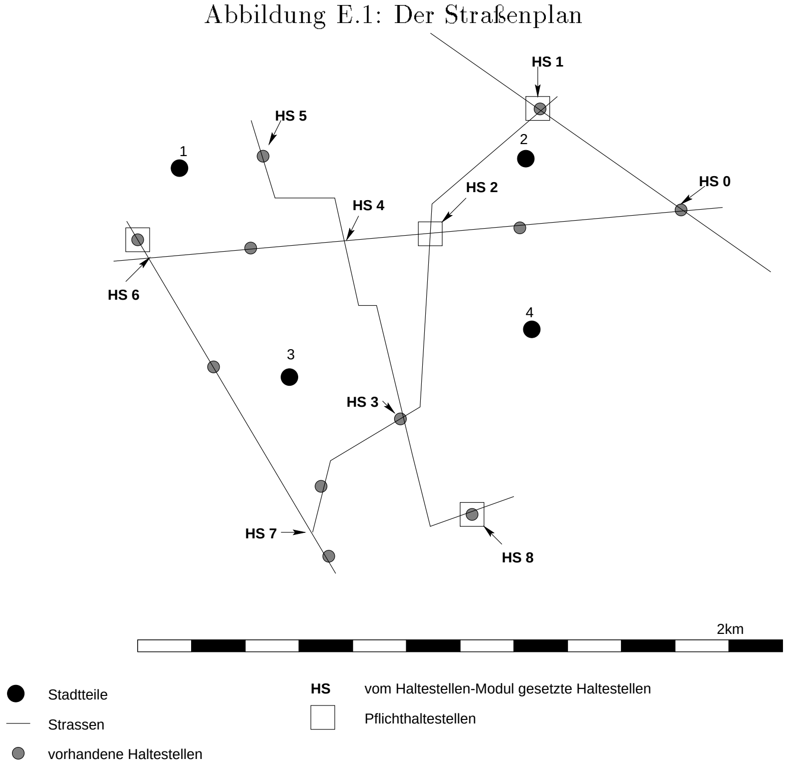
Der Straßenplan

Der Straßenplan besteht aus insgesamt fünf Straßen, die sowohl Kurvenverläufe, als auch Kreuzungen mit anderen Straßen beinhalten.(siehe Abbildung E.1)

Datenstruktur:

Liste von Listen, die Punkte enthalten. Dabei stellt eine Liste eine Straße dar und die Liste von Punkten die Koordinanten, die den Straßenverlauf in GRASS beschreiben:
list<list<CPunkt>>

Abbildung E.1: Der Straßenplan



Werte:

Die Pflichthaltestellen

Im Testszenario sind vier Pflichthaltestellen eingebunden worden und gleichmäßig über das Gebiet verteilt worden.

Abbildung E.2: Die Straßenplanliste

Datenstruktur:

Liste von Punkten, die die Haltestellenkoordinaten darstellen:
list<CPunkt>

Die vorgegebene Haltestellen

Im Testgebiet sind neun Haltestellen vorhanden. Sie sind durchnummeriert, beginnend bei 0.

Datenstruktur:

Liste von Punkten, die die Haltestellenkoordinante darstellen:
list<CPunkt>

Die Pendlermatrix

Die Pendlermatrix wurde willkürlich festgelegt. Die Werte beziehen sich auf die Pendlerbewegungen zwischen den Stadtteilen.

Datenstruktur:

Eine 4 - mal - 4 Matrix aus der LEDA-Bibliothek ([LEDA4.2]):
matrix

Werte:

	1	2	3	4
1	10	23	17	21
2	15	9	4	11
3	7	12	20	5
4	13	7	13	8

Die Pendlerliste:

Die Pendlerliste ist eng mit der Pendlermatrix verbunden und mappt die Indices der Matrix auf die Koordinanten des Geo - Informationssystems um.

Datenstruktur:

Eine Liste von Punkten, die die Koordinaten der Stadtteile darstellen:
list<CPunkt>

Werte:

Stadtteil	1	2	3	4
Koordinaten	2577500,5717150	2578400,5716950	2578400,5716950	2578200,5716400

Die Fahrtwunschmatrix

Die Fahrtwunschmatrix bezieht sich auf die elf schon im Testgebiet vorhandenen Haltestellen. Die Werte stellen die Anzahl Reisender von einer Haltestelle zu einer anderen dar.

Datenstruktur:

Eine 11 - mal - 11 Matrix aus der Leda - Bibliothek:
matrix

Werte:

Die Werte werden zufällig festgelegt. Dies ist möglich, da die neuen Haltestellen auch festgelegt sind (siehe unten) und somit hartkodiert weitergeleitet werden.

Die Fahrtwunschliste

Die Fahrtwunschliste hat dieselbe Aufgabe, wie die Pendlerliste oberhalb. Die Matrix-Indices der Fahrtwunschmatrix müssen auf das Koordinatensystem abgebildet werden.

Datenstruktur:

Eine Liste von Punkten , die Koordinanten darstellen:
list<CPunkt>

Werte:

Haltestelle	1	2	3	4
Koordinaten	2577050,5716850	2577500,5716800	2577600,5717100	2577800,5717150
Haltestelle	5	6	7	8
Koordinaten	2578200,5716800	2578850,5716800	2577900,5716450	2577200,5716500
Haltestelle	9	10	11	
Koordinanten	2577450,5716250	2578250,5716150	2577400,5716050	

Die Bevölkerungsdichtenliste

Die Bevölkerungsdichte bezieht sich auf der feinsten Ebene maximal auf die Stadtteile. D.h. für jeden Stadtteil muß ein Bevölkerungsdichte - Wert vorhanden sein.

Datenstruktur:

Eine Liste von double Werten, die die Bevölkerungsdichte angeben.
list<double>

Werte:

Die Werte werden auch hier, analog zu den Werten der Fahrtwunschmatrix, zufällig erzeugt. Die Bevölkerungsdichte ist bei der Haltestellenberechnung relevant, da aber die Haltestellen in diesem Systemtest hartkodiert sind, ist nur die Übergabe zu kontrollieren.

Die Toleranzgrenze

Der Wert, den ein Reisender maximal bereit ist zu einer Haltestelle zu laufen.

Werte:

double, 300 Meter

Das maximale Einzugsgebiet einer Haltestelle

Das Einzugsgebiet, das eine Haltestelle maximal abdeckt.

Wert:

double, 200 Meter

Das minimale Einzugsgebiet einer Haltestelle

Das Einzugesgebiet, das eine Haltestelle mindestens abdeckt.

Wert:

double, 450 Meter

Verbindungsradius

Der Radius, in dem die Haltestellen durch kürzeste Wege miteinander verbunden werden.

Wert:

double, 600 Meter

Die neu gesetzten Haltestellen

Die Haltestellen, die vom entsprechenden Modul gesetzt werden, wurden für das Testszenario festgelegt. Dabei muß darauf geachtet werden, dass die Pflichthaltestellen gemäß der Anforderungen unter den neu gesetzten Haltestellen enthalten sind.

Datenstruktur:

Die Ergebnisrückgabe des Moduls Haltestellen geschieht durch übergabe des Wegenetzgraphen. In den Knoten sind dann die Haltestellenkoordinaten enthalten, sowie die Haltestellen-ID:
CWegenetzGraph
(siehe auch Der Wegenetzgraph)

Werte:

Haltestelle	0	1	2	3	4	5	6	7	8
Koordianten	2578850,5716800	2577800,5717150	2577950,5716800	2577900,5716450	2577750,5716800	2577600,5717100	2577150,5716750	2577300,5716200	2578250,5716150

Der Wegenetzgraph

Der Wegenetzgraph wird vom Haltestellen-Modul aufgebaut und vom Linienplan für seine Berechnungen gebraucht.

Datenstruktur:

Der Wegenetzgraph besteht aus CWegenetzknoten und CWegenetzkanten. Die Knoten enthalten die Haltestellen - Koordinaten und die Haltestellen - ID, sowie eine Priorität. An den Kanten stehen die Entfernungen in Form von Meterangaben, sowie eine Liste von Listen von Punkten, die den Straßenverlauf durch Koordinaten wiederspiegeln.

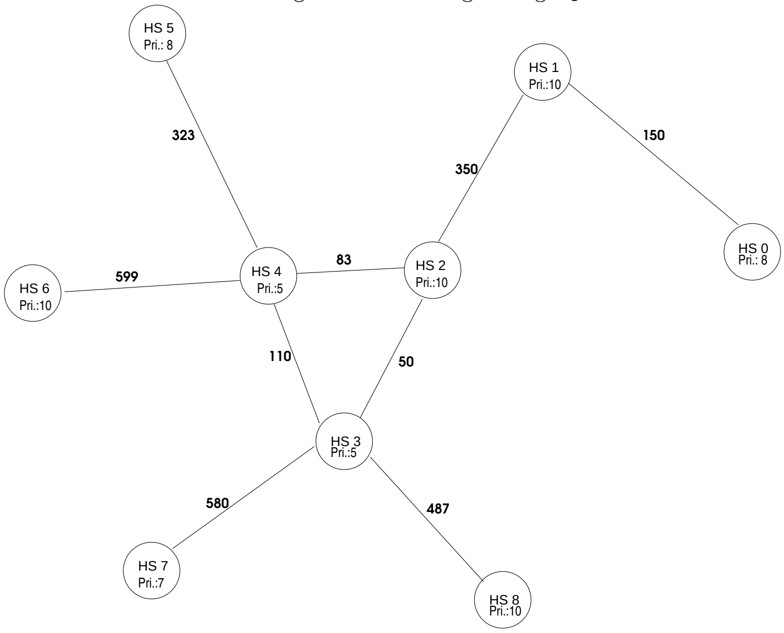
Werte:

Siehe E.3

E.5 Testfälle 1 bis 3

Zu diesen Testfällen existiert ein externes Dokument “Testfall 1 - 3”

Abbildung E.3: Der Wegenetzgraph



E.6 Testfälle 4 und 5

Zu diesen Testfällen existiert ein externes Dokument “Testfall 4 und 5

E.7 Testszenarien

E.7.1 Testszenario 1

Der Testlauf wurde abgebrochen

E.7.2 Testszenario 2

Der Testlauf wurde abgebrochen

E.7.3 Testszenario 3

Der Testlauf wurde abgebrochen

E.7.4 Testszenario 4

Testziel

Das Ziel dieses Tests ist es einen Wegenetzgraphen zu erzeugen, der ein minialer Spannbaum ist. Dazu wird der minimale Verbindungsradius auf 1 gesetzt.

Testgebiet

Herne

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 1000
- max. Einzugsgebiet: 1200
- Verbindungsradius: 1
- Entfernungsbestimmung: Luftlinie
- Haltestellenlage: kreuzungsnah
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 10

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 3000
- max Linienanzahl: 4

Fahrplan

- max Fahrzeuganzahl: 0
- Mindesttakt: 10
- Mindesthaltezeit: 1
- Mindestumsteigezeit: 0

Ergebnisse des Durchlaufes

Haltestellen

- Die Zufallszahl ist 991398752.
- Alle Knotenverbindungen liegen auf den Straßen und der Graph ist zusammenhängend.
- Es wurden 46 Haltestellen gesetzt.

Linienplan

ABSTURZ: Bei Berechnen des Linieplans stürzt das Programm mit einem Oracle - Fehler ab:

```
[DEBUG:CDBSchnittstelle.cpp:12905]
    [MetFund] speichernGuetemasseLinienplan
Oracle Fehler: Fehler in speichernGuetemasseLinienplan
ORA-01426: numeric overflow
```

E.7.5 Testszenario 5

Testziel

Das Ziel dieses Tests ist es, einen Komplettdurchlauf auf den gesamten Herne-Daten durchzuführen.

Testgebiet

Herne

Programmversion

3.7.2001, 15Uhr 50.

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 150
- max. Einzugsgebiet: 450
- Verbindungsradius: 1000
- Entfernungsbestimmung: Luftlinie
- Haltestellenlage: 0,5
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 10

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 3000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja

- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Fahrplan

- Mindesttakt: 15
- Mindesthaltezeit: 0
- Mindestumsteigezeit: 0

Ergebnisse des Durchlaufes

Haltestellen

- Die Zufallszahl ist 994168162.
- Alle Knotenverbindungen liegen auf den Straßen und der Graph ist zusammenhängend.
- Es wurden 412 Haltestellen gesetzt.

Linienplan

Die Berechnung stürzt bei der Basislinienwahl ab! Eine nähere Beschreibung befindet sich unter Anhang F, Fehler Nummer 26.

E.7.6 Testszenario 6

Testziel

Das Programm soll einen Gesamtdurchlauf auf den Herne-Daten durchführen.

Testgebiet

Herne

Programmversion

6.7.2001, 11:20 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 150
- max. Einzugsgebiet: 450

- Verbindungsradius: 1500
- Entfernungsbestimmung: Luftlinie
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 10

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 30000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Fahrplan

- Mindesttakt: 15
- Mindesthaltezeit: 0
- Mindestumsteigezeit: 2

Ergebnisse des Durchlaufes

Haltestellen

- Die Zufallszahl ist 994411754.
- Es wurden 409 Haltestellen gesetzt.
- Der Wegenetzgraph enthält 2824 Kanten.
- Laufzeit: ca. 40 Minuten.

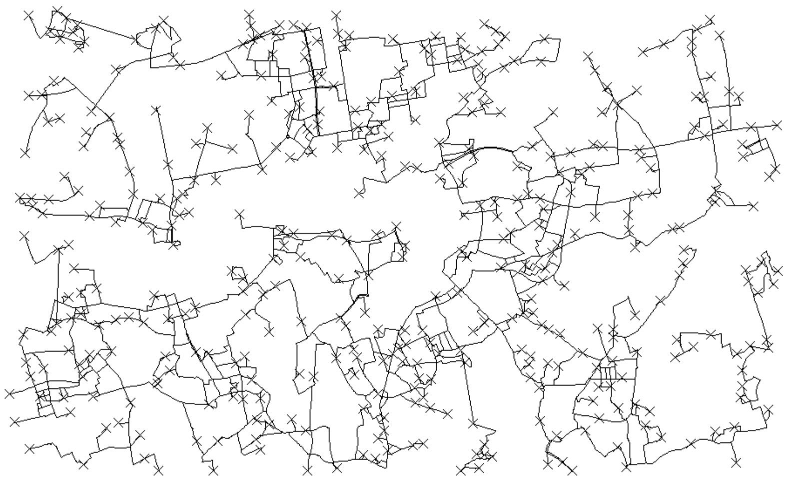
Linienplan

- Laufzeit: 2 Stunden 10 Minuten

Fahrplan

- Laufzeit: ca. 50 Minuten (inklusive Gütemaße berechnen)

Abbildung E.4: Der Wegenetzgraph des Testszenarios 6
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



E.7.7 Testszenario 7

Testziel

Das Programm soll einen Gesamtdurchlauf auf den Herne-Daten durchführen. (Wurde im Batch-Modus durchgeführt)

Testgebiet

Herne

Programmversion

10.7.2001, 14:25 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 800
- Verbindungsradius: 2000
- Entfernungsbestimmung: Luftlinie
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5

- Bevölkerungsraster: 10

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 30000
- Algorithmus: 1
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Fahrplan

- Mindesttakt: 15
- Mindesthaltezeit: 0
- Mindestumsteigezeit: 2

Ergebnisse des Durchlaufes

Haltestellen

- Der Wegenetzgraph besteht aus 272 Haltestellen die durch 1230 Kanten verbunden sind.

Linienplan

- Es wurden 278 Linien berechnet.
- Die Streckenkilometer des Linienplans betragen 2698,983 km.
- Es wurden 2064 Kanten gezählt.
- Die Anzahl der Umstiege insgesamt beträgt 121.536.
- Die Anzahl der Verbindungen mit Umsteigezwang liegt bei 35.099.
- Die Anzahl der Verbindungen insgesamt beträgt 73712.
- Die Reisezeit ohne Umstiege liegt bei 1.489.918 Minuten.
- Die Reisezeit mit Umstiegen hat eine Dauer von 1.854.526 Minuten.
- Die Anzahl der Gesamtfahrer ist 278.863.
- Die Anzahl der Direktfahrer ist 15.360.

- Die mittlere Reisezeit ist 25 Minuten.
- 47,6164 % aller Verbindungen sind keine Direktverbindungen.
- Der Direktfahreranteil ist 5,50808%.
- Die mittlere Mindestreisezeit ist 20 Minuten.
- Gesamte Streckenkilometer des betrachteten Gebietes betragen 1668,809 km.
- Die Reisezeitverlängerung beträgt 1.25 Minuten.
- Der Verknüpfungsgrad ist 11076.
- Der mittlerer Verknüpfungsgrad beträgt 14,3833%.

Von diesem Testfall existiert kein Fahrplan-Durchlauf, da die gesamten Daten nach dem Restart des Datenbank-Servers verlorengegangen sind.

E.7.8 Testszenario 8

Testziel

Mit diesem Test soll verstärkt die Berechnung der Gütemaße überprüft werden.

Testgebiet

Herne-Ausschnitt

Programmversion

10.7.2001, 14:45 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 800
- Verbindungsradius: 2000
- Entfernungsbestimmung: Luftlinie
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 10

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 30000
- Algorithmus: 1
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Fahrplan

- Mindesttakt: 15
- Mindesthaltezeit: 0
- Mindestumsteigezeit: 2

Ergebnisse des Durchlaufes

Von diesem Testfall existieren keine Ergebnisdaten, da die gesamten Daten nach dem Restart des Datenbank-Servers verlorengegangen sind.

E.7.9 Testszenario 10

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

11.7.2001, 18:00 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 250
- max. Einzugsgebiet: 650
- Verbindungsradius: 1200
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

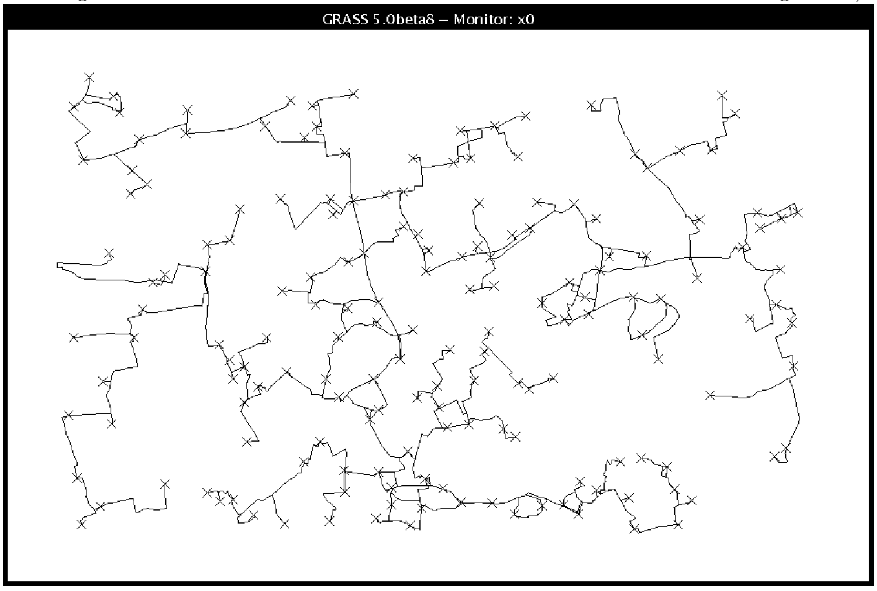
- max Gesamtlinienlänge: 120000
- max Linienlänge: 30000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 171 Haltestellen gesetzt.

Abbildung E.5: Der Wegenetzgraph des Testszenarios 10 (Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Linienplan

- Für den Linienplan wurde die Programmversion vom 13.7.2001, 15:00 Uhr genommen, da die voherige Version mit dem Algorithmus 2 eine Haltestelle nicht angebunden hatte.
- Es wurden 68 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 34 Minuten.
- Alle Haltestellen sind nun angebunden.
- Es wurden 66.636 Umstieg berechnet. Damit beträgt die durchschnittlich Anzahl der Umstiege 2,29.
- Der mittlerer Verknüpfungsgrad ist 6,47.

Fahrplan

- Der Linienplan kann mit 233 Bussen gefahren werden.
- Die netzbedingte Wartezeit liegt bei 5214 Minuten.
- Der Fahrplanwirkungsgrad beträgt 77,71%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 66,52 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 3,51.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 2,67.

E.7.10 Testszenario 11

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

11.7.2001, 18:00 Uhr

Parameter

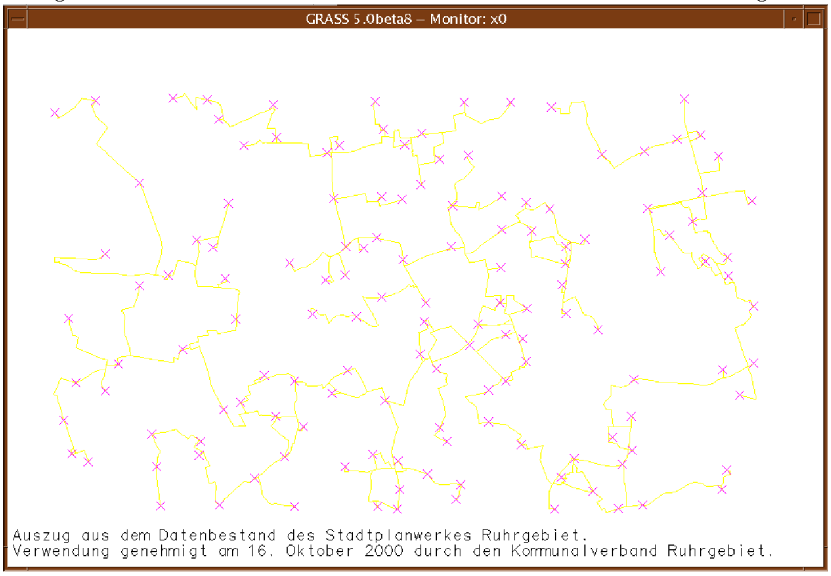
Haltestellen

- Algorithmus: Stochastisch mit Umgebungsprüfung
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 700
- Verbindungsradius: 1500
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

- max Gesamtlinienlänge: 100000
- max Linienlänge: 45000
- Algorithmus: 1
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Nein
- Linienendverknüpfung: Ja

Abbildung E.6: Der Wegenetzgraph des Testszenarios 11
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 140 Haltestellen gesetzt.

Linienplan

- Es sind drei Haltestellen nicht angebunden worden!
- Es wurden 67 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 38 Minuten.
- Es wurden 40975 Umstieg berechnet. Damit beträgt die durchschnittlich Anzahl der Umstiege 2,2.
- Der mittlerer Verknüpfungsgrad ist 10,6.
- Direktfahreranteil liegt bei 6,99%.

Fahrplan

- Der Linienplan kann mit 254 Bussen gefahren werden.
- Die netzbedingte Wartezeit liegt bei 6123 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 80,76%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 68,46 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 3,97.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 1,87.

E.7.11 Testszenario 12

Der Testlauf wurde abgebrochen

E.7.12 Testszenario 13

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Im Vordergrund steht der Test des Evolutionären Algorithmus mit der Plus - Strategie.

Testgebiet

Herne

Programmversion

12.7.2001, 18:00 Uhr

Parameter

Haltestellen

- Algorithmus: Evolutionär
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 750
- Verbindungsradius: 1750
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30
- Strategie: Plus
- Eltern: 3
- Kinder: 10
- Generationen: 140
- Güteunterschreitung: Ja - 10.000

Linienplan

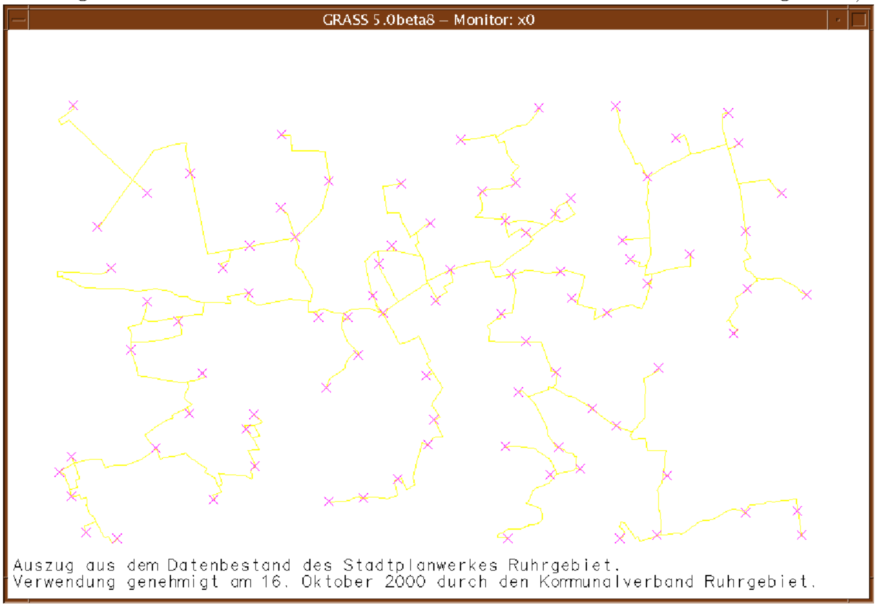
- max Gesamtlinienlänge: 80000
- max Linienlänge: 50.000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 89 Haltestellen erzeugt.

Abbildung E.7: Der Wegenetzgraph des Testszenarios 13
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Linienplan

- 30 Linien wurden erzeugt.
- Die mittlere Mindestreisezeit liegt bei 32 Minuten.
- Alle Haltestellen sind angebunden.
- Es wurden 8856 Umstieg berechnet. Damit beträgt die durchschnittlich Anzahl der Umstiege 1,13.
- Der mittlerer Verknüpfungsgrad ist 10,8.
- Direktfahreranteil liegt bei 9,83%.

Fahrplan

- Der Linienplan kann mit 133 Bussen gefahren werden.
- Die netzbedingte Wartezeit liegt bei 1371 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 84,1 %.

Globale Gütemaße

- Die durchschnittliche Anzahl der Busse beträgt 4,33.
- Die mittlere Reisezeit liegt bei 53,9 Minuten.
- Durchschnittlich befinden sich 2,64 Personen in einem Bus.

E.7.13 Testszenario 14

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Es soll hierbei ein minimaler Spannbaum erzeugt werden. Dies geschieht über den Parameter *Verbindungsradius*, der bei einem Wert von 1 einen minimalen Spannbaum berechnet.

Testgebiet

Herne

Programmversion

12.7.2001, 18:00 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch mit Umgebungsprüfung
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 250
- max. Einzugsgebiet: 650
- Verbindungsradius: 1
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

- max Gesamtlinienlänge: 8000
- max Linienlänge: 35000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

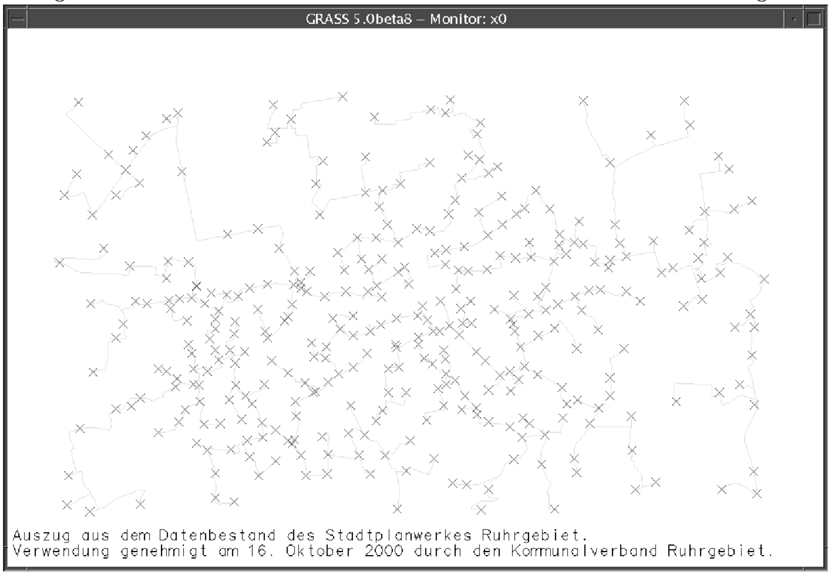
Haltestellen

- Es wurden 355 Haltestellen erzeugt.

Linienplan

- 82 Linien wurden erzeugt.
- Die mittlere Mindestreisezeit liegt bei 35 Minuten.
- Zwei Haltestellen wurden nicht angebunden.

Abbildung E.8: Der Wegenetzgraph des Testszenarios 14
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Fahrplan

- Auf dem Linienplan müssen 215 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 1725 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 68,19 %.

Globale Gütemaße

- Die Anzahl der Busse pro Linie beträgt 2,6.
- Die mittlere Reisezeit liegt bei 36,9 Minuten.
- Durchschnittlich befinden sich 24,86 Personen in einem Bus.

E.7.14 Testszenario 15

Der Testlauf wurde abgebrochen

E.7.15 Testszenario 16

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

12.7.2001, 16:30 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch mit Umgebungsprüfung
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 333
- max. Einzugsgebiet: 777
- Verbindungsradius: 2500
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 29

Linienplan

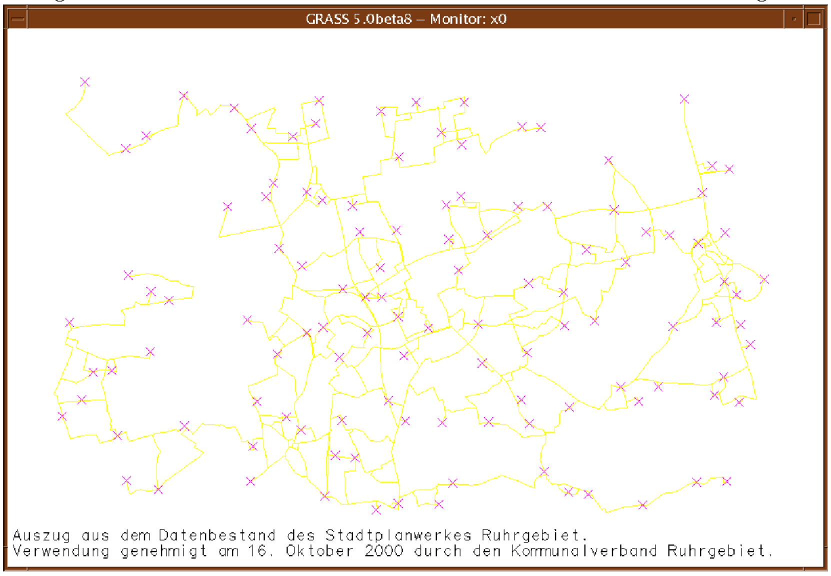
- max Gesamtlinienlänge: 8000
- max Linienlänge: 31113
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 118 Haltestellen gesetzt.

Abbildung E.9: Der Wegenetzgraph des Testszenarios 16
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Linienplan

- Es wurden 61 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 22 Minuten.
- Alle Haltestellen sind nun angebunden.
- Die Anzahl der Umstiege beträgt 18802. Damit ergibt sich eine durchschnittlich Anzahl der Umstiege von 6430.
- Der mittlerer Verknüpfungsgrad ist 17,51.
- Direktfahreranteil liegt bei 6,87%.

Fahrplan

- Auf diesem Linienplan müssen 345 Busse eingesetzt werden.
- Die netzbedingte Wartezeit liegt bei 14411 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 88,87 %.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 41,35 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 6,56.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 3,04.

E.7.16 Testszenario 17

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

12.7.2001, 16:30 Uhr

Parameter

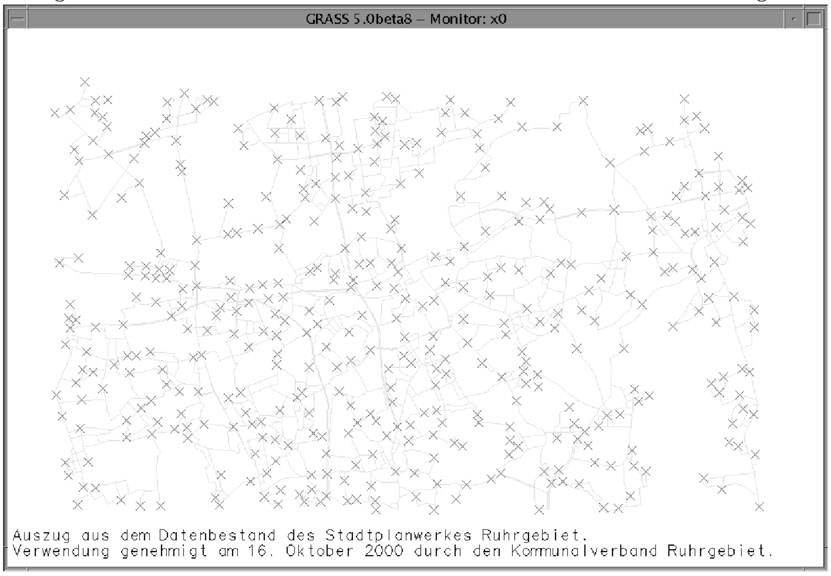
Haltestellen

- Algorithmus: Stochastisch mit Umgebungsprüfung
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 150
- max. Einzugsgebiet: 450
- Verbindungsradius: 2000
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 40

Linienplan

- max Gesamtlinienlänge: 80
- max Linienlänge: 25000
- Algorithmus: 1
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Abbildung E.10: Der Wegenetzgraph des Testszenarios 17 (Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 421 Haltestellen gesetzt.

Linienplan

- Es wurden 243 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 19 Minuten.
- Alle Haltestellen sind angebunden.
- Die Anzahl der Umstiege beträgt 366736.
- Der mittlerer Verknüpfungsgrad ist 10,33.
- Direktfahreranteil liegt bei 2,23%.

Fahrplan

- Auf diesem Linienplan müssen 1252 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 168839 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 87,51%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 44,02 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 5,15.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 0,93.

E.7.17 Testszenario 18

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

12.7.2001, 16:30 Uhr

Parameter

Haltestellen

- Algorithmus: Evolutionär
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 600
- Verbindungsradius: 1200
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

- max Gesamtlinienlänge: 80000
- max Linienlänge: 37582
- Algorithmus: 1
- Geschwindigkeit: 17 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja

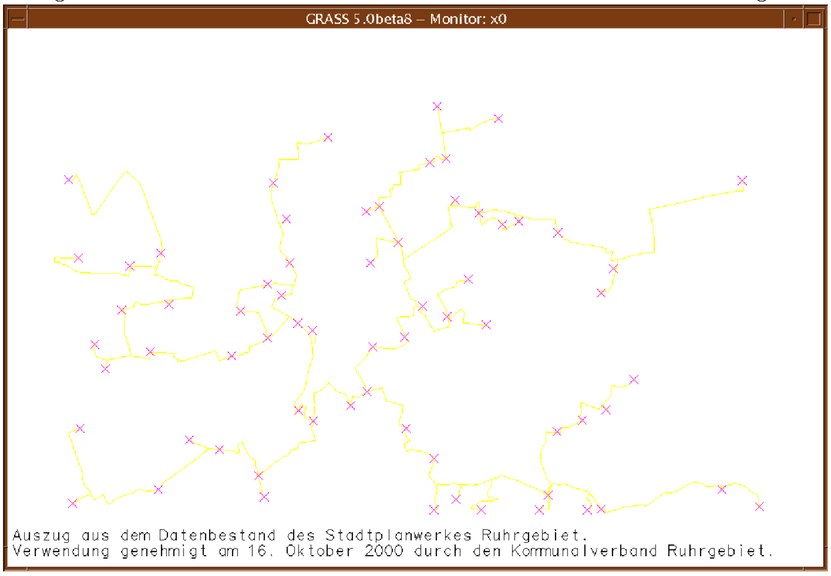
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 68 Haltestellen gesetzt.

Abbildung E.11: Der Wegenetzgraph des Testszenarios 18
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Linienplan

- Es wurden 20 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 27 Minuten.
- Alle Haltestellen sind angebunden.
- Die Anzahl der Umstiege liegt bei 4062. Damit ergeben sich durchschnittlich 1940 Umstiege.
- Der mittlerer Verknüpfungsgrad ist 18,16.
- Direktfahreranteil liegt bei 14,85%.

Fahrplan

- Auf dem berechneten Linienplan müssen 87 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 784 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 85,67%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 58 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 4,35.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 3.

E.7.18 Testszenario 19

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

12.7.2001, 16:30 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 300
- max. Einzugsgebiet: 800
- Verbindungsradius: 1
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

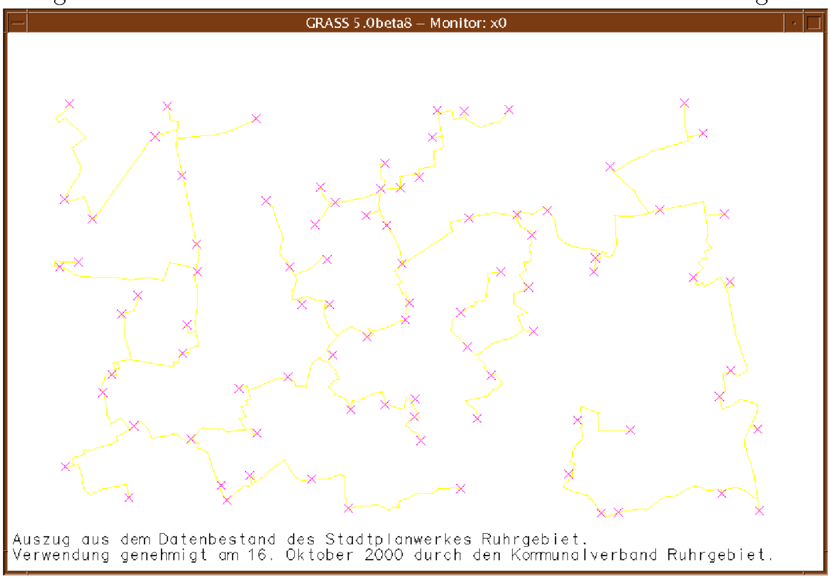
- max Gesamtlinienlänge: 8000
- max Linienlänge: 45000
- Algorithmus: 1
- Geschwindigkeit: 17 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

Haltestellen

- Es wurden 88 Haltestellen gesetzt.

Abbildung E.12: Der Wegenetzgraph des Testszenarios 19 (Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Linienplan

- Es wurden 18 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 36 Minuten.
- Alle Haltestellen sind nun angebunden.
- Die Anzahl der Umstiege liegt bei 8346. Damit ergeben sich durchschnittlich 1,09 Umstiege.
- Der mittlerer Verknüpfungsgrad ist 15,0.
- Direktfahreranteil liegt bei 14,36%.

Fahrplan

- Auf dem berechneten Linienplan müssen 88 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 405 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 89,9%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 65 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 4,89.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 4,36.

E.7.19 Testszenario 20

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund. Es sollen die in Herne real existiernende Haltestellen als Berechnungsgrundlage dienen. Dies soll verstärkt die Berechnung der Gütemaße testen.

Testgebiet

Herne

Programmversion

11.7.2001, 18:00 Uhr

Parameter

Haltestellen

- Algorithmus: Pflichthaltestellen
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 250
- max. Einzugsgebiet: 655
- Verbindungsradius: 1400
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

- max Gesamtlinienlänge: 80
- max Linienlänge: 60.000
- Algorithmus: 2
- Geschwindigkeit: 17 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

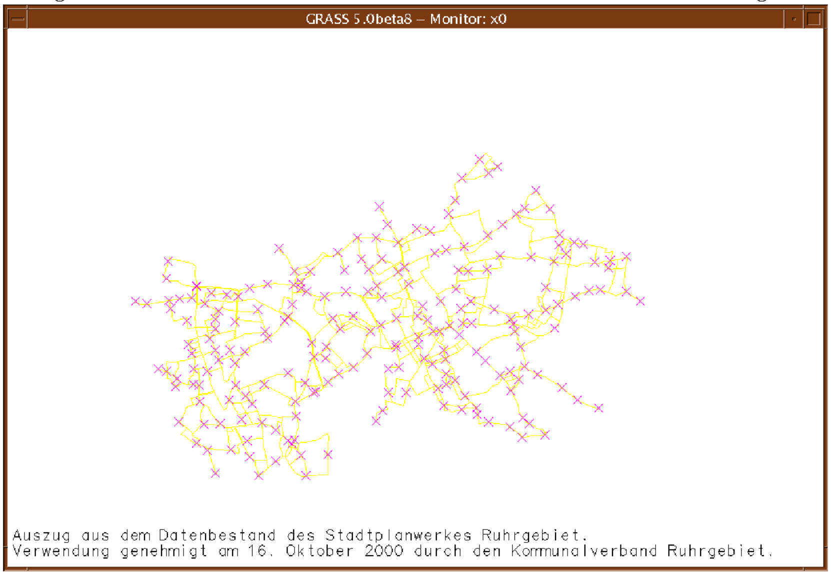
Haltestellen

- Es wurden 216 Haltestellen übernommen

Linienplan

- Die Linienanzahl beträgt 98
- Die mittlere Mindestreisezeit liegt bei 10 Minuten.
- Alle Haltestellen sind nun angebunden.
- Die Anzahl der Umstiege liegt bei 77295. Damit ergeben sich durchschnittlich 22285 Umstiege.
- Der mittlerer Verknüpfungsgrad ist 10,79.

Abbildung E.13: Der Wegenetzgraph des Testszenarios 20 (Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Fahrplan

- Auf dem berechneten Linienplan müssen 305 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 23189 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 72,27%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 34,21 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 3,11.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 2,1.

E.7.20 Testszenario 21

Testziel

Es wird ein Gesamtdurchlauf auf den Herne-Daten angestrebt. Ein Erzeugen von Daten für die Präsentation steht dabei im Vordergrund.

Testgebiet

Herne

Programmversion

16.7.2001, 10:00 Uhr

Parameter

Haltestellen

- Algorithmus: Stochastisch
- Zufallsgeneratorinitialisierung: -1
- min. Einzugsgebiet: 200
- max. Einzugsgebiet: 550
- Verbindungsradius: 1250
- Entfernungsbestimmung: Realweg
- Haltestellenlage: 1,0 (kreuzungsnah)
- Toleranzgrenze: 0,5
- Bevölkerungsraster: 30

Linienplan

- max Gesamtlinienlänge: 120000
- max Linienlänge: 40000
- Algorithmus: 2
- Geschwindigkeit: 15 km/h
- Basislinienwahl: Ja
- Eindeutige Linienverknüpfung: Ja
- Mehrdeutige Linienverknüpfung: Ja
- Aufbruchverknüpfung: Ja
- Linienendverknüpfung: Ja

Ergebnisse des Durchlaufes

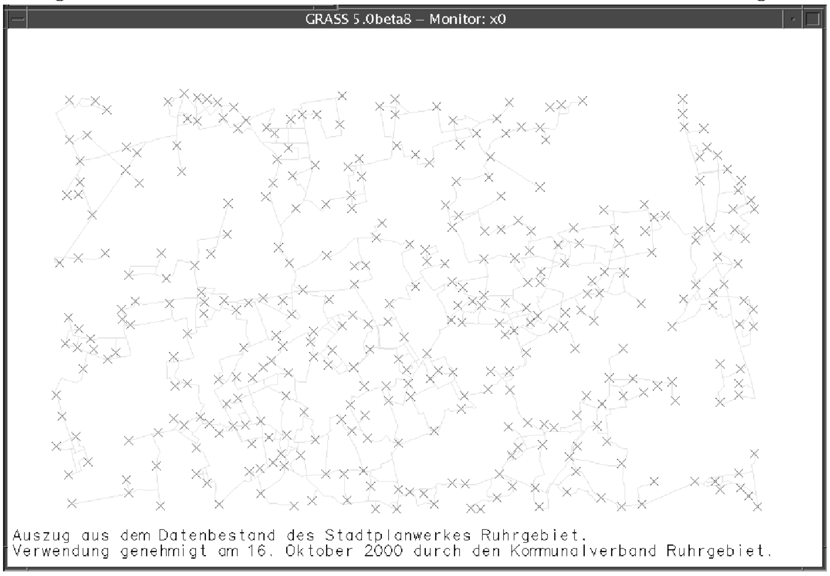
Haltestellen

- Es wurden 357 Haltestellen gesetzt.

Linienplan

- Es wurden 128 Linien berechnet.
- Die mittlere Mindestreisezeit liegt bei 24 Minuten.
- Alle Haltestellen sind nun angebunden.
- Die Anzahl der Umstiege liegt bei 364.120. Damit ergeben sich durchschnittlich 2,87 Umstiege.
- Der mittlerer Verknüpfungsgrad ist 6,06.

Abbildung E.14: Der Wegenetzgraph des Testszenarios 21
(Auszug aus dem Datenbestand des Stadtplanwerkes Ruhrgebiet. Verwendung genehmigt am 16. Oktober 2000 durch den Kommunalverband Ruhrgebiet.)



Fahrplan

- Auf dem berechneten Linienplan müssen 630 Busse eingesetzt werden.
- Die netzbedingte Wartezeit beträgt 47679 Minuten.
- Der Fahrplanwirkungsgrad liegt bei 0,17%.

Globale Gütemaße

- Die mittlere Reisezeit liegt bei 58,20 Minuten.
- Die durchschnittliche Anzahl der Busse beträgt 3,82.
- Die durchschnittliche Anzahl der Personen pro Bus liegt bei 12,34.

Anhang F

Fehlerdokument

Die folgende Auflistung der Fehler soll die Vorgehensweise der Projektgruppe bei der Fehlerbehebung dokumentieren. Sie ist dynamisch während der Implementierungs- und Testphase entstanden. Deshalb ist dieser Anhang umgangsprachlich gehalten und eventuell formal nicht vollkommen korrekt.

FehlerNr. 001

Beschreibung: Startet man den Linienplan erneut, nachdem er einmal durchgelaufen ist, stürzt dieser ab. (siehe Testszenario1)

Zuständigkeitsbereich: Linienplan

Verantwortlicher: Michael Schlüter

Status: Behoben. (Static Variablen in Methoden werden beim Löschen eines Objektes nicht gelöscht. Initialisierung eines Arrays in einem neuem Objekt wurde deshalb unterdrückt.)

FehlerNr. 002

Beschreibung: Der Linienplan beachtet die Einstellung max. Linienanzahl nicht

Zuständigkeitsbereich: Linienplan

Verantwortlicher: Reza

Status: Erledigt! (Der Parameter wird durch Helge aus den Einstellungen entfernt.)

FehlerNr. 003

Beschreibung: Die Fortschrittsanzeige geht beim Berechnen des Simplex-Tableaus nur bis 99,9%, obwohl die Berechnung beendet ist (siehe Testszenario 2).

Zuständigkeitsbereich: Fahrplan

Verantwortlicher: Michael Wintergerst

Status: Behoben.

FehlerNr. 004

Beschreibung: Die Anzahl benötigter Fahrzeuge ist 783 auf 27 Linien mit 57 Haltestellen (siehe Testszenario 2) .

Zuständigkeitsbereich: Fahrplan

Verantwortlicher: Frank

Status: Problem bekannt, Algorithmus wird neu entwickelt. Entwicklung abgeschlossen

FehlerNr. 005

Beschreibung: Das Linien - Fahrplan anzeigen dauert sehr lange und belegt die CPU zu 95% (siehe Testszenario 2) .

Zuständigkeitsbereich: Programm

Verantwortlicher: Helge

Status: Behoben durch schnelleren Algorithmus.

FehlerNr. 006

Beschreibung: Das Haltestellen-Modul stürzt bei der Wegenetzgraph-Erzeugung ab (siehe Testszenario 3)

Zuständigkeitsbereich: Haltestellen

Verantwortlicher: Marc

Status: Fehler durch Vergösserung der Sektoren (indirekt) behoben.

FehlerNr. 007

Beschreibung: Die Linieplan-Berechnung stürzt ab:

```
[DEBUG:CDBSchnittstelle.cpp:12905]
    [MetFund] speichernGuetemasseLinienplan
Oracle Fehler: Fehler in speichernGuetemasseLinienplan
    ORA-01426: numeric overflow
```

Nachtrag: Jin-Ha: Der Absturz ist auf einen Fehler im Wegenetzgraphen zurückzuführen. Alle Kanten haben Entfernung 0, wenn man den Verbindungsradius auf 1 stellt. Das ergibt 0-Werte bei einigen Gütemaßen im Linienplan, die wiederum in einem Quotienten unendlich ergeben. Wahrscheinlich ist die Datenbank nicht in der Lage, dieses abzuspeichern.

Zuständigkeitsbereich: Linienplan ->Haltestellen

Verantwortlicher: Jin-Ha->Marc

Status: Erledigt.

FehlerNr. 008

Beschreibung: Bei Verbindungsradius 1 in den Einstellungen des Haltestellenmoduls wird ein Wegenetzgraph erstellt, der an den Kanten die Entfernung 0
trägt. Dies führt später zu einem Absturz im Linienplanmodul, da die Entfernung dort unbedingt größer als 0 sein muss. Für nähere Details
(siehe Fehler 7)

Zuständigkeitsbereich: Haltestelle

Verantwortlicher: Marc

Status: Fehler durch explizites Setzen der Entfernungen beim Zusammenhängen des Wegenetzgraphen behoben.

FehlerNr. 009

```
Beschreibung
[DEBUG:CGUI.et:625] [MetHilf] Lege Quelle-Ziel-Matrix an
Bus error
```

Tritt unregelmäßig auf und ist nicht reproduzierbar.

Zuständigkeitsbereich: Programm / GUI

Verantwortlicher: Helge

Status: Bekannt

FehlerNr 010

Beschreibung: Die Ausgabe des Fahrplans stimmen nicht (Testfall 1). Lässt man diesen Fahrplan anzeigen, so stellt man fest, dass Linie -1 über andere
Haltestellen als Linie 1 führt (ebenso alle weiteren Linienpaare). Die negativen und positiven Linien müssten doch über die gleichen Haltestellen fahren (1 als Hinfahrt und -1 als Rückfahrt).

Zuständigkeitsbereich: Fahrplan

Verantwortlicher: Mike

Status: Ist kein Fehler, sondern eine Fehlinterpretation der Ausgabe.

FehlerNr. 011

Beschreibung: Das Haltestellen-Modul ist durchgelaufen. Bei Einsammeln des Threads stürtzt das Programm ab:

```
HS: Bewertung = 529.474312
HS: Verkehrsbedarfsmatrixerstellung
CKontrolle: MODUL HALTESTELLEN hat waehrend Laufzeit
    UNBEKANNTEN FEHLER GEWORFEN!!!!!!
```

```
Haltestelle beendet.
[DEBUG:CKontrolle.cpp:595]

    [MetFund] CKontrolle: Haltestellen-Thread wurde gestoppt
Segmentation fault
```

Die Zufallszahl ist 991920275. Bei einem sofortigen Neustart tritt der Fehler erneut auf

Zuständigkeitsbereich: Haltestelle

Verantwortlicher: Marc

Status: Gütemaße-Klasse bisher nicht vorhanden und daher Aufrufe selbiger bis auf weiteres entfernt.

FehlerNr 012

Beschreibung: Wenn das Programm mit Testfall 2 gestartet wird, stürzt es bei der Linienplanber echnung mit einem LEDA-Error ab:

```
Speichere QuelleZiel-Matrix in der Datenbank...
[DEBUG:CTestklasse2.cpp:468]
    [MetFund] Testdaten von den Haltestellen
    als Grundlage des Linienplans gespeichert
LEDA ERROR HANDLER new_edge(v,w): v not in graph
```

Zuständigkeitsbereich: Tester / Linienplan

Verantwortlicher: Tim oder Jörg oder JinHa

Status: Erledigt.

FehlerNr. 013

Beschreibung: Absturz des Haltestellen - Moduls auf Herne mit den Einstellungen aus Testszenario 4

```
[DEBUG:CParameterHaltestelle.cpp:6]
    [Objekt ] [CParameterHaltestelle]
[DEBUG:CStrassennetzStochastischUmgebung.cpp:16]
    [Objekt ] CStrassennetzStochastischUmgebung()
Segmentation fault
```

Zuständigkeitsbereich: Haltestellen

Verantwortlicher: Marc

Status: Fehler durch implizites Setzen von Fortschritt im Konstruktor behoben.

FehlerNr. 014

Beschreibung: Absturz des Linienplan-Moduls auf Herne und den Testfällen direkt nach der Abschlussverknüpfung

```
[DEBUG:CAbschlussverknuepfung.cpp:749]

    [MetFund] CAbschlussverknuepfung::durchfuehrenAufbruchsverknuepfung()
[DEBUG:CAbschlussverknuepfung.cpp:669]

    [Objekt ] CAbschlussverknuepfung::erstellenAllerEndBasislinien()
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :1 8 6 4 3 0
[DEBUG:CAbschlussverknuepfung.cpp:697]
    [MetFund] Linie :1
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :2 4 3 1
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :3 4 3 2
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :5 5 6 4
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :7 7 6 4
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :8 2 3 1
[DEBUG:CAbschlussverknuepfung.cpp:685]
    [MetFund] betrachtete Linie :9 1 3 0
[DEBUG:CAbschlussverknuepfung.cpp:741]
    [rest ] alle Moeglichen Basislinien erstellt.
Bus error
```

Zuständigkeitsbereich: Linienplan

Verantwortlicher: Reza

Status: Behoben.

FehlerNr. 015

Beschreibung: Das Fahrplan-Modul bekommt vomProgramm als für die Haltezeiten immer den Wert 0 übergeben

Zuständigkeitsbereich: Programm

Verantwortlicher: Tim

Status: Korrigiert, indem die Daten nun direkt bei der GUI abgeholt werden.

FehlerNr. 016

Beschreibung: Die Mindesthaltezeiten, die vom User eingegeben werden, stimmen nicht mit den Werten, die der Fahrplan bekommt, überein.

Zuständigkeitsbereich: Programm

Verantwortlicher: Tim oder Helge

Status: Korrigiert, indem die Daten nun direkt bei der GUI abgeholt werden.

FehlerNr. 017

Beschreibung: Wenn nach der Berechnung der Haltestellen der Menüpunkt „Alle Haltestellen zeigen" angewählt wird, erhält man einen leeren GRASS-Monitor. Zudem ist der angezeigte Maßstab 1:1000 km. Es befindet sich am Nullpunkt und am oberen Rand jeweils eine Haltestelle.

Zuständigkeitsbereich: Programm / GUI

Verantwortlicher: Helge

Status: Behoben! Nullpunkt-Haltestelle (Dummy-Haltestelle; benötigt für GRASS) wird ignoriert.

FehlerNr. 018

Beschreibung: Löschen der Projekte funktioniert nicht. Der Fehler scheint bei CFortschritt zu liegen, da die letzte Ausgabe

```
CFortschritt wird konstruiertist.
```

Zuständigkeitsbereich: GUI

Verantwortlicher: Alexander

Status: Behoben.

FehlerNr. 019

Beschreibung: Das Haltestellen-Modul stürzt mit der folgenden Meldung ab:

```
[DEBUG:CDBSchnittstelle.cpp:8892]

    [MetHilf] Lege urspruengliche Quelle-Ziel-Matrix an
[DEBUG:CDBSchnittstelle.cpp:14402]
    [MetHilf] Lege Pendler-Matrix an
[DEBUG:CDBSchnittstelle.cpp:14836]
    [MetHilf] Lege Pendlerliste an
[DEBUG:CParameterHaltestelle.cpp:6]
    [Objekt ] [CParameterHaltestelle]
[DEBUG:CKontrolle.cpp:500]
```

```
    [MetFund] HaltestellenMODUL erzeugt
[DEBUG:CStrassennetzStochastischUmgebung.cpp:16]
    [Objekt ] CStrassennetzStochastischUmgebung()
[DEBUG:CDBSchnittstelle.cpp:8625]
    [MetHilf] Lege Quelle-Ziel-Matrix an
[DEBUG:CStrassennetz.cpp:175]
    Zufallszahl = 993488245
[DEBUG:CStrassennetz.cpp:430]
    [AlgoStE] Pflichthaltestellen = 0
Segmentation fault
```

Zuständigkeitsbereich: Haltestellen

Verantwortlicher: Marc

Status: Gemeldet.

FehlerNr. 020

Beschreibung: Das Haltestelle-Modul ist beendet, und die Fortschrittsanzeige steht erst bei 75%

Zuständigkeitsbereich: Haltestellen oder GUI

Verantwortlicher: Alexander

Status: Behoben.

FehlerNr. 021

Beschreibung: Das Haltestellenmodul stürzt nach der Erstellung des Wegenetzgraphen mit folgender Fehlermeldung ab:

```
CStrassennetz.cpp:401:
    failed assertion ‘Is_Connected(*Wegenetzgraph)’
```

Zuständigkeitsbereich: Haltestellen

Verantwortlicher: Marc

Status: Behoben.

FehlerNr. 022

Beschreibung: Das Haltestellenmodul stürzt beim Speichern der Gütemaße ab.

Zuständigkeitsbereich: Gütemaße

Verantwortlicher: Jörg

Status: Behoben. (setzenQuelleZielMatrix wurde von den Haltestellen nicht aufgerufen.)

FehlerNr. 023

Beschreibung:

```
[DEBUG:CZusammenhangskomponenten.cpp:460]
    [AlgoStE] Erstellen des Fahrplans!
[DEBUG:CZusammenhangskomponenten.cpp:1033]
    [Objekt ] Aufbau des Umlaufgraphen
[DEBUG:CZusammenhangskomponenten.cpp:1056]
    [MetFund] Erstellen Knoten Fahrgastfahrten
[DEBUG:CZusammenhangskomponenten.cpp:1133]

    [MetFund] Kanten einfuegen Vergleich End- und Startknoten.
[DEBUG:CZusammenhangskomponenten.cpp:1199]
    [Objekt ] Fuellen der Arrays.
Bus error
```

Zuständigkeitsbereich: Fahrplan

Verantwortlicher: Michael Wintergerst

Status: Behoben

FehlerNr. 024

Beschreibung: Bei der Basislinienwahl kommt ein LEDA-Error:

```
array:: entry index out of range cout sagte:116:
CStartverknuepfung::waehlenBasislinienMichael2()
```

(siehe Anhang E.7, Testszenario 5)

Zuständigkeitsbereich: Linienplan

Verantwortlicher: Michael

Status: Behoben (vor dem Benutzen des Arrays wird getestet, dass es keine Haltestellennummer gibt, die größer als die Anzahl der Haltestellen ist).

FehlerNr. 025

Beschreibung: Es gibt Haltestellen, die durch den Linienplan mit keiner Linie versorgt werden

Zuständigkeitsbereich: Linienplan

Verantwortlicher: Michael

Status: Behoben.

Anhang G

Glossar

Arc Bezeichnung einer Vektorlinie unter GRASS. Ist definiert als eine Folge von Punkten.

Ausgangslösung Gültiger (wahrscheinlich) nicht optimaler Fahrplan, Ausgangsheuristik.

Basislinie Die Basislinie ist ein zentrales Element im Algorithmus von Sonntag, der im Linienplanmodul benutzt wird. Aufbauend auf den Basislinien werden die Linien zusammengesetzt. Eine Basislinie besteht aus zwei Kanten und den drei daran anliegenden Haltestellen.

Direkte Bindung Gibt an, dass an einer Haltestelle der Umstieg von einer angegebenen Linie zu einer anderen angegebenen Linie ohne/mit geringer Umstiegszeit verbunden sein soll.

Direktfahrt / Direktfahrer Ein Direktfahrer ist ein Fahrgast, der an seinem Ziel ankommt, ohne umzusteigen. Da ein Umstieg die Attraktivität einer Verbindung vermindert, es aber auf der anderen Seite unmöglich ist, für alle Verbindungen eine Direktfahrt zu verwirklichen, muss ein Kompromiss gefunden werden. Sonntag versucht in seinem Algorithmus den Anteil der Direktfahrer zu maximieren.

Einzugsgebiet einer Haltestelle Für das Einzugsgebiet einer Haltestelle existieren zwei entscheidende Werte: Das minimale und der maximale Einzugsgebiet einer Haltestelle. Dies sind die Werte, die angeben, welchen Bedarf eine Haltestelle minimal bzw. maximal befriedigt. Die Werte sind als zwei Radien im die Haltestelle zu interpretieren.

Fahrtwunschmatrix Siehe Quelle-Ziel-Matrix.

Fahrtwunschpunkt Ein Fahrtwunschpunkt ist ein Punkt, von den ausgehend oder auch hingehend Fahrtwünsche existieren. In den Eingabedaten sind dies die Haltestellen aus der Datenbasis. Bei den Ausgaben entspricht dies den berechneten Haltestellen samt den verwendeten Pflichthaltestellen.

Haltestellenpriorität Jede Haltestelle bekommt eine Priorität zugewiesen, die die Bedeutung dieser Haltestelle ausdrückt. Der Wertebereich reicht von 1 bis 10. Je höher dabei die Priorität, desto bedeutender ist die Haltestelle.

Layer Ein Layer wird vom GIS bereitgestellt. Es handelt sich dabei um eine Ebene, die eine Menge von geografischen Objekten aufnehmen kann. Die verschiedenen geografischen Elemente, wie Straßen und Flüsse, lassen sich so bei der grafischen Anzeige in einem Fenster übereinander legen.

Maximalgerüst Maximaler Spannbaum eines bewerteten Graphen.

Mindestumsteigezeit Mindestdauer, die ein Fahrgast benötigt, um an einer bestimmten Haltestelle die Linie zu wechseln.

Mindesthaltezeit Mindestdauer, die ein Bus an einer Haltestellen warten muss (zum Ein- und Auszeigen), bis er seine Fahrt wieder aufnehmen kann.

Netz der Transportverbindungen Multigraph der zwischen zwei Haltestellen so viele Kanten enthält wie dort Linien entlang fahren.

Netzbedingte Wartezeit Wartezeit die über die Mindestwartezeit und Mindestumsteigezeit hinausgeht.

Pendlermatrix Eine Matrix mit der Anzahl der Personen, die von einem Stadtteil in den anderen wollen.

Pflichthaltestellen Haltestellen, die unbedingt zu berücksichtigen sind, wie z.B. vor Schulen, Denkmäler, Gewerbegebiete, etc. Diese können vom Benutzer manuell gesetzt werden. Intern erhalten sie die Priorität 10.

Pflichtlinien Linien, die unbedingt zu berücksichtigen sind und die nicht verändert werden dürfen. Diese können vom Benutzer manuell bestimmt werden.

Planquadrat Als Planquadrat wird das Gebiet bezeichnet, in dem die Haltestellen zu verteilen sind. Daraus resultiert auch direkt, welche Pfade für die Linienplanung nur in betracht kommen.

Quelle-Ziel-Matrix Vollständige Matrix zwischen allen Haltestellen mit Angabe der jeweilige Reisendenanzahl. Synonym der Fahrtwunschmatrix.

Reisezeitaufwand Summe (bezüglich aller Haltestellenbeziehungen) der Produkte aus Anzahl Reisender und Reisezeit.

Site Ein einzelner Punkt unter GRASS, sofern er nicht Teil einer Arc ist.

Straßennetz Das Netz aller Straßen, die GRASS zur Verfügung stellt.

Straßenplan Das Straßennetz, reduziert auf die kürzesten Wege zwischen allen Haltestellen in einem vorgegebene Radius um die Haltestellen herum.

Taktzeit Die Taktzeit gibt an, wie oft pro Zeitintervall (normalerweise 1 Stunde) ein Bus auf einer Linie fahren soll.

Transportkettengraph Der Transportkettengraph (TKG) lässt sich aus dem Netz der Transportverbindungen dadurch ableiten, dass für jede Kante ein Knoten erzeugt wird, der somit die Fahrtzeit zwischen zwei Haltestellen symbolisiert. Knoten werden dann durch eine gerichtete Kante verbunden, wenn eine Verbindung hinsichtlich des gegebenen Liniennetzes möglich ist. Die Bewertung der sich ergebenen Kanten spiegelt die netzbedingte Wartezeit (zusätzlich multipliziert mit dem entsprechenden Verkehrsstrom) wieder. Dieses bedeutet, dass die Bindungen den Warteaufwand angeben.

Toleranzgrenze Stellt eine Distanz in Metern dar, die die potentiellen Fahrtgäste bereit sind maximal zu einer Haltestelle zu laufen.

(Haltestellen-)Verbindungsradius Gibt den Wert an, der den Radius um eine Haltestelle definiert, in dessen Umkreis alle Haltestellen durch kürzeste Wege mit der Haltestelle aus dem Mittelpunkt.

Verkehrsbedarfsmatrix Diese Matrix gibt die Anzahl der Reisenden zwischen den einzelnen, vom Haltestellen-Modul gesetzten Haltestellen an. Sie wird während der Haltstellen-Berechnung erzeugt und vom Linienplan-Modul benötigt.

Verkehrsstrom Anzahl Reisender auf einer bestimmten Strecke bezüglich einer bestimmten Linie.

Wegenetzgraph Ein Graph, in dessen Knoten die Haltestellenkoordinaten stehen. Zudem wird die Haltestelle über eine eindeutige Ziffer (ID) identifiziert und ist mit einer Haltestellenpriorität versehen. Zudem gibt es die Möglichkeit die haltestelleneigene Umsteigezeit in den Knoten festzuhalten. Die Verbindung der Knoten geschieht aufgrund der Benutzereingabe "Verbindungsradius", der für jede Haltestelle den Radius definiert, in dem alle Haltestellen, die mit der Haltestelle im Mittelpunkt verbunden werden, liegen müssen. An den Kanten steht der Straßenzug und die Entfernung der Haltestellen. Dabei ist der kürzeste Weg als Stecke berücksichtigt. Desweiteren befinden sich die Reise- oder Fahrtzeit, der Verkehrsfluss, die Mindestwartezeit in einer Haltestelle, sowie eine Liste von Buslinien, die über die jeweilige Kante fahren an den Kanten des Wegenetzgraphen. Bei letzterem wird für jede Buslinie die Ankunfts- und die Abfahrtszeit, der Linienfluss, und der Takt einer Linie gespeichert.

Zyklomatische Zahl Überbestimmtheit bezüglich eines Gerüstes einer Zusammenhangskomponente eines Graphen, d.h. Differenz zwischen Kantenanzahl und Knotenanzahl - 1.

Literaturverzeichnis

[Bussieck] Bussieck, M.-R.; Winter, T.; Zimmermann, U.T.; Schlußbericht: Optimale LInienführung und Routenplanung in Verkehrssystemen (Schienenverkehr);Literatur `http://www.math.tu-bs.de/mo/projects/lop/literature.html`;

[Dienst78] Dienst, Hartmut; Linienplanung im spurgeführten Personenfernverkehr mit Hilfe eines heuristischen Verfahrens; Diss. Braunschweig; 1978

[EA99] Evolutionäre Algorithmen; Universität Dortmund; Fachbereich Informatik; Lehrstuhl 11; Evolutionaere Algorithmen `http://ls11-www/bb/jahresbericht98-99/node11.html`;1981

[Fowler98] Fowler, Martin; Scott, Kendall: UML konzentriert; 1998; Addison-Wesley

[GRASS5] Geo-Informationssystem GRASS (Geographic Resources Analysis Support System), Release 5, GRASS `http://www.geog.uni-hannover.de/grass/index.html`

[Groetschel96] Grötschel, M.; Löbek, A.;Völker, M.: Optimierung des Fahrzeugumlaufs im Öffentlichen Nahverkehr, Berlin 1996

[Gruhn98/99] Gruhn, Volker; Vorlesungsunterlagen "Softwaretechnologie"; Wintersemester 1998/1999; LS10 `http://ls10-www.cs.uni-dortmund.de/`

[Gue92] Güting, Ralf Hartmut; Datenstrukturen und Algorithmen; B.G. Teubner. Stuttgart; 1992

[IEEE-829] Testdokumentation nach der IEEE-829, Vorlesung Softwaretechnologie, Wintersemester 1998/1999, Universität Dortmund, `http:://ls10-www.cs.uni-dortmund.de/`

[Klemt78] Klemt, Wolf-Dieter: Untersuchungen zur Nachfrageorientierten Netzerzeugung für den öffentlichen Personennahverkehr auf der Grundlage eines Planungsinformationssystems; Diss. Berlin; 1978

[Krug87] Krug, Stephan: Ein interaktives Programmsystem zur Angebotsplanung für den liniengebundenen öffentlichen Personennahverkehrs, Braunschweig 1987

[Lewandowski] Lewandowsky, Stefan; Optimayation Criteria for Public Transportation Services; 9th IFAC Symposium Control in Trasportation Systems 2000; Proceedings; TH Braunschweig; 2000

[LEDA4.2] Max-Planck Institut für Informatik, Library of Efficient Datatypes and Algorithms, Release 4.2, LEDA `http://www.mpi-sb.mpg.de/LEDA/`

[Maack78] Maack, Udo E.; Graphentheoretische Untersuchungen zur Berechnung nachfrageorientierter Netze des öffentlichen Personennahverkehrs auf der Grundlage eines Planungsinformationssystems; Diss. Berlin; 1978

[PG369-Handbuch] PG369; HaLiFa Handbuch; Fachbereich Informatik; Universität Dortmund; 2001

[PG369-Internetadresse] PG369-Internetadresse `http://ls4-www.informatik.uni-dortmund.de/Lehre/00-46369/index.html`

[PG369-Klassendokumentation] PG369; Klassendokumentation;Fachbereich Informatik; Universität Dortmund; 2001.

[ScHeFe94] Schöneburg, Heinzmann, Feddersen; Genetische Algorithmen und Evolutionsstrategien; Addison-Wesley; Bonn 1994; ISBN: 3-89319-493-2

[SimonisDiss] Simonis, Carsten; Die nachfrageorientierte Optimierung von Omnibuslinien im Stadtbereich durch Verknüpfung von Teilstrecken nach unterschiedlichen Modellansätzen; Diss. Aachen; 1981

[Simonis81] Simonis, Carsten; Wall, H.: Omnibusnetze zur Erschließung von Verdichtungsräumen und deren Randzonen, Bonn 1981

[Sonntag77] Sonntag, Herbert; Linienplanung im öffentlichen Nahverkehr; Diss. Berlin; 1977

[TEA24] Tool for Evolutionary Algorithms, Version 2.4 small, Lehrstuhl für Systemanalyse; Fachbereich Informatik; Universität Dortmund; 2001

[Testdatendokument] PG369; Testdokument; Fachbereich Informatik; Universität Dortmund; 2001

[Voelker01] Völker, Marcus; Busliniennetze optimal gestalten - Ein multikriterieller Algorithmus zur rechnergestützten Planung; Der Nahverkehr, Ausgabe 10/2001, S. 12-17, Alba Fachverlag, Düsseldorf

[Wegener] Wegener, Ingo; Grundvorlesung Datenstrukturen; Vorlesungsskript der Universität Dortmund; 1998

[Weigand81] Weigand, Werner; Graphentheoretisches Verfahren zur Fahrplangestaltung in Transportnetzen unter Berücksichtigung von Pufferzeiten mittels Interaktiver Rechentechnik; Diss. Braunschweig; 1981

[Zeitplan] Zeitplan der PG 369 für das Sommersemester, zu finden auf [PG369-Internetadresse]

[Zimmermann] Zimmermann, W.; Operations Research, Quantitative Methoden zur Entscheidungsvorbereitung; 9. Ed. Oldenbourg; 1999