Suchalgorithmen und Interaktionstechniken für Fahrplan-Informationssysteme

Ralf Möller Lothar Hotz

1. Juli 1991

Zusammenfassung

Dieser Bericht beschreibt Suchalgorithmen zur Bestimmung von Verkehrsverbindungen in einem Nahverkehrssystem. KI-spezifische Suchtechniken werden mit Optimierungsmethoden aus anderen Disziplinen (z.B. Optimierungstheorie) kombiniert. Mithilfe dieser Verfahren wurde ein anwendungsnaher Prototyp eines Fahrplan-Informationssystems realisiert. Es werden die Konzepte der objektorientierten Benutzungsschnittstelle vorgestellt, die einem Benutzer verschiedene Ausschnitte des Nahverkehrssystems in verschiedenen Maßstäben visualisiert.

Abstract

This paper describes search algorithms to find optimal connections in a local traffic system. AI search techniques are combined with optimization methods developed in other disciplines (e.g. optimization theory) to realize a near-application prototype of a time-table information system. We present concepts of the user interface which visualizes the net of the traffic system in different views and at different scales.

Inhaltsverzeichnis

Abbildungsverzeichnis

Das Projekt FIS - ein Überblick

Das Projekt „Fahrplan-Informationssysteme“ (FIS) wird am „Labor für Künstliche Intelligenz“ (LKI) der Universität Hamburg in Zusammenarbeit mit der Firma „Hamburger Berater Team“ (HBT) durchgeführt. Es gliedert sich in zwei Phasen. In der ersten Phase wurde mittels einer Prototypimplementation die grundlegende Konzeption eines FIS entwickelt. In der zweiten Phase werden die Ansätze zur automatischen Suche verfeinert und implementiert. Dieser Bericht entstand zu Beginn der zweiten Projektphase. Das Ziel ist es, ein weitgehend serienreifes System zu entwickeln. Die Abbildungen 0.1 und 0.2 geben einen Überblick über den Projektverlauf.

Projektleitung:

Hans-Joachim Habermann (HBT)
Prof. Dr. Bernd Neumann (LKI)

Projektmitarbeiter:

FIS1 - erste Projektphase von April bis September 90

Michael Malsch (HBT)
Ralf Möller (LKI)
Thomas Schnudt (HBT)

FIS2 - zweite Projektphase von Januar bis Oktober 91

Lothar Hotz (LKI)
Michael Malsch (HBT)
Ralf Möller (LKI)
Dr. Friedemann Weik (HBT)

Förderung

Das Projekt FIS wurde durch die Wirtschaftsbehörde der Hansestadt Hamburg gefördert.

Überblick über die Projektarbeiten

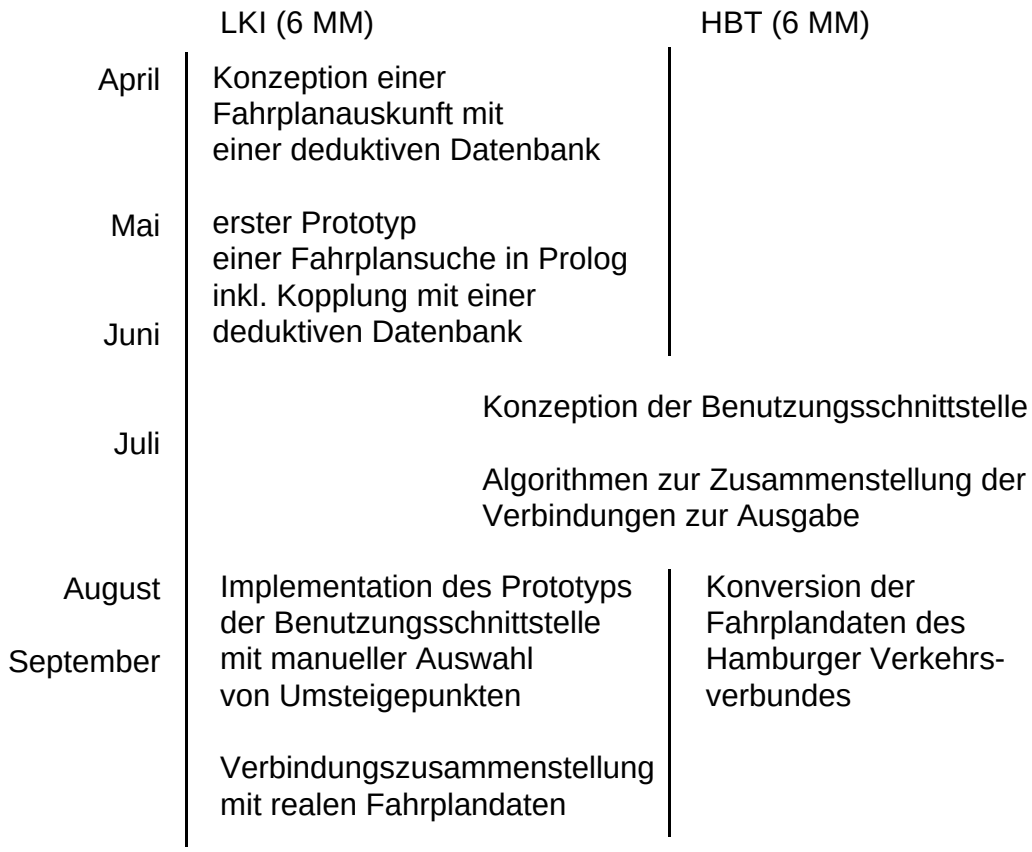

Abbildung 0.1: Überblick über die Projektarbeiten (1. Projektphase).

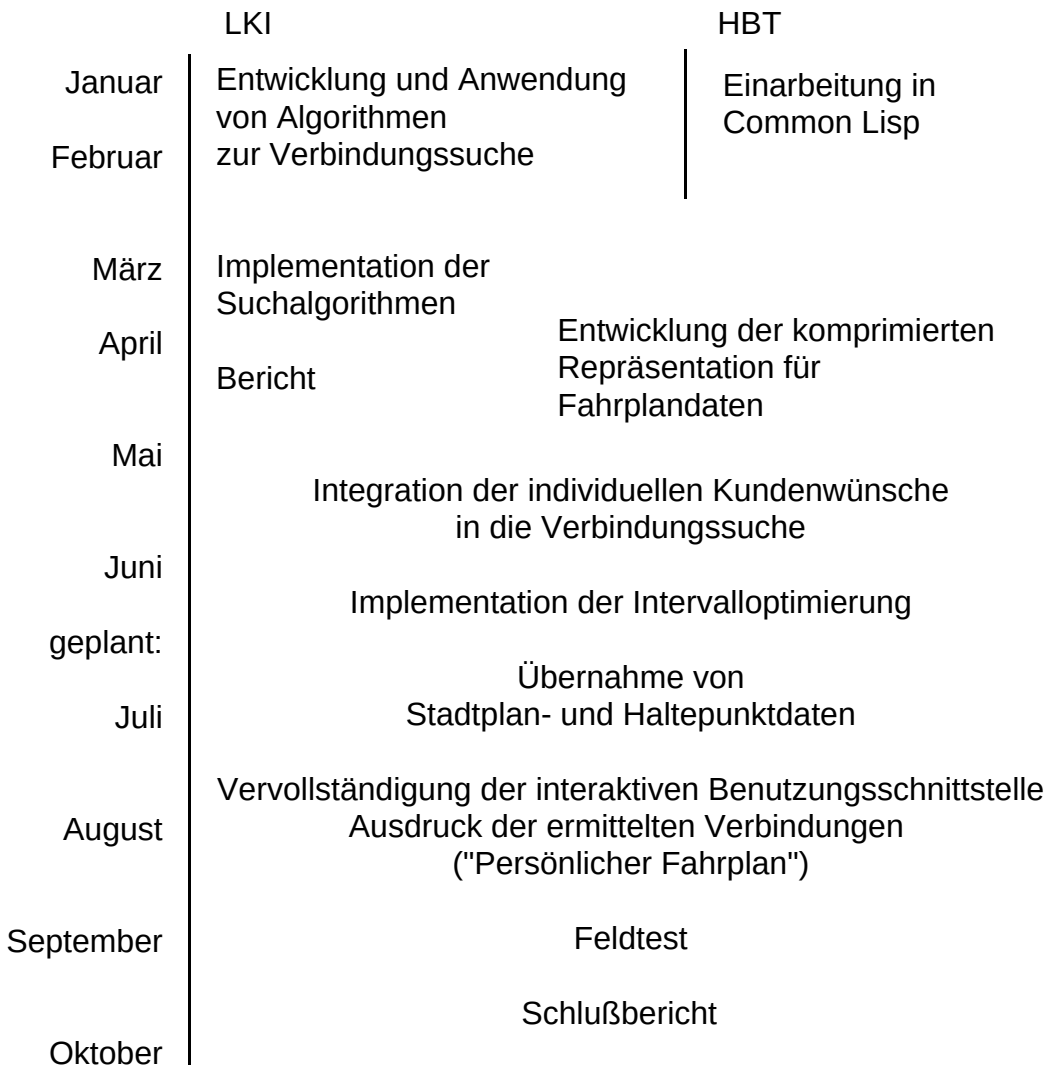

Abbildung 0.2: Überblick über die Projektarbeiten (2. Projektphase).

Kapitel 1

Einleitung

Zu den Aufgaben des „Labors für Künstliche Intelligenz“ (LKI) an der Universität Hamburg gehört die anwendungsnahe Forschung im Bereich der Künstlichen Intelligenz (KI). Für das in diesem Bericht beschriebene Projekt FIS, das sich mit Informationssystemen für Fahrpläne beschäftigt, wurde angestrebt, KI-Methoden einzusetzen, um Fahrplanauskünfte über einen großen, komplexen Fahrplan-Datenbestand geben zu können. Dem LKI wurde durch eine Zusammenarbeit mit dem Hamburger Verkehrsverbund (HVV) ein Datenbestand zugänglich, der etwa die Hälfte der Linien des Hamburger Nahverkehrsystems beschreibt. Durch die Verwendung dieser realen Daten ist sichergestellt, daß die im Projekt FIS entwickelten Verfahren realistisch erprobt werden können.

Das im Projekt FIS konzipierte Auskunftssystem hat als Zielgruppe noch nicht den Fahrgast am Bahnhof, sondern vielmehr die Mitarbeiter einer Telefonauskunft, wie sie z.B. beim HVV existiert. Es handelt sich also um eine Nutzergruppe, die als Spezialisten in Bezug auf die Fahrplanauskunft gelten. In diesem Rahmen wurde das FIS bewußt eher als entscheidungsunterstützendes System konzipiert, weil die Fragen von auskunftsuchenden Kunden vielfach recht vage sind und erst nach Rückfragen eine Anfrage an das Auskunftssystem gestellt werden kann. Eine Übertragung der bei diesem Ansatz im Projekt FIS gewonnenen Erkenntnisse auf ein Informationssystem, das in einen Fahrkartenautomaten integriert werden könnte, scheint jedoch möglich. Folgende Auskunftsformen sollen durch den FIS-Prototypen berücksichtigt werden:

- Verbindung von A nach B mit vorgegebener Ankunfts- bzw. Abfahrtszeit,
- Auskünfte bzgl. vom Kunden genannter Linien,

- Ausdruck und Zusendung von Fahrplanauskünften („Persönlicher Fahrplan“).

In diesem Bericht gehen wir auf die beiden letzten Punkte nicht näher ein. Es handelt sich dabei um vergleichsweise einfache Datenbasisanfragen und entsprechende Ausgabemechanismen.

Wie wird nun ein Kundenberater bei der Aukunft unterstützt? Aufgrund von ungenauen Schilderungen einiger Kunden muß es dem Auskunftgebenden möglich sein, Ortsangaben näher einzugrenzen. Bei der HVV-Auskunft wird hierfür ein Stadtplan verwendet. Für ein Fahrplan-Informationssystem scheint es also angebracht, dem FIS-Bediener Stadtplandaten zur Orientierung anzubieten. Mit Hilfe der graphischen Darstellung von Stadtplandaten, die auch die Verkehrslinien beinhalten, können Start- und Ziel- sowie Umsteigestationen interaktiv ausgewählt werden. Das FIS sucht aus der Fahrplandatenbasis gemäß den Angaben bzgl. des Ankunfts- oder Abfahrtszeitintervalls entsprechende Fahrten heraus und stellt die ermittelten Verbindungen innerhalb eines Dialogfensters dar. Nun ist die manuelle Auswahl der „Fahrtstrecke“ nicht immer eindeutig, wenn etwa die Fahrtzeit oder die Anzahl der Umsteigevorgänge minimiert werden soll. Innerhalb des Projektes FIS wurde daher eine Such- und Optimierungskomponente entwickelt, die einen Kundenberater bei der Auskunft unterstützt und verschiedene Bewertungskriterien (z.B. Fahrtzeit, Anzahl der Umsteigevorgänge, Bevorzugung von Bussen gegenüber U-Bahnen) zuläßt.

Wie in den nachfolgenden Abschnitten deutlich wird, sind gerade durch die Menge der zugrundeliegenden Daten besondere Anforderungen an die Suchalgorithmen sowie an die Komponenten zur graphischen Darstellung definiert. Bekannte heuristische KI-Suchverfahren werden daraufhin betrachtet, inwieweit sie, entsprechend angepaßt und eingesetzt, in der Fahrplandomäne Anwendung finden können. Es werden jedoch auch andere, nicht direkt in den Bereich KI einzuordnende Verfahren untersucht und angewandt.

In dem hier vorgestellten FIS-Prototypen werden Regelhaftigkeiten beim Einlesen der Fahrplandaten automatisch ermittelt. Dadurch wird eine erhebliche Speicherplatzersparnis erzielt. Die Repräsentation der Fahrplandaten beim HVV nutzt (bisher) keine Regelhaftigkeiten aus[1], sondern repräsentiert alle Fahrten explizit (extensionale Repräsentation).

Nach der Analyse der Fahrplandomäne in Kapitel 2 bildet die Beschreibung der eingesetzen Suchalgorithmen in Kapitel 3 den Kern dieser Arbeit. Anschließend wird die Repräsentation der Fahrplandaten skizziert. Nach der Darstellung der Präsentations- und Interaktionskomponenten schildert Kapitel 6 die Themen der aktuellen Arbeiten, die das Fahrplan-Informationssystem

[1]Der HVV bezieht die Fahrplandaten aus der Einsatzplanung der Fahrzeuge.

vervollständigen. Anschließend werden in Kapitel 7 Ansätze zur Ermittlung von Verbindungen in anderen bisher entwickelten Auskunftssystemen vorgestellt. Der Bericht schließt mit einer Zusammenfassung der Ergebnisse.

Kapitel 2

Problemanalyse

Um einen Überblick über die Probleme der Fahrplanauskunft zu bekommen, müssen zunächst einmal die Domäne und die zugrundeliegenden Daten genauer untersucht werden.

Im Hamburger Nahverkehrssystem gibt es mehrere *Verkehrsmitteltypen*: S- und U-Bahnen, Busse, Schnellbusse, Nachtbusse, Stadtbusse sowie Fährschiffe. Eine *Bewertung* der Verkehrsmittel spiegelt die Qualität der Beförderung wider, kann jedoch nicht unabhängig vom Fahrgast, d.h. statisch, vorgenommen werden. Obwohl im Normalfall S-Bahnen schneller sind und einen höheren Fahrkomfort bieten, halten manche Kunden Busse wegen des sich in Reichweite befindlichen Fahrers für die bessere Alternative. Zu bedenken ist, daß nicht alle Verkehrsmittel zu jeder Zeit fahren. So gibt es während der Nachtzeit meist nur Nachtbusse, die - und das ist ein wichtiger Punkt - eventuell nicht auf den gleichen Strecken wie die Busse zur Tageszeit verkehren. Nach Angaben der HVV-Vertreter erfolgt z.B. bei den Bussen durchaus keine regelmäßige Abfahrt, etwa im Fünf-Minuten-Takt. Dieses gilt insbesondere für Busfahrten in die Stadtrandbezirke.

Hier wird deutlich, daß die Fahrpläne direkt in die Verbindungssuche mit einbezogen werden müssen. Das Problem der Fahrplanauskunft in einem Nahverkehrssystem läßt sich i.a. nicht zerlegen in eine Routensuche gefolgt von einer Auswahl der zeitlich passenden Verbindung. Um eine Routensuche ohne Berücksichtigung der Fahrpläne überhaupt durchführen zu können, müssen Fahrt- und Wartezeiten geschätzt werden (siehe unter „Verbindungsvorbewertung“ in Kapitel 7). Fehlerhafte Schätzungen bedingen nicht nur, daß sich der Suchaufwand erhöht, sondern führen dazu, daß eine optimale Lösung (im Sinne der Vorgaben) nicht gefunden wird.

Zu den Besonderheiten und (für die automatische Suche) problematischen Fällen eines Nahverkehrssystems gehören spezielle Linienführungen. Im Hamburger Nahverkehrssystem sind sowohl Ringlinien als auch Schleifen- bzw.

Q-Linien zu berücksichtigen (Abbildung 2.1). Durch Linienformen dieser Art sind besondere Anforderungen an die Suchalgorithmen gestellt (siehe Kapitel 3).

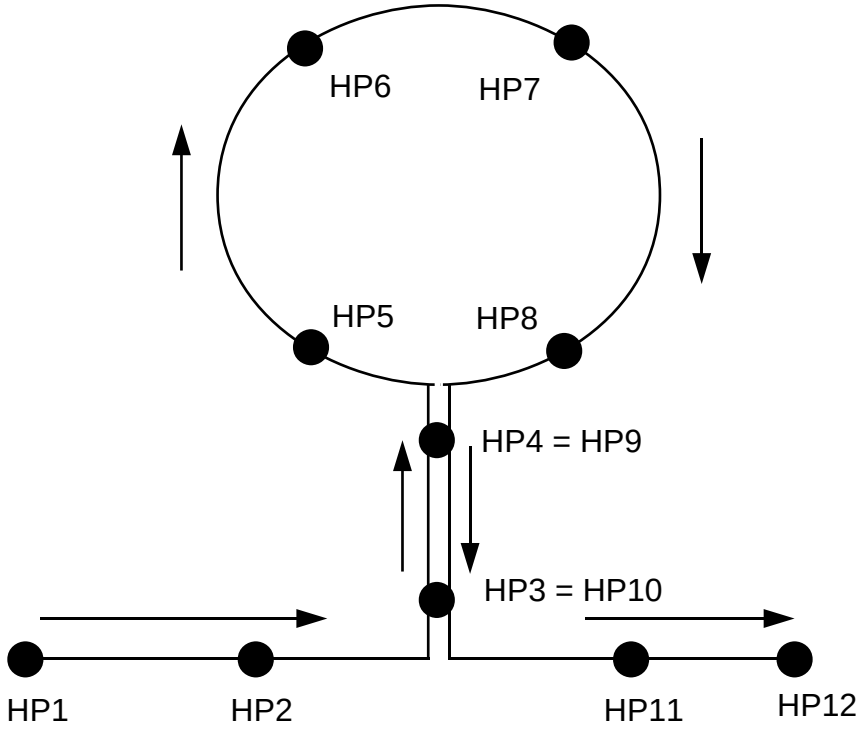

Abbildung 2.1: Beispiel für eine Q-Linie.

2.1 Modellierung der Domäne

Innerhalb eines Fahrplanauskunftssystems sind verschiedene Aspekte des Nahverkehrs zu modellieren. Zum einen ist die verkehrstechnische Sicht für die Suche, zum anderen ist die geographische Sicht für die Benutzungsschnittstelle relevant. Um diese Aspekte sauber einführen zu können, werden zunächst einige Begriffe definiert.

Verkehrstechnische Aspekte

Wir fassen Bahnhöfe verschiedener Größe (auch S- und U-Bahnhöfe) sowie Bus- und Straßenbahnhaltestellen unter dem Begriff *Haltepunkt* zusammen. Haltepunkte werden durch *Fahrten* von Nahverkehrsmitteln verschiedener

Linien verbunden. Unter einer *Linie* wird eine Menge von Fahrten mit gleichem Verkehrsmitteltyp unter einem bestimmten Namen zusammengefaßt. Beispiele sind etwa: S- und U-Bahn-Linien (S3, U1, U2) oder Buslinien (120, 124). Wir teilen die Fahrten einer Linie auf verschiedene *Unterlinien* auf. Alle Fahrten einer Unterlinie bedienen mit jeder Fahrt die gleichen Haltepunkte in der gleichen Reihenfolge. Häufig modellieren Unterlinien z.B. Verzweigungen oder Teilstrecken einer Linie. Manche Unterlinien lassen sich als „Rückfahrt" von anderen Unterlinien interpretieren. Für die Fahrplansuche ist diese Unterscheidung allerdings irrelevant.

Ein *Umsteigeknoten* faßt eine Menge von Haltepunkten zusammen, zwischen denen ein Fahrgast mit vertretbarem Aufwand zu Fuß verkehren kann. Ein Beispiel ist ein Verkehrsknoten wie der Hauptbahnhof, der in Hamburg aus mehreren Unterbahnhöfen (Hauptbahnhof-Süd, Hauptbahnhof-Nord, Hauptbahnhof-ZOB[1], etc.) besteht. Man beachte, daß auch innerhalb eines Haltepunktes zwischen verschiedenen Linien nicht unerhebliche Fußwege berücksichtigt werden müssen. Weiterhin ist auch denkbar, z.B. bei Schleifenlinien, bei einem Haltepunkt in eine andere Fahrt der gleichen (Unter-)Linie umzusteigen. In diesem Sinne sind auch alle „einzelnen" Haltepunkte als (formale) Umsteigeknoten modelliert. Umsteigeknoten für einzelne Haltepunkte ermöglichen weiterhin eine einfache Integration der Fußwegezeiten vom eigentlichen Start bzw. Ziel bis zu einem Haltepunkt des Nahverkehrssystems in die Optimierung (siehe Abschnitt 3.2).

Das *Verkehrsnetz* wird als eine Menge von Umsteigeknoten aufgefaßt, die über Linien miteinander verbunden sind.

Geographische Aspekte

Geographische Informationen sind für eine Darstellung von z.B. Linien und Haltepunkten innerhalb der Benutzungschnittstelle relevant. Zu den geographischen Aspekten der Modellierung gehören Ortsangaben für Haltepunkte und Linienzüge (angenähert durch Stützpunkte). Ein wichtiger Gesichtspunkt bei der Modellierung der Linienverläufe ist die geometrische Repräsentation von Unterlinien und der einfache Zugriff auf Teilabschnitte (Abschnitte zwischen zwei Haltepunkten). Teilabschnitte werden z.B. bei der graphischen Darstellung einer Verbindung benötigt. Zur Orientierung ist es notwendig, neben Straßen(zügen) auch markante und öffentliche Gebäude zu repräsentieren. Die Repräsentationsformen sollten verschiedene Detaillierungsgrade unterstützen.

Es bietet sich an, geographische Informationssysteme (GIS), die für vie-

[1]ZOB = Zentraler Omnibusbahnhof

le Städte entwickelt werden, einzusetzen. Diese Systeme könnten weiterhin Daten über Lage und Form von Gewässern, Waldgebieten und andere geographische Einheiten enthalten, die damit auch für das FIS zur Verfügung stünden. Die Erfassung und Aktualisierung solcher Daten im großen Stil nur für ein Fahrplaninformationssystem ist vermutlich zu aufwendig.

2.2 Das Verbindungssproblem

Was ist nun die Aufgabe des Fahrplanauskunftssystems? Wir betrachten hier die Aufgabe, bzgl. einer vorgegebenen Abfahrts- bzw. Ankunftszeit eine Verbindung (oder auch mehrere Verbindungen) von einem Starthaltepunkt zu einem Zielhaltepunkt zu ermitteln. Eine *Verbindung* ist dabei definiert als eine Folge von Fahrten (einschließlich Zeitangaben und entsprechenden Zu- bzw. Aussteigehaltepunkten). Der Zusteigehaltepunkt der ersten Fahrt bildet den Starthaltepunkt der Verbindung. Der Aussteigehaltepunkt einer Fahrt ist entweder identisch mit dem Zusteigehaltepunkt der folgenden Fahrt, oder der Zusteigehaltepunkt der folgenden Fahrt kann „zu Fuß“ erreicht werden. Der Aussteigehaltepunkt der letzten Fahrt ist der Zielhaltepunkt der Verbindung. Als *Route* oder auch *Strecke* sei die Folge der Umsteigeknoten der Verbindung (ohne Zeitangaben) definiert.

2.3 Ein Beispiel für verschiedene Verbindungsmöglichkeiten

Um einen Eindruck von der Problematik der Verbindungsbestimmung zu vermitteln, zeigen wir Ausschnitte des Hamburger Nahverkehrssystems mit Hilfe der Benutzungsschnittstelle des FIS-Prototypen (Abbildung 2.2). Weitere Aspekte der Benutzungsschnittstelle werden in Kapitel 5 vorgestellt.

Ein Kunde möchte von „Am Weinberg“ bis „Holstenhofweg“ fahren. Es wird ein Abfahrtszeitintervall 10:00 - 11:00 Uhr angegeben. In jedem Fall sind zwei Umsteigevorgänge notwendig. Das Dialogfenster zeigt Auschnitte des Verkehrsnetzes: links eine Übersichtsdarstellung, rechts zwei vergrößerte Darstellungen von Startbereich (oben) und Zielbereich (unten). Der Auskunftgebende kann zwischen den zwei parallel verlaufenden diagonalen Linien wählen. Auf dem ersten Teilabschnitt verkehren mehrere Busse, auf dem zweiten Abschnitt fährt die U1 („obere“ Strecke) oder die S4 („untere“ Strecke). Anschließend stehen wieder mehrere Buslinien zur Verfügung.

Bei einer manuellen Auswahl von Linien kann auch ein erfahrener FIS-Bediener bei fast gleichwertigen Alternativen nicht immer eindeutig entschei-

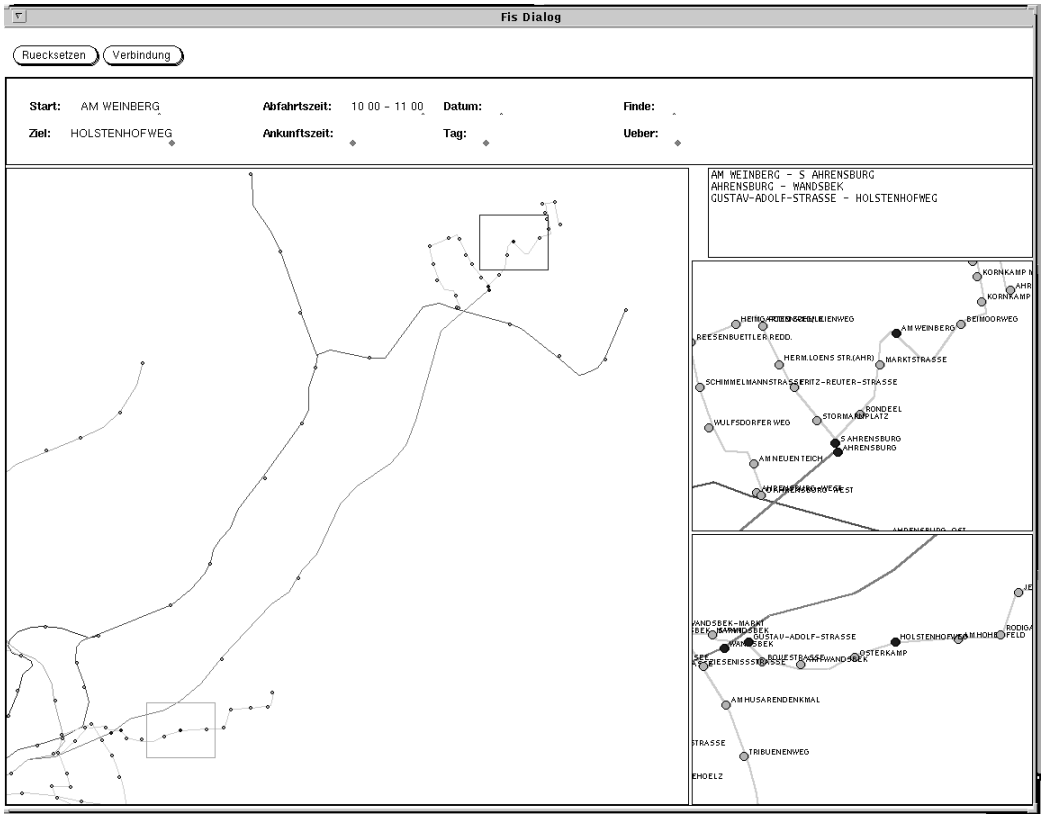

Abbildung 2.2: Verbindungsalternativen.

den, welche die zeitlich kürzere oder nach weiteren Kriterien bessere ist. In Kapitel 5 zeigen wir, daß die Fahrten über die „obere“ diagonal verlaufende Strecke ca. 60 Minuten, über die „untere“ Strecke jedoch nur ca. 40 Minuten dauern.[2] Einmal hängt die Fahrtzeit sehr von den Anschlüssen nach dem Umsteigen ab, zum anderen müßte graphisch dargestellt werden, ob in dem angegebenen Abfahrtszeitintervall überhaupt ein Verkehrsmittel auf einer Linie fährt. Dies ist z.B. bei Buslinien in den Stadtrandbezirken nicht unbedingt der Fall.

Durch die Präsentation des Stadtplans mit Hilfe der Graphikkomponente von FIS kann der Auskunftgebende eine Menge von Start- und Zielhaltepunkten, die von den Ortsangaben des Kunden mit vertretbarem Aufwand zu Fuß erreichbar sind, eingrenzen. Eventuell können bestimmten Orten auch schon Haltepunkte fest zugeordnet werden. Zu beachten ist jedoch, daß der gewählte Haltepunkt i.a. von der bestimmten „optimalen“ Verbindung ins Zielgebiet abhängt. Stärker noch: der Fußweg zu einem Haltepunkt in der Umgebung kann auch die Verbindungssuche beeinflussen.

Man wünscht sich bei der Kundenberatung Unterstützung durch ein Planungs- und Suchverfahren, das nach vorgegebenen Kriterien (z.B. „behindertenfreundlich“) Verbindungen ermittelt. Dieses System sollte mit der Interaktionskomponente gekoppelt sein. In dem in diesem Bericht vorgestellten Fahrplanauskunftssystem werden bei der automatischen Verbindungssuche folgende Optimalitätskriterien berücksichtigt:

- Minimierung der Fahrtzeit,
- Minimierung der Zahl der Umsteigevorgänge.

Algorithmen zur Verbindungssuche werden im nächsten Kapitel vorgestellt. Die Integration von weiteren, individuellen Kriterien schildert Kapitel 6.

[2] Erfahrene Berater in einer Kundenauskunft können dieses in diesem Fall vielleicht noch abschätzen. Es gibt jedoch auch Beispiele, in denen die Verhältnisse noch komplizierter sind. Dies ist z.B. der Fall, wenn lange Wartezeiten beim Umsteigen entstehen.

Kapitel 3

Graphensuchalgorithmen und ihre Anwendung

In den sechziger Jahren wurden heuristische Suchverfahren als Lösung für bestimmte Klassen von KI-Problemen untersucht. Die Problemlösung wird dazu als schrittweise Bewegung in einem Zustandsraum aufgefaßt (siehe z.B. [24]). Jedem Zustand (Knoten) im Zustandsraum wird durch eine problemspezifische Nachfolgerfunktion eine Menge von Nachfolgezuständen zugeordnet. Der Zustands- bzw. Problemraum stellt meist einen (gerichteten) Graphen dar. Die Kanten, die von einem Knoten ausgehen, werden als Alternativen betrachtet; es handelt sich also um einen ODER-Graphen. Eine Problemlösung ist dann ein Pfad (eine Folge von Kanten) von einem Startzustand (Startknoten) über Zwischenzustände zu einem Zielzustand. Zustände oder auch Zustandsübergänge sind mit sog. Kosten assoziiert. Kosten können verschiedene Problemaspekte modellieren. Bei dem Problem des Traveling-Salesman z.B. werden Städte durch Knoten, Verbindungen durch Kanten und Entfernungen durch Kantenkosten modelliert. Ein optimaler Pfad ist ein Pfad, bei dem die Summe der Kantenkosten minimal ist. Durch Kantenkosten werden also problemspezifische Informationen in die Graphensuche integriert, die Suche nach einem Zielzustand wird damit durch Domänenwissen gesteuert. Andernfalls droht auch schon bei „kleinen“ Problemen eine kombinatorische Explosion. Für kombinatorische Suchprobleme in Graphen existieren verschiedene wohlbekannte Algorithmen, die wir im nächsten Abschnitt kurz skizzieren, um anschließend die problemgerechte Anpassung an die hier vorliegenden Domäne zu schildern.

3.1 Der A*-Graphensuchalgorithmus

Ein allgemeiner domänenunabhängiger Graphensuchalgorithmus zur Bestimmung von Lösungen, d.h. Pfaden mit minimalen Kosten, in ODER-Graphen heißt A*. Der Algorithmus wird in den meisten KI-Lehrbüchern detailliert vorgestellt [22], [24]. In diesem Bericht wird daher nur das Grundkonzept grob skizziert, um die verwendeten Bezeichner einzuführen. Der Algorithmus sucht, von einem Startknoten ausgehend, einen optimalen Pfad zu einem Zielknoten. Problemspezifische Datenstrukturen (bzw. Instanzen hiervon), die die Zustände des Problems modellieren, sind Bestandteil der zugeordneten Suchgraphknoten, werden aber von A* als atomar angesehen. Die innere Struktur der Problemzustände wird im Standard-A*-Algorithmus nicht betrachtet.

Der Suchraum wird schrittweise durchwandert, indem Suchgraphknoten nach bestimmten Kriterien ausgewählt und ihre direkten Nachfolger durch eine problemspezifische Nachfolgerfunktion generiert werden (der Suchraum bzw. die Knoten werden „expandiert"). Hierzu werden zwei Mengen (Listen) von Knoten verwaltet: eine für noch zu expandierende Knoten (OPEN) und eine für bereits expandierte Knoten (CLOSED). Die Knoten auf der OPEN-Liste sind nach den geschätzten Kosten sortiert, die den Knoten zugeordnet sind. Es wird bei jedem Iterationsschritt der Knoten mit den aktuell niedrigsten Kosten auf der OPEN-Liste expandiert. Die Kosten f eines Knotens setzen sich aus zwei Komponenten zusammen (siehe Abbildung 3.1): $f = g + h$.

Der Summand g beschreibt dabei die Kosten für den (aktuell) besten Pfad vom Startknoten zum aktuellen Knoten, der Summand h beschreibt entsprechend die optimalen Kosten vom aktuellen Knoten zum Zielknoten. Während die Kosten g immer bekannt sind, läßt sich h meist nur durch eine Schätzfunktion h' bestimmen. Damit ergibt sich: $f' = g + h'$. Falls nun für alle Knoten $h' \leq h$ gilt, so werden die Kosten zum Ziel niemals überschätzt. Wenn man einen Schätzer h' findet, der diese Bedingung erfüllt, ist gewährleistet, daß der Algorithmus A* die optimale Lösung (gemäß den Vorgaben) findet. Für eine genauere Analyse des Algorithmus wird auf [22] verwiesen. Zur Berechnung des Schätzers h' für unser Problem siehe Abschnitt 3.3.

Beim Expandieren der Knoten (Durchwandern des Suchgraphen) kann der Fall eintreten, daß ein schon betrachteter Knoten erneut erzeugt wird. Dies zeigt an, daß man über einen anderen Pfad im Suchgraphen auch zu diesem Knoten gelangt. Falls der Algorithmus einen weiteren Weg zu einem Knoten findet und die Kosten dieses Pfades geringer sind (g liefert für den neu entdeckten Pfad einen günstigeren Kostenwert), so bedeutet dies, daß der neue Pfad zu diesem Knoten zu bevorzugen ist. Aus diesem Grunde führt der Algorithmus alle betrachteten Knoten in den beiden Listen OPEN und CLO-

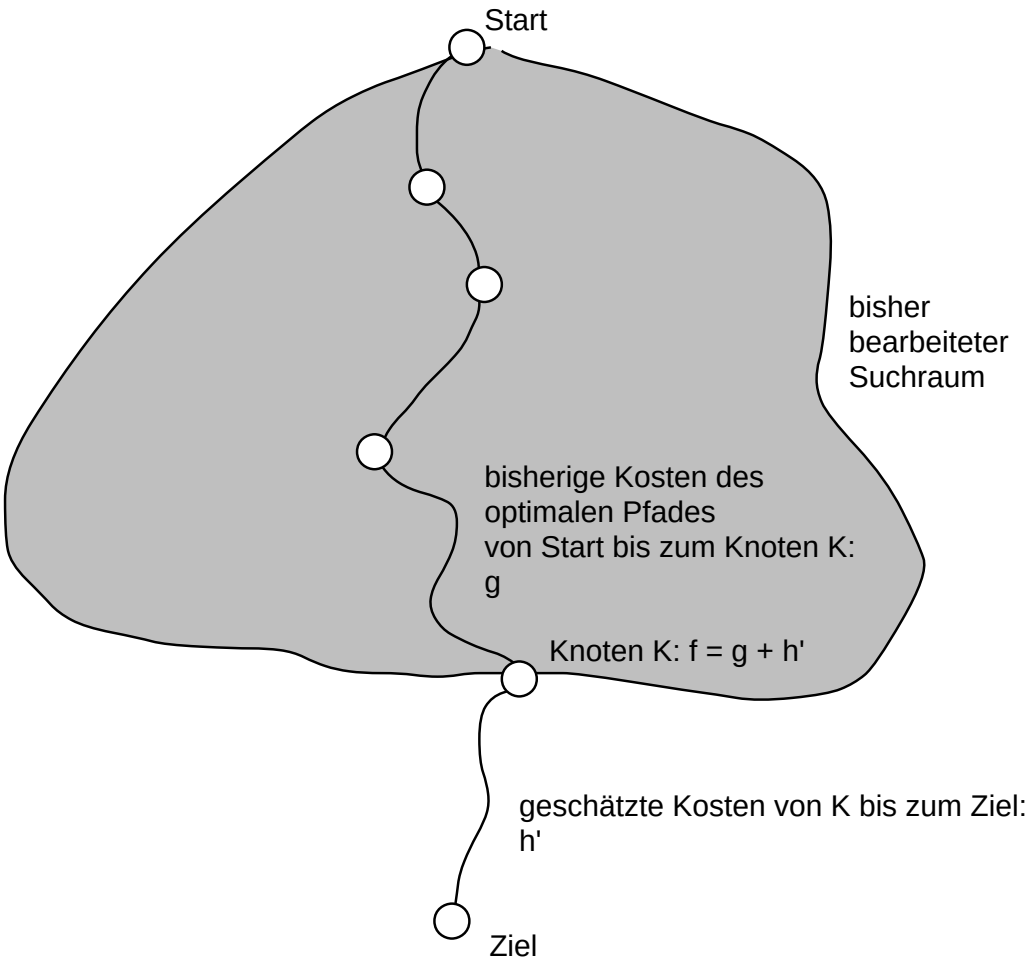

Abbildung 3.1: Zusammensetzung der Kosten eines Knotens aus einem bekannten und einem geschätzten Anteil.

SED. Für jeden Nachfolgerknoten muß also überprüft werden, ob er schon existiert. Es gibt zwei Möglichkeiten: entweder befindet sich der Knoten auf der OPEN-Liste, muß also ggf. noch expandiert werden, oder er befindet sich auf der CLOSED-Liste, ist also schon expandiert. Sollte letzteres der Fall sein, wird der neue Kostenwert an die Nachfolger des sich schon auf der CLOSED-Liste befindlichen Knotens weiterpropagiert, da sich die günstigeren Kosten auch auf die nachfolgenden Knoten auswirken. Zur Änderung der Verzeigerung zum jeweiligen Vorgänger siehe [22].[1]

3.2 Suche nach einer „optimalen Verbindung" mit Hilfe des A*-Algorithmus

Bei der Suche im Problemgraphen unterscheiden wir in der FIS-Domäne zwischen zwei Fällen: Wird ein Abfahrtszeitpunkt vorgegeben, ergibt sich eine Art Vorwärtssuche; bei Vorgabe eines Ankunftszeitpunktes liegt eine Rückwärtssuche vor. Betrachten wir zur Erläuterung nur die Vorwärtssuche. Die Rückwärtssuche ergibt sich analog (siehe aber Abschnitt 3.5).

Durch den Algorithmus A* wird nur der Grundrahmen zur Suche bereitgestellt. Dieses Gerüst muß nun domänenabhängig angewendet werden. Dabei werden den Konstrukten des A* (Suchraum, Knoten, Expansion und Kosten) problemspezifische Interpretationen zugeordnet. Ein besonderes Merkmal unseres Vorgehens ist die Berücksichtigung der Fahrtzeiten bei der Suche. Es werden verschiedene Optimierungen vorgestellt, um die Anzahl der expandierten Knoten zu begrenzen.

Beschreibung des Suchraums

Wie sieht nun der durch das FIS betrachtete Suchraum aus? Einmal bildet das Verkehrsnetz einen Graphen aus der Menge aller Umsteigeknoten U, die dann benachbart sind, wenn es eine Linie zwischen ihnen gibt (Abbildung 3.2). Dieser Graph wird zur Übersicht im FIS-Dialog ausschnittsweise dargestellt (vgl. Abschnitt 2.3). Die Umsteigeknoten werden hierbei gemäß ihrer geographischen Anordnung plaziert. Zur Übersicht reicht es aus, die Kanten dieses Verkehrsnetzgraphen als ungerichtete Kanten darzustellen.

Neben diesen ungerichteten Kanten, die die Linienstruktur des Verkehrsnetzes widerspiegeln, unterscheiden wir noch zwei weiteren Arten von Kan-

[1] Die Verwaltung der Pfade innerhalb einer Graphensuche wird durch den Grundalgorithmus GRAPHSEARCH bereitgestellt. Die Aufteilung der Kosten eines Knotens in zwei Komponenten g und h wird durch den Algorithmus A vorgenommen. Wenn gilt $h' \leq h$, dann heißt der Algorithmus A*.

ten. Die erste Kantenart (gestrichelt) ergibt sich direkt aus der Topologie des Nahverkehrssystems und besagt, daß es eine Unterlinie gibt, die nach dem Ausgangsknoten der Kante als nächsten Knoten den Endpunkt der Kante bedient.

Diese Kanten seien als *Linienkanten* bezeichnet. Da zwei Umsteigeknoten durch mehr als eine Unterlinie direkt verbunden sein können, handelt es sich bei den Linienkanten um Multi-Kanten. Man beachte, daß Linieninformationen (d.h. die „Zusammenfassung“ bestimmter auf einer Unterlinie liegender Knoten) hier nicht vollständig repräsentiert sind. Wird dieser Graph für die Suche herangezogen, kann durch die lokale Expansion nicht auf die (durchgehenden) Linien Bezug genommen werden; unnötiges Umsteigen ist bei naivem Vorgehen die Folge (vgl. Abschnitt 3.2 und Abbildung 3.5).

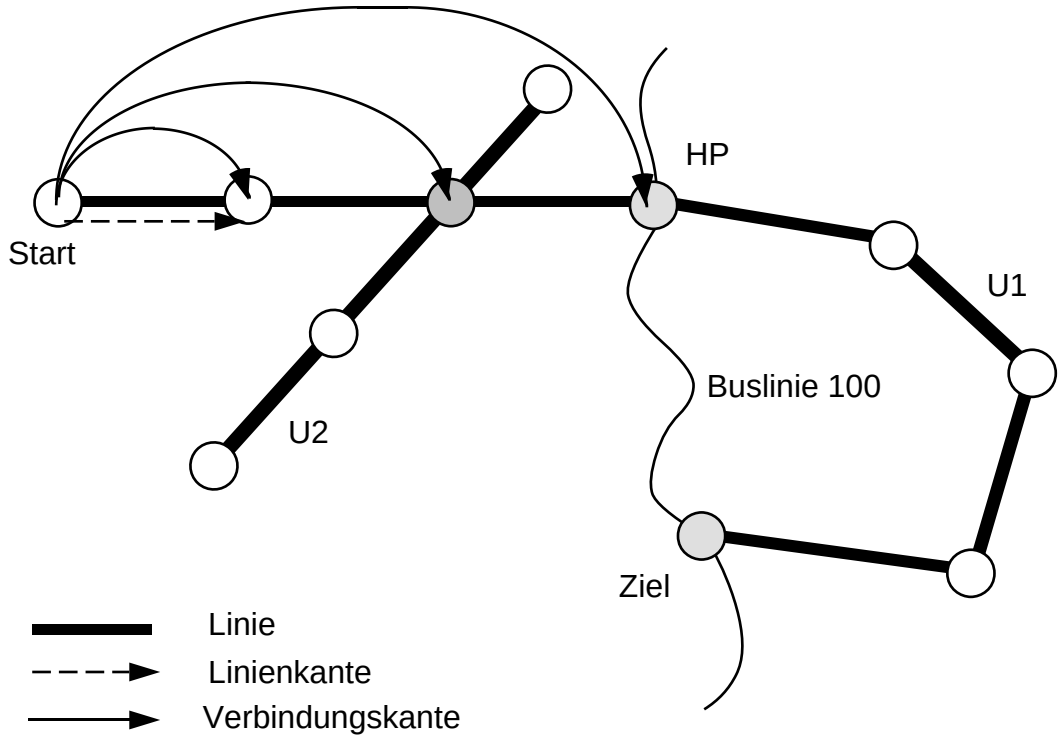

Abbildung 3.2: Linien und Verbindungskanten.

Die zweite Kantenart (durchgezogene Darstellung) beschreibt, daß ein Umsteigeknoten am Kantenendpunkt ohne Umsteigen vom Umsteigeknoten am Kantenausgangspunkt erreicht werden kann. Diese Kanten werden *Verbindungskanten* genannt. Beide Kantenarten sind gerichtet.

Man kann die Umsteigeknoten, die direkt durch eine Unterlinie verbun-

den sind, auch als Menge repräsentieren. In dieser Sichtweise ergeben sich die Umsteigemöglichkeiten von einer Unterlinie zur anderen als Schnitt dieser Mengen. Betrachten wir das Beispiel in Abbildung 3.3. Dort wird beschrieben, daß vom Knoten *Start* alle Umsteigeknoten, die der Menge $U1$ angehören, direkt erreicht werden können; u.a. auch das Ziel. Dieses wird auch von der Buslinie 100 bedient. Es wird das gleiche Verkehrsnetz wie in Abbildung 3.2 repräsentiert. Diese Struktur läßt sich nach [7] auch als Hypergraph deuten. In einem Hypergraphen wird die Graphen zugrundeliegende binäre Relation über Knoten, auf beliebige Teilmengen von diesen verallgemeinert. Die Menge $U1$ beispielsweise bildet eine Kante des Hypergraphen. Eine Hyperkante repräsentiert die von einem Knoten aus direkt erreichbaren Umsteigeknoten und entspricht damit einer Schar von Verbindungskanten, deren jeweilige Endknoten von einer Unterlinie verbunden werden. Die Verbindungskanten, die von einem Knoten ausgehen, sind also implizit durch die Haltepunkte der Unterlinien gegeben, die den Knoten bedienen.

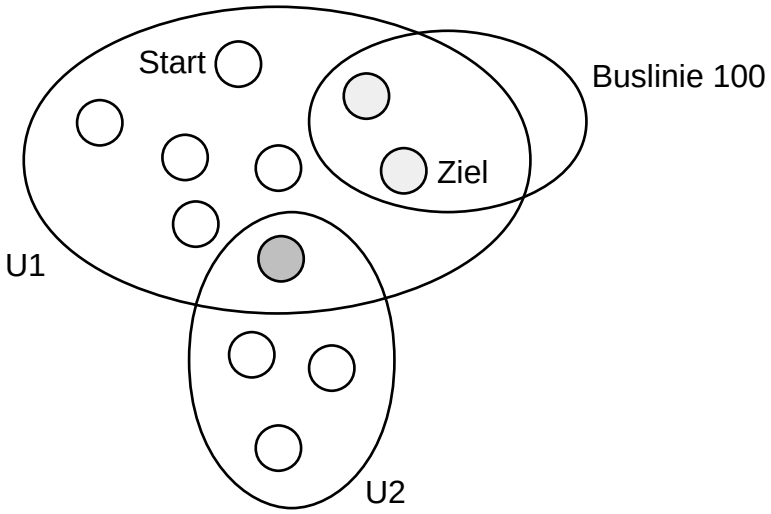

Abbildung 3.3: Der Suchraum des FIS interpretiert als Hypergraph.

Die Kostenbeschriftung der Linienkanten ist die minimale Fahrtdauer (bestimmt über alle Fahrten) vom Ausgangsumsteigeknoten zum Endumsteigeknoten. Die Kosten der Verbindungskanten können nicht a priori definiert werden, sondern ergeben sich während der Verbindungssuche aus den Fahrplänen und dem Zeitpunkt, an dem ein betreffender Umsteigeknoten frühestmöglich verlassen (bei Angabe einer Abfahrtszeit) oder spätestmöglich erreicht (bei Angabe einer Ankunftszeit) werden kann. Primär werden die Kosten der Verbindungskanten also durch die aktuelle Fahrtzeit vom jeweili-

gen Ausgangsumsteigeknoten zum Endumsteigeknoten der Kante bestimmt. Wie die Fahrtzeiten bei der Nachfolgergenerierung berücksichtigt werden, schildert der nächste Abschnitt.

Bestimmung der Nachfolger eines Suchraumknotens

Wie sehen nun die von dem Algorithmus A* bearbeiteten Knoten im Detail aus? Ein Problemzustand wird beschrieben durch:

- einem Umsteigeknoten,
- einer Zeitangabe (frühestmögliche Abfahrtszeit bzw. spätestmögliche Ankunftszeit, je nach Suchrichtung),
- einem Haltepunkt, an dem der Umsteigeknoten erreicht wurde, und
- einer Unterlinie, mit der dieser Haltepunkt des Umsteigeknotens erreicht wird.

In Abschnitt 2.1 wurde ein Umsteigeknoten als eine Menge von Haltepunkten definiert, die ohne besonderen Aufwand zu Fuß untereinander erreicht werden können. Wir gehen davon aus, daß die Daten zur Bestimmung der Umsteigeknoten einschließlich der Fußwegzeiten zur Verfügung stehen. Die Fußwegzeiten seien gegeben durch Tupel der Form: eingehende Unterlinie, angefahrener Haltepunkt, ausgehende Unterlinie, Ausgangshaltepunkt, Fußwegzeit.

Die Nachfolger eines Suchgraphknotens (bzw. eines Problemzustands) werden wie folgt bestimmt: Für jeden Haltepunkt des dem Suchgraphknoten zugeordneten Umsteigeknotens und für jede Unterlinie, die diesen Haltepunkt bedient, wird die frühestmögliche Abfahrt zu jedem über die Verbindungskanten ermittelten Nachfolge-Umsteigeknoten bestimmt (der Fahrplanzugriff wird in Abschnitt 4.1 beschrieben). Daraus ergibt sich eine Menge von Tupeln bestehend aus einem Umsteigeknoten, einer Ankunftszeit, einer Unterlinie und einem Haltepunkt, an dem die Unterlinie den Umsteigeknoten erreicht. Der Haltepunkt ist wichtig für die Berechnung der Fußwegzeit (innerhalb des Umsteigeknotens), welche die Abfahrtszeit der Expansion beeinflußt. Für jeden Umsteigeknoten filtert man das Tupel mit der frühesten Ankunftszeit heraus. Diese Tupel definieren dann die neuen Problemzustände. Jedem dieser Tupel wird dann durch die A*-Verwaltung ein Suchgraphknoten zugeordnet, durch die Kostenfunktion bewertet und in die OPEN-Liste entsprechend einsortiert. In die Bewertung der Knoten gehen verschiedene Kostenfaktoren ein. Als Einheit der Metrik der Kosten verwenden wir die Zeit. Als wichtigste Komponente geht die Zeitangabe des Knoten ein (umgerechnet auf eine

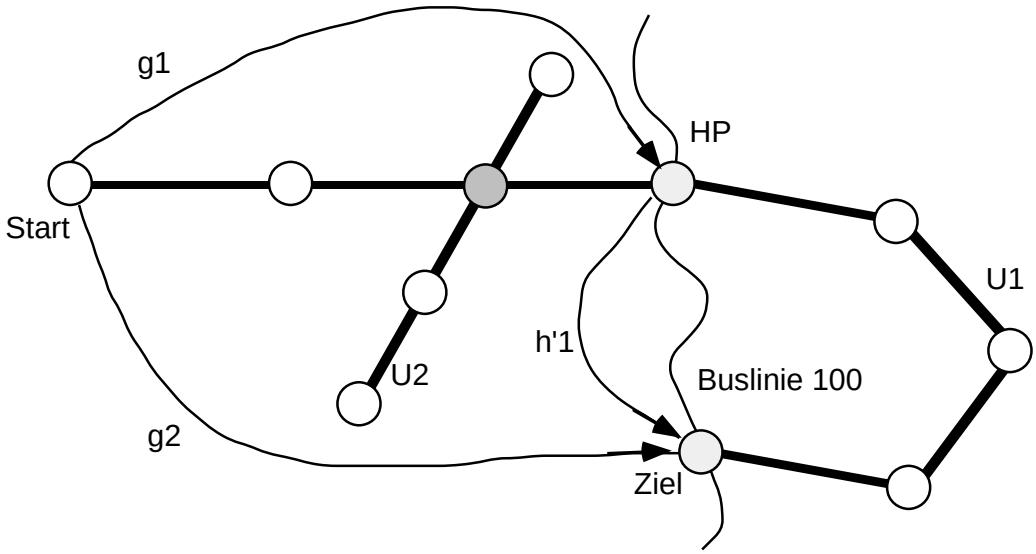

Abbildung 3.4: Verwaltung von Alternativen

Fahrtzeit). Weitere Kosten-„Faktoren“ werden in Abschnitt 3.2 und Kapitel 6 eingeführt.

Man beachte, daß durch die in diesem Abschnitt vorgestellte Expansionsart schon nach genau einer Expansion der Zielknoten gefunden wird, wenn er vom Startknoten aus direkt über eine Verbindungskante erreicht werden kann. Das heißt jedoch gemäß der A*-Verwaltung nicht, daß der aktuelle Weg zum Zielknoten sofort als optimaler Weg akzeptiert wird. Wenn über einen anderen Knoten durch die Abschätzung noch eine bessere Verbindung potentiell möglich ist, so stellt die Bewertung und Sortierung der Knoten gemäß A* sicher, daß dieser noch vor dem Zielknoten expandiert wird. Eventuell wird dann der Zielknoten über diesen betrachteten Knoten noch mit einer günstigeren Eintragung in der Agenda vermerkt (siehe Abbildung 3.4). Dieses ist der Fall, wenn gilt: $g_2 > g_1 + h'_1$.

Es hat sich als nachteilig erwiesen, die Nachfolge-Umsteigeknoten über die Linienkanten zu ermitteln. Wir verdeutlichen dies an einem Beispiel.

Wie schon einführend gesagt, unterscheidet man in einem zu modellierenden Verkehrsnetz evtl. zwischen verschiedenen Verkehrmitteltypen, die zwar auf der gleichen „Strecke“ fahren, aber verschiedene Geschwindigkeiten ermöglichen sollen (Eilzüge vs. Nahverkehrszüge oder Schnellbusse vs. „normale“ Busse). Zur Beschleunigung werden einige Haltepunkte der Strecke von den Eilzügen oder Schnellbussen nicht bedient (Abbildung 3.5). Wenn nun in dem in Abbildung 3.5 skizzierten Verkehrsnetz der dem Haltepunkt HP1

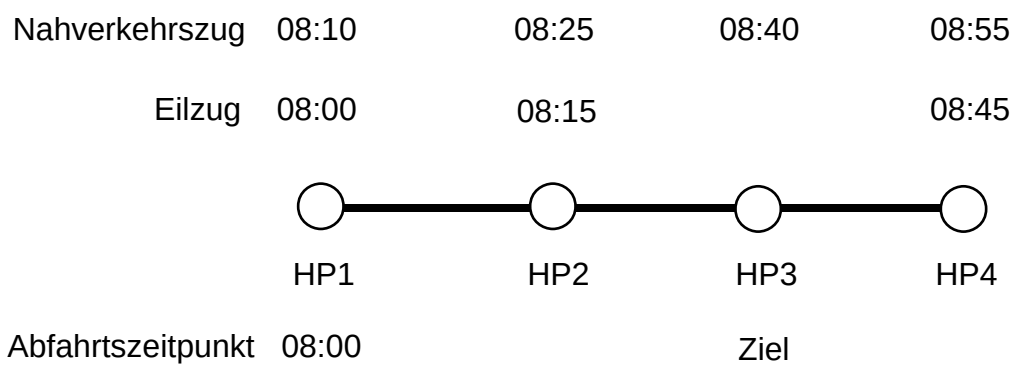

Abbildung 3.5: Beispiel für eine fehlerhafte Nachfolgergenerierung.

zugeordnete Suchgraphknoten expandiert werden soll (bei einer Abfahrtszeit von 8:00 Uhr) und Haltepunkt HP3 als Ziel angegeben wird, so entstehen Probleme, wenn man, bedingt durch die Linienkanten, nur einen Untergraphen bestehend aus den Knoten HP1 und HP2 betrachtet. Bei einer lokalen Expansion von HP1 könnte der Nachfolgehaltepunkt HP2 sowohl von einem Eilzug als auch von einem Nahverkehrszug bedient werden. Ist nun der Eilzug der frühestmögliche Nachfolger, so werden bei der lokalen Expansion die „Nahverkehrszugnachfolger“ bzgl. HP1 nicht generiert, der Zielknoten HP3 also evtl. nicht oder nur durch (unnötiges) Umsteigen (bei HP2) erreicht. Man müßte also auch einen Nachfolger für die Nahverkehrslinie generieren und verwalten. Die in diesem einfachen Beispiel geschilderte Problematik tritt auch bei sich verzweigenden Unterlinien auf. Bei der oben geschilderten, von uns gewählten Expansionsform der Knoten bereitet diese Konstellation keine Probleme, da HP3 ebenso als Nachfolger generiert wird (mit dem Nahverkehrszug als Unterlinie, mit der der Nachfolgerknoten erreicht wird). Bei der Expansionsform über die Verbindungskanten müssen allerdings weitaus mehr Suchgraphknoten auf der OPEN-Liste verwaltet werden. Symptomatisch ist etwa die Expansion eines Knotens, der einen Hauptbahnhof repräsentiert. Es werden sehr viele Nachfolger generiert, da vom Hauptbahnhof über die Verbindungskanten viele Umsteigeknoten erreicht werden. Die Generierung dieser Nachfolgerknoten erfolgt auch, wenn das Ziel direkt über eine Linienkante vom Hauptbahnhof erreicht werden kann.

Minimierung der Anzahl der Umsteigevorgänge

Umsteigevorgänge werden von jedem Fahrgast als unangenehm empfunden, wenn durch das Umsteigen nicht eine angemessene Fahrtzeitverkürzung er-

reicht wird. In unserer Anwendung des A*-Suchalgorithmus entspricht jede Expansion des Suchgraphen einem Umsteigevorgang. Daher kann man Umsteigevorgänge unterdrücken, indem man A-priori-Expansionskosten einführt.[2] Durch diese Modellierung wird die Minimierung von Umsteigevorgängen auf einfache Weise in das graphentheoretische Modell integriert. Wird ein Umsteigen mit Kosten von fünf Minuten bestraft, so werden alle Verbindungen vorgezogen, bei denen die Fahrtzeit nur fünf Minuten länger ist (sofern nicht noch andere Kosten einbezogen werden).

Integration von Fußwegzeiten in die Optimierung

In den Anfragen von Kunden werden häufig nicht direkt Haltepunktnamen, sondern vielmehr Straßennamen oder Namen von z.B. öffentlichen Gebäuden genannt. Wenn eine geographische Datenbasis in das FIS integriert ist, können aufgrund von Koordinatenangaben von Straßen, Gebäuden und Haltestellen die Fußwegezeiten vom angegebenen Startort (Zielort) zu den in Frage kommenden umliegenden Haltepunkten abgeschätzt werden.

Um auch die Fußwegzeiten vom Start zu den umliegenden Haltepunkten in die Suche integrieren zu können (vgl. Abschnitt 2.3), werden die Haltepunkte in der in Frage kommenden Startregion zu einem virtuellen Umsteigeknoten zusammengefaßt. Die Fußwegzeiten haben also einen ähnlichen Status wie die Fußwegzeiten in einem normalen Umsteigeknoten. Analoges gilt für die Fußwegzeiten von den potentiellen Haltepunkten in der Zielregion zum eigentlichen Ziel. Ein ähnliches Vorgehen bei der Behandlung einer Start- und Zielregion im Gegensatz zu einem Start- und Zielknoten wird auch in [23] vorgeschlagen.

Domänenspezifische Besonderheiten bedingt durch die Behandlung der Zeit

Wie schon im vorigen Abschnitt angedeutet, enthalten die Kosten g nicht nur die reinen Fahrtkosten, sondern auch Umsteige-Strafkosten, Verkehrsmittel-Bewertungskosten etc. Genaueres hierzu schildert Kapitel 6. Wenn im Standard-A*-Algorithmus ein weiterer Pfad zu einem Knoten auf der CLOSED-Liste gefunden wird, der bzgl. g einen günstigeren Wert hat, so werden diese neuen Kosten an die Nachfolger propagiert (siehe Abschnitt 3.1). Dieses ist in unserem Fall nicht ohne weiteres möglich, da durch die Nachfolgerfunktion nur Nachfolger bzgl. der frühestmöglichen (bzw. spätestmöglichen) Abfahrtszeit (bzw. Ankunftszeit) generiert werden. Wenn sich durch den neuen

[2] A-priori-Expansionskosten werden zu g addiert.

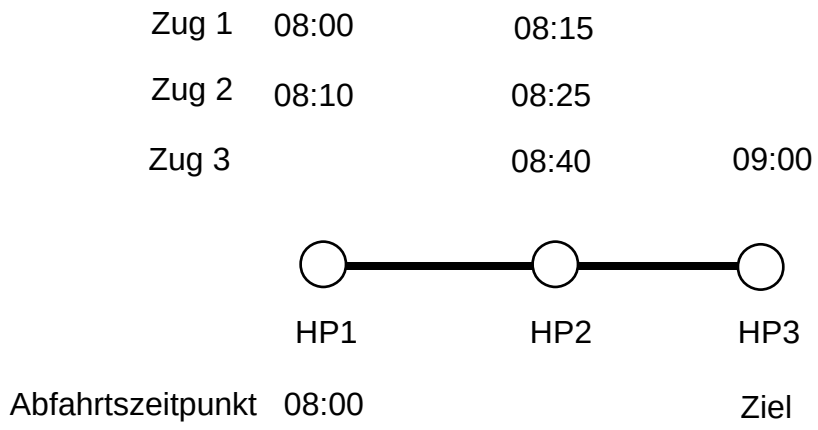

Abbildung 3.6: Zusammenwirken von Vorwärts- und Rückwärtssuche.

Pfad eine andere frühestmögliche Abfahrtszeit (Vorwärtsuche) bzw. eine andere spätestmögliche Ankunftszeit ergibt, entsteht in der FIS-Domäne evtl. ein völlig anderer Suchgraph.[3] Dieses gilt schon bei einer Differenz von nur einer Minute. Sollte man feststellen, daß ein schon expandierter Suchgraphknoten doch über einen günstigeren Pfad erreicht werden kann als bisher ermittelt[4], so bleibt i.a. nur die Möglichkeit, den Suchraum ausgehend von diesem Knoten erneut aufzuspannen. Es erfolgt also keine Propagierung der Kosten, sondern es wird ein neuer Suchgraphknoten auf der OPEN-Liste erzeugt. Der auf der CLOSED-Liste gefundene Knoten kann einschließlich aller seiner Nachfolger (auf der OPEN- oder der CLOSED-Liste) aus der Betrachtung herausgenommen werden, sofern man nur an einer (optimalen) Lösung interessiert ist.

Bei der reinen Vorwärtssuche kann durch die Suchstrategie, zunächst die frühestmögliche Abfahrt zu wählen, eine unnötige Wartezeit bei einem Umsteigeknoten auftreten. Abbildung 3.6 zeigt ein Beispiel anhand eines einfachen Fahrplans. Angenommen, der Abfahrtszeitpunkt bzgl. HP1 sei vom Auskunftsuchenden mit 8:00 Uhr angegeben. Das Suchverfahren würde den Zug 1 wählen (frühestmögliche Abfahrt) und um 8:40 Uhr mit dem Zug 3 ab HP2 weiterfahren. Es ergibt sich bei HP2 eine Umsteigewartezeit von 25 Minuten. Offensichtlich würde ein guter Fahrplanberater vorschlagen, nicht um 8:00 Uhr zu starten, sondern mit dem Zug 2 erst um 8:10 Uhr loszu-

[3]Die Sprechweise „entsteht ein anderer Suchgraph“ spiegelt die intensionale Sicht des (unendlichen) Suchgraphen wieder. Man kann auch sagen, es wird ein anderer Teilgraph relevant, der bisher nicht berücksichtigt wurde.

[4]Der Suchgraphknoten befindet sich dann auf der CLOSED-Liste.

fahren. Es ergibt sich die gleiche Ankunftszeit. Durch die Vorwärtssuche wird also zunächst nur die frühestmögliche Ankunft am Ziel bestimmt. Aufgrund dieser Information kann nun in einem Rückwärtssuchlauf bzgl. dieser frühestmöglichen Ankunftszeit eine spätestmögliche Abfahrtszeit bestimmt werden. Man kann sich leicht überlegen, daß eine ähnliche Situation auch bei einer Rückwärtssuche auftreten kann. Im Anschluß an eine Rückwärtssuche muß also durch eine Vorwärtssuche noch die Fahrtzeit minimiert werden. Das Verfahren braucht nicht angewendet zu werden, wenn schon beim „ersten Durchgang" die Wartezeit beim Umsteigen vernachlässigbar ist.

Die Abhängigkeit der Verbindung von der Fahrtzeit bildet einen wesentlichen Unterschied zu Navigationssystemen (siehe z.B. [21]). Dies ist in den von der Zeit abhängenden Kantenkosten begründet. Damit können optimale Verbindungen i.a. nicht „im voraus" ermittelt werden. Graphalgorithmen wie z.B. Ford-Moore [5], die Verbindungen von einem Startknoten zu allen anderen Knoten berechnen, oder Matrixalgorithmen, die alle minimalen Verbindungen berechnen, können daher nicht für die Verbindungssuche zur Anwendung kommen. Ein Matrixalgorithmus kann jedoch für die Berechnung des Schätzers für h' verwendet werden (vgl. Abschnitt 3.3).

Optimierung der Graphensuche

Um die Suchzeiten einzugrenzen, sollte man die Graphensuche optimieren. Dabei liegt das Hauptmerkmal in der Begrenzung des Suchraums auf Erfolg versprechende Knoten. Man muß zwischen zwei Arten der Optimierung unterscheiden. Zum einen gilt es, optimale Verbindungen zu bestimmen (und dafür garantiert der A*-Algorithmus), zum anderen ist es aber auch notwendig, den hierzu verwendeten Algorithmus, d.h. das Verfahren, das die Verbindungen ermittelt, handhabbar zu machen. Will man den Algorithmus A* optimieren, muß darauf geachtet werden, daß weiterhin eine optimale Lösung (Verbindung) gefunden wird. Andererseits ist es wünschenswert, möglichst wenige Knoten als Nachfolger eines zu expandierenden Knotens zu generieren, um OPEN nicht unnötig zu füllen. Ist eine obere Schranke für die Kosten von Start zum Ziel bekannt, so kann der Suchraum verkleinert werden, ohne Optimalitätskriterien zu verletzen (vgl. [23]). Darunter fallen im FIS z.B. solche Umsteigeknoten, die zwar durch eine Verbindungskante erreicht werden können, für die es aber keinen Eintrag im Fahrplan gibt (z.B. weil die Linie nur sonntags fährt); die Kosten sind größer als jede obere Schranke. Wählt man das h' eines gerade zu expandierenden Knotens als obere Schranke, schließt man so alle Nachfolger aus, die weiter vom Ziel entfernt sind. Da aber nur eine Schätzung vorliegt, darf man die Knoten nicht gänzlich verwerfen; stattdessen werden sie in einer zweiten Liste zurückgestellt (s.u.).

Man kann durch Auswahl aus OPEN auf das Ziel fokussieren, indem nicht nur nach f sortiert wird, sondern bei gleichem f zusätzlich h' als zweiter Sortierschlüssel verwendet wird. Anstelle von f werden dann bei der Auswahl aus OPEN die einzelnen Summanden g und h' betrachtet [23]. In unserem Fall kann man aus den Knoten, die über eine Linie erreicht werden, denjenigen mit kleinstem h' (bei gleichem f) wählen. Er kommt dem Ziel aufgrund der Schätzung am nächsten. Dies reicht aber noch nicht aus, um unerwünschte Expansionen zu vermeiden. Die Nachfolger haben gleiches oder größeres f und würden gemäß der Ordnung von OPEN hinten angestellt, die durch h' zurückgesetzen Knoten werden trotzdem gewählt. Die Suchgraphknoten, für die es bereits einen Nachfolger gibt (auf der Liste CLOSED), der mit kleinerem h' und gleicher Linie erreicht wird, können also aus OPEN herausgenommen werden. So kann die Expansion wie gewünscht fortfahren. Um aber keine mögliche Verbindung zu unterdrücken (z.B. dadurch, daß aufgrund der Fahrpläne keine Verbindung über die gewählten Knoten zu finden ist), müssen die Knoten zwischengespeichert werden (OPEN2). Soll ein Knoten aus OPEN gewählt werden, muß OPEN2 nach näher am Ziel liegenden Knoten durchsucht werden. h' des ersten Knotens von OPEN darf nicht größer sein als h' des ersten Knotens von OPEN2.

Man erkennt, daß diese Betrachtung einiger Konstrukte von A* lediglich für unseren Problembereich relevant ist. Es werden z.B. Linien- und Zeitinformationen als spezielle Merkmale der Suchgraphknoten und nicht (nur) der Problemzustände betrachtet und bei der Verwaltung der Suchgraphknoten berücksichtigt. Damit wird eine zielfokussierte Suche erreicht, ohne eine optimale Lösung zu gefährden.

Um die Suche zu optimieren, ist es sinnvoll, mit einer möglichst genauen Abschätzung zu arbeiten. Im bisherigen System werden für die Berechnung der minimalen Fahrtzeiten zwischen zwei benachbarten Knoten *alle* Fahrten herangezogen. Da aber die Verbindungen stark von der gewünschten Abfahrtszeit abhängen, sollte die Schätzfunktion auf diese abgestimmt sein. Dies kann erreicht werden, indem unterschiedliche Schätzer für verschiedene Fahrpläne erstellt werden. Es bieten sich z.B. Schätzer für Nachtfahrten (Nachtbusse fahren im Unterschied zu Bussen, die am Tage verkehren, andere Wege in anderen Rhythmen), Wochenend- und Werktagspläne an. Die Suche könnte so weiter verbessert werden. Der nächste Abschnitt schildert, wie eine Schätzfunktion bestimmt werden kann.

3.3 Das Gauß-Jordan-Verfahren

Für den A*-Algorithmus wird eine Funktion benötigt, die die optimale Fahrtzeit zwischen zwei beliebigen Umsteigeknoten abschätzt. Auf geographischen Informationen basierende euklidische Schätzer (etwa die Entfernung zu einem Zielhaltepunkt anhand der Luftlinie) haben den Nachteil, daß Fahrtzeiten z.B. bei bestimmten „mäandernden“ Buslinien evtl. erheblich unterschätzt werden. Durch Unterschätzungen wird wiederum der Suchraum unnötig vergrößert.

Die Schätzung sollte daher besser aus den zur Verfügung stehenden Fahrplaninformationen erfolgen. Um die Kosten von jedem Knoten zu jedem potentiellen Zielknoten ermitteln zu können, muß jeder Knoten mit jedem anderen über eine Kante mit entsprechender Kostenbeschriftung verbunden werden. In Abschnitt 3.2 wurde der Begriff der Linienkante und deren Kostenbeschriftung eingeführt (vgl. Abbildung 3.2). Diese Kanten bestimmen die minimalen Kosten (minimalen Fahrtdauern) zwischen zwei über eine Linie verbundenen Umsteigeknoten, wobei der Kantenendknoten direkt nach dem Kantenausgangsknoten bedient wird. Um nun einen Schätzer zu gewinnen, werden in den Graphen neue Kanten zwischen nicht über eine Linienkante direkt verbundenen Knoten eingefügt. Es wird ein Algorithmus benötigt, der die minimalen Kosten dieser noch nicht beschrifteten Kanten ermittelt.

Grundsätzlich läßt sich für diese Abschätzung auch der Algorithmus A* einsetzen. Ausgehend von jeden Umsteigeknoten wird mittels A* zu jedem potentiellen Zielknoten der kürzeste Weg bestimmt. Als vorläufiger Schätzer h'_1 zur Bestimmung des endgültigen Schätzers h' könnte der euklidische Abstand der Umsteigeknoten verwendet werden. Hierzu werden geographische Daten (Koordinaten) für die Haltepunkte benötigt. Für das Hamburger Nahverkehrssystem stehen uns die Koordinaten der Haltepunkte bzw. Umsteigeknoten (noch) nicht zur Verfügung, so daß für h'_1 der Wert 0 eingesetzt werden muß. Damit arbeitet A* wie ein Breitensuchverfahren [22]. Dieses Vorgehen ist bei der Größe und bei der engen Vermaschung des Suchraumes nicht handhabbar.

Es gibt jedoch andere, hierfür geeignete Verfahren z.B. aus der Optimierungstheorie (Dynamische Programmierung). Grundidee dabei ist, den durch die Umsteigeknoten und die Linienkanten definierten Graphen (Abbildung 3.2) als Adjazenzmatrix zu repräsentieren. Die Einträge a_{ij} dieser Matrix enthalten die minimalen Kosten (Fahrtzeiten) zwischen den Knoten i und j, wenn es eine Linienkante zwischen ihnen gibt. Kanten einer direkten Schleife ($i = j$) sind für unsere Abschätzung gleich 0. Wenn zwei Knoten nicht direkt über eine Linienkante verbunden sind, wird für a_{ij} ∞ eingesetzt. Diese Ausgangsmatrix wird nun weiterverarbeitet bis die gewünschten minimalen

Kosten ermittelt sind. Durch „Matrixmultiplikation“ (statt der Multiplikation wird die Addition, anstelle der Addition wird der Minimum-Operator verwendet) erhält man nacheinander Matrizen, die die kürzesten Pfade über einen, über zwei usw. Knoten beschreiben.

Ein solcher Algorithmus mit der Zeitkomplexität $O(n^3 log(n))$ wird in [1] skizziert. Für das Prototyp-FIS liegt n (Anzahl der Umsteigeknoten) zur Zeit bei 1286, wobei für das Hamburger Verkehrssystem im Endstadium etwa 2600 erwartet wird. Für $n = 1286$ hätte unsere erste Implementation dieses Algorithmus mehrere Stunden gebraucht. In einem Übersichtsartikel über Pfadprobleme in Graphen stellt Rote [25] u.a. das Gauß-Jordan-Verfahren vor. Dieser Algorithmus hat die Zeitkomplexität $O(n^3)$ und außerdem noch den Vorteil, daß er mit einer einzigen Matrix arbeitet (Platzkomplexität $O(n^2)$). In unserer Implementation auf einer SUN SPARCstation 1+ (in C) beträgt die Rechenzeit für $n = 1286$ ca. 35 Minuten.

Um das im FIS für die Berechnung der kürzesten Pfade zwischen allen Umsteigeknoten verwendete Gauß-Jordan-Verfahren zunächst informell zu verdeutlichen, betrachten wir das oben genannte Breitensuchverfahren. Der Grund für die Ineffizienz dieses Verfahrens liegt in dem Mangel, Zwischenergebnisse nicht weiter zu verwenden. Jeder Pfad wird ohne die Nutzung vorhergehender Ergebnisse berechnet. Sei z.B. *abcde* ein kürzester Pfad, so wird seine Berechnung nicht auf den kürzesten Pfad *abcd* zurückgeführt.

Genau dies nutzt die Methode der sukzessiven Approximation nach [1] aus. Seien a_{ij} die Einträge der Adjazenzmatrix A des Liniengraphen. Die minimalen Kosten x_{ij} zwischen beliebigen Knoten i und j sind nun gegeben durch $x_{ij} = min_{1 \leq k \leq n}(a_{ik} + x_{kj}, i \neq j)$. Der minimale Pfad zwischen den Knoten i und j ist also das Minimum der minimalen Kantenkosten von i zu allen Nachbarn k, plus der Kosten des minimalen Pfades von k nach j. x_{ij} ist damit auf ein bereits bekanntes Zwischenergebnis zurückgeführt.

Im Gauß-Jordan-Verfahren werden nun nicht zuerst alle Additionen ($a_{ik}+ x_{kj}$) ausgeführt und darüber das Minimum gebildet (was der oben erwähnten mehrfachen Matrixmultiplikation mit der Zeitkomplexität $O(n^3 log(n))$ entsprechen würde). Stattdessen wird das Minimum über dem Ergebnis der vorhergehenden Minimierung und den Kosten des um einen Knoten erweiterten Pfades ($a_{ik} + x_{kj}$) berechnet. Die Summanden brauchen daher nicht gespeichert zu werden. Das berechnete Minimum kann anstelle des bisherigen treten. Dies entspricht einer Änderung der Markierungen von Kanten im zur Matrix gehörenden Graphen. Das initiale Minimum ist durch A gegeben. Der Kern des Algorithmus sieht wie folgt aus:

for $k = 1$ to n do
    for $i = 1$ to n when $i \neq k$ do

for $j = 1$ to n when $j \neq k$ do
$a_{ij} := min(a_{ij}, a_{ik} + a_{kj})$

Das Verfahren berechnet also einen optimalen Pfad bezüglich der minimalen Fahrtdauer von jedem Umsteigeknoten zu jedem potentiellen Zielumsteigeknoten. Wenn ein neuer Umsteigeknoten hinzukommt, muß das Verfahren allerdings erneut für alle Umsteigeknoten angewendet werden, es arbeitet nicht inkrementell.

3.4 Kombinatorische Phänomene

Wie bereits erwähnt, ist die Fahrtroute von der Abfahrtszeit abhängig. Es ist jedoch zu beachten, daß die bzgl. der vorgegebenen Randbedingungen ermittelte Route eventuell schon bei geringfügig späterer Abfahrtszeit erheblich anders verläuft. Folgende Beispiele sollen dies verdeutlichen.

FIS ermöglicht es, eine Menge von Zielhaltepunkten anzugeben. Dies ist nützlich, wenn z.B. zwei Haltepunkte nahe beieinander liegen, aber von verschiedenen Linien bedient werden. Schon kleine Zeitverschiebungen führen so zu unterschiedlichen Verbindungen. Betrachtet man folgende, vom System berechneten Verbindungen für die Strecke von „Hochkamp“ nach „Wraust“ oder „Wraust, Johannssenstift“:

```
   8:30                  8:40                  8:50
Hochkamp              Hochkamp              Hochkamp
S1                    S1                    S1
Hbf                   Hbf                   Hbf
124                   120                   124
Wraust                Wraust, Johannssenstift  Wraust
```

„S1“ bezeichnet eine S-Bahnlinie, „120“ und „124“ sind Bezeichnungen von Buslinien. Während dieses Beispiel noch Regelmäßigkeiten erkennen läßt, sind diese bei unterschiedlichen Verkehrstagen nicht mehr zu finden. Für die Strecke „Rausdorfer Str. Mitte“ bis „Ohlstedt“ mit der frühestmöglichen Abfahrtszeit von 9 Uhr (in beiden Fällen!) erhält man:

```
  Werktags              Sonntags
   9:04                  9:57
Rausdorfer Str. Mitte  Rausdorfer Str. Mitte
364                   237
Trittau               Trittau
333                   333
```

Merkenstrasse	Glinde Markt
U3	263
Hbf	Wandsbek-Markt
U1	U1
Ohlstedt	Ohlstedt

Sonntags ist die Verbindung über die Busse schneller, da die U-Bahn weniger häufig fährt. In Wandsbek bekommt man eine U1, die am Hauptbahnhof nicht zu erreichen wäre.[5] Es ist also keinesfalls möglich, nur die Umsteigeknoten für eine Verbindung zu berechnen und dann diese für verschiedene Abfahrtszeiten zu verwenden. Durch die optimale Suche des A* ist gewährleistet, daß die dem Fahrplan inhärente Kombinatorik beachtet wird.

3.5 Intervalloptimierung

Die Beispiele in dem vorhergehenden Abschnitt machen deutlich, daß die Suche nach einer optimalen Verbindung ausgehend z.B. von einem Abfahrtszeitintervall nicht unproblematisch ist. Einmal liegt das Problem vor, die beste Verbindung innerhalb eines gegebenen Zeitintervalls zu finden („Ich will zwischen 8 und 9 Uhr ankommen."). Zum anderen sollen für den persönlichen Fahrplan alle Verbindungen innerhalb eines Intervalls ermittelt werden. Je nach Abfahrtszeit ergeben sich unterschiedliche Fahrtrouten. Es bieten sich zwei Lösungsmöglichkeiten an.

Die erste Möglichkeit besteht darin, als Startknoten des Suchgraphen alle Abfahrten in dem Intervall als entsprechende Suchgraphknoten auf die OPEN-Liste zu setzen. Der A*-Algorithmus wird dann alle Knoten in der oben geschilderten Weise expandieren und die jeweils günstigsten Verbindungen ermitteln. Der Nachteil liegt darin, daß bei dieser Lösung die in Abschnitt 3.2 geschilderte Problematik der späteren Abfahrtsmöglichkeit bei gleicher Ankunftszeit nicht berücksichtigt wird.

Die zweite Variante expandiert zunächst die frühestmögliche Abfahrt in dem Intervall zur Bestimmung der frühestmöglichen Ankunft vollständig durch, um anschließend hierzu durch Rückwärtsexpansion die spätestmögliche Abfahrt zu bestimmen. Diese kann evtl. später als die frühestmögliche Abfahrt liegen. Diese Abfahrtszeit ist die Abfahrtszeit einer ersten Lösung und bildet die untere Grenze eines neuen Abfahrtszeitintervalls. Das skizzierte Verfahren wird nun auf dieses neue Intervall angewendet, usw. Das

[5] Wer die Ergebnisse wirklich nachprüfen will, sehe in den Winterfahrplan 88/89 des HVV. Man beachte, daß sonntags die Abfahrt erst um 9:57 Uhr erfolgt, obwohl 9:00 Uhr als potentielle Abfahrtszeit angegeben wurde.

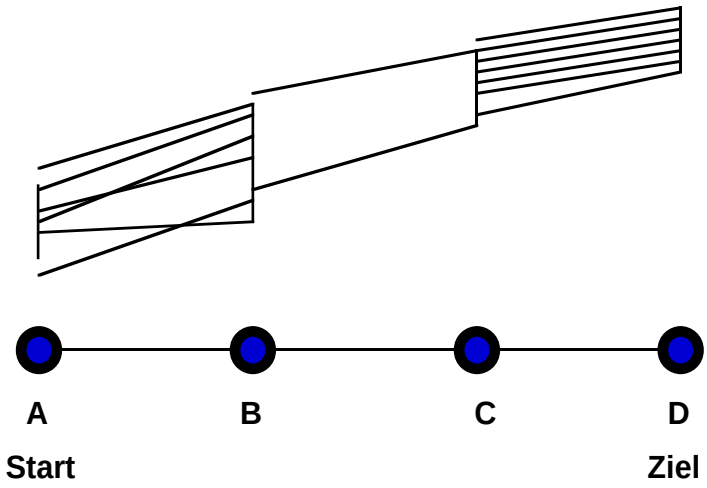

Abbildung 3.7: Intervallbehandlung bei Vorwärtssuche.

Verfahren terminiert, wenn das Intervall die Länge 0 hat. Die ermittelten Lösungen werden verglichen und diejenige mit den niedrigsten Kosten wird als beste in dem angegebenen Intervall ermittelt.

Beide Lösungen erhöhen den Suchaufwand erheblich, so daß eine Intervalloptimierung nicht bezüglich eines langen Zeitintervalls erfolgen kann, sondern nur bezüglich einiger Abfahrtsmöglichkeiten nach der frühestmöglichen Abfahrt und vielleicht auch einiger davor.

Zu jeder dieser Abfahrten von einem Umsteigeknoten liefert der A*-Algorithmus einen minimalen Pfad durch den Suchgraphen bestehend u.a. aus einer Sequenz von Umsteigeknoten. Im Sinne einer umfassenden Auskunft ist es angebracht, diese nun schon einmal berechneten Informationen dem Auskunftgebenden auch anzuzeigen. Insbesondere bei Nachfragen von Fahrgästen („Warum ist das denn günstiger als der Weg über ...“) sind auf diese Weise bessere Auskünfte bzw. Erklärungen möglich. Dies kann verknüpft werden mit einer Darstellung eines Ausschnittes des Fahrplanes nach Abbildung 5.3. Es handelt sich dabei um eine wenig zeitkritische Datenbasis-Zugriffsoperation. Wir sprechen daher nicht von einer Suche, sondern von einer Zusammenstellung von Fahrten.

Das Verfahren zur Bestimmung dieser Fahrplanausschnitte wird in den Abbildungen 3.7 und 3.8 skizziert. In der Vertikalen sind jeweils Zeitintervalle, in der Horizontalen die Verbindung dargestellt. Wir ordnen den Zeitintervallen jeweils die Namen der korrespondierenden Haltepunkte zu. Die Verbindungslinien zwischen den Zeitintervallen stellen Fahrten dar. Betrachten wir zunächst die Vorwärts-Zusammenstellung. Die Zusammenstellung bei vorge-

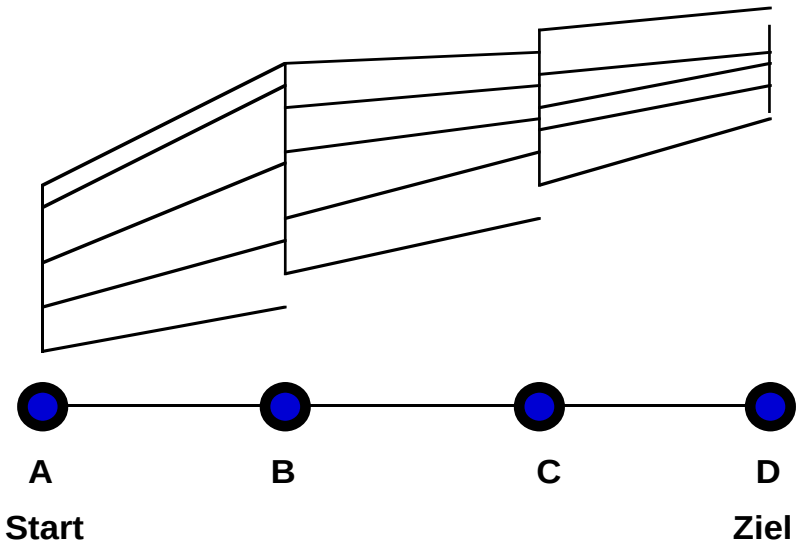

Abbildung 3.8: Intervallbehandlung bei Rückwärtssuche.

gebenem Ankunftsintervall (in Rückwärtsrichtung) erfolgt wiederum analog.

Ausgehend von einem vorgegebenen Abfahrtszeitintervall bei A werden alle Fahrten zum nachfolgenden Haltepunkt B einschließlich der spätestmöglichen Fahrten vor dem Intervall und der frühestmöglichen danach gesammelt. Durch die Ankunftszeiten dieser Fahrten bei B ist ein neues Zeitintervall definiert. Es werden alle Fahrten zu C, die in diesem Intervall B verlassen und nach C führen, gesammelt. Weiterhin wird die frühestmögliche Fahrt nach diesem Intervall betrachtet.[6] Dieses Verfahren wird fortgesetzt, bis das Ziel erreicht ist. Die Fahrten werden, wie in Abbildung 5.3 gezeigt, angeordnet. Für jede Lösung im Sinne der Intervalloptimierung (s.o.) kann ein solches Dialogfenster angezeigt werden. Die jeweils beste Lösung wird hervorgehoben. Zur Bestimmung der Anordnung und Synchronisation der Fahrten zu entsprechenden Verbindungen siehe Abschnitt 5.2.

3.6 Resümee

In den vorigen Abschnitten wurden zwei unterschiedliche Ansätze zur Bestimmung von optimalen Pfaden in Graphen vorgestellt. Der Gauß-Jordan-Algorithmus bestimmt den minimalen Pfad für alle Knotenpaarkombinatio-

[6] Die spätestmögliche davor kommt nicht in Betracht, da hierfür offensichtlich kein „Zubringer“ existiert.

nen. Der Algorithmus eignet sich zur Bestimmung eines Schätzers für die Fahrtzeit zu jedem potentiellen Ziel; er ist naturgemäß aufwendig (Zeitkomplexität $O(n^3)$). Dieses ist jedoch in der FIS-Anwendung nicht problematisch, da er in einer Vorlaufphase („off-line“) angewendet wird.

Weiterhin haben wir gezeigt, daß es möglich ist, einen wohlverstandenen Algorithmus zur Graphensuche (A*) anzuwenden, um unter Berücksichtigung von Fahrt- und Umsteigewartezeiten einen „kürzesten“ Pfad von einem Startknoten zu einem Zielknoten („on-line“) zu finden. Der vorangegangene Abschnitt machte deutlich, daß ein kombinatorisch vollständiges Vorgehen in dieser Anwendung notwendig ist, um gewünschte optimale Verbindungen bestimmen zu können. Fahrgastspezifische Optimalitätskriterien werden in Kapitel 6 geschildert.

Aufwendige Operationen wie etwa Intervalloptimierung können optional eingesetzt werden. Wenn ein hoher Durchsatz erforderlich ist, sollte nur ein Abfahrts- bzw. Ankunftszeitpunkt (im Gegensatz zu einem Intervall) angegeben werden. Ein Beispiel wäre der Auftrag eines Werkes, in dem für alle Mitarbeiter (> 2000) Nahverkehrsverbindungen zur An- und Abreise jeweils zum Schichtbeginn bzw. Schichtende für eine Fahrplanperiode „über Nacht“ automatisch berechnet und ausgedruckt werden sollen.[7] Gleiches gilt für die Verbindungen von Fahrkartenabonnenten.

Allgemeine Heuristiken eines erfahrenen Auskunftgebenden wie etwa „Fahren Sie mit dem Bus zur nächsten U-Bahn-Station, dann bis ins Zielgebiet mit der U-Bahn, und von da ab nehmen Sie dann den Bus“[8] haben wir nicht in das Suchverfahren integriert. Sie sind erstens i.a. nicht korrekt (optimal) und zweitens nicht immer einsetzbar. Wenn U-Bahnen nicht verkehren (z.B. in der Nacht), muß eine kombinatorische Suche ohnehin durchgeführt werden (Nachtbusse). Die Integration von räumlichem Wissen mittels Heuristiken dieser Art ist sogar äußerst komplex, weil man die Übergangszeiten z.B. von einem Nachtbus zur tagsüber verkehrenden U-Bahn berücksichtigen muß. Weiterhin ist nicht sichergestellt, daß ein solches Vorgehen überhaupt Geschwindigkeitsvorteile bei der Verbindungsbestimmung bieten kann.

[7] Dieses ist durchaus realistisch und wird vom HVV in Erwägung gezogen.

[8] Wir zitieren gerne ein Bonmot eines HVV-Vertreters, der diese Strategie als die „Bringe-die-Fahrgäste-möglichst-schnell-unter-die-Erde-Strategie“ bezeichnet hat.

Kapitel 4

Interne Datenrepräsentation

Nachdem die vorangehenden Kapitel einen Überblick über die Suchalgorithmen gegeben haben, betrachten wir nun die Fahrplanrepräsentation. Eine Repräsentation von Daten, gerade in einer so zeitkritischen Domäne wie der Fahrplanauskunft, kann nicht ohne Berücksichtigung der Verarbeitungsprozesse erfolgen. Verarbeitungsprozesse sind in erster Linie Anfrageoperationen. Ein Beispiel ist die Bestimmung der frühestmöglichen Abfahrt von einem Haltepunkt gemäß einer vorgegebenen Zeitangabe. In Abschnitt 2 haben wir auf die Möglichkeit und Notwendigkeit einer Kompression der Fahrplandaten hingewiesen. In dem in diesem Bericht vorgestellten Prototypsystem wird etwa die Hälfte der Daten des Hamburger Nahverkehrssystems verwaltet:

- 1442 Haltepunkte,
- 1286 Umsteigeknoten,
- 102 Linien,
- 697 Unterlinien,
- 16772 Fahrten mit im Mittel ca. 20 bedienten Haltepunkten.

Bei der Repräsentation der Fahrplandaten sind also zwei (nicht voneinander unabhängige) Anforderungen zu koordinieren: auf der einen Seite werden effizienten Zugriffsoperationen benötigt, auf der anderen Seite ist auf möglichst gute Ausnutzung des Speicherplatzes zu achten.

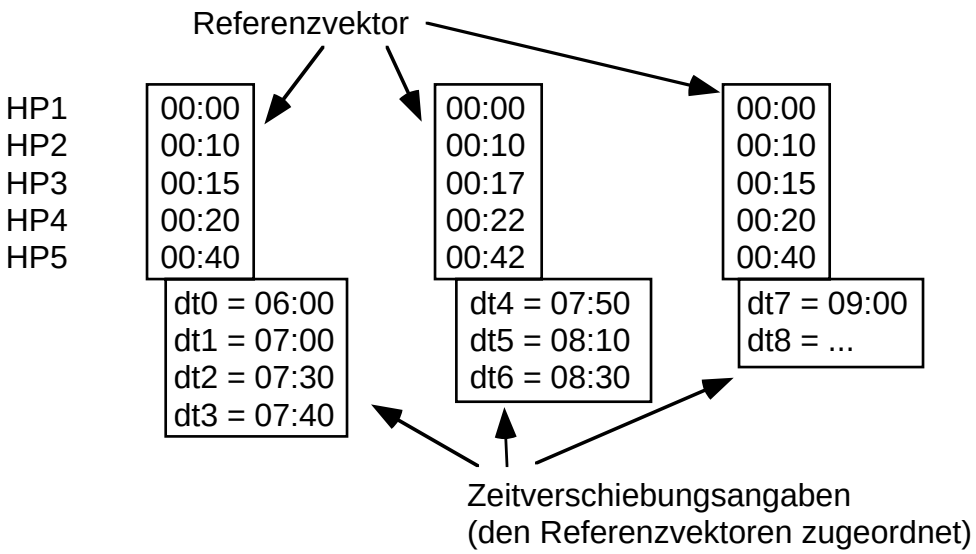

Abbildung 4.1: Skizze des Repräsentationsschemas für Fahrplandaten

4.1 Komprimierte Darstellung der Fahrplandaten

Das grundlegende Repräsentationsschema für die Fahrplandaten ist in Abbildung 4.1 skizziert. Die Darstellung zeigt den Fahrplan einer Unterlinie, die fünf Haltepunkte (HP1 bis HP5) in der durch die Indizes angedeuteten Reihenfolge bedient. Durch sogenannte Referenzvektoren wird beschrieben, wie lange ein Verkehrsmittel für den Weg von einem Haltepunkt zum nachfolgenden braucht. Die Zeiteintragungen in einem Referenzvektor beginnen jeweils bei 0:00 Uhr beim ersten Haltepunkt. Die Zeitangaben der folgenden Haltepunkte repräsentieren die jeweilige normierte Ankunftszeit des Verkehrsmittels an den Haltepunkten. Die Fahrtzeiten zwischen den einzelnen Haltepunkten sind jedoch nicht immer gleich, so daß mehrere Referenzvektoren verwendet werden müssen. In der Hauptverkehrszeit sind z.B. Busse durch die höhere Verkehrsdichte meist langsamer als zu verkehrsärmeren Zeiten. Referenzvektoren kennzeichnen also jeweils gewisse Klassen von Fahrten.

Zur Festlegung der tatsächlichen Ankunftszeit bei einem Haltepunkt ist jedem Referenzvektor eine Menge von Zeitverschiebungsangaben dt zugeordnet. Eine Fahrt ergibt sich aus der Addition der Zeiten aus einem Referenzvektor und einer Zeitverschiebungsangabe dt.

Im Beispiel in Abbildung 4.1 werden drei Referenzvektoren mit den zugeordneten Zeitverschiebungsangaben dargestellt. Es sei angenommen, daß die

Verschiebungsangaben von links nach rechts größer werden. Die Fahrten, die sich aus dem zweiten Referenzvektor ergeben, dauern zwei Minuten länger (längere Fahrtzeit zwischen HP2 und HP3). Man beachte, daß der dritte Referenzvektor wieder mit dem ersten identisch ist und daher nur einmal repräsentiert werden muß.

Zusätzlich zu der hier geschilderten Grundidee muß noch berücksichtigt werden, daß bei einer Verschiebungsangabe dt evtl. ein Aufenthalt in einem Haltepunkt repräsentierbar sein muß. Weiterhin enthalten die Zeitverschiebungsangaben dt Informationen über den Verkehrstag (z.B. Werktag, Samstag, Sonntag, Feiertag). Der HVV-Fahrplan unterscheidet zwischen mehr als zehn Arten von Verkehrstagen. Die interne Datenrepräsentation aller Fahrplandten benötigt bei der hier skizzierten Repräsentationsform inklusive einer Zielschätzmatrix ca. 6 Megabyte Speicher.

Beim Zugriff auf die Fahrplandaten ist ein Haltepunkt H und eine Zeit t gegeben. Unter Berücksichtigung der Zeitverschiebungsangaben werden die Fahrten durchlaufen, wobei auf den Referenzvektor mit Hilfe des zu H gehörenden Indexes zugegriffen wird. Bei der Vorwärtssuche sind Fahrten gesucht, die frühestmöglich nach t an dem Haltepunkt H abfahren, im Falle der Rückwärtssuche solche, die spätestmöglich vor t an dem Haltepunkt H ankommen.

Weiterhin muß darauf geachtet werden, daß bei bestimmten Linienführungen (vgl. Abschnitt 2) in den Referenzvektoren für mehrmals bediente Haltepunkte mehrfache Zeiten eingetragen sind. Zu Haltepunkt H gehören dann mehrere Indizes, die nach und nach berücksichtigt werden. Allerdings brauchen keine Fallunterscheidungen bzgl. der speziellen Linienführung (Q-Linie, Kreis etc.) gemacht zu werden.

4.2 Standard-Datenbankanschluß

Massendaten werden üblicherweise mit Datenbanksystemen verarbeitet. Auch der Fahrplan-Datenbestand des HVV wird in einem relationalen Datenbanksystem verwaltet. Bisherige Ansätze von automatisierten Auskunftssystemen (siehe Kapitel 7) haben gezeigt, daß der direkte Zugriff auf die Daten in der Datenbank bei der Verbindungssuche zu langsamen Antwortzeiten führen kann.[1]

Bei der heuristischen Graphensuche wird der Suchfokus sehr häufig gewechselt (z.B. „Best-First-Suche“). Dieses ist problematisch bei einem Datenbankzugriff, der einen Engpaß darstellt. Bei einem FIS kann nicht, wie es

[1]Bei den bisherigen Systemen wird das Suchverfahren ad hoc an das Datenrepräsentationsschema angepaßt; ein Weg, den wir nicht beschreiten wollen.

in „klassischen“ Anwendungen der Fall ist, ein gewisser Datenbestand nach einem Suchkriterium in den Hauptspeicher geladen, verarbeitet und das Ergebnis anschließend wieder in die Datenbank eingetragen werden. Es werden während der Suche Fahrplaninformationen bzgl. beliebiger Linien und Haltepunkte benötigt. Eine angepaßte physikalische Datenbankorganisation ist schwerlich möglich, durch lange Zugriffszeiten werden auch die Suchprozesse erheblich verlangsamt. „Ineffizienzen“ beim Zugriff sind nicht nur ärgerliche, einfach zu behebende „Implementationsprobleme“, sondern stellen das ganze Projekt in Frage, da in einem Suchraum wie dem des Hamburger Nahverkehrssystems eine Suchanfrage bei langsamen Datenzugriffen leicht einige Minuten dauern kann. Anfragezeiten von mehreren Minuten sind besonders bei einer (telefonischen) Beratung nicht tolerierbar.

4.3 Deduktive Datenbanken

In der ersten Projektphase wurde ein deduktives, in ein Prolog-System integriertes Datenbanksystem (siehe auch [8]) verwendet, um in einer Prototypimplementation des A*-Algorithmus nach [2] die Möglichkeit einer Integration einer deduktiven Datenbank zu testen. Deduktive Datenbanken sind relationale Datenbanken, die jedoch bestimmte Relationen virtuell verwalten [18]. Virtuelle Relationen werden intensional durch „Regeln“ definiert, welche die Bedingungen spezifizieren, die für Tupel gelten, die der Relation angehören. Die „Regeln“ werden dabei in einer logischen Sprache formuliert. Wir hatten die Hoffnung, durch diese Art der intensionalen Modellierung Fahrplandaten und insbesondere die angesprochenen Regelhaftigkeiten der Fahrten einfach kodieren zu können.

Bei der Kopplung von relationalen und logischen Formalismen ist zu beachten, daß bei Datenbankanwendungen die Relationen meist als solche verknüpft werden, während bei einer logischen Sicht mehr die Verarbeitung von Tupeln im Vordergrund steht. Der Zugriff auf Tupel ist nur effizient, wenn sie im Hauptspeicher gehalten werden. Sind sie, wie bei Datenbanksystemen erforderlich, in einer Datei (eventuell in Cache-Speichern) gespeichert, so sollte der Zugriff durch besondere Selektionsprädikate unterstützt werden. Diese Vorgehensweise ist aus der Datenbankwelt wohlbekannt: Zugriffe auf relational repräsentierte Datenbestände werden meist durch Sprachen wie SQL (Structured Query Language) unterstützt. SQL ermöglicht eine Spezifikation von mächtigen Zugriffsprädikaten. Das folgende Beispiel gibt nur einen kleinen Eindruck: `SELECT EMPLOYEE WHERE NAME = 'MEYER' AND SALARY > 5000`. Man beachte, daß der Junktor `AND` des Zugriffsprädikats eine logische Verknüpfung darstellt. Bei einer Kopplung von Logikbe-

weisern und Datenbanksystemen müssen also beim Zugriff auf Terme, die in einer Datenbank gehalten werden, auch Möglichkeiten zur Spezifikation solcher Zugriffsprädikate vorgesehen werden. Um nun Transparenz beim Zugriff zu ermöglichen, also die explizite Angabe von Zugriffsprädikaten zu vermeiden, werden solche Zugriffe über Regeln (Hornklauseln) verdeckt. Im Sinne der Logik ist es dann gleichwertig, eine Menge von Hornklauseln zum Beweis einer Formel heranzuziehen oder die entsprechenden Fakten. Wenn also die Domäne nahelegt, bestimmte, in konventionellen Systemen explizit gespeicherte Tupel durch Regeln erst „zur Laufzeit“ zu berechnen und dieses transparent gestaltet werden kann, dann kann eventuell eine erhebliche Speichereinsparung erzielt werden. Dieses ist einer der wesentliche Gesichtspunkte der Kopplung von logischen Systemen mit Datenbanksystemen: statt einer extensionalen Modellierung im klassischen Datenbanksystem, können bestimmte Teile intensional mit logischen Formeln, die den Tupelraum erst zur Laufzeit nach vorgegebenen Parametern aufspannen, repräsentiert werden. Eine Modellierung von Relationen mithilfe von Regeln (Hornklauseln) bietet also ggf. eine erhebliche Speicherersparnis.

Das Äquivalent zu dem `SELECT`-Ausdruck in SQL zur Berechnung der Relation `high-wages-employee` (s.o.) sieht in Hornklauselform wie folgt aus:

```
high-wages-employee(Name) :-
   employee(Name, Salary, Address, ...),
   Salary > 5000.
```

Eine Beweisanfrage im Sinne der Logik, in der ermittelt werden soll, ob ein Angestellter `'MEYER'` ein Spitzenverdiener ist, wird durch den Term `high-wages-employee('MEYER')` formuliert. Das Prolog-System verwendet zum Beweis die obige Regel. Innerhalb des Regelrumpfes wird für `Name` `'MEYER'` eingesetzt. Nach dem Beweis von `employee('MEYER', Salary, Address, ...)` ist an `Salary` das Gehalt von `'MEYER'` gebunden. Nun wird der Term `Salary > 5000` überprüft, wobei für `Salary` der durch den Beweis des `employee`-Terms erhaltene Gehaltsterm eingesetzt wird.

Eine Anfrage `high-wages-employee(Who)` liefert für `Who` alle Angestellten der virtuellen Relation `high-wages-employee`. Hier wird deutlich, daß die obige Definition der Relation bei dieser Anfrageart den Nachteil hat, erst alle Elemente der Relation `employee` zu betrachten, nur um durch das Prädikat `>` wenige herauszufiltern.[2] In dem im Projekt FIS zu Testzwecken verwendeten deduktiven Datenbanksystem kann das Prädikat `high-wages-employee` wie folgt definiert werden:

[2] Der Compiler führt hier keine Optimierungen durch.

```
high-wages-employee(Name) :-
   retr_tup(employee, [Name, Salary | Rest], Salary > 5000).
```

Diese Notationsform hat den Nachteil, daß `employee` explizit als Datenbankrelation verwendet wird (`retr_tup` steht für „retrieve tuple“). Die relationale Modellierungsweise des Datenbankteils ist außerdem recht unflexibel bei der Darstellung von dynamischen Datenstrukturen (z.B. Listen beliebiger Länge, siehe die Listen mit Zeitverschiebungsangaben in Abbildung 4.1).

Der Prolog-Beweiser läßt sich in gewisser Weise als Suchverfahren interpretieren (Backtracking). Der Beweis-Mechanismus von Prolog läßt sich allerdings nicht direkt zur Graphensuche verwenden. Es sind Verwaltungsinformationen über die expandierten Knoten notwendig (OPEN- und CLOSED-Listen), um eine Endlosrekursion zu verhindern. In einer Implementation der Graphensuche von Bratko [2] werden diese Listen mehr oder weniger geschickt durch die an Backtracking orientierte Such- bzw. Beweismaschinerie des Prolog-Systems verwaltet. Bei der Suche findet ein häufiger Fokuswechsel statt, der nicht einfach in (reinem) Prolog ausgedrückt werden kann. Weiterhin scheint die komplizierte Prolog-Implementation einer Erweiterung und einem Ausbau der Suchsystems abträglich zu sein.

Es handelt sich bei Prolog um einen rückwärtsverkettenden Top-down-Beweiser, der eine Tiefensuche mit Backtracking durchführt. Zur Kopplung von logischen und Datenbanksystemen wurde auch ein Bottom-up-Vorgehen vorgeschlagen. Bottom-up-Beweisverfahren sind vergleichbar mit vorwärtsverkettenden Inferenzsystemen. Sie haben meist den Nachteil der fehlenden Zielfokussierung, bieten aber den Vorteil, daß eine „Linksrekursion“ (vgl. LL und LR Parser) keine Probleme bereitet. Als Abhilfe wurde vorgeschlagen, die Terme (Regeln), die das Beweisverfahren verwendet, unter Berücksichtigung des zu beweisenden Terms (des Ziels) so umzuschreiben, daß eine Zielfokussierung auch in das Bottom-up-Beweisverfahren integriert wird (Rule-Rewriting). Näheres hierzu findet sich in [4] und [3]. Beispiele enthält auch [18]. Rewriting-Verfahren sind für eine Anwendung, die über ein Prototypstadium hinausgehen soll, noch nicht weit genug ausgereift.

4.4 Zusammenfassung der Datenrepräsentationsaspekte

Zusammenfassend läßt sich sagen, daß Standard-Datenbanken aufgrund von Performanzproblemen nicht verwendet werden können. Auch Ansätze mit einer deduktiven Datenbank und einem logischen Programmiersystem erwiesen sich als nicht tragfähig. Der Beweiser eines logischen Programmiersystems

(insbesondere Prolog) beinhaltet einen impliziten Suchmechanismus. Leider ist dieser nicht direkt für eine Graphensuche einsetzbar. Die an sich elegante Kopplung des logischen Systems mit einer relationalen Datenbank (deduktive Datenbank) wird durch den Nachteil der recht unflexiblen relationalen Modellierung wieder aufgewogen. Weiterhin zeigte das Prototyp-System einer deduktiven Datenbank noch keine ausreichende Geschwindigkeit beim Zugriff auf Daten auf Sekundärspeichern.

Aufgrund dieser Erfahrungen haben wir uns dafür entschieden, die Fahrplandaten bei komprimierter Datenhaltung im virtuellen Speicher zu verwalten. Da weiterhin eine benutzerfreundliche Schnittstelle unter Verwendung von Graphik zu entwickeln war und sich für diesen Bereich ein objektorienter Programmierstil besser eignet, wurde als Implementationssprache für das FIS Common Lisp mit dem Common Lisp Object System [26], [16] gewählt.[3] Konzepte der Benutzungsschnittstelle werden im nächsten Kapitel vorgestellt.

[3]In diesem Zusammenhang wäre die Integration einer objektorientierten Datenbank für CLOS-Instanzen eine interessante Erweiterung. Ein solches Programmodul stand uns nicht zur Verfügung.

Kapitel 5

Präsentation und Interaktion

Zur Interaktion mit dem Auskunftssystem FIS steht dem Kundenberater eine graphische Benutzungsschnittstelle zur Verfügung. Wir erläutern deren Funktionalität anhand einiger Beispiele.[1] Weiterhin schildern wir einen Algorithmus, der es ermöglicht, Fahrten mehrerer Teilstrecken innerhalb eines Dialogfensters zur Verbindungsbeschreibung synchronisiert darzustellen. Anschließend geben wir einen Überblick über die System-Architektur.

5.1 Konzeption der Benutzungsschnittstelle

Nehmen wir an, ein Kunde will von „Kellinghusenstraße“ bis „Veddel“ fahren. Er möchte ca. um 10:00 Uhr abfahren (Abbildung 5.1). Das System bietet dem Auskunftgebenden eine Detailübersicht über das Start- und das Zielgebiet. Die Detailübersicht soll im FIS-Endsystem auch Straßen und andere Stadtplaninformationen umfassen.[2]

In diesem Beispiel ist die Wahl des Umsteigehaltepunkts (Sternschanze) recht eindeutig. Das FIS schlägt zwei verschiedene Verbindungen vor (Abbildung 5.2).[3]

Es erwies sich als günstig, gleich einen kleinen Fahrplanausschnitt in Form von mehreren Verbindungen zu ermitteln, da Kunden häufig auch nach der

[1] Dieser Bericht stellt kein Handbuch zur Benutzung des FIS dar. Interaktionstechniken werden daher nur geschildert, wenn sie von allgemeinem Interesse sind und besondere Erfahrungen mit dem Prototypen kennzeichnen.

[2] Man beachte, daß die Darstellung auf dem Bildschirm in Farbe erfolgt, die verschiedenen Linien also eindeutig voneinander unterschieden werden können. Die Schwarz-Weiß-Darstellung in diesem Bericht kann nur einen ungefähren Eindruck vermitteln. Der Auskunftgebende hat die Möglichkeit, sich Linienverläufe hervorheben zu lassen.

[3] Die Verbindungsbestimmung erfolgt nach dem Anklicken der Schaltfläche (button) „Verbindung“. Die Darstellung der Fahrtzeit erfolgt in Minuten in dem unteren Fenster.

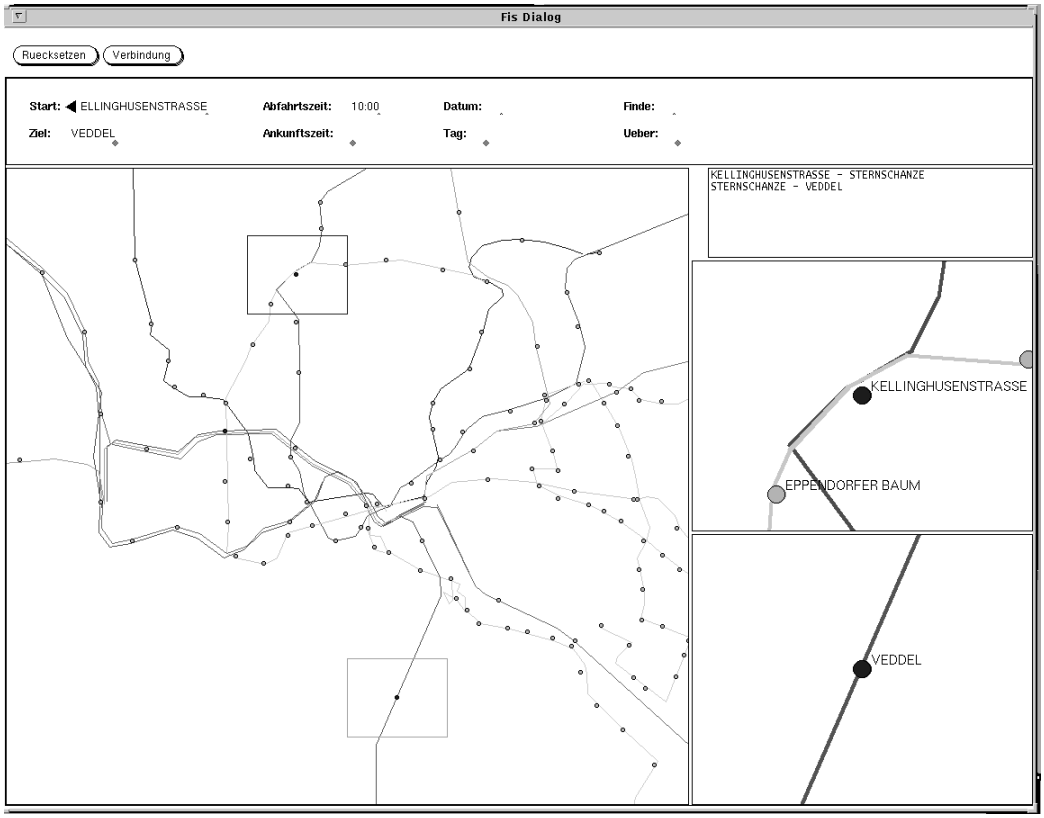

Abbildung 5.1: FIS-Eingabefenster, Übersichtsdarstellung des relevanten Ausschnitts des Gesamtsystems sowie Detail-Darstellungen für den Start- und den Zielbereich.

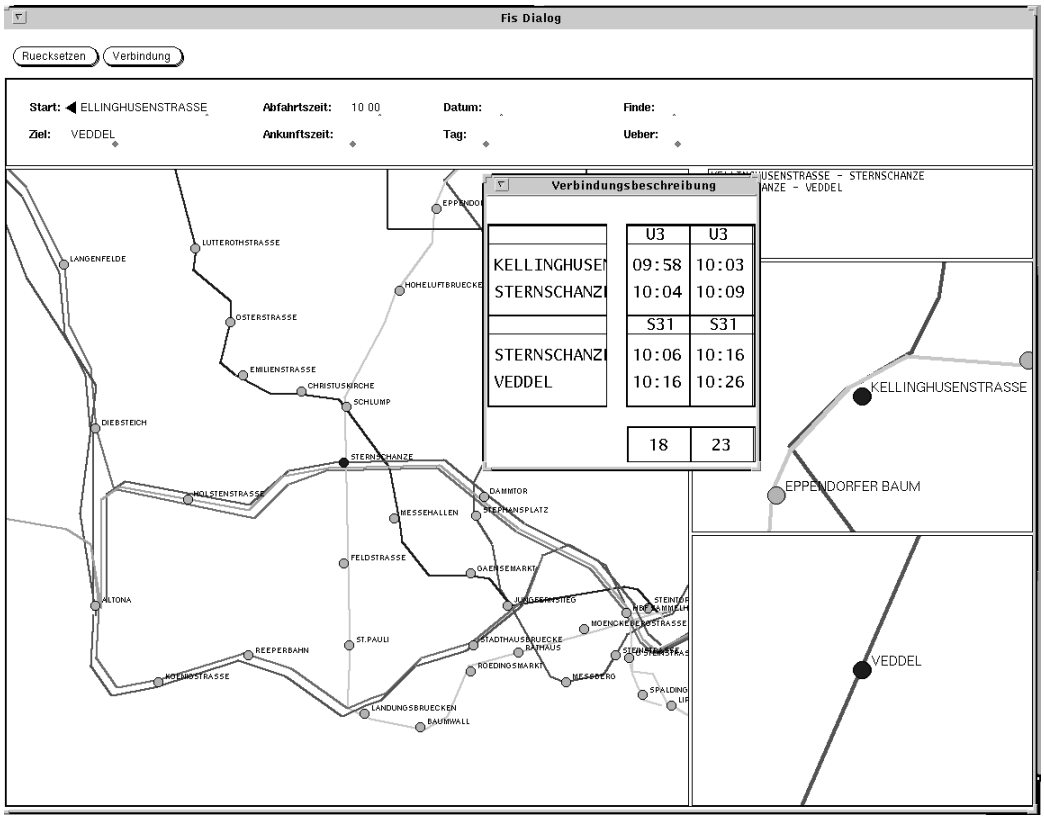

Abbildung 5.2: Verbindungsbeschreibung.

zeitlich nächsten Möglichkeit fragen. Gibt man einen Abfahrtszeitpunkt an, wird eine Fahrt vor dem Zeitpunkt und eine Fahrt nach dem Zeitpunkt herausgesucht. Es ist auch möglich, ein Abfahrtsintervall anzugeben. In diesem Fall werden alle Abfahrten in dem Intervall sowie die erste davor und die nächste danach ermittelt (siehe Abschnitt 3.5).

Ein weiteres Beispiel haben wir schon in Kapitel 2.3 betrachtet. In dem Beispiel will ein Kunde zwischen 10:00 - 11:00 Uhr von „Am Weinberg“ bis „Holstenhofweg“ fahren. Nach Abbildung 2.2 gibt es zwei Alternativstrecken. Die zeitlich schnellsten Verbindungen sind über die „untere“ diagonal verlaufende Strecke möglich. Die Verbindungsbeschreibung ist in Abbildung 5.3 dargestellt.[4] Die Fahrten wurden, entsprechend zusammengefaßt, in Spalten geordnet. Es brauchen nicht alle Fahrten sofort angezeigt zu werden, einige werden zunächst zurückgehalten. Dieses wird durch ein kleines Dreieck angezeigt. Zwischen 8:23 Uhr und 9:45 Uhr fährt die S4 (oder eine anderere Linie) also mindestens noch einmal.

Falls sich nun im Beratungsgespräch herausstellt, daß ein Kunde eventuell eine längere Umsteigezeit wünscht, so kann man die „versteckte“ weitere Fahrt durch Anklicken hervorholen. In Abbildung 5.4 wurde der Pfeil in der dritten Spalte unterste Zeile angeklickt. Statt in die Linie 163 wird nun in die Linie 263 gewechselt.

Folgerichtig erscheint nun ein Linkspfeil, der auf eine vorige Fahrt hinweist. Die nachfolgenden Fahrten einer Spalte werden beim Anklicken eines Pfeils in einer weiter oben stehenden Zeile entsprechend neu synchronisiert. Die entsprechende Fahrtdauer wird in einer abgesetzten Zeile weiter unten dargestellt. Die günstigste Fahrt in dem angegebenen Abfahrtszeitintervall dauert also mindestens 30, mit verlängerter Umsteigezeit etwa 36 Minuten. Ist dieses aber wirklich die schnellste Verbindung? Abbildung 5.5 zeigt die Zusammenstellung der Fahrten auf der Alternativstrecke. In diesem Beispiel war die geometrisch kürzere Variante auch die zeitlich schnellere. Dieses muß jedoch nicht immer der Fall sein.

Zur Orientierung innerhalb einer Stadtplandarstellung ist es notwendig, die Darstellungsfläche diagonal verschieben zu können (panning). Hierfür erwiesen sich die zunächst vorgesehenen, weit verbreiteten Rollbalken als ungeeignet, weil sie keine diagonale Verschiebung ermöglichen. Es wurde eine andere Lösung realisiert. Der Benutzer kann durch Drücken der Umschalttaste (shift) und einem Linksklick mit der Maus[5] einen Punkt fokussieren (entspricht einem ggf. diagonalen Verschieben). Das Verschieben des Darstellungsausschnitts ist zur schnelleren Manipulation mit einem Zooming

[4] Die Darstellung der Verbindungen ist durch Mausinteraktion rollbar.

[5] Es wird eine Dreiknopfmaus verwendet.

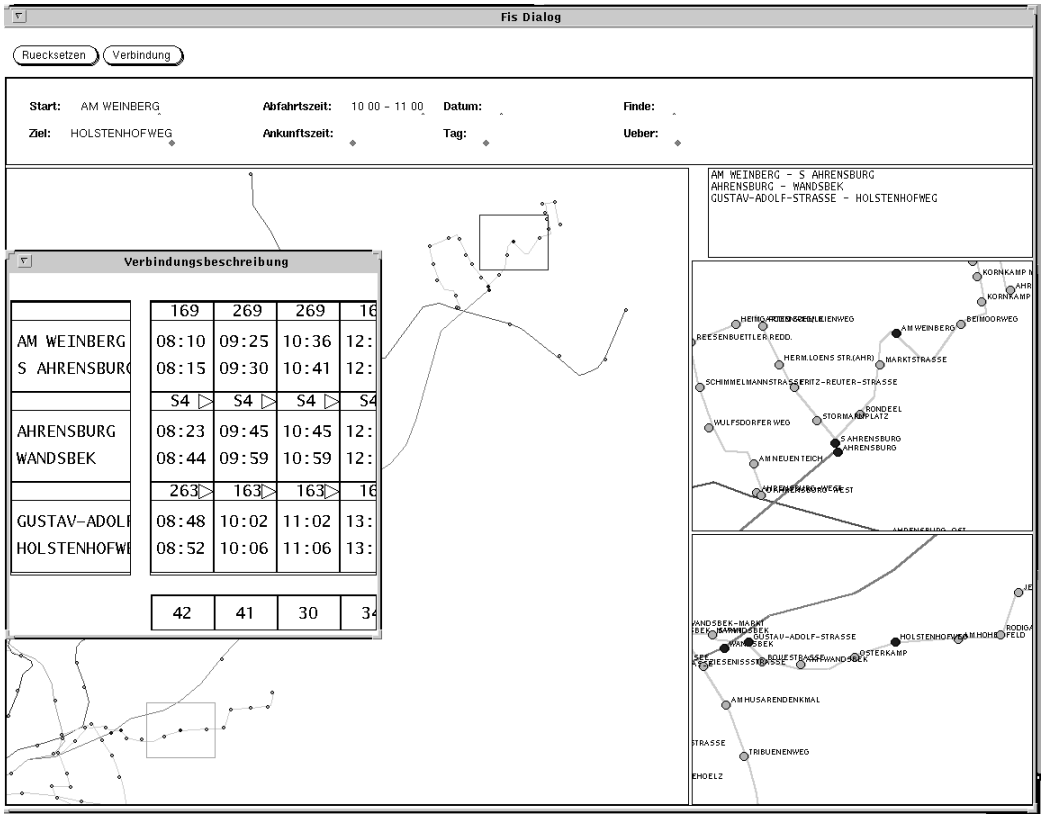

Abbildung 5.3: Erste Alternative: S-Bahn.

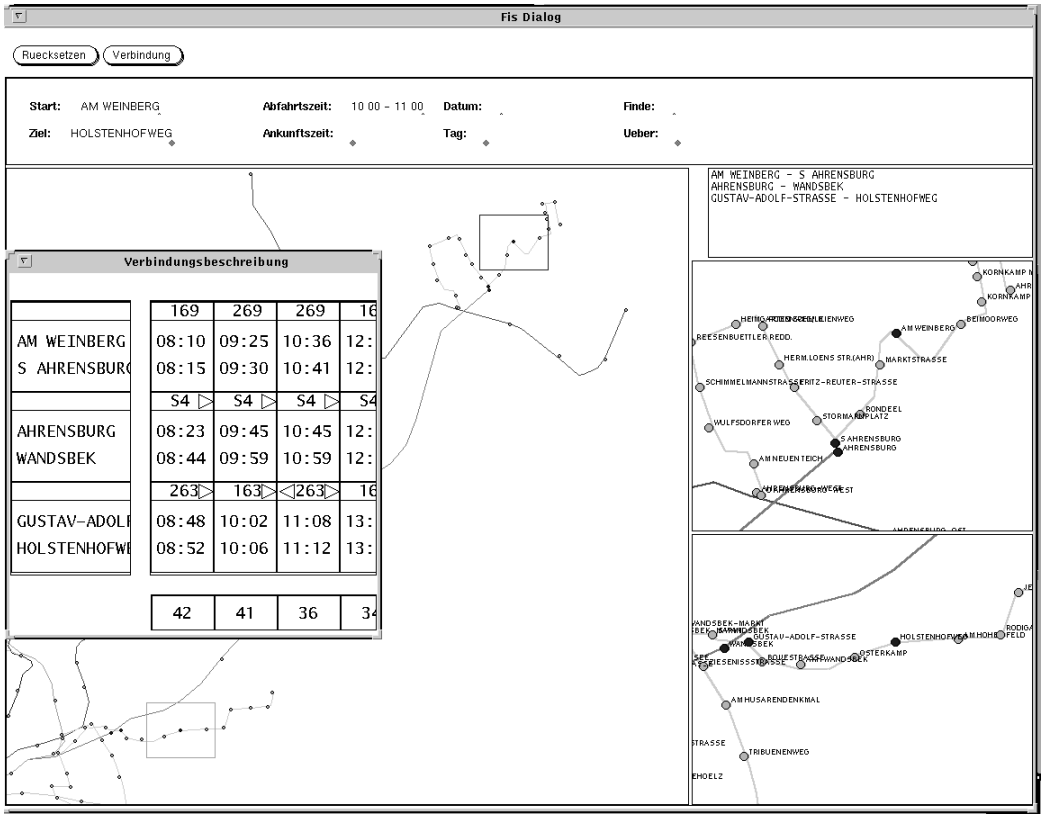

Abbildung 5.4: Auswahl einer längeren Umsteigezeit.

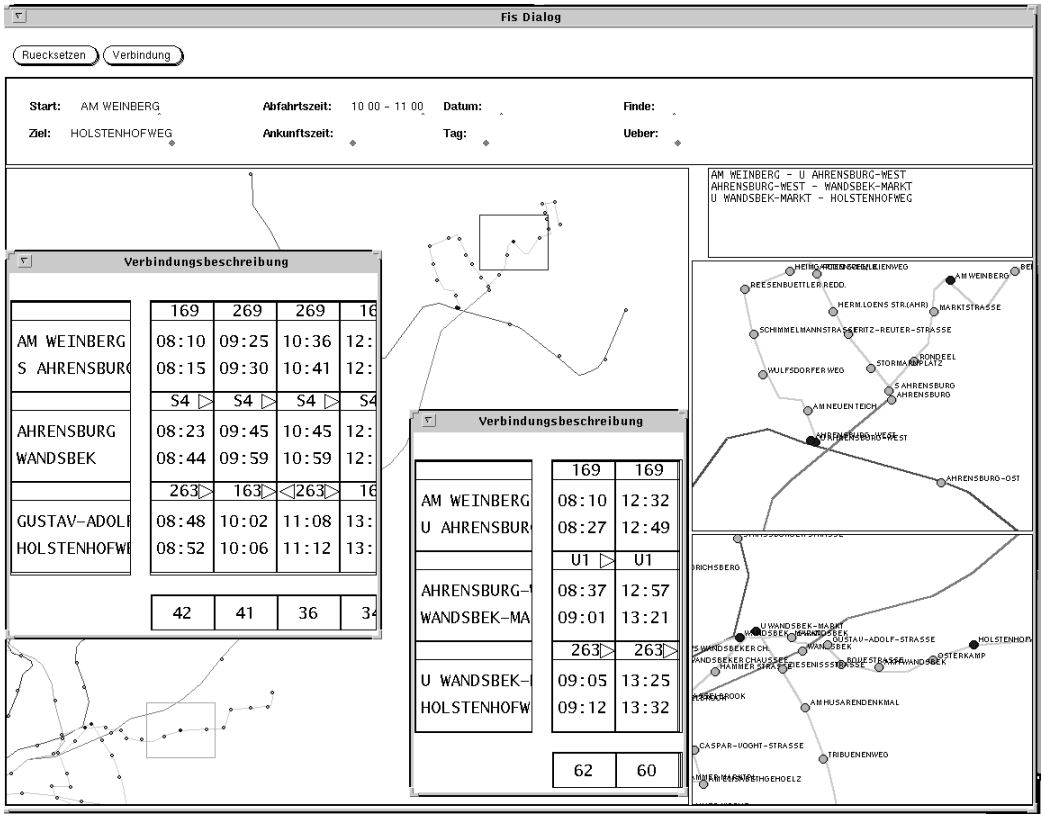

Abbildung 5.5: Zweite Alternative: U-Bahn.

kombinierbar. Wird statt der Umschalttaste die Steuerungstaste (control) während eines Linksklicks betätigt, wird der angeklickte Punkt fokussiert und die Darstellung detaillierter vorgenommnen (zoom in). Eine Vergröberung (zoom out) wird durch Gedrückthalten der Steuerungstaste (control) bei einem Rechtsklick vorgenommen. Diese Interaktionsform ist nicht für Anfänger geeignet, ermöglicht aber einem Experten ein schnelles Arbeiten. Die Interaktionsformen sollen auch bei erhöhtem Lernaufwand für Fortgeschrittene so schnell wie möglich handhabbar sein. Bei Verwendung einer spezielleren Graphikbibliothek können weitere Interaktionstechniken zum Einsatz kommen. Hierzu gehören Lupenfunktionen und animiertes Verschieben (on-line panning, on-line zooming).

Zur Eingabe von Namen (z.B. Haltepunkten) mit der Tastatur (vgl. Abbildung 5.1) wurde eine Vervollständigung (completion) mittels eines Abgleichs (matching) mit allen dem System bekannten Namen der geographischen Datenbasis realisiert. Weiterhin wird eine einfache Rechtschreibkorrektur eingesetzt, die Buchstabendreher, Auslassung und das Hinzufügen eines einzelnen Buchstabens korrigiert.

5.2 Zusammenstellung von Verbindungen

Die Verteilung und Synchronisierung der Fahrten zur Darstellung innerhalb einer Verbindungsbeschreibung ist ein nicht-triviales Anordnungsproblem.[6] Wir erläutern den Kern des Algorithmus anhand eines Beispiels, in dem ein Abfahrtszeitintervall von 10:00 Uhr bis 10:30 vorgegeben ist. Wir bezeichnen die Tripel bestehend aus einer Linie, einer Abfahrtszeit und einer Ankunftszeit als Deskriptoren. Die Deskriptoren wurden gemäß Abschnitt 3.5 ermittelt.

Zur Erläuterung des Algorithmus zur Verbindungszusammenstellung betrachten wir das nachfolgende Beispiel.[7] Alle Deskriptoren führen intern einen Index, der den Anfangswert `0` erhält (in Klammern dargestellt).

```
S21 (0)   S21 (0)   S21 (0)
7:00      10:00     13:00
7:30      10:30     13:30

U1 (0)    U1 (0)    U1 (0)    U1 (0)
8:00      10:05     13:35     14:00
```

[6] Die Grundidee des Anordnungsalgorithmus stammt von Michael Malsch und Thomas Schnudt (HBT).

[7] Die Fahrplandaten sind fiktiv.

```
8:30     10:35     14:25     14:30

123 (0)   124 (0)   123 (0)   124 (0)
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30
```

Beginnend mit der letzten Fahrt wird für alle Fahrten der letzten Teilstrecke - unter Berücksichtigung der Umsteigezeit - die jeweils vorhergehende Fahrt der davorliegenden Teilstrecke ermittelt. Die auf diese Weise ermittelten Fahrten der Teilstrecken erhalten die laufende Nummer der Fahrt der letzten Fahrtstrecke (im folgenden als Fahrtnummer bezeichnet). Dieses ist im ersten Schritt die `4`. Wenn es sich nicht um die erste Verbindung handelt, wird der Index ausgehend von dem gerade numerierten Deskriptor eines Teilstücks auf alle Deskriptoren übertragen, die einen Anschluß zum letzten Deskriptor der jeweils vorigen Teilstrecke ermöglichen. Im nachfolgend dargestellten ersten Schritt wurde in der zweiten Deskriptorzeile (zweites Teilstück der Route) der Index 4 auch auf den Deskriptor mit dem Stern übertragen.

```
Schritt 1:
S21 (0)   S21 (0)   S21 (4)
7:00     10:00     13:00
7:30     10:30     13:30

U1 (0)    U1 (0)    U1 (4)*   U1 (4)
8:00     10:05     13:35     14:00
8:30     10:35     14:25     14:30

123 (0)   124 (0)   123 (0)   124 (4)
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30

Schritt 2:
S21 (0)   S21 (0)   S21 (3)
7:00     10:00     13:00
7:30     10:30     13:30

U1 (0)    U1 (0)    U1 (3)    U1 (3)
8:00     10:05     13:35     14:00
8:30     10:35     14:25     14:30

123 (0)   124 (0)   123 (3)   124 (4)
```

```
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30

Schritt 3:
S21 (2)   S21 (0)   S21 (3)
7:00     10:00     13:00
7:30     10:30     13:30

U1 (0)    U1 (2)    U1 (3)    U1 (3)
8:00     10:05     13:35     14:00
8:30     10:35     14:25     14:30

123 (0)   124 (2)   123 (3)   124 (4)
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30

Schritt 3:
S21 (1)   S21 (0)   S21 (3)
7:00     10:00     13:00
7:30     10:30     13:30

U1 (1)    U1 (2)    U1 (3)    U1 (3)
8:00     10:05     13:35     14:00
8:30     10:35     14:25     14:30

123 (1)   124 (2)   123 (3)   124 (4)
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30
```

Sind bis hinab zur Fahrtnummer 1 alle Fahrtenfolgen ermittelt, ist die Synchronisation abgeschlossen. Die Verbindungen ergeben sich aus den Deskriptoren der ersten Teilstrecke, die nicht den Index `0` haben. In dem obigen Beispiel wurden zwei Verbindungen ermittelt. Deskriptoren ab der zweiten Teilstrecke, die nicht einen Index einer Verbindung tragen, werden auf `0` gesetzt.

```
S21 (1)   S21 (0)   S21 (3)
7:00     10:00     13:00
7:30     10:30     13:30

U1 (1)    U1 (0)    U1 (3)    U1 (3)
```

```
8:00     10:05     13:35     14:00
8:30     10:35     14:25     14:30

123 (1)   124 (0)   123 (3)   124 (0)
9:00     11:00     15:00     17:00
9:30     11:30     15:30     17:30
```

Die mit dem Index `0` beschrifteten Deskriptoren werden mit dem jeweils links stehenden Deskriptor ungleich `0` zusammengefaßt. Damit ergibt sich folgende interne Anordnung:

```
S21 (1)   S21 (0)            S21 (3)
7:00     10:00              13:00
7:30     10:30              13:30

U1 (1)    U1 (0)             U1 (3)    U1 (3)
8:00     10:05              13:35     14:00
8:30     10:35              14:25     14:30

123 (1)   124 (0)            123 (3)   124 (0)
9:00     11:00              15:00     17:00
9:30     11:30              15:30     17:30
```

In einem Dialogfester zur Verbindungsbeschreibung wie in Abbildung 5.5 ergeben sich zwei Spalten und drei Zeilen. Die am weitesten links stehenden Deskriptoren jeder Verbindung werden zunächst angezeigt. Außer in dem Kästchen rechts oben wird bei der Darstellung ein kleines nach rechts gerichtetes Dreieck eingeblendet, um anzudeuten, daß es noch Alternativen innerhalb der Verbindungsspalte gibt. Wählt ein Benutzer z.B. in der ersten Spalte und ersten Zeile die zweite Alternative, so gibt es innerhalb der zusammengestellten Verbindung keine Anschlüsse. Daher wird mit einem großen Rechtspfeil auf die nächste Verbindung (Spalte) verwiesen. Ein Beispiel für eine solche Situation zeigt Abbildung 5.6.

5.3 System-Architektur und Implementation

Der FIS-Prototyp wurde auf einer SUN SPARCstation 1+ mit „Harlequin Common Lisp“ unter Verwendung der X-Window-Schnittstellen CLX und CLUE (Common Lisp User Interface Enviroment) sowie des Objektsystems

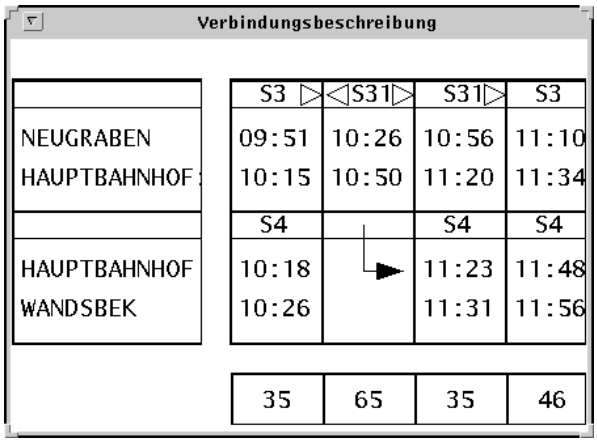

Abbildung 5.6: Zusammenstellung von Verbindungen.

CLOS (Common Lisp Object System) entwickelt. Der Gauß-Jordan-Algorithmus ist in C realisiert.[8] Weiterhin wurden zur Gestaltung der Fenster der Modul „Harlequin Toolkit“ verwendet. Die Modellierung und Darstellung der Linien und Haltepunkte wurde durch einen Modul zur objektorientierten Modellierung von graphischen Formen vorgenommen („Pictures“). Es ist geplant, auch eine Version für „Allegro Common Lisp“ unter Verwendung des Moduls CLIO (Common Lisp Interactive Objects) anstelle des Pakets „Harlequin Toolkit“ zu erstellen.

Noch ein Wort zu den Antwortzeiten, wenn bezüglich der Fahrtdauer und bezüglich der Anzahl der Umsteigevorgänge optimiert wird: Die Beantwortung des Beispiels aus Abbildung 5.5 dauert eine halbe Sekunde. Am Ende befinden sich 39 Knoten auf der OPEN-Liste und 13 Knoten auf der CLOSED-Liste. Über 50 Prozent aller Anfragen werden in weniger als drei Sekunden beantwortet. Für verschiedene Haltepunktkombinationen wurden mit dem in diesem Bericht beschriebenen FIS-Prototypen mittlere Suchzeiten von ca. 10 Sekunden ermittelt (jeweils für einen Vorwärtssuchvorgang). Die Suchzeiten variieren, je nach Paging-Rate dauert der Suchvorgang erheblich länger. Die Repräsentation der Fahrpläne umfaßt eine Datenkompression, um akzeptable Suchzeiten zu gewährleisten. Der hierdurch erhöhte Aufwand ist eine Konzession an die Verwendung von realen Massendaten.

[8] Die Matrixzugriffe erfolgen direkt über Adreßzeiger, nicht über Indizes. Adreßarithmetik in bezug auf Matrixzugriffe läßt sich in Common Lisp nicht realisieren, ist aber wegen der benötigten Geschwindigkeit notwendig, auch wenn der Algorithmus in einem Vorlauf und nicht während der eigentlichen Verbindungssuche abläuft.

Kapitel 6

Aktuelle Arbeiten

Die Entwicklung des Fahrplan-Informationssystems ist zur Zeit der Berichterstellung noch nicht abgeschlossen. Dieses Kapitel gibt einen Überblick über die verfolgten Arbeiten und die Ideen zur Gestaltung der noch fehlenden Optimierungs- und Interaktionskomponenten.

6.1 Optimalitätskriterien

In einem öffentlichen Nahverkehrssystem ist es wichtig, individuelle Kundenwünsche bei der Verbindungssuche und -optimierung zu berücksichtigen. Dieses erhöht einerseits die Akzeptanz eines automatischen Auskunftssystems, zum anderen kann eine ständige manuelle Suche bei Sonderwünschen vermieden werden. Besondere Wünsche der Nahverkehrskunden drücken sich aus in individuellen Optimalitätskriterien:

- Vermeidung bestimmter Objekte (z.B. Tunnel, bestimmte Haltepunkte),
- Bevorzugung bestimmter Verkehrsmittel (z.B. Bus vor S-Bahn),
- behindertengerechte Umsteigezeiten und -haltepunkte,
- Ansteuerung bzw. Passieren bestimmter Objekte (Sightseeing),
- Berücksichtigung von kurzfristigen Zugverspätungen,
- Berücksichtigung von Verspätungen durch (Dauer-)Baustellen und Umlenkungen.

Es ist nicht ausgeschlossen, daß in Zukunft bisher noch nicht bedachte, weitere Kriterien hinzukommen. Um diese Kriterien berücksichtigen zu können und eine modulare Erweiterung zu gewährleisten, wird es notwendig sein, Bewertungskriterien und ihre Auswirkungen auf die Suche explizit zu machen.

Eine naheliegende Möglichkeit ist, alle Kriterien über die Kantenkosten des Suchgraphen zu verrechnen. Verschiedene Kosten werden auf „Zeitkosten" umgerechnet. Eine Bestrafung mit 5 Zeiteinheiten (etwa Minuten) einer Tunnelstrecke läßt sich dahingehend deuten, daß die Tunnelstrecke vermieden wird, wenn man auf einer anderen Stecke z.B. nur 5 Minuten länger fährt. Ein Nachteil bei diesem Vorgehen ist, daß die Schätzfunktion, die ja zunächst nur Zeitkosten berücksichtigt, bei weiteren „Kostenfaktoren" immer schlechter wird. Damit erhöht sich der Suchaufwand erheblich.

Bei der Integration der verschiedenen Kostenfaktoren muß untersucht werden, inwieweit noch vorhersagbare Ergebnisse produziert werden. Wird genau eine Lösung durch Integration verschiedenster Kostenfaktoren bestimmt, so läßt sich meist nicht einfach deuten, inwiefern diese Lösung besser ist als z.B. die zweitbeste Lösung. Weiterhin können sich Tunnelstrecken und touristisch attraktive Strecken in Konkurrenz zueinander befinden. Die Frage ist, inwieweit sich hier bestimmte quantitative Gewichtungen eignen. Man sollte allerdings beachten, daß jeder Expansion des Suchraum in unserem Ansatz einem Umsteigevorgang entspricht. Es müssen daher zur Bestimmung der Gesamtbewertung einer Verbindung nur wenige Kostenfaktoren verrechnet werden, wenn man von maximal etwa fünf Umsteigevorgängen ausgeht.

Es gibt weitere Probleme bei der Generierung von Nachfolgern im A*-Algorithmus. Hierzu ein Beispiel: Die Bevorzugung von bestimmten Verkehrsmitteln (z.B. Bussen) läßt sich nicht ohne weiteres mit Hilfe eines Kostenfaktors ausdrücken. Wenn man nur Busfahrten als Nachfolger zuließe, wäre nicht sichergestellt, daß überhaupt eine Verbindung gefunden werden kann. Es muß also der frühestmögliche Nachfolger allgemein sowie der frühestmögliche Busnachfolger generiert werden (beide können identisch sein).

Um Nachfragen von Kunden beantworten zu können, ist es evtl. günstiger, verschiedene Routen nach jeweils einem der oben aufgeführten individuellen Optimierungskriterien (einschließlich einer Umsteigeminimierung) zu bestimmen und zum Vergleich aufzulisten.

Weitere praktische Versuche und theoretische Arbeiten sind notwendig, um das FIS in dieser Hinsicht zu vervollständigen.

6.2 Interaktive Umsteigeknotenauswahl vs. automatische Suche

Die automatische Suche kann mit einer interaktiven Dialogform verknüpft werden. Gefundene Verbindungen werden innerhalb der Übersichtsdarstellung visualisiert. Es ist dem Auskunftgebenden daher besser möglich, dem Kunden die vorgeschlagene Fahrtroute zu schildern. Die Erfahrung zeigt, daß man entspannter fahren kann, wenn man die Strecke kennt, man also z.B. weiß, daß der nächste Umsteigeknoten erst nach drei weiteren Stationen kommt. Auf Wunsch könnte für einen Kunden ein Ausdruck der Fahrtroute erstellt werden.[1]

Durch Mausinteraktion kann der Auskunftsgebende auf einfache Weise obligatorische Zwischenziele anwählen. Die Aufgabe des automatischen Suchsystems besteht dann darin, die Teilstücke optimal zu verbinden.[2]

6.3 Berücksichtigung von Tarifzonen

Tarifzoneninformationen lassen sich auf verschiedene Weisen in ein Fahrplan-Informationssystem integrieren (mit ansteigender Komplexität):

- Integration in die graphische Darstellung,
- Bestimmung des Fahrpreises zu einer bestimmten Verbindung,
- Einbeziehung in die Optimierung (nach dem Motto: „Für die Tarifzonen X, Y und Z habe ich eine Monatskarte!“).

6.4 Präsentation und Repräsentation von Stadtplandaten

In den vorangegangenen Kapiteln wurde die Verwendung der Darstellung eines Stadtplanes zur Orientierung herausgestellt. Neben dem Auskunftgebenden in einer Telefonzentrale können auch Kunden von einer Stadtplandarstellung der Start- und Zielregion profitieren, wenn diese zusammen mit einer schriftlichen Zusammenstellung der Anfrageergebnisse übergeben bzw. zugestellt werden. In Hamburg wird eine Digitalisierung der analogen Stadtkarten der Baubehörde in Form einer „Digitalen Stadtgrundkarte“ in den

[1] Der Ausdruck könnte auch per Post zugestellt werden.

[2] Wir nehmen an, daß die Reihenfolge der Zwischenstationen eindeutig festgelegt ist.

nächsten Jahren zur Verfügung stehen. Weiterhin gibt es private Anbieter von digitalen Stadtplandaten.

Um den Speicherbedarf (z.B. in einer Datenbank) gering zu halten, werden geographische Objekte durch Linienzüge in einem geeigneten Koordinatensystem (z.B. Gauß-Krüger-Koordinaten) repräsentiert.[3] Diese Daten müssen für einen menschlichen Betrachter innerhalb des FIS entsprechend aufbereitet und dargestellt werden. Hierzu sind weitere Attribute notwendig. Für Straßen z.B. sollte eine Klassifizierung (Autobahn, Ring-, Haupt-, oder Nebenstraße, Fußgängerzone, etc.) sowie Hausnummern bzgl. der Teilabschnitte bei der Modellierung berücksichtigt werden. Auch Haltepunkte sollten gemäß ihrer Bedeutung und Aufgabe klassifiziert werden. Für markante oder auch öffentliche Gebäude könnten Piktogramme verwendet werden.

Im bisherigen System sind nur die U- und S-Bahnlinien sowie einige Buslinien repräsentiert. Der Stadtplan-Datenbestand wird in Kürze ausgebaut und auf Straßendaten erweitert. Die Darstellung von Stadtplandaten sollte so erfolgen, daß der Betrachter sich gut orientieren kann. Ähnlich wie in gezeichneten offiziellen Stadtplänen sollten wichtige Straßen gegenüber Nebenstraßen durch eine breitere, farbig betonte Dartellung hervorgehoben werden. Eingeblendete Hausnummern ermöglichen eine genauere Start bzw. Ziellokalisierung. Zu berücksichtigen ist, daß das elektronische Medium gestattet, beliebige Objekte aus- oder einzublenden, so daß eine situationsangepaßte Darstellung von Stadtplandaten möglich wird.

Zu den besonderen Darstellungstechniken gehört z.B. die Darstellung von Teilbereichen durch eine „Fisheye"-Vergrößerung. Im Rahmen der Orientierung innerhalb des Fahrplan-Informationssystems erscheint aber eine Fisheye-Verzerrung das Schätzen von Entfernungen erheblich zu erschweren [13], [17].

Ein Zooming der Darstellung kann nicht durch eine einfache Skalierung realisiert werden. Je nach Zoomfaktor werden verschiedene Darstellungstechniken zum Einsatz kommen. Zu berücksichtigen sind der Detaillierungsgrad der Darstellung, die verwendeten Schriftgrößen sowie die Plazierung von Schriftzügen (Objektzuordnung vs. Ausnutzung von Freiflächen, siehe z.B. [14]). Um eine bessere Orientierung zu gewährleisten, bietet es sich an, Darstellungen so zu verzerren, daß z.B. bestimmte, sehr detailliert darzustellende Gebiete auf Kosten von Freiflächen (Gewässer, Waldgebiete) vergrößert werden. Wichtige, bekannte Straßenzüge können übertrieben breit dargestellt

[3] In dem in diesem Bericht vorgestellten Prototypen wurde eine vorläufige geographische Schnittstelle verwendet. Die geographischen Daten der gezeigten Linien wurden mit einem speziell entwickelten Zeichenprogramm erfaßt.

werden.

6.5 Resümee

Die vorangehenden Abschnitte vermitteln einige Ideen zur Erweiterung der Such- und Präsentationsverfahren. Um die (Teil-)Bewertungen zur Bestimmung der optimalen Verbindung nachvollziehen zu können, wird es notwendig sein, die für die Teilabschnitte einer Verbindung vergebenen Bewertungsfaktoren mit ihren Werten zu protokollieren. Mithilfe des A*-Algorithmus können dann weitere (bezüglich der angesetzten Bewertungskriterien weniger optimale) Verbindungen (Lösungspfade) berechnet werden. Werden die Teilbewertungen protokolliert, so können die ermittelten Verbindungen verglichen werden. Diese Entwicklungen werden in einem Anschlußbericht detaillierter dargestellt.

Kapitel 7

Vergleich mit bisherigen Ansätzen

Bisher entwickelte automatische Fahrplanauskunftssysteme für den Nahverkehr verwenden unterschiedliche Verfahren zur Bestimmung von Verbindungen. Im folgenden werden die Suchverfahren einiger Systeme betrachtet und mit dem Ansatz des im vorliegenden Bericht beschriebenen Systems FIS verglichen. Systeme, die für die automatische Fahrplanauskunft entwickelt wurden, sind u.a. AFI (Automatisches Fahrplaninformationssystem) [10], dessen Nachfolger STAFI (Standardisiertes Auskunfts- und Fahrplaninformationssystem) [15], ARIADNE[1] [11] und EFA [20].

Mit Ausnahme von EFA gehen die genannten Systeme von einem mehrstufigen, hierarchischen Verfahren aus. Mehrstufig bedeutet, daß in einem ersten Schritt (mehrere) Routen (Folgen von Umsteigeknoten) ohne Fahrplaninformationen nur aus der Linientopologie des Verkehrsnetzes ermittelt werden. Danach werden in einem weiteren Schritt für eine festgelegte Anzahl solcher Routen die Fahrtzeiten bestimmt, um die zeitkürzeste Verbindung zu ermitteln.

Betrachten wir die Bestimmung der Routen im ersten Schritt anhand des Systems AFI. Es werden nacheinander alle Routen mit direkten Verbindungen bzw. einmaligem bis dreimaligem Umsteigen ermittelt. Von diesen werden 20 Routen anhand einer Verbindungsvorbewertung ausgewählt. Bezüglich dieser ausgewählten Routen werden dann die tatsächlichen Fahrtzeiten ermittelt. Die Verbindungsvorbewertung orientiert sich an Schätzungen für die voraussichtliche Fahrtzeit und für die mittlere Umsteigezeit (u.a. in Abhängigkeit von den Verkehrsmitteltypen der Teilstrecken). Als mittle-

[1] Da es sich bei ARIADNE um ein kommerzielles System handelt, konnte nichts über die realisierten Suchverfahren in Erfahrung gebracht werden.

re Umsteigezeit von einer S-Bahn zu einer anderen wird etwa eine Zeit von ca. 6 Minuten angesetzt. Auswirkungen von Änderungen dieses Parameters können nur schwer abgeschätzt werden. Weiterhin geht in die Verbindungsvorbewertung die Anzahl der Umsteigevorgänge ein. Genaueres zur Verrechnung dieser Schätzwerte schildert [10]. Greschner schildert in [9], wie eine Auswahl von verschiedenen Verbindungen durch ein regelorientiertes System ausgedrückt werden kann.

Als Grund für die Einteilung der Suche in Stufen wird angegeben [10], daß durch die Speicherung des Fahrplans auf einer (langsamen) Magnetplatte die Ermittlung der realen Fahrtzeiten nur für ausgewählte Routen erfolgen kann.

Die Einführung von Vorbewertungen, die das Fehlen von Fahrplaninformationen in der ersten Stufe kompensieren sollen, führt dazu, daß nicht in jedem Fall die optimale Verbindung gefunden wird. Problematisch sind die Tageszeiten, in denen z.B. die Nachtbusse noch und die S-Bahnen schon fahren (oder umgekehrt). Eine optimale Verbindung kann i.a. nur gefunden werden, wenn neben der räumlichen Dimension auch die Fahrtzeit für alle Möglichkeiten direkt bei der Routensuche betrachtet wird (vgl. Abschnitt 3.4). Ein solches Verfahren wird einstufig genannt, die zeitlichen und räumlichen Dimensionen einer Verbindung werden nicht getrennt ermittelt. Im FIS und in EFA [19] wird dieser Ansatz gewählt.

In EFA kommt ein abgewandeltes Breitensuchverfahren zum Einsatz. In die Bewertung der Knoten geht die Anzahl der Umsteigevorgänge und die Fahrtzeit ein. Statt die Suche durch einen aus den Fahrtzeiten gewonnenen Schätzer zu fokussieren, wie im System FIS realisiert, wird die Suche durch räumliche Merkmale begrenzt. Für jeden potentiellen Start- und Zielknoten wird dabei eine Menge von Tarifzonen vorgegeben.[2] Während der Suche werden nur Haltepunkte bzw. Umsteigeknoten betrachtet, die in diesen Tarifzonen liegen.

Durch die Festlegung auf zulässige räumliche Flächen wird es nicht möglich sein, auf einfache Weise zusätzliche Bewertungsmaßstäbe, wie Sonderwünsche einiger Kunden (vgl. Abschnitt 6.1), einzubeziehen. Zum Beispiel könnte eine touristisch attraktive Verbindung durch eine nicht „erlaubte“ Tarifzone führen. Die im FIS verwendete Kostenfunktion schließt nicht von vornherein Haltepunkte aus. Sonderwünsche können durch Änderung der Kostenparameter berücksichtigt werden.

Der Unterschied zwischen EFA und FIS liegt in der unterschiedlichen Steuerung des Suchverfahrens. Bei EFA wird dazu eine etwas willkürliche

[2]Nach welchen Kriterien die Menge von Tarifzonen ermittelt wird, wurde nicht klar ausgesagt.

Einschränkung des Suchraums erzeugt. FIS verwendet hingegen mit dem A*-Algorithmus eine allgemeine, fundierte Best-First-Graphensuche, die aufgrund eines Schätzers die Suche fokussiert, den Suchraum als solchen aber nicht grundsätzlich verkleinert. Eine schlechte Schätzung führt lediglich zu längeren Suchzeiten und nicht zu falschen Lösungen (vgl. Abschnitt 3.1). Durch die bei FIS realisierte Anwendung des A* werden problemspezifische Merkmale berücksichtigt, ohne seine Eigenschaften zu verändern. Bei den im System AFI eingesetzten Schätzern zur Verbindungsvorbewertung (s.o.) ist nicht sichergestellt, daß auch bei einem schlechten Schätzer die gleiche Lösung gefunden wird (wenn auch mit erhöhtem Aufwand).

Einen weiterer Unterschied des Systems FIS zu allen genannten Systemen bildet die graphische Aufbereitung der Netz- und Fahrplaninformationen, wodurch dem Kundenberater ein besserer Überblick ermöglicht wird. Durch die Integration von Stadtplandaten wird es möglich, Verbindungen nicht nur zwischen Haltestellen zu ermitteln (wie z.B. bei EFA), sondern auch Straßennamen bei der Auskunft zu berücksichtigen. Die Routensuche kann dann alle Haltestellen, die mit einem zumutbaren Fußweg zu erreichen sind, bestimmen, um dann mit dem in Abschnitt 3.2 geschilderten Verfahren unter Einbeziehung der Fußwege eine entsprechend optimale Verbindung zu ermitteln.

Kapitel 8

Zusammenfassung

In diesem Bericht haben wir gezeigt, daß die Verbindungssuche innerhalb eines Nahverkehrssystems mit einem Graphensuchverfahren behandelt werden kann. Der Standard-Suchalgorithmus A* für ODER-Graphen ermöglicht, entsprechend in der Domäne eingesetzt, akzeptable Suchzeiten. Schätzfunktionen werden mit Hilfe eines Verfahrens aus der Optimierungstheorie gewonnen (Gauß-Jordan-Algorithmus). Sowohl bei der Bestimmung des Schätzers als auch bei der A*-Suche werden nur zeitliche Informationen verwendet. Die Integration von individuellen Kundenwünschen in die Verbindungssuche befindet sich zum Zeitpunkt des Entstehens dieses Berichts in der Entwicklung. Die Integration einer Komponente zur Berücksichtigung von Tarifzonen erscheint möglich.

Um eine Beratung eines Kunden z.B. innerhalb einer Telefonauskunft vorteilhaft zu gestalten, wurde eine graphische Benutzungsoberfläche konzipiert. Stadtplandaten werden übersichtlich dargestellt und ermöglichen gezielte Rückfragen an den auskunftssuchenden Kunden. Präsentationsformen für Stadtplandaten werden im weiteren Projektverlauf verfeinert.

Danksagung

Wir danken den Projektmitarbeitern der Partnerfirma „Hamburger Berater Team“ (HBT) Herrn Michael Malsch, Herrn Thomas Schnudt sowie Herrn Dr. Friedemann Weik für die gute Zusammenarbeit und für viele eingebrachte Ideen. Unser Dank geht auch an den Projektleiter Herrn Prof. Dr. Bernd Neumann, der in Diskussionen mehrere „Initialzündungen“ von Ideen ausgelöst hat und durch die sorgfältige Durchsicht von frühen Fassungen des vorliegenden Berichts wesentlich zum Gelingen dieser Arbeit beigetagen hat, sowie auch an den weiteren Projektleiter Herrn Hans-Joachim Habermann (HBT), der das Projekt initiiert hat.

Weiterhin gilt unser Dank auch den Mitarbeitern des „Labors für Künstliche Intelligenz“, insbesondere Herrn Thorsten von Stein, für viele Diskussionen und konstruktive Kritik. Wir danken Frau Dr. Gudula Retz-Schmidt und Herrn Rainer Joswig für wertvolle Hinweise zu einer früheren Version dieser Arbeit.

Dankend erwähnt werden soll auch das Engagement der HVV-Vertreter, Herrn Lamla und Herrn von Rumohr, die für Diskussionen bereitstanden und es ermöglichten, die Fahrplan-Domäne besser kennenzulernen.

Literaturverzeichnis

[1] Belmann, R., Dreyfus, S., *Applied Dynamic Programming*, Princeton, NJ: Princeton University Press, 1962.

[2] Bratko, I., *PROLOG Programming for Artificial Intelligence*, Addison-Wesley, Reading, 1986.

[3] Bry, F., Query Evaluation in Recursive Databases: Bottom-up and Top-down Reconciled, in: Proc. *1. Conf. on Deductive and Object-Oriented Databases (DOOD'89)*, Kyoto, Japan, 4.-6. Dezember 1989.

[4] Ceri, S., Gottlob, G., Tanca, L., *What You Always Wanted to Know About Datalog (And Never Dared to Ask)*, IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1, März 1989, pp. 146-166.

[5] Domschke, W. *Kürzeste Wege in Graphen: Algorithmen, Verfahrensvergleiche* Köndigstein/Ts.:Hain, 1972.

[6] Genesereth, M., Nilsson, N., *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Los Altos, California, 1987.

[7] Gondran, M., Minoux, M., *Graphs and Algorithms*, John Wiley and Sons, 1984.

[8] Gray, P.M.D., Lucas, R.J. (Hrsg.), *Prolog and Databases: Implementations and New Directions*, Ellis Horwood, Computer Science Series, 1988.

[9] Greschner, G., Simons, A., *MOBILE-Fahrplanauskunft mit Künstlicher Intelligenz*, Verkehr und Technik, Heft 5/1988.

[10] Hamburger Verkehrsverbund, SNV Studiengesellschaft Nahverkehr mbH, *Entwicklung und Erprobung eines Automatischen Fahrplaninformations- und Auskunftssystems für den Verkehrsraum des Hamburger Verkehrsverbundes AFI*, Forschungs- und Entwicklungsvorhaben des Bundesministers für Forschung und Technik, 1983.

[11] Hannoversche Consulting für Verkehrswesen, Transporttechnik und Elektronische Datenverarbeitung, *ARIADNE*, Interner Bericht 1989.

[12] Harary, F., *Graph Theory*, Addison-Wesley, Reading, Mass., 1969.

[13] Hollands, J.G., Cary, T.T., Mathews, M.L., McCann, C.A., *Presenting a Graphical Network: A Comparison of Performance Using Fisheye and Scrolling Views*, in: Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Salvendy, G. Smith, M.J. (Eds.), Elsevier Science Publishers B.B., Amsterdam, 1989, pp. 313-321.

[14] Jones, C.B., *Cartographic Name Placement with Prolog*, in: IEEE Computer Graphics & Applications, September 1989, pp. 36-47.

[15] Kaufhold, H., Pasquay, F., Pfau, H., von Rumohr, V., *Automatische Fahplanauskunft*, Der Nahverkehr, Heft 6/1989.

[16] Keene, S., *Object-Oriented Programming in Common Lisp*, Addison-Wesley, 1989.

[17] Leung, Y.K., *Human-Computer Interface Techniques for Map-Based Diagrams*, in: Designing and Using Human-Computer Interfaces and Knowledge Based Systems, Salvendy, G. Smith, M.J. (Eds.), Elsevier Science Publishers B.B., Amsterdam, 1989, pp. 361-369.

[18] Manthey, R., Gallaire, H., Nicolas, J.-M., Can we reach a uniform paradigm for deductive query evaluation?, in: *„Wissensbasierte Systeme“*, Brauer, W., Freksa, C. (Hrsg.), Proceedings 3. Internationaler KI-Kongreß, München, Oktober 1989, Informatik Fachberichte, Springer Verlag, 1989, pp. 17-32.

[19] Mentz, H.-J., *Wegewahl in Raum und Zeit*, Verkehr und Technik, Heft 1/1986.

[20] Mentz, H.-J., Oldenbürger, H.-D., *Der Persönliche Fahrplan*, Verkehr und Technik, Heft 6/1988.

[21] Neukirchner, E.P. *Fahrerinformations- und Navigationssystem*, in: Informatik Spektrum 14, 1991, pp. 65-68.

[22] Nilsson, N., *Principles of Artificial Intelligence*, Palo Alto, Calif.: Tioga, 1980.

[23] Pearl, J., *Heuristics - Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley, 1984.

[24] Rich, E., *Artificial Intelligence*, McGraw Hill, New York, 1983.

[25] Rote, G., *Path Problems in Graphs*, in: Computing, 7, 1990, pp. 159-189.

[26] Steele, G.L., *Common Lisp the Language - Second Edition*, Digital Press, 1990.