Berechnung kürzester Wege in Straßennetzen mit Wegeverboten

Bei der Fakultät für Bauingenieur- und Vermessungswesen
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.)
eingereichte Dissertation

vorgelegt von
Wolfgang Schmid
Hinterriedstraße 21
71272 Renningen

Stuttgart 2000

Für Tanja

Prüfungskommission:

Hauptberichter:	Prof. Dr. - Ing. habil. Dieter Fritsch
Mitberichter:	Prof. Dr. rer. nat. Volker Claus
	Prof. Drs. mult. Erik W. Grafarend

Tag der mündlichen Prüfung: 25.07.2000

Inhaltsverzeichnis

Allgemeine Bezeichnungen

In der Arbeit halte ich mich an folgende allgemeine Bezeichnungen:

a, b	Indizes für Fallunterscheidungen
aga	Anzahl der gemeinsamen Anfangskanten
$\alpha(e)$	Anfangsknoten von e
$\beta(e)$	Gewicht von e
$B(i)$	kürzester-Wege-Baum nach dem Schritt i
$d_e(u, v)$	Euklidischer Abstand von u nach v
$d(u, v)$	Länge des kürzesten Weges von u nach v
$\delta(w)$	Länge des Weges w
E	Kantenmenge
E_0	Menge aller Kanten, von welchen aus ein Wegeverbot beginnt
e, e', f	Kanten aus E
e_{ij}	Kante aus E mit $\alpha(e_{ij}) = v_i$ und $\omega(e_{ij}) = v_j$
eka	Anzahl der gemeinsamen Endkanten
e_i^p	gesplittete Kante Nr. i des Wegeverbotes p
$e_{i,e}^p$	gesplittete Kante, die p mit dem Ausgangsgraphen verbindet
$e_{i,j}^{p,p'}$	gesplittete Kante, die p mit p' verbindet
G, G^i	Graphen
g, g_i^w	Kantenbijektionen
g_v	Knotenbijektionen
h	Bijektion
h_E	Kantenbijektion
h_V^p	Knotenbijektion – von p erzeugt
h_1^p	Kantenbijektion der gesplitteten Wege – von p erzeugt
h_2, h_3	Kantenbijektionen der Schritte 2 und 3
i, j, k, l, m, n	Indizes
ι	Einbettung – kanonische Injektion
$I[v]$	Indexnummer von v
K	Kreuzung
λ	leerer Weg
M	Menge aller Kanten, in welche bedingt abgebogen werden darf
N	Menge aller Kanten, in welche abgebogen werden darf
N^e	Menge aller Kanten, in welche von der Kante e aus abgebogen werden darf
$N(i)$	Nachbarschaftsliste nach dem Schritt i

$\omega(e)$	Endknoten von e
P	Menge der Wegeverbote
p	Wegeverbot
$[\![p]\!]$	Menge aller Wegeverbote mit der Anfangskante von p
Π	Projektion
π_i	Projektion auf die i-te Komponente
ψ	Abbildung, die den Verlauf eines Weges im Wegegraphen aufzeigt
q	Index für gemeinsame Kanten
$R(i)$	Menge aller Knoten, die weder in $B(i)$ noch in $N(i)$ sind
s	Startknoten
T	Menge aller Abbiegeverbote
t	Abbiegeverbot
u, v, v_i	Knoten aus V
V	Knotenmenge
V_a, V_i, V_r	äußere Knotenmenge, innere Knotenmenge, Randknotenmenge
v_k^{ij}	Der Knoten v_k wurde über e_{ij} in den kürzeste Wege Baum aufgenommen
W	Menge aller Wege eines Graphen
$W\backslash\backslash P$	Menge aller erlaubten Wege
$[w]$	Menge aller Wege, welche die Anfangskante von w haben
w, w'	Wege aus W
$W(u, v)$	Menge aller Wege von u nach v
X, z_i	Polygonzüge
X_i	Punkt Nr. i des Polygonzuges X
Y_i	Präknoten
Z	Menge von Polygonzügen
z	Zielknoten

Kapitel 1

Einführung

Autonome Routenplanungssysteme gewinnen in den letzten Jahren zunehmend an Bedeutung im praktischen Einsatz.

Ein Test der Zeitschrift ADAC Motorwelt vom April 1997 (Ausgabe 4 Seite 36) hat die sechs gängigsten Planungssysteme auf dem deutschen Markt untersucht. Zur allgemeinen Überraschung haben die sechs Systeme bei ein und derselben Route durch München sechs teilweise unterschiedliche Lösungen angeboten.

Dieses Ergebnis erlaubt den Schluß, daß die Algorithmen, Kontroll- und Programmstrukturen, die den verwendeten Programmen zugrunde liegen, durchaus noch nicht den notwendigen Standard erreicht haben.

Das grundlegende Konzept der Routenplanung besteht im Finden eines kürzesten Weges. Optionale Erweiterungen des Modells, die weitere Parameter wie 'Schnellster Weg', 'Umweltfreundlichkeit', 'Geringste Belastung' usw. in Kombination verwenden, werden hier bewußt außer acht gelassen.

Eine Lösung für das dringendste Problem soll hier beschrieben werden - ein Algorithmus, der den Minimalstandard der Lösung des kürzesten Wege Problems soweit erweitert, daß ein zukünftiger Test nicht mehr sechs unterschiedliche und damit für den einzelnen Anwender suboptimale Lösungen hervorbringt, sondern wirklich den kürzesten Weg.

Wie lassen sich diese suboptimalen Lösungen erklären?

Es gibt in einer Großstadt an ca. 40 - 50 % der Kreuzungen ein Abbiege- oder ein Wegeverbot. Diese Abbiege- und insbesondere die Wegeverbote werden teilweise von den gängigen Routenplanungssystemen falsch oder gar nicht interpretiert.

In der Praxis werden Straßennetze durch Graphen repräsentiert, da Graphen die in der Informatik am häufigsten verwendete Abstraktion darstellen [Tur96].

Daher untersucht diese Arbeit kürzeste Wege in Graphen mit Abbiege- und Wegeverboten.

Überraschenderweise gibt es bisher keine grundlegenden graphentheoretischen Analysen zu diesem Problem.

1.1 Verallgemeinertes optimales Routing

Das Straßennetz wird als gerichteter bewerteter Graph interpretiert, wobei die Gewichte der Kanten zum Beispiel deren Länge oder die Zeitspanne repräsentieren, die benötigt wird, um sich von einem Anfangsknoten zu einem Endknoten zu bewegen. Die Aufgabe, eine Menge von bewegten Objekten (z.B. Kfz) in einem Straßennetz von einem Start- zu einem Zielknoten zu führen, nennt man ein Routingproblem ([Kad95]).

Eine mögliche Lösung dieses Problems ist es den kürzesten Weg zwischen dem Start und Zielknoten zu berechnen und diesen dann zu befahren (statisches Routing). Dieses Verfahren kann versagen, wenn auch im Falle von Störungen (Baustellen, Stauungen, Sperrungen usw.) das Ziel in kurzer Zeit erreicht werden soll. Daher muß die Möglichkeit vorgesehen werden im Störungsfalle kürzeste Wege neu zu berechnen und dann diese zu nutzen oder es müssen Ersatzwege (z. B. k kürzeste Wege) von vornherein berechnet und gespeichert werden (dynamisches Routing).

Es kommt darauf an, die Wege der Verkehrsteilnehmer so anzugeben, daß sich diese möglichst wenig überschneiden. Kommen zwei Autos gleichzeitig an einer Kreuzung an, so muß eventuell eines warten, bevor es weiterfahren kann. Dadurch kann es zu größeren Verzögerungen kommen.

Ist ein Streckenabschnitt blockiert, erschwert passierbar oder überlastet, sind eventuelle Störungen unvermeidbar. Für diesen Fall sollen die Autos auf eine Alternativroute umgeleitet werden können, deren Länge nur unwesentlich vom kürzesten Weg abweicht. Zu diesem Zweck benötigt man von einem Startknoten S aus nicht nur den kürzesten Weg zu einem Zielknoten Z, sondern auch den zweitkürzesten bzw. k–kürzesten Weg [SB77].

Routingverfahren unterteilt man in statische und dynamische Verfahren. Ein statisches Verfahren verwendet die Gewichte der einzelnen Kanten, um kürzeste Wege zu berechnen. Diese Methode ist von Nachteil, wenn der kürzeste Weg eine sich zeitlich verändernde Größe darstellt (z.B. durch Stauungen, Baustellen, ...). Das Verfahren erkennt keine Alternativrouten, wenn man sich in einem überlasteten Streckenabschnitt befindet und ist damit nicht flexibel.

Dynamische Routingverfahren schalten bei übermässiger Belastung (die unter Umständen zu großen Verzögerungen führt) auf andere Wege um (siehe [Kad95]). Sie bestimmen die kuerzesten Wege periodisch die kürzesten Wege periodisch neu und passen sich dadurch an veränderte Straßenverhältnisse an (siehe [KPSKPC95], [LHPS86]). Für jede Kante wird die von einem Auto tatsächlich benötigte Zeit gemessen und der Kante als neues Gewicht zugeordnet. Beim verteilten Routing wird bei jedem Knoten eine Routingtabelle, in der jedem Ziel die Nachbarkreuzung zugeordnet wird, über die der kürzeste Weg verläuft. Jedes Fahrzeug wird mittels der Routingtabelle immer an die richtige Nachbarkreuzung geleitet. Die notwendige Flexibilität ist damit gewährleistet.

Der Aufbau der Datenbank und die Abstraktion von Straßennetzwerken für Computer fordern eine Verwendung von Graphen für das Problem des verallgemeinerten optimalen Routing.

Bewegen wir uns in diesem System dynamisch, so können wir das vorhandene Datenpotential entsprechend unserer Forderung nach einem kürzesten Weg viel besser nutzen.

Aufgrund der oben beschriebenen Eigenschaften ist das statische System als das Schwächere anzusehen. Eine Verwendung eines dynamischen Systems ist damit Grundvoraussetzung für ein verallgemeinertes optimales Routing und langfristig sollen deshalb die statischen Systeme von den dynamischen Systemen abgelöst werden.

1.2 Zielsetzung der Arbeit

Aufgrund zunehmender Verkehrsdichte wird eine effiziente Routenplanung inklusive Ausweichrouten immer wichtiger. Obwohl alle auf dem Markt befindlichen Planer auf das Kartenmaterial zweier Hersteller zurückgreifen, unterscheiden sich die errechneten Routen erheblich.

An dieser Stelle setzt diese Arbeit an. Durch die Integration der Abbiege- und Wegeverbote in bekannte kürzeste Wege Berechnungen, unter dem Dach der Graphentheorie und mit dem GDF-Format als grundlegender Informationsquelle, werden neue Leistungsmerkmale für Routenplaner erschlossen.

Zur Erreichung dieses Ziels werden mehrere Ansätze streng mathematisch definiert, entsprechend in Sätzen ausformuliert und anschließend bewiesen.

Folgendes Schema wird dabei angewendet:

1. Einführung in den Sachverhalt
2. Grundlegende Definitionen
3. Erklärung des Algorithmus
4. Formale Ausarbeitung des Algorithmus
5. Beweis der Gültigkeit und ggf. der Minimalität

Auf diese Weise erreicht man die angestrebte starke Verallgemeinerung. Dabei sind die Algorithmen so allgemein gehalten, daß sie auch in der Praxis sehr selten vorkommende Fälle wie z.B. sich selbst überlappende Wegeverbote berücksichtigen (siehe 5.6.4).

Ein weiteres Ziel ist die Implementation des Algorithmus zur Berechnung kürzester Wege in Graphen mit Abbiege- und Wegeverboten in ein Pascal-Programm. Die Programmiersprache Pascal wurde aufgrund der einfachen Lesbarkeit des Quellcodes gewählt und gewährleistet eine einfache Übertragung der Programmcodes in andere Computersprachen. Diese Implementation ermöglicht es, die gefundenen Ergebnisse praktisch zu testen und auszuwerten.

1.3 Aufbau der Arbeit

Im zweiten Kapitel werden die graphentheoretischen Grundlagen eingeführt. Als erstes wird der Begriff 'Graph' erklärt. Alle anderen Begriffe bauen auf der Definition eines Graphen auf. Die einzelnen Definitionen werden in logischer Abfolge nacheinander dargestellt.

Im dritten Kapitel werden zwei Verfahren zur Berechnung kürzester Wege, vier Modifikationen zur Bewältigung großer Datenmengen und ein Ansatz zur Berechnung zweitkürzester Wege dargestellt. Der Algorithmus von D'Esopo und der $A*$-Algorithmus werden im Detail beschrieben und bewiesen. Diese sind zur Berechnung kürzester Wege in von Straßenverkehrsdaten erzeugten Graphen besonders effizient. Ihre Vorteile gegenüber dem Algorithmus von Dijkstra und den Tripelalgorithmen (vgl. Abschnitte A.1 und A.3) werden herausgearbeitet.

Im vierten Kapitel werden kürzeste Wege in Graphen mit Abbiegeverboten berechnet. Dabei wird zuerst ein naheliegender Ansatz widerlegt. In Folge werden fünf andere Verfahren zur Berechnung kürzester Wege in Graphen mit Abbiegeverboten vorgestellt. Im einzelnen sind dies: Die Methode der Kantenaufnahme, bei welcher nicht mehr die Knoten, sondern die Kanten als Ziele interpretiert werden, die Methode der mehrfachen Knotenaufnahme, bei der Knoten unter gewissen Bedingungen mehrfach als Ziel auftreten können, die Methode des Ziehens neuer Kanten, bei welcher Kreuzungen, an welchen Abbiegeverbote existieren, 'überbrückt' werden und die Methode des verbotsorientierten Knotensplittings und knotenorientierte Netzwerke, bei welchen gewisse Kreuzungen vervielfacht werden. Bei diesen Lösungsansätzen wird entweder der Algorithmus oder der Graph so verändert, daß kürzeste Wege berechnet werden können.

Das fünfte Kapitel stellt den zentralen Teil der Arbeit dar. Hier wird die Methode des verbotsorientierten Knotensplittings zur Berechnung kürzester Wege in Graphen mit Abbiegeverboten auf Wegeverbote erweitert. Anschließend erfolgt der Beweis ihrer Gültigkeit und Minimalität.

Im sechsten Kapitel erfolgt eine programmtechnische Ausarbeitung inklusive einer Beschreibung der Datenformate DXF und GDF. Es wird ein Algorithmus zur Berechnung kürzester Wege in Straßennetzen mit Wegeverboten vorgestellt. Dazu wird durch Datenabstraktion aus vorgegebenen oder selbst erstellten Verkehrsdaten ein Graph mit Abbiegeverboten aufgebaut, der das Straßennetz repräsentiert. Dieser Graph wird nach einem Verfahren aus Kapitel 4 zu einem Wegegraph erweitert. Dies ist ein Graph, in dem im Gegensatz zum Ausgangsgraphen nur noch erlaubte Wege möglich sind. Im Wegegraph werden dann die kürzesten Wege berechnet und graphisch dargestellt.

Im siebten Kapitel wird dem Leser ein Ausblick gegeben, wie die erfolgte Forschung in Zusammenhang mit Ergebnissen anderer Disziplinen gebracht werden kann. Speziell wird dabei auf einen Algorithmus zur Berechnung k-kürzester Wege und auf einen modifizierten Algorithmus zur Berechnung kürzester Wege in Graphen mit Wegeverboten eingegangen. Ein Zukunftsausblick schließt die Betrachtungen ab.

Im Anhang A werden die zwei Standardverfahren (Tripelalgorithmus und Algorithmus von Dijkstra) zur Berechnung kürzester Wege vorgestellt. Der Anhang B beinhaltet Beispiele zu allen Bereichen der Arbeit und Anhang C enthält einige Listings zu den beschriebenen Programmen.

1.4 Danksagung

Diese Arbeit wurde angefertigt mit der finanziellen Unterstützung der Stiftung ‘Besinnung und Ordnung’, für deren großzügiges Stipendium ich mich herzlich bedanken möchte. Ich danke meinen Betreuern Herrn Professor Dieter Fritsch und Herrn Professor Volker Claus für die interessante Aufgabenstellung und viele hilfreiche Gespräche. Bedanken möchte ich mich außerdem bei Herrn Dipl. Inf. Bernd Ansel für Tips im Bereich Management, Herrn Dipl. Ing. Alexander Bauer für Anregungen in den Bereichen Implementation und Verkehrswesen (speziell Modelleisenbahnbau über Wohnzimmertüren), Herrn Dr. rer. nat. Gerd Brüchert für die mathematische Unterstützung über 700 km Distanz, bei Herrn Dipl. Inf. Friedhelm Buchholz für Diskussionen und Anregungen im Bereich Graphentheorie, bei Herrn Dipl. Ing. Markus Englich für die Beratung in bei Softwareproblemen, bei Dr. Ing. Volker Walter für die Einführung in das GDF–Datenformat und bei Tanja Schmid MA für orthographischen Beistand.

Kapitel 2

Grundlagen der Graphentheorie

In diesem Kapitel sollen die notwendigen mathematischen Grundlagen gelegt werden, auf welchen die Arbeit aufbaut. Graphen sind die in der Informatik am häufigsten verwendeten Abstraktionen und entsprechend vielfältig definiert. Um Inkonsistenzen zu vermeiden, werden wir uns konsequent an die Graphdefinition nach [Nol76] halten, die speziell auf Straßennetze mit Multikanten ausgerichtet ist. Die Definitionen 2.9 bis 2.12 sind speziell für diese Arbeit entwickelt worden. Sofern nicht anders vermerkt, sind alle betrachteten Graphen endlich, das heißt, sie enthalten nur endlich viele Knoten und endlich viele Kanten.

Definition 2.1 *(siehe auch [Nol76] Seite 12 ff) Ein* **Graph** *ist ein Quadrupel* (V, E, α, ω)*, bestehend aus einer nichtleeren Menge* V **(Knotenmenge)***, einer nichtleeren Menge* E **(Kantenmenge)***, einer Abbildung* $\alpha : E \to V$*, die jeder Kante* e *ihren* **Anfangsknoten** $\alpha(e)$ *zuordnet, und einer Abbildung* $\omega : E \to V$*, die jeder Kante* e *ihren* **Endknoten** $\omega(e)$ *zuordnet. Mit den Abbildungen* α *und* ω *wird der Graph zu einem* **gerichteten** *Graphen.*

Eine Kante e mit $\alpha(e) = \omega(e)$ heißt **Schlinge**. Zwei Kanten e, e' heißen **parallel**, wenn $\alpha(e) = \alpha(e')$ und $\omega(e) = \omega(e')$. Gibt es zu einer Kante e eine oder mehrere parallele Kanten, so nennt man e auch eine **Multikante**. Falls es in einem Graphen keine parallelen Kanten gibt, sprechen wir von einem parallelenfreien Graphen; hier kann die Kantenmenge E in $V \times V$ eingebettet werden. Die Einbettung ι wird definiert als $\iota(e) := (\alpha(e), \omega(e))$.

Definition 2.2 *(siehe auch [Nol76] Seite 40 ff) Ein Graph* (V, E, α, ω) *zusammen mit einer Abbildung* $\beta : E \to \mathbb{R}$ *heißt* **gewichteter** *Graph. Falls* $\beta(e) \in {\mathbb{R}_0}^+$*, dann heißt* $\beta(e)$ *die Länge der Kante* e*. Ist* $\beta(e) \geq 0$ *für alle* e*, dann heißt der Graph* **positiv***.*

Ist $(V, E, \alpha, \omega, \beta)$ ein parallelenfreier gewichteter Graph, so kann man β erweitern zu einer Abbildung $\beta' : V \times V \to \mathbb{R} \cup \{\infty\}$ durch

$$\beta'_{uv} := \beta'(u, v) := \begin{cases} \beta(e) & \text{falls } \alpha(e) = u \text{ und } \omega(e) = v \quad \text{für ein } e \in E \\ \infty & \text{sonst.} \end{cases}$$

In diesem Falle genügt es also, (V, β') anzugeben, um den Graphen zu beschreiben. Die Matrix $\beta'_{uv \in V}$ heißt **Adjazenz-** oder **Nachbarschaftsmatrix** des Graphen (siehe dazu auch [Cla94]).

Definition 2.3 *(siehe auch [AHU83] Seite 50 ff oder [Har74] Seite 20 ff) Sei $G = (V, E, \alpha, \omega)$ ein Graph. Eine Kante e heißt* **bidirektional***, wenn zu e ein $e' \in E$ existiert mit $\alpha(e) = \omega(e')$ und $\omega(e) = \alpha(e')$; ist der Graph gewichtet, so soll außerdem $\beta(e) = \beta(e')$ gelten. Eine Kante, für die dies nicht gilt, heißt* **unidirektional***. Sind alle Kanten bidirektional, so heißt der Graph* **ungerichtet***.*

In den folgenden Graphdarstellungen werden Knoten immer als Kreise mit inliegender Nummer, unidirektionale Kanten als Pfeile und bidirektionale Kanten als Linien dargestellt.

Definition 2.4 *(siehe auch [Har74] Seite 208 ff oder [Nol76] Seite 28 ff)*
Es seien $G = (V, E, \alpha, \omega)$ ein Graph und $e_1, ..., e_n \in E$. Das n-Tupel $(e_1, e_2, ..., e_n)$ heißt **Weg** *(von $\alpha(e_1)$ nach $\omega(e_n)$), wenn $\omega(e_i) = \alpha(e_{i+1})$ für alle $i = 1, ..., n-1$ gilt. Ist $w = (e_1, e_2, ..., e_n)$ ein Weg, dann heißt e_1 die* **Anfangskante***, e_n die* **Endkante***, $\alpha(e_1)$ der* **Anfangsknoten** *und $\omega(e_n)$ der* **Endknoten** *des Weges. Die Menge aller Wege in einem Graphen heißt W.*

Ein Weg $w' = (e'_1, , ..., e'_m)$ heißt **Teilweg** *des Weges w, wenn ein Index i mit $1 \leq i \leq n-m+1$ existiert mit $e_{i+j-1} = e'_j$ für $j = 1, ..., m$.*

Die Abbildung $\pi_i : W \to E$ mit $\pi_i(e_1, ..., e_n) := e_i$ heißt **Projektion** *auf die i-te Komponente für $1 \leq i \leq n$. Der Graph G heißt* **zusammenhängend***, wenn zu je zwei Knoten $u, v \in V$ immer mindestens ein Weg von u nach v existiert.*

Für $n = 0$ definieren wir den leeren Weg λ. Dieser besitzt weder Anfangsknoten noch Endknoten. Ein Weg $(e_1, ..., e_n)$ heißt **geschlossen**, wenn sein Anfangsknoten gleichzeitig sein Endknoten ist, das heißt $\alpha(e_1) = \omega(e_n)$. Wenn G parallelenfrei ist, kann ein Weg $(e_1, e_2, ..., e_n)$ auch eindeutig durch seine $n + 1$ Knoten $(\alpha(e_1), \alpha(e_2), ..., \alpha(e_n), \omega(e_n))$ dargestellt werden.

Sei $G = (V, E, \alpha, \omega, \beta)$ ein gewichteter Graph, W die Menge aller Wege in G, dann kann die Abbildung β auf Wege erweitert werden durch

$$\delta : W \to \mathbb{R} \qquad \delta(e_1, ..., e_n) := \sum_{i=1}^{n} \beta(e_i). \tag{2.1}$$

Man nennt $\delta(w)$ die **Länge des Weges** w.

Sind G ein gewichteter Graph ohne geschlossene Wege mit negativer Länge, $u, v \in V$, $W(u, v)$ die Menge aller Wege von u nach v, dann heißt ein Weg w **kürzester Weg** von u nach v, wenn gilt

$$\delta(w) \leq \delta(w') \quad \text{für alle } w' \in W(u, v). \tag{2.2}$$

In Graphen mit geschlossenen Wegen mit negativer Länge gibt es im allgemeinen keine kürzesten Wege.

Definition 2.5 *(siehe auch [AHU83] Seite 52 ff oder [Nol76] Seite 60 ff) Ein Graph* $G = (V, E, \alpha, \omega)$ *heißt* **Baum**, *wenn*

1. *G besitzt keine geschlossenen Wege.*
2. *Es gibt einen ausgezeichneten Knoten* v' *mit* $\omega(e) \neq v'$ *für alle* $e \in E$. v' *heißt* **Wurzel**.
3. *Jeder andere Knoten ist Endknoten von genau einer Kante.*
4. *Zu jedem* $v \in V$ *existiert ein Weg von* v' *nach* v.

Sei eine Kante e in einem Baum gegeben, dann heißt $\alpha(e)$ **Vater** *von* $\omega(e)$ *und* $\omega(e)$ **Sohn** *von* $\alpha(e)$. *Ein Knoten ohne Söhne heißt* **Blatt**. *Ein Baum, bei dem jeder Knoten maximal zwei Söhne besitzt, heißt* **Binärbaum**.

Bemerkung 1: Sei $v \in V$, dann ist der Weg w von v' nach v eindeutig. Die Anzahl der Kanten dieses Weges heißt **Höhe** des Knotens v. Das Maximum der Höhen aller Knoten heißt **Höhe des Baumes**.

Bemerkung 2: Der Verlauf des kürzesten Weges zwischen zwei Knoten ist im allgemeinen nicht eindeutig (es kann mehrere kürzeste Wege zwischen zwei Knoten geben). Entscheidet man sich eindeutig für einen kürzesten Weg zwischen zwei Knoten, so kann die Menge aller kürzesten Wege von einem Startknoten aus als Baum dargestellt werden. Dieser Baum heißt **kürzester-Wege-Baum**.

Satz 2.1 *Sei* $G = (V, E, \alpha, \omega, \beta)$ *ein positiver, zusammenhängender, ungerichteter Graph mit* $\beta(e) = \beta(e')$ *für alle bidirektionalen Kanten* e, e'. *Wir definieren eine Abstandsfunktion* $d : V \times V \to \mathbf{R}^+$ *durch* $d(u, v) := \delta(w)$ *als Länge eines kürzesten Weges von u nach v und* $d(v, v) := 0$. *Mit dieser Definition ist* (V, d) *ein halbmetrischer Raum. Gilt sogar* $\beta(e) > 0$ *für alle* $e \in E$, *so ist* (V, d) *ein metrischer Raum (siehe dazu auch [Jun90]).*

Beweis:

1. $d(u, v) \geq 0$ und $d(v, v) = 0$ sind klar nach Definition. Falls $\beta(e) > 0$ für alle $e \in E$, folgt aus $d(u, v) = 0$ sofort $u = v$.
2. $d(u, v) = d(v, u)$ folgt aus der Ungerichtetheit (wegen $\beta(e) = \beta(e')$ für bidirektionale e, e').
3. $d(u, w) \leq d(u, v) + d(v, w)$ ist klar nach der Dreiecksungleichung für min.

■

Definition 2.6 *(siehe [Nol76] Seite 29 ff) Seien* $w_1 = (e_1^1, ..., e_n^1)$ *und* $w_2 = (e_1^2, ..., e_m^2)$ *zwei Wege, dann ist* w_1 *mit* w_2 **konkatenierbar** *oder* **zusammensetzbar**, *wenn* $\omega(e_n^1) = \alpha(e_1^2)$ *gilt. Den zusammengesetzten Weg* $(e_1^1, ..., e_n^1, e_1^2, ..., e_m^2)$ *bezeichnen wir mit* $w_1 + w_2$.

Für λ gilt: $w_1 + \lambda = \lambda + w_1 = w_1$. λ ist das neutrale Element der Konkatenation. Die so erklärte Konkatenation ist assoziativ, das heißt, für drei Wege w_1, w_2 und w_3 gilt $(w_1 + w_2) + w_3 = w_1 + (w_2 + w_3)$, aber die Konkatenation ist nicht kommutativ. Ferner gilt $\delta(w_1 + w_2) = \delta(w_1) + \delta(w_2)$.

Sind $G = (V, E, \alpha, \omega)$ ein parallelenfreier Graph, $w = (e_1, ..., e_n)$ ein nichtleerer Weg in G und e eine Kante von $\omega(e_n)$ nach $v \in V$, dann setzen wir $w + v := w + (e) = (e_1, ..., e_n, e)$ ('Anhängen eines Endknotens').

Definition 2.7 *(siehe [Nol76] Seite 31 ff) Sei $G = (V, E, \alpha, \omega)$ ein Graph, dann heißt ein Graph $G' = (V', E', \alpha', \omega')$* **Teilgraph** *von G, wenn $V' \subseteq V$, $E' \subseteq E$, $\alpha|_{E'} = \alpha'$, $\omega|_{E'} = \omega'$ (und gegebenenfalls $\beta|_{E'} = \beta'$, wenn die Graphen gewichtet sind).*

Sei $E' \subseteq E$. Ein Knoten $v \in V$ heißt **innerer Knoten** von E', wenn (1) v Anfangs– oder Endknoten mindestens einer Kante $e' \in E'$ ist, und (2) es keine Kante $e \in E \backslash E'$ gibt, die v als Anfangsknoten oder Endknoten hat. $v \in V$ heißt **Randknoten** von E', wenn v Anfangs– oder Endknoten einer Kante $e' \in E'$ und Anfangs– oder Endknoten einer Kante $e \in E \backslash E'$ ist. $v \in V$ heißt **äußerer Knoten** von E', wenn es keine Kante $e' \in E'$ gibt, die v als Anfangs– oder Endknoten hat. Bezeichnet man die Menge der inneren Knoten von E' mit $V_i(E')$, die Menge der Randknoten von E' mit $V_r(E')$ und die Menge der äußeren Knoten von E' mit $V_a(E')$, dann ist offensichtlich $V = V_i(E') \cup V_r(E') \cup V_a(E')$ eine Partition von V. Setzt man noch $V(E') := V_i(E') \cup V_r(E')$, dann ist $G(E') := (V(E'), E', \alpha|_{E'}, \omega|_{E'})$ ein Teilgraph von (V, E, α, ω).

Eine Kantenmenge E' heißt **wegweise zusammenhängend**, wenn zu je zwei Kanten $e \neq e'$ aus E' ein Weg $w = (f_1, ..., f_m)$ von $\omega(e)$ nach $\alpha(e')$ existiert mit $\{f_1, ..., f_m\} \subseteq E'$.

Definition 2.8 *Ein Graph heißt* **planar** *oder* **plättbar**, *wenn es eine Darstellung des Graphen in der Ebene gibt, so daß sich die Kanten höchstens in ihren Eckpunkten schneiden.*

Auf eine genauere mathematische Definition möchte ich verzichten. Weitere Informationen zu planaren Graphen finden sich z. B. in [NiC88] und [KRRS94].

Satz 2.2 Eulerscher Polyedersatz *Gegeben sei ein konvexes Polyeder mit n Ecken, m Kanten (jedes Paar bidirektionaler Kanten wird einfach gezählt) und g Gebieten, so gilt:*

$$n + g = m + 2.$$

Der Beweis wird mit vollständiger Induktion nach m geführt und befindet sich zum Beispiel in [Tur96]. Ein zusammenhängender planarer Graph kann als Polyeder im weiteren Sinn gedeutet werden. Deshalb folgt aus dem Eulerschen Polyedersatz eine nützliche Abschätzung für planare Graphen:

Satz 2.3 *Sei G ein zusammenhängender, parallelenfreier, planarer, ungerichteter Graph mit $n \geq 3$ Knoten und $m \geq 2$ Kanten, dann gilt*

$$m \leq 3n - 6.$$

Beweis: Jede Kante trägt zur Abgrenzung von genau zwei Gebieten bei, und jedes Gebiet wird von mindestens drei Kanten (da G parallelenfrei ist) begrenzt. Damit gilt $3g \leq 2m$. Mit dem Eulerschen Polyedersatz folgt die Behauptung. Dieser Satz kann durch Addition der einzelnen Gleichungen auch für nicht zusammenhängende planare Graphen bewiesen werden. ∎

Definition 2.9 *Sei $G = (V, E, \alpha, \omega)$ ein Graph. Ein Weg (e_1, e_2) heißt ein* **Abbiegeverbot** *bei $\omega(e_1)$, wenn die Sequenz $.., e_1, e_2, ..$ in einem Weg ausdrücklich nicht vorkommen darf. Liegen mehrere Abbiegeverbote vor, so faßt man diese zu einer Menge $T \subseteq W$ zusammen ('turn prohibitions').*

Analog nennt man für $n \geq 2$ einen Weg $(e_1, ..., e_n)$ ein **Wegeverbot**, *wenn die Sequenz $..., e_1, ..., e_n, ...$ in Wegen nicht vorkommen darf und faßt die Wegeverbote zu einer Menge $P \subseteq W$ zusammen ('path prohibitions'). Es sei*

$$W \backslash\backslash P := \{w \in W \mid w \text{ enthält kein } p \in P \text{ als Teilstück } \}$$

die Menge der unter Beachtung der Wegeverbote P zugelassenen Wege. Insbesondere kann jedes Abbiegeverbot (e_1, e_2) als spezielles Wegeverbot gesehen werden.

Sei $G = (V, E, \alpha, \omega)$ ein Graph, dann ist auf der Menge aller Wege W eine Äquivalenzrelation $\sim$ definiert durch

$$w = (e_1, ..., e_n) \sim w' = (e'_1, ..., e'_m) \quad :\Leftrightarrow \quad e_1 = e'_1. \tag{2.3}$$

$\sim$ ist reflexiv, da $w \sim w$ für alle $w \in W$, $\sim$ ist symmetrisch, da offensichtlich $w \sim w' \Rightarrow w' \sim w$, und $\sim$ ist transitiv, da $w \sim w'$ und $w' \sim w'' \Rightarrow w \sim w''$. Zu jedem $w \in W$ gehört die sogenannte **Äquivalenzklasse** $[w]$, das ist die Menge aller zu w in Relation stehenden Wege:

$$[w] := \{w' \in W \mid w' \sim w\}. \tag{2.4}$$

Ist $P \subseteq W$ eine Menge von Wegeverboten, so kann man analog eine Äquivalenzrelation auf P definieren, deren Äquivalenzklassen wir mit $[\![p]\!]$ bezeichnen. Hierbei gilt $[\![p]\!] = [p] \cap P$.

Definition 2.10 *Sei $G := (V, E, \alpha, \omega)$ ein Graph, P eine Menge von Wegeverboten, $w = (e_1\ , ...,\ e_n) \in W$ und $\tilde{P} \subseteq [\![w]\!]$. Für jedes $\tilde{p} = (\tilde{e}_1, ..., \tilde{e}_m) \in \tilde{P}$ sei*

$$aga(\tilde{p}, w) := \max\{k \mid e_i = \tilde{e}_i \text{ für } i = 1, ..., k\}$$

die Anzahl der gemeinsamen Anfangskanten von $\tilde{p}$ und w; ferner sei

$$Aga(\tilde{P}, w) := \max\{aga(\tilde{p}, w) \mid \tilde{p} \in \tilde{P}\}$$

die maximale Anzahl gemeinsamer Anfangskanten von Elementen von $\tilde{P}$ und w.
Falls $aga(p', w) = Aga(\tilde{P}, w)$ gilt, nennt man p' einen **maximalen Anfangsweg** *von w in $\tilde{P}$. Existiert kein Weg dieser Art, so sei $p' := \lambda$ und $Aga(\tilde{P}, w) := 0$.*

Definition 2.11 *Sei $G := (V, E, \alpha, \omega)$ ein Graph und seien $w = (e_1\ ,\ ...,\ e_n)$, $w' = (e'_1, ..., e'_m) \in W$, dann heißt w'* **Endweg** *von w, wenn ein Index $q \geq 0$ existiert, so daß*

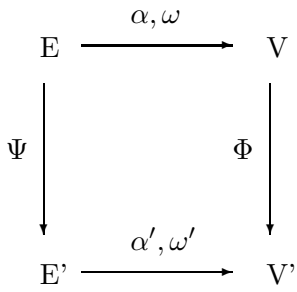
$$e_{q+i} = e'_i \text{ für } 1 \leq i \leq n - q.$$

Jeder Weg ist Anfangsweg und Endweg von sich selbst. Nach dem Ende von w kann w' noch weitere Kanten besitzen. Existieren mehrere Indizes q dieser Art, so soll immer das minimale q gewählt werden. Wir definieren $eka(w, w') := n - q$. Der Weg w' beginnt innerhalb des Weges w. Die Wege verlaufen dann $n - q$ Kanten gemeinsam bis zum Ende von w. w' geht nach dem Ende von w eventuell noch weiter. Es ist möglich, daß w und w' gegenseitig voneinander Endwege sein können, ohne daß sie identisch sind – siehe dazu Beispiel B.1 und die Abschnitte 5.6.3 und 5.6.4.

Definition 2.12 *Sind $G = (V, E, \alpha, \omega)$ und $G' = (V', E', \alpha', \omega')$ zwei Graphen, dann nennen wir ein Paar (Φ, Ψ) von Abbildungen, $\Phi : V \to V'$ und $\Psi : E \to E'$, einen* **Homomorphismus** *von G nach G', wenn sie Anfangs– und Endknoten erhaltend sind, das heißt, wenn*

$$\alpha'(\Psi(e)) = \Phi(\alpha(e)) \quad \text{und} \quad \omega'(\Psi(e)) = \Phi(\omega(e)) \qquad \text{für alle } e \in E. \tag{2.5}$$

Die Gleichungen (2.5) kann man auch kurz in die Form $\alpha' \circ \Psi = \Phi \circ \alpha$ und $\omega' \circ \Psi = \Phi \circ \omega$ fassen, was besagt, daß das folgende Diagramm kommutiert:



Aus der Erhaltung von Anfangs– und Endknoten folgt, daß jeder Weg $w = (e_1, ..., e_n) \in W$ auf einen Bildweg $\Psi(w) := (\Psi(e_1), ..., \Psi(e_n)) \in W'$ führt. In diesem Sinne induziert $\Psi : E \to E'$ eine Abbildung $W \to W'$, die wir ebenfalls mit Ψ bezeichnen. Ist G' ein Graph mit Abbiege– oder Wegeverboten P' und G ein Graph ohne Abbiege– oder Wegeverbote, dann heißt G eine **wegeverbotsfreie Erweiterung** oder **Wegegraph** von G', wenn ein Homomorphismus (Φ, Ψ): $G \to G'$ existiert mit

(VE 1) Ψ, Φ sind surjektiv,
(VE 2) zu jedem Weg $w' \in W' \backslash\backslash P'$ existiert ein Weg $w \in W$ mit $\Psi(w) = w'$,
(VE 3) für jeden Weg $w \in W$ gilt $\Psi(w) \in W' \backslash\backslash P'$.

Gelegentlich werden wir einen Graphen (V, E, α, ω) durch Hinzunahme von neuen Kanten $\tilde{E}$ bzw. neuen Knoten $\tilde{V}$ erweitern. Dies soll immer heißen, daß $V, E, \tilde{V}, \tilde{E}$ paarweise disjunkt sind. Die Anzahl der Elemente einer Menge M werden wir mit $|M|$ bezeichnen.

Kapitel 3

Kürzeste Wege in einem Graphen

In der Literatur (z. B. [Tur96, 241-242]) werden verschiedene Probleme der Berechnung kürzester Wege diskutiert. Beim Problem der kürzesten Wege unterscheiden wir:

1. Von nur einem Startknoten aus wird der kürzeste Weg zu nur einem Zielknoten gesucht. Dies wird in der Literatur als 'Source Target Shortest Path Problem' (**STSP**) beschrieben.
2. Von nur einem Startknoten aus werden die kürzesten Wege zu allen Knoten des Graphen gesucht. In der Literatur kommt dies unter dem Stichwort 'Single Source Shortest Path Problem' (**SSSP**) vor. Bei Umorientierung der Kanten oder ungerichteten Graphen ist dieses Problem äquivalent zu: Von allen Knoten eines Graphen wird der kürzeste Weg zu nur einem Zielknoten gesucht.
3. Zwischen allen Paaren von Knoten werden die kürzesten Wege gesucht. Diese Aufgabenstellung wird auch 'All Pair Shortest Path Problem' (**APSP**) genannt.

Bis heute ist kein Algorithmus bekannt, der das STSP Problem im 'worst case' in geringerer Zeitkomplexität löst als das SSSP. Jeder Algorithmus, der das SSSP Problem löst, löst durch $|V|$-fache Anwendung auch das APSP. In den Abschnitten A.1 und A.3 befinden sich der Dijkstra-Algorithmus zur Lösung des SSSP und die sogenannten Tripelalgorithmen zur Lösung von APSP.

In diesem Kapitel werden wir uns mit weiteren Algorithmen beschäftigen, die das SSSP lösen. Zur Berechnung kürzester Wege in Graphen, die von Straßenverkehrsnetzen analog zum Abschnitt 6.5 erzeugt sind, werden wir in Folge vier Betrachtungen durchführen. Insbesondere werden zwei Algorithmen untersucht, die spezielle Eigenschaften von Straßennetzen berücksichtigen und dort geringe Rechenzeiten benötigen. Diese Eigenschaften sind am Ende der einzelnen Abschnitte erläutert.

3.1 Der Algorithmus von D'Esopo

Der Algorithmus von D'Esopo wird z. B. in [Pol61] beschrieben. Dieser Algorithmus zur Lösung des SSSP ist in Straßennetzen besonders effizient, da diese ebenen Charakter besitzen. Der große Vorteil des Algorithmus von d'Esopo gegenüber dem Dijkstra-Algorithmus ist die Vermeidung einer eventuell aufwendigen Minimumsuche. Siehe dazu den letzten Absatz von 3.1 und Abschnitt A.2.

Gegeben sei ein positiver Graph $G = (V, E, \alpha, \omega, \beta)$. Gesucht werden alle kürzesten Wege von einem Startknoten $s \in V$ aus. Jeder Knoten v des Graphen bekommt eine Indexnummer $I[v]$ und eine Distanz $d(v)$, die den kürzesten bisher bekannten Abstand zum Startknoten bezeichnet. Der Index entscheidet, ob ein neuer Knoten untersucht werden soll, oder ob eine Korrektur notwendig ist. Das Kontrollregister beinhaltet den Index des Knotens, der gerade betrachtet wird. Dieser heißt *aktueller* Knoten und wird mit a bezeichnet. Der Index i beinhaltet den kleinsten Index des Knotens, der in einem Erweiterungsschritt 3 eine neue Distanz zugewiesen bekommen hat. Die am Ende des Algorithmus markierten Kanten ergeben den Kürzeste-Wege-Baum.

Erster Schritt:
Der Startknoten bekommt den Index 1 und die Distanz 0, alle anderen Knoten erhalten den Index 0 und die Distanz ∞. Das Kontrollregister enthält den Wert 1. Fahre mit dem zweiten Schritt fort.

Zweiter Schritt:
Suche alle Endknoten von Kanten, die den aktuellen Knoten als Anfangsknoten besitzen. Gibt es solche Knoten, so bekommt jeder dieser Knoten, der noch Index 0 besitzt den nächsten verfügbaren Index zugeteilt, und wir fahren mit dem dritten Schritt fort. Gibt es solche Knoten nicht, so gehe zu Schritt 4.

Dritter Schritt:
Wir betrachten alle Kanten e, für die $\alpha(e) = a$ gilt und bilden $D := \beta(e) + d(a)$. Folgende drei Möglichkeiten sind denkbar:

1. $d(\omega(e)) = \infty$: Der angeschlossene Knoten besitzt also noch keine Distanz. Dann definieren wir $d(\omega(e)) := D$. Die Kante e wird markiert. Fahre mit der nächsten Kante fort, die in a beginnt. Existiert keine, so gehe zu Schritt 4.

2. $d(\omega(e)) \leq D$: Hier ist nichts zu tun, da bereits ein kürzerer Weg bekannt ist. Fahre mit der nächsten angrenzenden Kante fort und, falls keine existiert, gehe zu Schritt 4.

3. $d(\omega(e)) > D$: Wiederum weisen wir seiner Distanz den Wert der Summe $d(\omega(e)) := D$ zu. Markiere die Kante e und entferne die Markierung von der Kante, die $\omega(e)$ mit dem (kürzeste) Wegebaum verbindet. Zusätzlich betrachten wir noch den Index $I[\omega(e)]$.

   Gilt $I[\omega(e)] < I[a]$ und $I[\omega(e)] < i$, so setze $i := I[\omega(e)]$. Wiederhole Schritt 3 mit der nächsten angrenzenden Kante und, falls keine existiert, gehe zu Schritt 4.

Vierter Schritt:
Wenn ein Index i im Fall 3.3 gespeichert wurde, so weisen wir dessen Wert dem Kontrollregister

zu. Wenn nicht, erhöhen wir dies um 1 und bestimmen dazu den aktuellen Knoten a. Gibt es keinen derartigen Knoten mehr, endet der Algorithmus, sonst Schritt 2.

Die Gültigkeit des Algorithmus folgt aus der Tatsache, daß die Menge aller kürzesten Wege von einem Knoten aus als Baum aufgefaßt werden kann. Wird ein 'Fehler' in dem bereits berechneten Kürzeste-Wege-Baum gefunden, so wird dieser im Schritt 3.3 korrigiert.

In einem Straßennetz münden in einer Kreuzung in der Regel 4 Straßen. Da jetzt jeder Knoten nur wenig benachbarte Knoten besitzt, wird der Indexzähler nur langsam größer. Normalerweise erscheint ein Knotenindex höchstens zwei Mal im Kontrollregister. In diesem Falle wäre die Zeitkomplexität des Algorithmus $O(|V|)$. Besonders effizient ist der Algorithmus in Verbindung mit einer Einschränkung des Graphen (Abschnitt 3.3.4). Im schlimmsten Fall ist dessen Zeitkomplexität jedoch $O(|V|^2)$, da jeder Knoten maximal $|V|$ mal betrachtet werden muß.

3.2 Der $A*$-Algorithmus

In diesem Abschnitt werden wir den $A*$-Algorithmus zur Lösung des SSSP (siehe auch [Tur96, 256 - 265]) betrachten. Dieser Algorithmus ist eine Erweiterung des Algorithmus von Dijkstra (siehe Abschnitt A.1). Der $A*$-Algorithmus verändert nur die Reihenfolge der Aufnahme der Knoten in den Kürzeste-Wege-Baum.

Definition 3.1 *Sei $G = (V, E, \alpha, \omega, \beta)$ ein positiver Graph, $f : V \times V \to \mathbb{R}_0^+$ eine Schätzfunktion für den Abstand $d : V \times V \to \mathbb{R}_0^+$. Eine solche Funktion f heißt* **konsistent***, wenn für alle Kanten $e \in E$ und alle $z \in V$ gilt:*

$$f(\alpha(e), z) \leq \beta(e) + f(\omega(e), z) \text{ und } f(z, z) = 0.$$

Satz 3.1 *Sei $G = (V, E, \alpha, \omega, \beta)$ ein positiver Graph und f eine konsistente Schätzfunktion. Ferner seien $x, y, z \in V$ und w ein Weg von x nach y, dann gilt:*

$$\begin{aligned}
&(1) \quad f(x,z) \leq \delta(w) + f(y,z) && (3.1)\\
&(2) \quad f(x,z) \leq d(x,y) + f(y,z) && (3.2)\\
&(3) \quad f(x,y) \leq \delta(w) && (3.3)\\
&(4) \quad f(x,z) \leq d(x,z) && (3.4)
\end{aligned}$$

Eine Schätzfunktion mit der Eigenschaft (4) *heißt* **zulässig**.

Beweis: Wir beweisen zunächst (1) durch vollständige Induktion nach der Anzahl n der Kanten von w. Für $n = 1$ ist $w = (e)$ mit $\alpha(e) = x$ und $\omega(e) = y$. Es folgt

$$\begin{aligned}
f(x,z) &= f(\alpha(e), z) && \text{wegen } x = \alpha(e)\\
&\leq \beta(e) + f(\omega(e), z) && \text{da } f \text{ konsistent ist}\\
&= \delta(w) + f(y,z) && \text{weil } w = (e) \text{ und } y = \omega(e).
\end{aligned}$$

Ist nun $w = (e_1, ..., e_n)$, dann setze $x' := \omega(e_1) = \alpha(e_2)$ und $w' := (e_2, ..., e_n)$. Damit gilt:

$$\begin{array}{lll} f(x,z) & \leq \delta(e_1) + f(x',z) & \text{Fall } n=1 \text{ angewendet auf } x, x', z \\ & \leq \delta(e_1) + \delta(w') + f(y,z) & \text{nach Induktionsannahme angewendet auf } x', y, z \\ & = \delta(w) + f(y,z) & \text{wegen } w = (e_1) + w' \end{array}$$

was zu zeigen war. Da (1) für beliebige Wege von x nach y gilt, ist diese Bedingung insbesondere auch für einen kürzesten Weg erfüllt, woraus (2) folgt. (3) folgt nun aus (1) mit $y = z$ und (4) folgt aus (2) mit $y = z$. ∎

Gegeben sei ein positiver Graph $G = (V, E, \alpha, \omega, \beta)$ mit konsistenter Schätzfunktion f. Gesucht ist der kürzeste Weg zwischen zwei Knoten $s, z \in V$. Wie beim Algorithmus von Dijkstra wird auch beim $A*$-Algorithmus schrittweise ein Kürzeste-Wege-Baum B erschaffen. Bei der Auswahl der in diesen Baum aufgenommenen Knoten spielt dabei nicht nur die Länge des bereits zurückgelegten Weges, sondern auch die geschätzte Länge des Restweges eine Rolle.

Algorithmus:

Sei $B(0) := \{s\}$ und $B(i)$ der Kürzeste-Wege-Baum nach dem Schritt i. Für $B(i)$ bestimmen wir das

$$\min\{d(s, \alpha(e)) + \beta(e) + f(\omega(e), z) \mid e \in E, \alpha(e) \in B(i),\ \omega(e) \notin B(i)\}. \tag{3.5}$$

Die Addition von $f(\omega(e), z)$ ist der entscheidende Unterschied zum Dijkstra-Algorithmus. In $B(i+1)$ werden diejenigen Endknoten $\omega(e)$aufgenommen, für die dieses Minimum angenommen wird. Wenn der Knoten z erreicht ist, endet der $A*$-Algorithmus.

Satz 3.2 *Der $A*$-Algorithmus berechnet den kürzesten Weg zwischen s und allen Knoten $v \in V$, sofern sie erreichbar sind.*

Beweis: Wir beweisen ähnlich wie beim Dijkstra–Algorithmus die folgende Aussage durch vollständige Induktion nach i. Für $i = 0$ ist nichts zu zeigen.

Induktionsschluß: Sei $v = \omega(e)$ ein im $i+1$-ten Schritt aufgenommener Knoten und $d(s, \alpha(e)) + \beta(e)$ die vom Algorithmus 'berechnete Entfernung' von s nach v. Falls wir für einen beliebigen Weg w' von s nach v zeigen können, daß $d(s, \alpha(e)) + \beta(e) \leq \delta(w')$, so ist die vom Algorithmus 'berechnete Entfernung' tatsächlich gleich $d(s, v)$. Sei also w' ein Weg von s nach v und e' die erste Kante von w' mit der Eigenschaft $\alpha(e') \in B(i)$ und $\omega(e') \notin B(i)$ (eine solche Kante existiert, da andernfalls $v \in B(i)$ wäre). Wir zerlegen w' in Teilstücke durch $w' = w'_1 + (e') + w'_2$ und erhalten

$$\begin{array}{lll} \delta(w') & = \delta(w'_1) + \delta(e') + \delta(w'_2) & \text{Weglänge ist additiv} \\ & = \delta(w'_1) + \beta(e') + \delta(w'_2) & (e') \text{ ist einelementig} \\ & \geq \delta(w'_1) + \beta(e') + f(\omega(e'), v) & \text{nach Satz 3.1(3)} \\ & \geq d(s, \alpha(e')) + \beta(e') + f(\omega(e'), v) & \text{wegen } \alpha(e') \in B(i) \\ & \geq d(s, \alpha(e)) + \beta(e) + f(\omega(e), v) & \text{Minimumbildung des Algorithmus} \\ & = d(s, \alpha(e)) + \beta(e) & \omega(e) = v \end{array}$$

■

Bemerkungen: Sei $f(u,v) = 0$ für alle $u, v \in V$, so ist f eine konsistente Schätzfunktion, da $\beta(e) \geq 0$ und der $A*$-Algorithmus ist der Dijkstra-Algorithmus. Auch die Funktion $f(u,v) = d(u,v)$ für alle $u, v \in V$ ist eine konsistente Schätzfunktion. Wird diese gewählt, so betrachtet der Algorithmus zuerst alle Knoten, die auf einem kürzesten Weg von s nach z liegen. Die Geschwindigkeit des Algorithmus hängt also wesentlich von der Wahl der Funktion f ab. Für die schlechteste Schätzfunktion $f \equiv 0$ ist die Zeitkomplexität wie beim Dijkstra-Algorithmus $O(|V| * \log|V| + |E|)$, bei der besten Schätzfunktion $f(u,v) = d(u,v)$ liegt der Aufwand bei $O(|V|)$, wenn der kürzeste Weg zwischen allen Knotenpaaren eindeutig ist.

Statt eines Zielknotens z kann im $A*$-Algorithmus auch eine Menge von Zielknoten angegeben werden. Diese Erweiterung bleibt für unsere Anwendung aber ohne Bedeutung.

Ist nur eine zulässige, nicht aber eine konsistente Funktion f bekannt, so kann auf ähnliche Weise ein Kürzeste-Wege-Algorithmus definiert werden.

Satz 3.3 *Sei $G = (V, E, \alpha, \omega, \beta)$ ein positiver Graph und sei eine weitere Gewichtsfunktion $\beta' : E \to \mathbb{R}_0^+$ definiert mit $\beta'(e) \leq \beta(e)$ für alle $e \in E$, dann ist die Funktion*

$$f : V \times V \to \mathbb{R}_0^+, \quad f(u,v) := d'(u,v)$$

eine konsistente Schätzfunktion, wobei $d'(u,v)$ der kürzeste Weg von u nach v in $(V, E, \alpha, \omega, \beta')$ ist.

Beweis: Es ist $f(v,v) = d'(v,v) = 0$ für alle $v \in V$ nach Definition des kürzesten Weges. Sei $e \in E$, $z \in V$ beliebig gewählt, dann gilt

$$\begin{array}{lll} f(\alpha(e), z) & = d'(\alpha(e), z) & \text{Definition von } f \\ & \leq \beta'(e) + d'(\omega(e), z) & \text{Dreiecksungleichung für kürzeste Wege} \\ & = \beta'(e) + f(\omega(e), z) & \text{Definition von } f \\ & \leq \beta(e) + f(\omega(e), z) & \text{Definition von } \beta'. \end{array}$$

■

In Graphen, die wie im Abschnitt 6.5 definiert sind, kann also eine konsistente Schätzfunktion

$$f(u,v) := d_e(u,v) \quad \text{für alle } u, v \in V$$

definiert werden. Damit kann die Luftlinienentfernung $d_e(u,v)$ zweier Punkte zur Berechnung kürzester Wege herangezogen werden, was in der Praxis Bedeutung hat. Der $A*$-Algorithmus existiert in vielen Varianten (siehe dazu auch [Huc97] und [DeP85]).

Ein weiteres in der Praxis verbreitetes Verfahren zur Berechnung kürzester Wege sind genetische Algorithmen (siehe [SHF94]). Diese Methode berechnet zuerst einige Wege zwischen Start und Zielknoten um dann durch ein Verfahren (Crossover-Verfahren oder Mutationsverfahren) aus zwei Wegen einen kürzeren zu 'kreuzen' und zu lange Wege 'sterben' zu lassen. Auf diese Weise werden viele kurze Wege berechnet, die aber nicht den kürzesten Weg enthalten müssen. Je mehr Generationen man errechnet, desto genauer wird das Ergebnis.

3.3 Kürzeste Wege in modifizierten Graphen

Dieser Abschnitt ist in Zusammenarbeit mit Herrn Friedhelm Buchholz (Institut für Informatik der Universität Stuttgart) entstanden und basiert im Wesentlichen auf [BC96].

Um große Datenmengen in kurzer Zeit verwalten zu können und um reelle Gegebenheiten besser berücksichtigen zu können, betrachten wir folgende Ansätze:

1. Der Ausgangsgraph wird verändert, in der Regel verkleinert, um die Anzahl der in Betracht kommenden Knoten zu minimieren. Häufig werden hierbei Knoten und Kanten zusammengefaßt oder Kanten und Knoten gar nicht in Betracht gezogen. Die entstehenden Ergebnisse weichen in der Regel nicht oder nur minimal von der tatsächlich richtigen Lösung ab.
2. Hierarchischer Ansatz: Der Algorithmus sucht den kürzesten Weg vom Start– und Zielort zu einem zentralen, ausgedünnten zusammenhängenden Teilgraph – etwa der Autobahn – und berechnet dann nur noch den kürzesten Weg auf diesem Teilgraph. Auch hier können Fehler auftreten, die gerade bei der Berechnung von 'sehr nahe' beieinanderliegenden Knoten zu offensichtlich unbrauchbaren Ergebnissen führen.
3. Gewisse kürzeste Wege zwischen häufig frequentierten Knoten werden im voraus berechnet und abgespeichert.
4. Gewisse Knoten, die wahrscheinlich nicht im kürzesten Weg auftreten, werden nicht betrachtet.

3.3.1 Das Rathausmodell

Hier werden Teile des Graphen (z. B. Städte) zu einem Knoten (dessen Rathaus) zusammengefaßt. Damit genügt das Rathausmodell dem Ansatz 1. Sei $G = (V, E, \alpha, \omega)$ ein Graph, $\tilde{E} \subseteq E$ wegweise zusammenhängend, $\tilde{V} := V(\tilde{E}) = V_i(\tilde{E}) \cup V_r(\tilde{E})$ die Menge der zu $\tilde{E}$ gehörigen inneren Knoten und Randknoten (vgl. Def. 2.7), ferner sei

$$\begin{array}{lll} E_1 & := \{e \in E \mid \alpha(e) \in \tilde{V} \text{ und } \omega(e) \in \tilde{V}\} & \text{'Innere Straßen'} \\ E_2 & := \{e \in E \mid \alpha(e) \notin \tilde{V} \text{ und } \omega(e) \in \tilde{V}\} & \text{'Einfallstraßen'} \\ E_3 & := \{e \in E \mid \alpha(e) \in \tilde{V} \text{ und } \omega(e) \notin \tilde{V}\} & \text{'Ausfallstraßen'}, \end{array}$$

dann sind E_1, E_2 und E_3 paarweise disjunkt und $\tilde{E} \subseteq E_1$. Wir definieren einen neuen Knoten $\tilde{v}$, der den Teilgraph $(\tilde{V}, E_1, \alpha|_{E_1}, \omega|_{E_1})$ repräsentiert, und setzen

$$\begin{array}{ll} V' := (V \backslash \tilde{V}) \cup \{\tilde{v}\}, & E' := E \backslash E_1 \\ \alpha' : E' \to V', & \alpha'(e) = \begin{cases} \tilde{v} & \text{falls } e \in E_3 \\ \alpha(e) & \text{falls } e \in E' \backslash E_3 \end{cases} \\ \omega' : E' \to V', & \omega'(e) = \begin{cases} \tilde{v} & \text{falls } e \in E_2 \\ \omega(e) & \text{falls } e \in E' \backslash E_2 \end{cases} \end{array}$$

Mit dieser Definition ist $G' := (V', E', \alpha', \omega')$ ein Graph. Wenn G parallelenfrei ist, verliert G' im allgemeinen diese Eigenschaft.

Ist G ein gewichteter Graph mit Gewichtsfunktion β, so kann G' folgende Gewichtung zugewiesen werden: Wähle einen Knoten $v_r \in \tilde{V}$ (das Rathaus). Den Kanten aus E_2 und E_3 soll ein neues Gewicht zugewiesen werden: Sei $e_2 \in E_2$, so folgt aus dem wegweisen Zusammenhang von $\tilde{E}$, daß Wege von $\alpha(e_2)$ nach v_r existieren. Wir definieren $\beta'(e_2) := d(\alpha(e_2), v_r)$. Für $e_3 \in E_3$ definieren wir analog $\beta'(e_3) := d(v_r, \omega(e_3))$. Für Kanten $e \in E'\backslash(E_2 \cup E_3)$ ist $\beta'(e) := \beta(e)$. Die zu β' gehörende Entfernungsfunktion d' majorisiert d:

$$d(u, v) \leq d'(u, v) \quad \text{für alle } u, v \in V\backslash\tilde{V}$$

Der maximale Fehler pro zusammengefaßtem Knoten ist bei

$$d'(u, v) - d(u, v) \leq \max_{v' \in \tilde{V}} d(v', v_r) + \max_{v'' \in \tilde{V}} d(v_r, v'').$$

Iteriert man den Prozeß des Zusammenfassens von Teilen des Graphen, so erhöht sich der Fehler additiv. Das Verfahren ist besonders effizient, wenn $\tilde{E}$ viele innere Knoten besitzt.

3.3.2 Überbrückung von Regionen

Sei $G = (V, E, \alpha, \omega, \beta)$ ein gewichteter Graph, $\tilde{E} \subseteq E$ wegweise zusammenhängend, $\tilde{V}, E_1, E_2, E_3$ wie im vorangehenden Absatz über das Rathausmodell definiert, ferner $V_r := V_r(\tilde{E}) = V_r(E_1)$ und $V_i := V_i(\tilde{E}) = V_i(E_1)$ die Menge aller Rand– bzw. inneren Knoten von E_1. Wir berechnen alle kürzesten Wege aller Randknoten untereinander, ziehen zwischen ihnen neue Kanten und weisen diesen das Gewicht des berechneten kürzesten Weges zu. Alle inneren Knoten und alle Kanten aus $\tilde{E}$ werden weggelassen.

Für $u, v \in V_r$ gilt des wegweisen Zusammenhanges wegen, daß Wege zwischen ihnen existieren. Zu u, v definieren wir eine neue Kante $e_{u,v}$ mit $\alpha'(e_{u,v}) := u$, $\omega'(e_{u,v}) := v$ und $\beta'(e_{u,v}) := d(u, v)$. Die Menge aller so definierten Kanten heiße E_4. Sei $E' := (E\backslash E_1) \cup E_4$ und $V' := V\backslash V_i$, α', ω' und β' seien wie oben bzw. im Ausgangsgraphen definiert, dann ist $G' := (V', E', \alpha', \omega', \beta')$ ein gewichteter Graph und $d'(u, v) = d(u, v)$ für alle $u, v \in V'$.

Dieses Verfahren eignet sich besonders gut zur Überbrückung von Ortschaften. Es ist nur sinnvoll, wenn die Anzahl der Randknoten relativ klein, die Anzahl der inneren Knoten aber relativ groß ist. Die Anzahl der neu zu ziehenden Kanten liegt bei $(|V_r| * (|V_r| - 1))$ und wächst damit quadratisch mit der Anzahl der Randknoten.

Die Überbrückung von Regionen ist ein Verfahren von Ansatz 1 und 3, da gewisse kürzeste Wege vorher berechnet werden. Der große Vorteil besteht darin, daß (sofern Start– und Zielknoten im verkürzten Graph noch existieren) die Ergebnisse exakt sind und nachvollzogen werden können. Eine genaue Angabe über die Rechenzeitersparnis kann auf Grund der Allgemeinheit des Ansatzes nicht gegeben werden. Das Rathausmodell und die Methode der Überbrückung von Regionen wird in Beispiel B.2 angewandt.

3.3.3 Der n-Level Graph

Dieser Ansatz ist abgeleitet von [CRP93] - entsprechende Ansätze sind auch an anderer Stelle bereits in den achziger Jahren verwirklicht worden. Zu Grunde liegt die Idee, daß verschiedene Straßentypen (Autobahn, Bundesstraße, Landstraße, ...) bezüglich der Fahrtzeit pro km eine Art Hierarchie [Hec84] bilden und daß es lohnenswert erscheint, sich auf hierarchisch möglichst hochstehenden Straßen zu bewegen.

Definition 3.2 *Sei $G = (V, E, \alpha, \omega, \beta)$ ein gewichteter wegweise zusammenhängender Graph, $G^k = (V^k, E^k, \alpha^k, \omega^k, \beta^k)$, $k = 0, ..., n$ eine endliche Sequenz von wegweise zusammenhängenden Graphen, wobei G^k Teilgraph von G^{k-1} ist und $G^0 = G$, dann heißt G ein n-***Level Graph*** *mit den Ebenen G^k.*

Ziel des n– Level–Algorithmus ist es nur noch, vom Startknoten v^s aus den kürzesten Weg zu einem Knoten v_1^s der nächsthöheren Ebene zu finden. Von diesem Knoten aus suchen wir wieder den kürzesten Weg in die nächsthöhere Ebene, bis wir einen Knoten v_n^s aus G^n erreicht haben. Auf die gleiche Weise (mit umorientierten Kanten) suchen wir einen Weg, der von einem Knoten v_n^z aus G^n zum Zielknoten v^z führt. Da G^n wegweise zusammenhängend ist, existiert ein Weg von v_n^s nach v_n^z, und somit ist ein Weg zwischen v^s und v^z gefunden - im allgemeinen ist dies jedoch nicht der kürzeste.

In der Realität sind die verschiedenen Ebenen etwa mit Straßentypen wie Autobahn, Bundesstraße oder Landstraße zu vergleichen. Bei dieser Definition wäre die Ebene nach dem untersten Straßentyp benannt, der in ihr vorkommt. In der Verkehrsplanung werden bei n-Level Graphen häufig alle kürzesten Wege der n-ten Ebene vorher berechnet und gespeichert, so daß der so berechnete kürzeste Weg zwischen zwei Knoten bereits beim Erreichen des n-ten Levels bekannt ist.

Benötigt man eine höhere Genauigkeit für den kürzesten Weg zwischen zwei Knoten, so kann die oberste Ebene weggelassen werden. Die Ergebnisse bis zur Stufe $n-1$ können übernommen werden. Nun muß noch der kürzeste Weg zwischen v_{n-1}^s und v_{n-1}^z berechnet werden. Mit jedem Weglassen einer Ebene erhöht sich die Genauigkeit und auch der Berechnungsaufwand bei großen Graphen. Damit verwendet der n–Level–Algorithmus die Ansätze 2 und 3. Hierarchische Graphen können auch durch Separationsansätze erzeugt werden (siehe hierzu [LiT79], [Buc00] und [Schl97]).

3.3.4 Beschränkung auf Teilgraphen

Beim Berechnen kürzester Wege wollen wir nur noch einen Teilgraphen betrachten, der höchstwahrscheinlich den kürzesten Weg enthält (Ansatz 4). Dieser Teilgraph ist aber abhängig von Start- und Zielpunkt.

Sei $G = (V, E, \alpha, \omega, \beta)$ ein gewichteter Graph, der von einer Polygonzugmenge analog zur Definition 6.1 erzeugt wurde, und sei $\beta(e)$ die Länge des Polygonzuges, der e erzeugt. Gesucht ist

ein kürzester Weg zwischen einem Startknoten S und einem Zielknoten Z. Sei $d(S,Z)$ die Länge eines kürzesten Weges von S nach Z und $d_e(S,Z)$ ihr euklidischer Abstand (Luftlinie), dann gilt nach dieser Definition $d(S,Z) \geq d_e(S,Z)$. Sei $\varepsilon > 0$ fest gewählt, dann wird ein Knoten K nur noch dann betrachtet, wenn er folgender Bedingung genügt:

$$d_e(S,K) \ + \ d_e(K,Z) \ \leq \ (1+\varepsilon) \ * \ d_e(S,Z). \tag{3.6}$$

Die in Frage kommenden Knoten liegen also auf dem Rand oder innerhalb einer Ellipse mit den Brennpunkten S und Z. Ihre große Halbachse hat dann die Länge $\frac{1}{2}d_e(S,Z) \cdot (1+\varepsilon)$ und die kleine Halbachse die Länge $\frac{1}{2}\sqrt{\varepsilon^2+2\varepsilon} \cdot d_e(S,Z)$.

Für dieses Verfahren kann keine Fehlerabschätzung angegeben werden, da hierbei der Fehler beliebig groß werden kann. Betrachtet man aber reale Straßennetze, so kann in Gebieten mit vielen Straßen ohne relevante Hindernisse wie große Seen oder hohe Berge davon ausgegangen werden, daß $d(S,Z) \approx 1,25 \cdot d_e(S,Z)$ gilt [Cla99]. Wählen wir z. B. $\varepsilon = 0.4$ und gehen wir davon aus, daß $d(S,Z) \leq d_e(S,Z) * (1+\varepsilon)$ gilt, so ist dieses Verfahren sogar exakt. Sei K ein Knoten des kürzesten Weges, dann gilt

$d_e(S,K) + d_e(K,Z)$	$\leq d(S,K) + d(K,Z)$	Definition von d_e
	$= d(S,Z)$	K liegt auf dem kürzesten Weg von S nach Z
	$\leq d_e(S,Z) \cdot (1+\varepsilon)$	nach Voraussetzung

und damit genügt K der Bedingung (3.6).

3.4 Zweitkürzeste Wege in einem Graphen

Oft stellt sich nicht nur das Problem, den kürzesten Weg zwischen zwei Knoten in einem zusammenhängenden Graphen zu finden, sondern auch den zweitkürzesten.

Definition 3.3 *Es seien $G = (V, E, \alpha, \omega, \beta)$ ein positiver zusammenhängender Graph, u und v in V, ferner $W(u,v)$ die Menge aller möglichen Wege mit Startpunkt u und Endpunkt v und $w := (e_1, ..., e_n)$ ein kürzester Weg von u nach v. w' heißt* **zweitkürzester Weg** *von u nach v, wenn gilt:*

$$\delta(w') \leq \delta(\tilde{w}) \text{ für alle } \tilde{w} \in W(u,v) \backslash \{w\}.$$

Algorithmus: Der Form halber erweitern wir den kürzesten Weg w um die Kante e_{n+1}, wobei $\beta(e_{n+1}) = 0$ und $\alpha(e_{n+1}) = \omega(e_{n+1}) = v$ gilt. Die Idee des Algorithmus ist es, entlang des kürzesten Weges nach dem zweitkürzesten Weg zu suchen. Hierzu benötigen wir die kürzesten Wege von jedem Knoten des Graphen hin zum Zielknoten.

1. Schritt: Vertausche die Orientierung der Kanten (der Anfangsknoten wird zum Endknoten und umgekehrt).

2. Schritt: Man berechne die Menge aller kürzesten Wege von v zu allen anderen Knoten des Graphen. Durch die Inversion der Orientierung werden alle kürzesten Wege zu v hin berechnet. Dieser Kürzeste-Wege-Baum enthält sicher den Startknoten u (Zusammenhang) und damit den kürzesten Weg von u nach v.

3. Schritt: Stelle die ursprüngliche Orientierung der Kanten wieder her.

4. Schritt: Nimm von jedem Anfangsknoten jeder Kante e_i des kürzesten Weges (auch von Start– und Zielknoten) jede Kante $e \neq e_i$ mit $\alpha(e) = \alpha(e_i)$ und berechne

$$\min_{i=1}^{n+1} \min_{e} \{d(u, \alpha(e)) + \beta(e) + d(\omega(e), v) \mid \text{ für } e \text{ gilt } \alpha(e) = \alpha(e_i) \text{ und } e \neq e_i\}$$

Gibt es keine Kanten dieser Art, so gibt es auch keinen zweitkürzesten Weg. Die Kante von w, für die dieses Minimum gebildet wird, heiße e_m, die (in diesem Fall) den kürzesten Weg verlassende Kante (e) heiße e'. Der kürzeste Weg von $\omega(e')$ nach v heiße $\tilde{w}$. Der zweitkürzeste Weg von u nach v ist also

$$w' := (e_1, ..., e_{m-1}) + e' + \tilde{w}.$$

Der Algorithmus kann auch auf Graphen mit Abbiegeverboten angewandt werden. Dabei ist darauf zu achten, daß die Vertauschung der Orientierung der Kanten bei den Abbiegeverboten berücksichtigt wird.

Häufig gilt, daß sich der zweitkürzeste Weg vom kürzesten Weg nur dadurch unterscheidet, daß man in eine kurze Straße abbiegt, dort wendet und dann wieder zum kürzesten Weg zurückgeht. Die Tatsache, daß hier der Knoten $\alpha(e')$ zweimal durchlaufen wird, ist im Algorithmus berücksichtigt, und zwar ohne einen virtuellen Knoten (siehe Abschnitte 4.5 und 4.6) erschaffen zu müssen. Möchte man diesen Umstand ausschließen, das heißt im zweitkürzesten Weg soll mindestens eine Kante des kürzesten Weges fehlen, so muß der Algorithmus modifiziert werden (siehe z. B. [Mack96]).

Beweis der Gültigkeit des Algorithmus: Angenommen, $w'' = (f_1, ..., f_N)$ ist ein von w verschiedener Weg von u nach v. Wir müssen $\delta(w'') \geq \delta(w')$ zeigen. Da $w'' \neq w$, gibt es einen Index i mit $f_1 = e_1, ..., f_{i-1} = e_{i-1}$ und $f_i \neq e_i$. Es folgt

$$\begin{aligned}
\delta(w'') &= \delta(f_1, ..., f_{i-1}) + \delta(f_i) + \delta(f_{i+1}, ..., f_N) && \text{Definition von } \delta \\
&= \delta(e_1, ..., e_{i-1}) + \beta(f_i) + \delta(f_{i+1}, ..., f_N) && \text{Definition von } i \\
&\geq d(u, \alpha(e_i)) + \beta(f_i) + d(\omega(f_i), v) && \text{Definition von } d \\
&\geq \delta(w') && \text{Minimumbildung}
\end{aligned}$$

■

Die Berechnung der k-kürzesten Wege befindet sich zum Beispiel in [Kno69], in [Huc96] oder in [AMMP90].

Kapitel 4

Berechnung kürzester Wege in Graphen mit Abbiegeverboten

Bei der Berechnung kürzester Wege sollen Abbiegeverbote berücksichtigt werden können. Dabei stellt sich heraus, daß das einfache 'Verbieten' des Abbiegens ohne weitere Modifikation des Graphen oder des Algorithmus nicht ausreicht. Kürzeste Wege in Graphen mit Abbiegeverboten können einen oder mehrere Knoten doppelt enthalten. Algorithmen, die auf einer Baumstruktur aller kürzesten Wege aufbauen, sind dieser Problematik aber nicht gewachsen. Nur durch Algorithmus– oder Graphveränderung können wir diesem Problem begegnen. Die Berechnung kürzester Wege erfolgt häufig auf Basis des Dijkstra-Algorithmus. Daher beschränken wir uns auf die Betrachtung des SSSP.

In bisher realisierten Kraftfahrzeug-Navigationssystemen sind optimale Routenplanungsverfahren auf der Basis von Wegeverboten nicht realisiert.

4.1 Die naive Methode

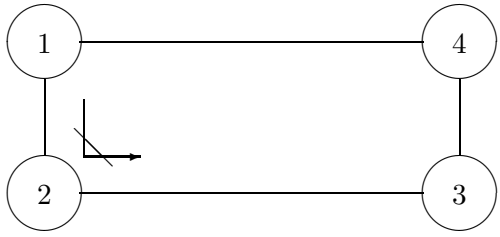
Sei $G = (V, E, \alpha, \omega)$ ein Graph mit den Abbiegeverboten T. Es ist nun naheliegend, während der Durchführung des Dijkstra-Algorithmus einen Knoten nur dann in die Nachbarschaftsliste bzw. in den Minimalbaum aufzunehmen, wenn er nicht über eine verbotene Kantenkombination erreicht werden würde.

Beispiel 4.1 Gegeben ist folgender Graph $G = (V, E, \alpha, \omega, \beta)$ mit Abbiegeverboten T:

$$
\begin{aligned}
V &= \{v_1, v_2, v_3, v_4\}, \\
E &= \{e_{12}, e_{14}, e_{21}, e_{23}, e_{32}, e_{34}, e_{41}, e_{43}\}, \\
T &= \{(e_{12}, e_{23})\},
\end{aligned}
$$

$$\beta(e) = \begin{cases} 1 & \text{falls } e \in \{e_{12}, e_{21}, e_{23}, e_{32}, e_{34}, e_{43}\} \\ 2 & \text{falls } e \in \{e_{14}, e_{41}\} \end{cases}$$

und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$ und $\omega(e_{ij}) = v_j$.



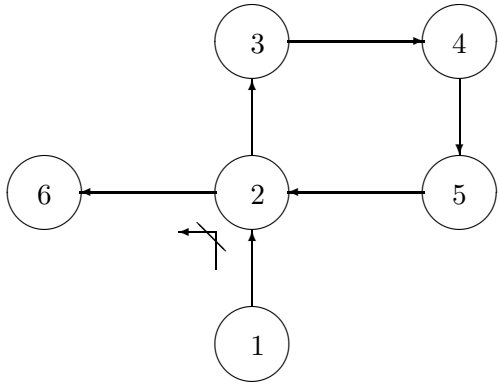
Der kürzeste Weg von v_1 nach v_3 müßte eigentlich über den Knoten v_2 gehen. Wegen des Abbiegeverbotes (e_{12}, e_{23}) verläuft dieser aber über v_4 und ist länger.

Problem: Das Bilden eines Minimalbaumes läßt es nicht zu, daß ein Knoten in ihm mehrmals vorkommt. Gewisse Konstellationen erzwingen sogar Wege, die mehrere Male ein- und denselben Knoten beinhalten. Der Algorithmus von Dijkstra sowie die Tripelalgorithmen sind der Problemstellung in dieser Form nicht gewachsen wie folgendes Beispiel zeigt:

Beispiel 4.2 Gegeben ist ein Graph $G = (V, E, \alpha, \omega, \beta)$ mit Abbiegeverboten T, mit

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\},$$
$$E = \{e_{12}, e_{23}, e_{34}, e_{45}, e_{52}, e_{26}\},$$
$$T = \{(e_{12}, e_{26})\},$$

den Kantengewichten $\beta(e) = 1 \quad$ für alle $e \in E$ und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$, $\omega(e_{ij}) = v_j$.



Der einzige und damit auch der kürzeste Weg von v_1 nach v_6 muß des Abbiegeverbotes wegen zweimal über den Knoten v_2 führen. Damit muß der Knoten v_2 zweimal in den Kürzeste-Wege-Baum aufgenommen werden, was der Dijkstra-Algorithmus in dieser Form nicht vorsieht.

Fazit: Um kürzeste Wege in Graphen mit Abbiegeverboten zu berechnen, müssen wir also entweder den Graphen oder den Algorithmus verändern. Beide Lösungen sind möglich. Wir unterscheiden in algorithmusverändernde *dynamische* Verfahren und in graphverändernde *statische* Verfahren. Bei der naiven Methode handelt es sich also um ein dynamisches Verfahren.

Bei statischen Verfahren darf ein kürzester-Wege-Algorithmus beliebig gewählt werden. Die Abbiegeverbote werden in einer Preprocessingphase in den Graph eingearbeitet. Dadurch werden statische Verfahren unflexibel gegenüber Veränderungen.

Der Nachteil der dynamischen Verfahren liegt in deren Algorithmusabhängigkeit und darin, daß die Abbiegeverbote weiterhin beachtet werden müssen. Dies können sehr viele sein ($0.5 \cdot |V|$ in Großstädten). Das Berechnen der kürzesten Wege wird dadurch verlangsamt. Der Vorteil der dynamischen Verfahren liegt in ihrer Flexibilität gegenüber Veränderungen, weil keine eventuell aufwendige Preprocessingphase erforderlich ist.

Der Nachteil der Unflexibilität wertet die statischen Verfahren gegenüber den dynamischen Verfahren ab. In Straßenverkehrsgraphen ist zum Beispiel bei temporären oder individuellen Abbiegeverboten Flexibilität geboten. Trotzdem werden wir uns verstärkt mit statischen Verfahren beschäftigen, da wir eines dieser Verfahren auf Wegeverbote erweitern wollen.

4.2 Methode der Kantenaufnahme

Dieses Standardverfahren befindet sich in [SB77]. Es ist eine der ersten Methoden zur Berechnung kürzester Wege in Straßennetzen mit Abbiegeverboten, die publiziert wurde. Es handelt sich hierbei um ein dynamisches Verfahren, welches den Algorithmus verändert, den Graphen aber gleich läßt.

Im Algorithmus werden nicht mehr die Knoten als Ziele betrachtet, sondern die Kanten, wobei eine Kante erreicht ist, wenn sie durchlaufen ist; die Kantenlänge wird zum Abstand Kante–Startknoten dazugezählt. Die Abbiegeverbote müssen natürlich weiterhin beachtet werden. Zur Berechnung kürzester Wege ist jeder entsprechende Algorithmus (auch Tripelalgorithmen) denkbar. Durch die Methode der Kantenaufnahme bleibt die Baumstruktur des Kürzeste-Wege-Baumes erhalten. Als Beispiel verwenden wir den Dijkstra-Algorithmus.

Beispiel 4.3 Gegeben sei der Graph G aus Beispiel 4.2. Wir wenden den Dijkstra-Algorithmus auf dessen Kanten an. Als Startknoten soll der Knoten v_1 gewählt werden, da von ihm aus das Abbiegeverbot eine Bedeutung hat. $B(i)$ sei der Kürzeste-Wege-Baum, $N(i)$ die Nachbarschaftsliste und $R(i)$ die weder in $B(i)$ noch in $N(i)$ aufgenommenen Kanten nach dem Schritt i.

1.Schritt:
$B(1) = \{e_{12}\}$. Dies ist die einzige Kante, deren Anfangsknoten v_1 ist. Gäbe es mehrere, so würden wir die Kante mit dem kleinsten Gewicht wählen. Alle Kanten, deren Anfangspunkt v_2 ist und in deren Richtung von e_{12} aus kein Abbiegeverbot besteht, werden in $N(1)$ aufgenommen: $N(1) = \{e_{23}\}$ und damit folgt für $R(1) = \{e_{26}, e_{34}, e_{45}, e_{52}\}$. e_{26} wird erst im Schritt 5 in die Nachbarschaftsliste aufgenommen.

2.Schritt:
Da sich nur ein Element in der Nachbarschaftsliste befindet (e_{23}), wird dieses in $B(2)$ aufgenommen. Für die anderen Mengen ergibt sich: $N(2) = \{e_{34}\}$ und $R(2) = \{e_{26}, e_{45}, e_{52}\}$.

Die weiteren Schritte laufen analog ab. Nach 6 Schritten ergibt sich also folgender kürzeste-Wege-Baum, wobei hier ausnahmsweise die Kanten als Knoten dargestellt werden:

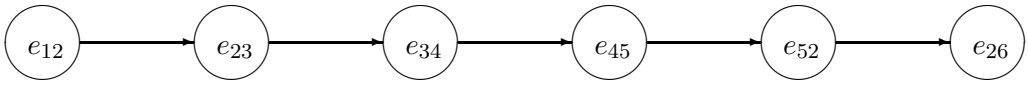


Abbildung 4.1 *Kürzeste–Wege–Baum des Graphen aus Beispiel 4.2, mit der Methode der Kantenaufnahme berechnet.*

Bewertung dieser Methode und Zeitabschätzung:

Der Kürzeste-Wege-Baum besitzt genau $|E|$ Knoten, was sich für $|V| << |E|$ wesentlich auf die Zeitkomplexität des Algorithmus auswirken kann. Beispiel B.3 zeigt, was passiert, wenn man diesen Graphen als ungerichteten Graphen betrachtet, und welche unnötigen Kanten mit aufgenommen werden. Die Anzahl der Schritte des durch Kantenaufnahme modifizierten Dijkstra-Algorithmus (bei dem die Nachbarschaftsliste in einem Heap organisiert ist) liegt deshalb bei $O(|E| \cdot \log(|E|))$.

Bei der Methode der Kantenaufnahme wird jeder Knoten (als Endknoten einer Kante) so oft in den Kürzeste-Wege-Baum aufgenommen, wie Kanten auf ihn weisen, selbst wenn sich an diesem kein Abbiegeverbot befindet. In Straßenverkehrsgraphen wird der Aufwand etwa vervierfacht, da auf jede Kreuzung in der Regel vier Straßen weisen [SB77].

4.3 Methode der mehrfachen Knotenaufnahme

Wie in Kapitel 4.1 erläutert, ist es bei der Berechnung kürzester Wege in einem Graph mit Abbiegeverboten wünschenswert, gewisse Knoten mehrfach in den Kürzeste-Wege-Baum auf-

zunehmen. Damit verliert dieser aber seine Baumstruktur. Um kürzeste Wege zu berechnen, modifizieren wir jetzt den Dijkstra-Algorithmus durch mehrfache Knotenaufnahme.

Diese Methode ist wie die Methode der Kantenaufnahme aus Abschnitt 4.2 ein dynamisches Verfahren, welches den Algorithmus und nicht den Graphen verändert. Dabei konzentrieren wir uns darauf, daß nicht mehr Elemente als unbedingt notwendig in den Kürzeste-Wege-Baum aufgenommen werden.

Wird ein Knoten in den Kürzeste-Wege-Baum über eine Kante e_0 aufgenommen, von welcher aus ein Abbiegeverbot existiert ($e_0 \in E_0$), so darf dieser Knoten noch ein weiteres Mal in den Kürzeste-Wege-Baum aufgenommen werden. Existiert kein derartiges Abbiegeverbot, so darf der Knoten nicht mehr aufgenommen werden.

Die Abbiegeverbote müssen weiterhin berücksichtigt werden, das heißt, für $(e_0, e_1) \in T$ darf $\omega(e_1)$ nicht in den Kürzeste-Wege-Baum aufgenommen werden, wenn dieser Knoten über einen Weg erreicht wird, der die Kantenfolge e_0, e_1 enthält. $\omega(e_0)$ darf aber kein weiteres Mal von der Kante e_0 aus aufgenommen werden, da sonst der Algorithmus in Endlosschleifen gerät.

Der Einfachheit halber möchte ich den Namen Kürzeste-Wege-Baum beibehalten und zur Unterscheidung die mehrfach aufgenommenen Knoten doppelt indizieren: v_i^j bedeutet, v_i wurde von der Kante e_j aus in den Baum aufgenommen.

Beispiel 4.4 Zur Veranschaulichung betrachten wir wieder den Graphen G aus Beispiel 4.2. Zur Berechnung der kürzesten Wege vom Startknoten v_1 aus verwenden wir wiederum den Dijkstra-Algorithmus. Nach dem Schritt i sei $B(i)$ der Kürzeste-Wege-Baum, $N(i)$ die Nachbarschaftsliste und $R(i)$ die Knoten, die nach $N(i)$ aufgenommen werden sollen und sich noch nicht in $B(i)$ befinden. Durch die Konstruktion des Algorithmus kann es Knoten geben, die sich sowohl in $B(i)$ als auch in $R(i)$ befinden.

1.Schritt: $B(1) := \{v_1\}$; damit gilt $N(1) = \{v_2\}$, $R(1) = \{v_2, v_3, v_4, v_5, v_6\}$.

2.Schritt: Der einzige Knoten, der in $B(2)$ aufgenommen werden kann, ist v_2. Da dies über die Kante e_{12} geschieht, aus der ein Abbiegeverbot besteht, kann v_6 nicht in $N(2)$ aufgenommen werden und v_2 muß noch ein weiteres Mal in $B(i)$ aufgenommen werden. Deshalb bleibt der Knoten v_2 in $R(2)$. $B(2) := \{v_1, v_2^{12}\}$, $N(2) = \{v_3\}$, $R(2) = \{v_2, v_4, v_5, v_6\}$.

3.Schritt: Wie im zweiten Schritt gibt es nur einen Knoten, der in $B(3)$ aufgenommen werden kann. Dies ist v_3. Damit gilt $B(3) := \{v_1, v_2^{12}, v_3\}$, $N(3) = \{v_4\}$, $R(3) = \{v_2, v_5, v_6\}$.

4.Schritt: $B(4) := \{v_1, v_2^{12}, v_3, v_4\}$, $N(4) = \{v_5\}$, $R(4) = \{v_2, v_6\}$.

5.Schritt: $B(5) := \{v_1, v_2^{12}, v_3, v_4, v_5\}$, $N(5) = \{v_2\}$, $R(5) = \{v_6\}$.

6.Schritt: Jetzt wird v_2 ein zweites Mal in den Kürzeste-Wege-Baum aufgenommen und zwar über die Kante e_{52}. Da aus dieser Richtung kein Abbiegeverbot existiert, wird v_2 kein drittes Mal in $B(i)$ aufgenommen werden.

7.Schritt: $B(7) := \{v_1, v_2^{12}, v_3, v_4, v_5, v_2^{52}, v_6\}$, $N(6) = R(6) = \emptyset$.

Der Kürzeste-Wege-Baum vom Knoten v_1 aus besitzt also folgende Struktur:

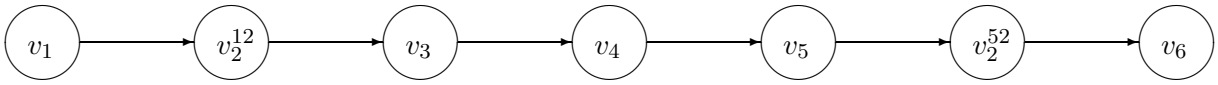


Abbildung 4.2 *Kürzester-Wege-Baum des Graphen aus Beispiel 4.2 mit der Methode der mehrfachen Knotenaufnahme berechnet.*

Bewertung dieser Methode und Zeitabschätzung:

Einer der wesentlichen Vorteile dieser Methode ist, daß sie völlig ohne Preprocessingphase auskommt. Ein Nachteil dieser Methode ist der Verwaltungsaufwand, der betrieben werden muß, um zu erkennen, welcher Knoten von welcher Kante aus bereits in den Kürzeste-Wege-Baum aufgenommen wurde.

Die Anzahl der in den Kürzeste-Wege-Baum aufgenommenen Elemente ist kleiner oder gleich $|V| + |T|$. Die Anzahl der Schritte des Dijkstra-Algorithmus liegt also bei

$$O((|V| + |T|) \cdot \log(|V| + |T|) + |E|).$$

Bemerkung: Es ist auch möglich, den in Abschnitt A.3 beschriebenen Tripelalgorithmus so zu verändern, daß kürzeste Wege in Graphen mit Abbiegeverboten berechnet werden können: Der Tripelalgorithmus wird so oft auf den Graph mit Abbiegeverboten angewandt, bis sich, im Vergleich zur Kürzeste-Wege-Matrix vom Schritt davor, nichts mehr verändert. Die Anzahl der Durchläufe steigt mit der Anzahl der Abbiegeverbote und liegt bei $O(|T|)$. Der Zeitaufwand für den Algorithmus liegt also bei

$$O(|V|^3 \cdot |T|).$$

4.4 Methode des Ziehens neuer Kanten

Wir wollen uns nun einem statischen Verfahren zur Lösung des STSP (Berechnung des kürzesten Weges zwischen einem fest vorgegebenen Startknoten und einem fest vorgegebenen Zielknoten) in Graphen mit Abbiegeverboten zuwenden, bei welchem (als eine Art Preprocessing) der Graph verändert wird, nicht aber der Algorithmus. Sei also $G^0 = (V, E^0, \alpha^0, \omega^0, \beta^0)$ ein Graph mit Abbiegeverboten T^0 und $u, v \in V$. Wir numerieren die Knoten von G^0 in irgendeiner Weise so, daß $V = \{v_1 = u, v_2, ..., v_N = v\}$. An dieser Stelle geht die feste Vorgabe von Start und Ziel ein. Um Wege der Länge 1 ausschließen zu können, erweitern wir den Graphen noch um einen neuen Zielknoten v' und um eine verbindende Kante e'_0 mit $\alpha(e'_0) := v$, $\omega(e'_0) := v'$ und $\beta(e'_0) := 0$ (in

den Beispielen wird aus Übersichtsgründen auf diese Erweiterung verzichtet). Jetzt konstruieren wir der Reihe nach für $k = 1, ..., N$ Graphen $G^k = (V, E^k, \alpha^k, \omega^k, \beta^k)$ mit Abbiegeverboten T^k und den folgenden Eigenschaften:

(1) G^k hat keine Abbiegeverbote an den Knoten $v_1, ..., v_k$; insbesondere ist G^N ein Graph ohne Abbiegeverbote.

(2) Ist $W^k(u, v)$ die Menge aller nicht verbotenen Wege von u nach v im Graph G^k, dann gibt es eine längenerhaltende Bijektion $W^{k-1}(u, v) \to W^k(u, v)$.

Konstruktion von G^k aus G^{k-1}: Es sei E die Menge aller Kanten e, die im Knoten v_k enden und für die ein Abbiegeverbot (e, f) existiert, d.h.

$$E = \{e \in E^{k-1} \mid \omega(e) = v_k \text{ und es gibt ein } (e, f) \in T^{k-1}\}.$$

Sei $E = \{e_1, ..., e_n\}$. Wir setzen $\tilde{G}^0 := G^{k-1}$ und konstruieren Graphen $\tilde{G}^1, \tilde{G}^2, ..., \tilde{G}^n =: G^k$ mit den Eigenschaften

(3) $\tilde{G}^j$ hat keine Abbiegeverbote an den Knoten $v_1, ..., v_{k-1}$ und für $i = 1, ..., j$ keine Abbiegeverbote $(e_i, *)$ am Knoten v_k; insbesondere hat $\tilde{G}^n$ keine Abbiegeverbote an den Knoten $v_1, ..., v_k$.

(4) Ist $\tilde{W}^j(u, v)$ die Menge aller Wege von u nach v im Graph $\tilde{G}^j$, dann gibt es eine längenerhaltende Bijektion $\tilde{W}^{j-1}(u, v) \to \tilde{W}^j(u, v)$.

Beachte, daß aus (3) und (4) sofort (1) und (2) folgen.

Konstruktion von $\tilde{G}^j$ aus $\tilde{G}^{j-1}$: Es sei $\tilde{G}^{j-1} = (V, \tilde{E}^{j-1}, \tilde{\alpha}^{j-1}, \tilde{\omega}^{j-1}, \tilde{\beta}^{j-1})$ mit Abbiegeverboten $\tilde{T}^{j-1}$. Es sei N^{e_j} die Menge der 'bezüglich e_j zulässigen' in v_k beginnenden Kanten, d.h.

$$N^{e_j} := \{f \in \tilde{E}^{j-1} \mid \tilde{\alpha}^{j-1}(f) = \tilde{\omega}^{j-1}(e_j) \text{ und } (e_j, f) \notin \tilde{T}^{j-1}\}.$$

Zu jedem $f \in N^{e_j}$ ziehen wir eine neue Kante $\tilde{f}$ von $\alpha(e_j)$ nach $\omega(f)$; sei $\tilde{F}_j := \{\tilde{f} \mid f \in F_j\}$. Wir definieren $\tilde{E}^j, \tilde{\alpha}^j, \tilde{\omega}^j$ und $\tilde{\beta}^j$ durch $\tilde{E}^j := \tilde{E}^{j-1} \cup \tilde{N}^{e_j} \backslash \{e_j\}$, sowie

$$\tilde{\alpha}^j : \tilde{E}^j \to V, \qquad \tilde{\alpha}^j(f) := \begin{cases} \tilde{\alpha}^{j-1}(f) & \text{falls } f \in \tilde{E}^{j-1} \backslash \{e_j\} \\ \tilde{\alpha}^{j-1}(e_j) & \text{falls } f \in \tilde{F}_j \end{cases} \tag{4.1}$$

$$\tilde{\omega}^j : \tilde{E}^j \to V, \qquad \tilde{\omega}^j(f) := \begin{cases} \tilde{\omega}^{j-1}(f) & \text{falls } f \in \tilde{E}^{j-1} \backslash \{e_j\} \\ \tilde{\omega}^{j-1}(g) & \text{falls } f = \tilde{g} \in \tilde{F}_j \end{cases} \tag{4.2}$$

$$\tilde{\beta}^j : \tilde{E}^j \to \mathbb{R}, \qquad \tilde{\beta}^j(f) := \begin{cases} \tilde{\beta}^{j-1}(f) & \text{falls } f \in \tilde{E}^{j-1} \backslash \{e_j\} \\ \tilde{\beta}^{j-1}(e_j) + \tilde{\beta}^{j-1}(g) & \text{falls } f = \tilde{g} \in \tilde{F}_j \end{cases} \tag{4.3}$$

Die neuen Abbiegeverbote in $\tilde{G}^j$ sind

$$\tilde{T}_1^j := \{(a, b) \in \tilde{T}^{j-1} \mid a \neq e_j\}, \qquad \tilde{T}_2^j := \{(\tilde{f}, g) \mid (f, g) \in \tilde{T}^{j-1}\}, \qquad \tilde{T}^j := \tilde{T}_1^j \cup \tilde{T}_2^j.$$

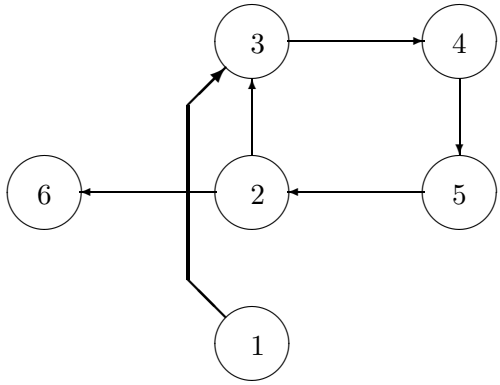
Der Graph wurde so verändert, daß die Kante e_j weggelassen wird – deshalb ist (3) trivial – und dafür zu jeder Kante $f \in F_j$ eine neue Kante $\tilde{f}$ von $\alpha(e_j)$ nach $\omega(f)$ erschaffen wurde. Die

Abbiegeverbote in $\tilde{T}_2^j$ müssen hinzugenommen werden, da sonst eventuell vorhandene Abbiegeverbote an zu v_k benachbarten Knoten umgangen werden. Warum gilt (4)? Wir betrachten drei Fälle: Falls der Weg $w \in \tilde{W}^{j-1}(u,v)$ die Kante e_j nicht enthält, dann wird w unverändert in $\tilde{W}^j(u,v)$ erscheinen. Falls $w \in \tilde{W}^{j-1}(u,v)$ die Kante e_j enthält, enthält er automatisch die Sequenz $.., e_j, f, ..$ für ein $f \in N^{e_j}$. Diese Sequenz wird längenerhaltend durch $\tilde{f}$ ersetzt, und w muß in dieser Form in $\tilde{W}^j(u,v)$ auftauchen.

Beispiel 4.5 Als Beispiel betrachten wir wieder den Graphen G aus Beispiel 4.2. Wir wählen als Startknoten den Knoten v_1 und als Zielknoten den Knoten v_6. Da nur ein Abbiegeverbot existiert und dieses sich am Knoten v_2 auswirkt, ist nur der Schritt von G^1 nach G^2 interessant. Für die anderen Schritte gilt $G^{i-1} = G^i$. Wir konstruieren $\tilde{G}^1$: Für die Menge E gilt $E = \{e_{12}\}$. Die Menge $N^{e_{12}}$ ist die Menge der Kanten, in die von der Kante e_{12} aus abgebogen werden darf, also gilt $N^{e_{12}} = \{e_{23}\}$. Die Kante e_{23} wird dupliziert und ihr Duplikat sei $\tilde{e}_{23}$. In $\tilde{G}^1$ gilt für $\tilde{e}_{23}$:

$$\tilde{\alpha}^1(\tilde{e}_{23}) = \alpha(e_{12}) = v_1 \quad \tilde{\omega}^1(\tilde{e}_{23}) = \omega(e_{23}) = v_3 \quad \tilde{\beta}^1(\tilde{e}_{23}) = \beta(e_{12}) + \beta(e_{23}) = 2$$

Es ergibt sich folgender neuer Graph, wobei die neue Kante fett abgebildet ist:



Der kürzeste und auch einzige Weg, um von v_1 nach v_6 zu gelangen, ist also ($\tilde{e}_{23}$, e_{34}, e_{45}, e_{52}, e_{26}).

Diese Methode kann sehr schnell sehr unübersichtlich werden, ungeachtet der Tatsache, daß zu jeder neu erschaffenen Kante ein ganzer Weg assoziiert werden muß. Da hier ein Graph erschaffen wurde, in dem die Abbiegeverbote nicht mehr benötigt werden, kann jeder beliebige Algorithmus zur Berechnung kürzester Wege verwendet werden. Ein wesentlicher Nachteil dieses Verfahrens ist, daß es nur das STSP löst.

4.5 Methode des verbotsorientierten Knotensplitting

Gegeben sei ein Graph $G = (V, E, \alpha, \omega)$ mit Abbiegeverboten T. Wir haben bei der Betrachtung der Methode der mehrfachen Knotenaufnahme erkannt, daß bei der Berechnung kürzester Wege gewisse Knoten mehrfach in den Kürzeste-Wege-Baum aufgenommen werden müssen. Dies sind Knoten, die von einer Kante aus erreicht werden, aus welcher ein Verbot existiert.

Sei nun $(e_1, e_2) \in T$ ein Abbiegeverbot und $v = \omega(e_1) = \alpha(e_2)$, so wird der Knoten v gesplittet und der Kante e_1 ein neuer Endpunkt, der neu erschaffene Knoten, zugewiesen. Die Kanten, die in v beginnen, und in deren Richtung der Weg von e_1 aus nicht verboten ist, werden ebenfalls gesplittet und erhalten als Anfangspunkt diesen neuen Knoten. Der neue Knoten kann also nur in erlaubten Richtungen verlassen werden und v kann über e_1 nicht mehr erreicht werden. Das Abbiegeverbot wird überflüssig. Damit handelt es sich hierbei um ein statisches Verfahren, welches wir im nächsten Kapitel auf Wegeverbote erweitern wollen.

Der Algorithmus des Computerprogramms erschafft zuerst alle virtuellen Knoten und weist sie den Kanten, aus deren Richtung ein Abbiegeverbot existiert, als Endpunkte zu. Im zweiten Schritt werden die Kanten gesplittet und diesen dann Endpunkte zugewiesen. Der Vorteil dieses Verfahrens besteht darin, daß zum Zeitpunkt des Kantensplittings bereits alle virtuellen Knoten erschaffen worden sind, und den Kanten gleich die richtigen Endknoten zugewiesen werden können. Um das Verfahren aber mittels vollständiger Induktion beweisen zu können, definieren wir gleich nach Erschaffen eines virtuellen Knoten dessen zugehörige Kanten und müssen damit rechnen, daß diesen im Verlauf des Algorithmus neue Endknoten zugewiesen werden.

4.5.1 Schrittweises Knoten/Kantensplitting

Wir definieren eine Äquivalenzrelation auf der Menge der Abbiegeverbote: Seien $t_1, t_2 \in T$ mit $t_i := (e_1^{t_i}, e_2^{t_i})$ $i = 1, 2$, dann gilt

$$t_1 \sim t_2 :\Leftrightarrow e_1^{t_1} = e_1^{t_2} \tag{4.4}$$

In jedem Schritt wollen wir gleich eine ganze Äquivalenzklasse von Abbiegeverboten betrachten.

Sei

$$E_0 := \{e \in E \mid \exists (e, e_2) \in T \text{ mit } e_2 \text{ beliebig } \in E\} \tag{4.5}$$

die Menge der Kanten, die durch die Projektion der Abbiegeverbote auf deren erste Komponente erzeugt wird. Jedes Element e_0 aus E_0 steht repräsentativ für eine der oben definierten Klassen. Jeder Endknoten einer Kante e_0 muß gesplittet werden. An den gesplitteten Knoten werden nur noch die Kanten angehängt, in die ein Weg von e_0 aus führen darf. Ziel ist es, dieses Knotensplitting schrittweise vorzunehmen, um dann zu zeigen, daß das Ergebnis unabhängig von der Reihenfolge der Kanten aus E_0 ist.

1. Schritt

Sei f_1 beliebig $\in E_0$. Definiere $N^{f_1} := \{e \in E \mid \alpha(e) = \omega(f_1) \text{ und } (f_1, e) \notin T\}$, einen neuen Knoten $v^{f_1} \notin V$ und neue Kanten $\tilde{E}^{f_1}$ mit $\exists$ eine Bijektion $h^{f_1} : \tilde{E}^{f_1} \to N^{f_1}$ und $\tilde{E}^{f_1} \cap E = \emptyset$. $T'^{f_1} := \{(f_1, e) \in T\}$.

Falls $E_0 \cap N^{f_1} \neq \emptyset$, so definieren wir

$$T^1 := \{(e_1, e_2) \in T \mid e_1 \in E_0 \cap N^{f_1}\}.$$

Zu jedem Abbiegeverbot $(e_1, e_2) \in T^1$ definieren wir ein weiteres Abbiegeverbot $({h^{f_1}}^{-1}(e_1), e_2)$. Die Menge aller so entstandenen Abbiegeverbote heiße $\tilde{T}^1$.

Falls $E_0 \cap N^{f_1} = \emptyset$ so sei $\tilde{T}^1 = \emptyset$.

Definition von G^{f_1}:

$$\begin{aligned} V^{f_1} &:= V \cup \{v^{f_1}\} \\ E^{f_1} &:= E \cup \tilde{E}^{f_1} \\ T^{f_1} &:= \tilde{T}^1 \cup (T \backslash T'^{f_1}) \end{aligned}$$

Die Abbildungen

$$\iota_v^{f_1} : \quad V \to V^{f_1} \qquad \iota_v^{f_1}(v) := v \in V^{f_1}$$

und

$$\iota_e^{f_1} : \quad E \to E^{f_1} \qquad \iota_e^{f_1}(e) := e \in E^{f_1}$$

heißen kanonische Injektionen von G nach G^{f_1}.
Die Abbildungen $\iota_v^{f_1}$ und $\iota_e^{f_1}$ sind injektiv, aber nicht unbedingt surjektiv, da v^{f_1} bzw. die Elemente aus $\tilde{E}^{f_1}$ keine Urbilder besitzen.

Die Abbildungen $\Pi_V^{f_1} : V^{f_1} \to V$:

$$\Pi_V^{f_1}(v) := \begin{cases} v & \text{für alle } v \in V \\ \omega(f_1) & \text{falls } v = v^{f_1} \end{cases}$$

und $\Pi_E^{f_1} : E^{f_1} \to E$:

$$\Pi_E^{f_1}(e) := \begin{cases} e & \text{für alle } e \in E \\ h^{f_1}(e) & \text{falls } e \in \tilde{E}^{f_1} \end{cases}$$

heißen kanonische Projektionen von G^{f_1} nach G.Wenn durch das Argument klar ist, ob $\Pi_V^{f_1}$ oder $\Pi_E^{f_1}$ gemeint ist, so schreiben wir stattdessen auch $\Pi^{f_1}(v)$ bzw. $\Pi^{f_1}(e)$.

Definition von $\omega^{f_1} : E^{f_1} \to V^{f_1}$

$$\omega^{f_1}(e) := \begin{cases} \omega(e) & \text{für alle } e \in E \backslash \{f_1\} \\ v^{f_1} & \text{falls } e = f_1 \\ \omega(h^{f_1}(e)) & \text{falls } e \in \tilde{E}^{f_1}. \end{cases}$$

Wegen $\omega(f_1) = v^{f_1}$ ist ω^{f_1} keine Fortsetzung von ω.

Definition von $\alpha^{f_1} : E^{f_1} \to V^{f_1}$

$$\alpha^{f_1}(e) := \begin{cases} \alpha(e) & \text{für alle } e \in E \\ v^{f_1} & \text{falls } e \in \tilde{E}^{f_1}. \end{cases}$$

Definition von $\beta^{f_1} : E^{f_1} \to \mathbb{R}_0^+$

$$\beta^{f_1}(e) := \begin{cases} \beta(e) & \text{für alle } e \in E \\ \beta(h^{f_1}(e)) & \text{falls } e \in \tilde{E}^{f_1}. \end{cases}$$

Bei den Definitionen von α^{f_1} und β^{f_1} liegen echte Fortsetzungen vor.

Trivialerweise gilt, daß

$$\beta^{f_1}(e) = \beta(\Pi_E^{f_1}(e)) \text{ für alle } E \in E^{f_1}.$$

Sei

$$G^{f_1} := (V^{f_1}, E^{f_1}, \alpha^{f_1}, \omega^{f_1}, \beta^{f_1}, T^{f_1}),$$

dann gilt: G^{f_1} ist ein Graph.

Weil ω^{f_1} keine Fortsetzung von ω ist, ist auch G kein Teilgraph von G^{f_1}.

i. Schritt

Sei $f_i \in E_0$ mit $f_i \notin \{f_1, f_2, ..., f_{i-1}\}$. Wir definieren

$$N^{f_i} := \{e \in E^{f_{i-1}} \mid \alpha^{f_{i-1}}(e) = \omega^{f_{i-1}}(f_i) \text{ und } (f_i, e) \notin T^{f_{i-1}}\}, \tag{4.6}$$

einen neuen Knoten $v^{f_i} \notin V^{f_{i-1}}$ und neue Kanten $\tilde{E}^{f_i}$, mit $\tilde{E}^{f_i} \cap E^{f_{i-1}} = \emptyset$. Zusätzlich soll eine Bijektion $h^{f_i} : \tilde{E}^{f_i} \to N^{f_i}$ existieren. $T'^{f_i} := \{(e_1, e_2) \in T^{f_{i-1}} \mid e_1 = f_i\}$.

Falls $E_0 \cap N^{f_i} \neq \emptyset$, so definieren wir analog zum ersten Schritt

$$T^i := \{(e_1, e_2) \in T^{f_{i-1}} \mid e_1 \in E_0 \cap N^{f_i}\}.$$

Zu jedem Abbiegeverbot $(e_1, e_2) \in T^i$ definieren wir ein weiteres Abbiegeverbot $({h^{f_i}}^{-1}(e_1), e_2)$. Die Menge aller so entstandenen Abbiegeverbote heiße $\tilde{T}^i$.

Falls $E_0 \cap N^{f_i} = \emptyset$, so sei $\tilde{T}^i = \emptyset$.

Definition von G^{f_i}:

$$\begin{aligned} V^{f_i} &:= V^{f_{i-1}} \cup \{v^{f_i}\} \\ E^{f_i} &:= E^{f_{i-1}} \cup \tilde{E}^{f_i} \\ T^{f_i} &:= \tilde{T}^i \cup (T^{f_{i-1}} \backslash T'^{f_i}) \end{aligned}$$

Definition der kanonischen Projektionen und Injektionen:

$\Pi_V^{f_i} : V^{f_i} \to V^{f_{i-1}}$:

$$\Pi_V^{f_i}(v) := \begin{cases} v & \text{für alle } v \in V^{f_{i-1}} \\ \omega(f_i) & \text{falls } v = v^{f_i} \end{cases}$$

$\Pi_E^{f_i} : E^{f_i} \to E^{f_{i-1}}$:

$$\Pi_E^{f_i}(e) := \begin{cases} e & \text{für alle } e \in E^{f_{i-1}} \\ h^{f_i}(e) & \text{falls } e \in \tilde{E}^{f_i} \end{cases}$$

$\iota_v^{f_i} : V^{f_{i-1}} \to V^{f_i}$:

$$\iota_v^{f_i}(v) := v \in V^{f_i}$$

$\iota_e^{f_i} : E^{f_{i-1}} \to E^{f_i}$:

$$\iota_e^{f_i}(e) := e \in E^{f_i}$$

Definition von $\alpha^{f_i} : E^{f_i} \to V^{f_i}$:

$$\alpha^{f_i}(e) := \begin{cases} \alpha^{f_{i-1}}(e) & \text{für alle } e \in E^{f_{i-1}} \\ v^{f_i} & \text{falls } e \in \tilde{E}^{f_i}. \end{cases}$$

Definition von $\beta^{f_i} : E^{f_i} \to \mathbb{R}$:

$$\beta^{f_i}(e) := \begin{cases} \beta(e) & \text{für alle } e \in E^{f_{i-1}} \\ \beta(h^{f_i}(e)) & \text{falls } e \in \tilde{E}^{f_i}. \end{cases}$$

Die Fortsetzungseigenschaft von α^{f_i} und β^{f_i} bleibt erhalten.

Wieder gilt

$$\beta^{f_i}(e) = \beta^{f_{i-1}}(\Pi_E^{f_i}(e)) \quad \text{für alle } e \in E^{f_i}.$$

Bei der Definition von $\omega^{f_i} : E^{f_i} \to V^{f_i}$ unterscheiden wir zwei Fälle:

1.Fall: f_i wurde bereits gesplittet, das heißt $\exists$ Indizes $i_1, .., i_k$ mit $f_i \in N^{f_{i_j}}$, und damit existieren Kanten $e_1, ..., e_k \in E^{f_{i-1}}$ mit $h^{f_{i_j}}(e_j) = f_i$ für alle $j = 1..k$.

$$\omega^{f_i}(e) := \begin{cases} \omega^{f_{i-1}}(e) & \text{für alle } e \in E^{f_{i-1}} \backslash (\{f_i\} \cup \bigcup_{j=1}^{k} {h^{f_{i_j}}}^{-1}(f_i)) \\ v^{f_i} & \text{falls } e \in \{f_i, f_{i_1}, ..f_{i_k}\} \\ \omega^{f_i}(h^{f_i}(e)) & \text{falls } e \in \tilde{E}^{f_i}. \end{cases} \tag{4.7}$$

2.Fall: $f_i \notin N^{f_j}, j = 1..i-1$

$$\omega^{f_i}(e) := \begin{cases} \omega^{f_{i-1}}(e) & \text{für alle } e \in E^{f_{i-1}} \backslash \{f_i\} \\ v^{f_i} & \text{falls } e = f_i \\ \omega^{f_i}(h^{f_i}(e)) & \text{falls } e \in \tilde{E}^{f_i}. \end{cases}$$

Die Definition von ω^{f_i} ist bei den Kanten aus $\tilde{E}^{f_i}$ rekursiv, da aber $h^{f_i}(e) \in E^{f_{i-1}}$ ist, kann keine Linksrekursion auftreten.

Mit dieser Definition ist $G^{f_i} := (V^{f_i}, E^{f_i}, \alpha^{f_i}, \omega^{f_i}, \beta^{f_i})$ ein Graph mit Abbiegeverboten T^{f_i}.

Beispiel 4.6 Als Beispiel für die Methode des verbotsorientierten Knotensplittings betrachten wir wieder den Graphen G aus Beispiel 4.2. Analog zum Beispiel 4.5 gilt $e_1^v = e_{12}$ und $N^{e_1^v} = e_{23}$. Wir definieren einen neuen Knoten $v_{2'}$ und eine neue Kante $e_{2'3}$. Die neuen Inzidenzabbildungen lauten $\alpha^{e_1^v}(e_{ij}) = v_i$ und

$$\omega^{e_1^v}(e) := \begin{cases} \omega(e) & \text{für alle } e \in E \backslash \{e_{12}\} \\ \mathbf{v_{2'}} & \mathbf{e = e_{12}} \\ v_3 & e = e_{2'3}. \end{cases}$$

Es ergibt sich folgender neuer Graph, wobei die neuen Elemente fett abgebildet sind:

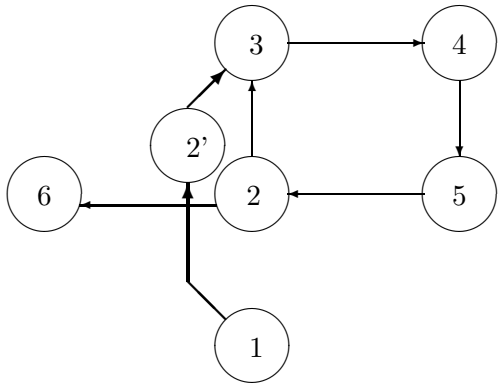


Abbildung 4.3 *Wegegraph mit der Methode des verbotsorientierten Knotensplittings*

Der kürzeste und auch einzige Weg, um von v_1 nach v_6 zu gelangen, ist also

$$(e_{12}, e_{2'3}, e_{34}, e_{45}, e_{52}, e_{26}).$$

4.5.2 Der Beweisanfang

Mit den folgenden Sätzen werden wir beweisen, daß der auf diese Weise erzeugte Graph tatsächlich ein minimaler Wegegraph ist.

Satz 4.1 *Sei* $G^{f_k} := (V^{f_k}, E^{f_k}, \alpha^{f_k}, \omega^{f_k}, \beta^{f_k}, T^{f_k})$ *ein Graph, auf den* k *Erweiterungsschritte angewendet wurden, und seien* E_0, N^{f_i} *wie oben definiert, dann gilt*

$$N^{f_i} \subseteq E \quad \textit{für alle } i = 1, ..., k$$

Beweis: Da $f_i \in E_0$ und $f_i \neq f_j$ für alle $j = 1, ..., i-1$ gilt, ist nach Definition von ω^{f_i}

$$\omega^{f_{i-1}}(f_i) = \omega(f_i) \in V.$$

Außerdem gilt nach der Definition von $\alpha^{f_{i-1}}$ für alle $e \in E^{f_{i-1}}$ mittels vollständiger Induktion:

$$e \in E \Leftrightarrow \alpha^{f_{i-1}}(e) \in V,$$

und für diesen Fall gilt $\alpha^{f_{i-1}}(e) = \alpha(e)$. Damit kommen bei der Definition von N^{f_i} nur Kanten aus E in Frage. ∎

Mit dem folgenden Satz zeigen wir, daß die kanonischen Projektionen surjektiv (trivial, da Grapherweiterung) und inzidenzerhaltend sind. Dies bedeutet unter anderem, daß jeder Weg im erweiterten Graphen als Bild der Projektion einen Weg im Ausgangsgraphen hat.

Satz 4.2 *Sei $G := (V, E, \alpha, \omega, \beta)$ ein Graph mit Abbiegeverboten T und $f \in E_0$. Sei $G^f := (V^f, E^f, \alpha^f, \omega^f, \beta^f)$ der zugehörige erweiterte Graph mit Abbiegeverboten T^f, dann gilt*

$$\alpha(\Pi_E^f(e)) = \Pi_V^f(\alpha^f(e)) \quad \textit{und} \quad \omega(\Pi_E^f(e)) = \Pi_V^f(\omega^f(e))$$

Beweis:

Sei $e \in E$, dann gilt

$$\begin{array}{lll} \alpha(\Pi_E^f(e)) & = \alpha(e) & \text{Definition von } \Pi_E^f \\ & = \Pi_V^f(\alpha^f(e)) & \text{Definition von } \Pi_V^f \text{ und } \alpha(e) \in V. \end{array}$$

Analoges gilt für ω, wenn $e \in E \backslash \{f\}$ ist. Für $e = f$ gilt:

$$\begin{array}{lll} \omega(\Pi_E^f(f)) & = \omega(f) & \text{Definition von } \Pi_E^f \\ & = \Pi_V^f(v^f) & \text{Definition von } \Pi_V^f \\ & = \Pi_V^f(\omega^f(f)) & \text{Definition von } \omega^f. \end{array}$$

Sei $e \notin E$, dann gilt

$$\begin{array}{lll} \alpha(\Pi_E^f(e)) & = \alpha(h^f(e)) & \text{Definition von } \Pi_E^f \\ & = \omega(f) & (f, h^f(e)) \text{ ist ein Weg} \\ & = \Pi_V^f(v^f) & \text{Definition von } \Pi_V^f \\ & = \Pi_V^f(\alpha^f(e)) & \text{Definition von } \alpha^f. \end{array}$$

$$\begin{array}{lll} \omega(\Pi_E^f(e)) & = \omega(h^f(e)) & \text{Definition von } \Pi_E^f \\ & = \omega^f(e) & \text{Definition von } \omega^f(e) \\ & = \Pi_V^f(\omega^f(e)) & \text{Definition von } \Pi_V^f \text{ und } \omega^f(e) \neq v^f. \end{array}$$

Analog und mit vollständiger Induktion beweist man die Inzidenzerhaltung für alle anderen Schritte und damit für die komplette Grapherweiterung. ∎

4.5.3 Die Unabhängigkeit der Reihenfolge

Satz 4.3 *Sei $G := (V, E, \alpha, \omega, \beta, T)$ ein Graph, E_0, N^f seien wie oben definiert und seien $f_1, f_2 \in E_0$ beliebig, dann gilt:*

Wenn man bei den Schritten 1 und 2 die Reihenfolge der Kanten f_1 und f_2 vertauscht, dann sind die Ergebnisgraphen isomorph. Siehe dazu auch Beispiel B.6

Beweis:
Seien f, g Kanten aus E_0. Im Fall a wird zuerst f, dann g, im Fall b zuerst g, dann f verwendet. Die erschaffenen Elemente aus den verschiedenen Fällen erhalten zur Unterscheidung die Indizes a bzw. b. Z.B. ist h_b^f die Bijektion von $\tilde{E}^f \to N^f$ im Fall b. Die Abbiegeverbote von der Kante g aus sind schon eingearbeitet worden.

1. Definition der Knotenbijektion $h_V : V_a^g \to V_b^f$

$$h_V(v) := \begin{cases} v & \text{für alle } v \in V \\ v_b^f & \text{falls } v = v_a^f \\ v_b^g & \text{falls } v = v_a^g \end{cases}$$

2. Definition der Kantenbijektion $h_E : E_a^g \to E_b^f$

$$h_E(e) := \begin{cases} e & \text{für alle } e \in E \\ h_b^{f\,-1}(h_a^f(e)) & \text{falls } e \in \tilde{E}_a^f \\ h_b^{g\,-1}(h_a^g(e)) & \text{falls } e \in \tilde{E}_a^g \end{cases}$$

Beide Abbildungen sind bijektiv:

h_V : klar nach Definition.

h_E : Nach Satz 4.1 gilt, daß N_a^f und N_b^f beide aus E sind. Sie sind also identisch unabhängig davon, ob neue Kanten definiert wurden oder nicht. Damit ist die Abbildung $h_b^{g\,-1} \circ h_a^g$ wohldefiniert und als Verkettung von Bijektionen bijektiv.

Verträglichkeit der Abbildungen β, α und ω:

1. $\beta_b^f(h_E(e)) = \beta_a^g(e)$:

   Für den Fall, daß $e \in E \subseteq E_a^g, E_b^f$ gilt trivialerweise $\beta_a^g(e) = \beta_b^f(h_E(e))$.
   Für $e \in \tilde{E}_a^f$ (und damit $h_E(e) \in \tilde{E}_b^f$) gilt

$$\begin{array}{lll} \beta_b^f(h_E(e)) & = \beta(h_b^f(h_E(e))) & \text{Def. von } \beta_b^f \\ & = \beta(h_b^f(h_b^{f\,-1}(h_a^f(e)))) & \text{Def. von } h_E \\ & = \beta(h_a^f(e)) & h_b^f \text{ ist Bijektion} \\ & = \beta_a^f(e) & \text{Def. von } \beta_a^f \\ & = \beta_a^g(e) & \text{weil } e \in \tilde{E}_a^f \end{array}$$

   Für $e \in \tilde{E}_a^g$ (und damit $h_E(e) \in \tilde{E}_b^g$) gilt

$$\begin{array}{lll} \beta_b^f(h_E(e)) & = \beta_b^g(h_E(e)) & \text{weil } e \in \tilde{E}_a^g \\ & = \beta(h_b^g(h_E(e))) & \text{Def. von } \beta_b^g \\ & = \beta(h_b^g(h_b^{g\,-1}(h_a^g(e)))) & \text{Def. von } h_E \\ & = \beta(h_a^g(e)) & h_b^g \text{ ist Bijektion} \\ & = \beta_a^g(e) & \text{Def. von } \beta_a^g \end{array}$$

2. $\alpha_b^f(h_E(e)) = h_V(\alpha_a^g(e))$:

   Wiederum ist im Fall $e \in E$ trivialerweise $\alpha_b^f(h_E(e)) = \alpha_a^g(e)$, denn die Anfangspunkte der Originalkanten werden in den einzelnen Schritten nicht verändert.

Für $e \in \tilde{E}_a^f$ (und damit $h_E(e) \in \tilde{E}_b^f$) gilt

$$\begin{array}{lll} \alpha_b^f(h_E(e)) & = v_b^f & \text{Def. von } \alpha_b^f \\ & = h_V(v_a^f) & \text{Def. von } h_V \text{ und} \\ h_V(\alpha_a^g(e)) & = h_V(\alpha_a^f(e)) & \text{da } e \in \tilde{E}_a^f \\ & = h_V(v_a^f) & \text{Def. von } \alpha_a^f \end{array}$$

Für $e \in \tilde{E}_a^g$ (und damit $h_E(e) \in \tilde{E}_b^g$) gilt

$$\begin{array}{lll} \alpha_b^f(h_E(e)) & = \alpha_b^g(h_E(e)) & \text{da } e \in \tilde{E}_a^g \\ & = v_b^g & \text{Def. von } \alpha_b^g \\ & = h_V(v_a^g) & \text{Def. von } h_V \text{ und} \\ h_V(\alpha_a^g(e)) & = h_V(v_a^g) & \text{Def. von } \alpha_a^g \end{array}$$

3. $\omega_b^f(h_E(e)) = h_V(\omega_a^g(e))$:

   Im Fall $e \in E \backslash E_0$ ist trivialerweise $\omega_b^f(h_E(e)) = \omega_a^g(e)$, denn die Endpunkte dieser Kanten werden in den einzelnen Schritten nicht verändert.

   Für $e = f$ gilt:

$$\begin{array}{lll} \omega_b^f(h_E(f)) & = v_b^f & \text{nach Def. von } \omega_b^f \\ h_V(\omega_a^g(f)) & = h_V(\omega_a^f(f)) & \text{da } f \in E \text{ und } f \neq g \\ & = h_V(v_a^f) & \text{Def. von } \omega_a^f \\ & = v_b^f & \text{Def. von } h_V \end{array}$$

   und für $e = g$ gilt analog:

$$\begin{array}{lll} \omega_b^f(h_E(g)) & = \omega_b^g(h_E(g)) & \text{da } g \in E \text{ und } g \neq f \\ & = v_b^g & \text{nach Def. von } \omega_b^g \\ h_V(\omega_a^g(g)) & = h_V(v_a^g) & \text{Def. von } \omega_a^g \\ & = v_b^g & \text{Def. von } h_V \end{array}$$

   Zum Beweis der Verträglichkeit für die Elemente aus $\tilde{E}^f$ bzw. $\tilde{E}^g$ unterscheiden wir wie bei der Definition von ω im zweiten Schritt mehrere Fälle:

   (a) Sei $f \in N_a^g = N_b^g$ und sei $e = h_a^{g-1}(f) \in \tilde{E}_a^g$, das heißt $h_E(e) \in \tilde{E}_b^g$ und $h_b^g(h_E(e)) = f$:

$$\begin{array}{lll} \omega_b^f(h_E(e)) & = \omega_b^g(h_E(e)) & \text{da } e \in \tilde{E}_a^g \\ & = \omega_b^g(h_b^{g-1}(h_a^g(e))) & \text{Def. von } h_E \\ & = \omega_b^g(h_b^{g-1}(f)) & \text{nach Voraussetzung} \\ & = \omega_b^g(h_b^g(h_b^{g-1}(f))) & \text{da } h_b^{g-1}(f) \in \tilde{E}_b^g \\ & = \omega_b^g(f) & h_b^g \text{ ist Bijektion} \\ & = v_b^g & \text{Def. von } \omega_b^g \end{array}$$

$$\begin{array}{lll} h_V(\omega_a^g(e)) & = h_V(\omega_a^g(h_a^{g-1}(f))) & \text{nach Voraussetzung} \\ & = h_V(v_a^g) & \text{Def. von } \omega_a^g \\ & = v_b^g & \text{Def. von } h_V \end{array}$$

(b) Sei $g \in N_a^f = N_b^f$ und sei $e = {h_a^f}^{-1}(g) \in \tilde{E}_a^f$, das heißt $h_E(e) \in \tilde{E}_b^f$ und $h_b^f(h_E(e)) = g$:

$$\begin{aligned}
\omega_b^f(h_E(e)) &= \omega_b^f({h_b^f}^{-1}(h_a^f(e))) && \text{Def. von } h_E \\
&= \omega_b^f({h_b^f}^{-1}(g)) && \text{nach Voraussetzung} \\
&= \omega_b^f(h_b^f({h_b^f}^{-1}(g))) && \text{da } {h_b^f}^{-1}(g) \in \tilde{E}_b^f \\
&= \omega_b^f(g) && h_b^f \text{ ist Bijektion} \\
&= v_b^f && \text{Def. von } \omega_b^f
\end{aligned}$$

$$\begin{aligned}
h_V(\omega_a^g(e)) &= h_V(\omega_a^f(e)) && \text{da } e \in \tilde{E}_a^f \\
&= h_V(\omega_a^f({h_a^f}^{-1}(g))) && \text{nach Voraussetzung} \\
&= h_V(v_a^f) && \text{Def. von } \omega_a^f \\
&= v_b^f && \text{Def. von } h_V
\end{aligned}$$

(c) Sei $e \in \tilde{E}_a^f$ und sowie e als auch $h_a^f(e)$ sind ungleich f und g, dann gilt:

$$\begin{aligned}
\omega_b^f(h_E(e)) &= \omega_b^f(h_b^f(h_E(e))) && \text{da } h_E(e) \in \tilde{E}_b^f \\
&= \omega_b^f(h_b^f({h_b^f}^{-1}(h_a^f(e)))) && \text{Def. von } h_E \\
&= \omega_b^f(h_a^f(e)) && h_b^f \text{ ist Bijektion} \\
&= \omega_b^g(h_a^f(e)) && \text{weil } e \text{ im 2. Schritt unverändert bleibt} \\
&= \omega(h_a^f(e)) && \text{da } h_a^f(e) \in E
\end{aligned}$$

$$\begin{aligned}
h_V(\omega_a^g(e)) &= h_V(\omega_a^f(e)) && \text{da } e \in \tilde{E}_a^f \\
&= h_V(\omega_a^f(h_a^f(e))) && \text{Def. von } \omega_a^f \\
&= \omega(h_a^f(e)) && \text{da } h_a^f(e) \in E
\end{aligned}$$

(d) Sei $e \in \tilde{E}_a^g$ und sowie e als auch $h_a^g(e)$ sind ungleich f und g, dann gilt analog:

$$\begin{aligned}
\omega_b^f(h_E(e)) &= \omega_b^g(h_E(e)) && \text{da } h_E(e) \in E_b^g \\
&= \omega_b^g(h_b^g(h_E(e))) && \text{da } h_E(e) \in \tilde{E}_b^g \\
&= \omega_b^g(h_b^g({h_b^g}^{-1}(h_a^g(e)))) && \text{Def. von } h_E \\
&= \omega_b^g(h_a^g(e)) && h_b^f \text{ ist Bijektion} \\
&= \omega(h_a^g(e)) && \text{da } h_a^g(e) \in E
\end{aligned}$$

$$\begin{aligned}
h_V(\omega_a^g(e)) &= h_V(\omega_a^g(h_a^g(e))) && \text{da } e \in \tilde{E}_a^g \\
&= h_V(\omega_a^f(h_a^g(e))) && \text{Def. von } \omega_a^f \\
&= \omega(h_a^g(e)) && \text{da } h_a^g(e) \in E
\end{aligned}$$

4.5.4 Die Beachtung der Abbiegeverbote

Satz 4.4 *Sei $G := (V, E, \alpha, \omega, \beta)$ ein endlicher Graph mit Abbiegeverboten T, $E_0 := \{e \in E \mid \exists (e, e_2) \in T\}$, $|E_0| = k$ und sei $G^{f_k} := (V^{f_k}, E^{f_k}, \alpha^{f_k}, \omega^{f_k}, \beta^{f_k}, T^{f_k})$ der aus G entstandene Graph, auf den die k Erweiterungsschritte angewendet wurden, v_1, $v_2 \in V$, dann gilt:*

Die im erweiterten Graphen berechneten kürzesten Wege sind die kürzesten Wege im Ausgangsgraphen unter Berücksichtigung der Abbiegeverbote.

Beweis:
Der Beweis wird indirekt und mit vollständiger Induktion über die Kanten aus E_0 geführt.

Induktionsanfang:
Sei nur die Kante $e_0 \in E_0 \Leftrightarrow$ alle Abbiegeverbote sind von der Form $(e_0, *)$. Annahme: Es existieren zwei Knoten $v_1, v_2 \in V$, so daß gilt: es existiert ein Weg von v_1 nach v_2, der kürzer ist als der vom Algorithmus errechnete Weg.

Um die Behauptung zum Widerspruch führen zu können, benötigen wir zunächst folgenden Satz:

Satz 4.5 *Zu jedem Weg* $(e_1, ..., e_k)$ *im Ausgangsgraphen* G *unter Berücksichtigung der Abbiegeverbote von* e_0 *aus existiert ein Weg* $(e'_1, ..., e'_k)$ *im erweiterten Graphen* G^{e_0} *mit:*

$$\begin{aligned}
\Pi_E^{e_0}(e'_i) &= e_i & i = 1..k \\
\beta^{e_0}(e'_i) &= \beta(e_i) & i = 1..k \\
\Pi_V^{e_0}(\omega^{e_0}(e'_i)) = \Pi_V^{e_0}(\alpha^{e_0}(e'_{i+1})) &= \omega(e_i) & i = 1..k-1 \\
\Pi_V^{e_0}\alpha^{e_0}(e'_1) &= \alpha(e_1) & \\
\Pi_V^{e_0}\omega^{e_0}(e'_k) &= \omega(e_k) & .
\end{aligned}$$

Beweis:

1 Fall: Für alle $i = 1..k$ gilt: $e_0 \neq e_i$:

Sei $e'_i := \iota^{e_0}(e_i)$ für alle i, dann gilt

$$\begin{aligned}
\omega^{e_0}(e'_i) &= \omega^{e_0}(\iota^{e_0}(e_i)) && \text{Def. von } \iota^{e_0} \\
&= \omega(e_i) && \text{da } e_i \in E \neq e_0 \\
&= \alpha(e_{i+1}) && (e_i) \text{ ist ein Weg} \\
&= \alpha^{e_0}(\iota(e_{i+1})) && \text{Def. von } \alpha^{e_0} \\
&= \alpha^{e_0}(e'_{i+1}) && \text{Def. von } \iota^{e_0}
\end{aligned}$$

2 Fall: e_0 kommt im Weg e_i ein oder mehrere Male vor:

Sei

$$e'_i := \iota^{e_0}(e_i) \quad \text{für alle } i, \quad \text{falls } e_0 \neq e_{i-1}.$$

Falls $e_0 = e_{i-1}$, so gilt, daß $e_i \in N^{e_0}$ ist, es existiert also $(h^{e_0})^{-1}(\iota^{e_0}(e_i)) \in \tilde{E}^{e_0}$. Nur an dieser Stelle gehen die Abbiegeverbote in den Beweis ein. Um die Wegeigenschaft in G^{e_0} zu erhalten, definieren wir

$$e'_i := (h^{e_0})^{-1}(\iota^{e_0}(e_i)), \quad \text{falls } e_0 = e_{i-1}.$$

Sei $e_i = \iota^{e_0\,-1}(e'_i) = e_0$. Für das oben definierte e'_{i+1} gilt nun:

$$\begin{aligned}
\omega^{e_0}(e'_i) &= \omega^{e_0}(\iota(e_0)) && \text{Voraussetzung} \\
&= v^{e_0} && \text{Def. von } \omega^{e_0} \\
&= \alpha^{e_0}(e'_{i+1}) && \text{da } e'_{i+1} \in \tilde{E}^{e_0}
\end{aligned}$$

Der Beweis für die anderen Kantenkombinationen verläuft analog zum Fall 1. Damit ist Satz 4.5 bewiesen.

Bemerkung:
Analog zeigt man, daß zu jedem Weg (e_j) $j = 1..k$ im Graphen $G^{f_{i-1}}$ unter der Berücksichtigung der Abbiegeverbote von f_i aus ein Weg e'_j $j = 1..k$ existiert mit den in Satz 4.5 beschriebenen Bedingungen. Auch die Konstruktion des Weges bleibt gleich.

Fortsetzung des Beweises von Satz 4.4:

Der berechnete kürzeste Weg von v_1 nach v_2 heiße $(b_1, ..., b_k)$, der tatsächlich kürzeste Weg heiße $(e_1, ..., e_l)$. $(b_i) \subseteq E^{e_0}$, $(e_j) \subseteq E$. Von diesem Weg existiert nach Satz 4.2 ein Bild unter der kanonischen Projektion in G. Annahme:

$$\sum_{j=1}^{l} \beta(e_j) < \sum_{i=1}^{k} \beta(\Pi^{e_0}(b_i)).$$

Zum Weg (e_j) existiert nach Satz 4.5 ein Weg $(e'_j) \subseteq E^{e_0}$ mit $\beta(e'_j) = \beta(e_j)$ $j = 1, ..., l$. Mit der Definition der kürzesten Wege folgt also:

$$\begin{array}{lll} \sum_{j=1}^{l} \beta(e'_j) & = \sum_{j=1}^{l} \beta(e_j) & \\ & < \sum_{i=1}^{k} \beta(\Pi^{e_0}(b_i)) & \text{Annahme} \\ & = \sum_{i=1}^{k} \beta^{e_0}(b_i) & \beta^{e_0} \text{ ist Fortsetzung von } \beta \\ & \leq \sum_{j=1}^{l} \beta(e'_j) & b_i \text{ ist kürzester Weg in } E^{e_0} \end{array}$$

Dies ist ein Widerspruch, also ist $\Pi^{e_0}(b_i)$ auch kürzester Weg in G.

Der Induktionsschritt verläuft analog zum Induktionsanfang unter der Verwendung der Bemerkung von Satz 4.5. ∎

4.5.5 Die Minimalität

Satz 4.6 *Sei $G = (V, E, \alpha, \omega)$ ein endlicher, gerichteter Graph mit Abbiegeverboten T, $G^f := (V^f, E^f, \alpha^f, \omega^f)$ der von G und T erzeugte Wegegraph, dann ist jede Kante notwendig, das heißt: Zu jeder Kante $e^f \in E^f$ existiert ein Weg $w = \{e_1^w, ..., e_n^w\} \subseteq E$, so daß gilt:*

Für alle Wege $w^f \subseteq E^f$ mit $\Pi^f(e_i^{w^f}) = e_i^w$ $1 \leq i \leq n$ und $\alpha^f(e_1^{w^f}) \in V$ gilt $e^f \in w^f$ (im Widerspruch zu (VE 2) in Definition 2.12).

Um alle Wege von G in G^f darstellen zu können, darf also keine Kante aus E^f weggelassen werden. Es handelt sich hierbei also um eine Art lokale Minimalität. Der Beweis, daß es sich um eine globale Minimalbedingung handelt, steht noch aus.

Die Bedingung $\alpha^f(e_1^{w^f}) \in V$ benötigen wir, damit bei jeder (kürzesten) Wegeberechnung unser Startknoten im erweiterten Graphen eindeutig bestimmt ist (siehe dazu Beispiel 4.7).

Beweis:

1. Sei $e^f \in E$, dann definieren wir $w^f := (e^f)$. Mit dieser Definition gilt $\alpha^f(e^f) \in V$ nach Satz 4.1, und alle neu erschaffenen Kanten besitzen Anfangspunkte in $V^f \backslash V$. Nach Satz 4.1 ist $e^f \in G^f$ die einzige Kante mit den Eigenschaften $\alpha^f(e^f) \in V$ und $\Pi^f(e^f) = e^f$. Würde man die Kante e^f aus G^f entfernen, so wäre der erlaubte Weg w^f in G^f nicht mehr möglich.

2. Sei $e^f \notin E$, dann gilt nach Satz 4.1: $\alpha^f(e^f) \notin V$, $\alpha^f(e^f)$ wurde also in einem Schritt z.B. dem Schritt Nr. j erzeugt. Damit gilt $\omega^f(f_j) = \alpha^f(e^f)$ und die Sequenz (f_j, e^f) ist ein Weg in E^f mit $\alpha^f(f_j) \in V$. Sei $w' = (e'_1, e'_2)$ ein Weg, für den $\Pi^{f_j}(e_1) = f_j$, $\Pi^{f_j}(e_2) = \Pi^{f_j}(e^f)$ und $\alpha^f(e_1) \in V$ erfüllt ist, dann gilt nach Satz 4.1 $e_1 = f_j$. e_2 ist nach Konstruktion im j. Schritt erschaffen worden, da $\alpha^f(e_2) = v^{f_j}$ (Wegeigenschaft) und die Anfangspunkte in den restlichen Schritten nicht mehr verändert werden. Damit gilt $\Pi^{f_j}(e_2), \Pi^{f_j}(e^f) \in N^{f_j}$ und

$$\begin{array}{lll} e^f & = (h^{f_j})^{-1}(\Pi^{f_j}(e^f)) & \text{Definition von } \Pi^{f_j} \\ & = (h^{f_j})^{-1}(\Pi^{f_j}(e_2)) & \text{Voraussetzung} \\ & = e_2 & \text{Definition von } \Pi^{f_j} \end{array}$$

   Wenn man also die Kante e^f aus G^f entfernen würde, so wäre der erlaubte Weg w mit der Bedingung $\alpha^f(e_1) \in V$ in G^3 nicht mehr möglich.

Beispiel 4.7 Die Bedingung $\alpha^f(e_1^{w^f}) \in V$ ist notwendig, damit der Anfangspunkt eines in den erweiterten Graphen übertragenen Weges eindeutig zu finden ist. Ohne diese Bedingung könnten in manchen erweiterten Graphen Kanten und Knoten weggelassen werden, ohne daß es Wege im Ausgangsgraphen gibt, die im erweiterten Graphen nicht mehr dargestellt werden können. Ein solcher Graph wird im folgenden vorgestellt:

Sei $G = (V, E, \alpha, \omega)$ mit Abbiegeverboten T und

$$V := \{v_1, ..., v_5\} \quad E := \{e_{12}, e_{13}, e_{14}, e_{15}, e_{21}, e_{31}, e_{41}, e_{51}\}$$

$$\alpha(e_{ij}) := v_i \quad \omega(e_{ij}) := v_j \tag{4.8}$$

$$T := \{(e_{21}, e_{13}), (e_{21}, e_{15}), (e_{31}, e_{14}), (e_{31}, e_{15})(e_{41}, e_{12}), (e_{41}, e_{13}), (e_{51}, e_{12}), (e_{51}, e_{14})\}$$

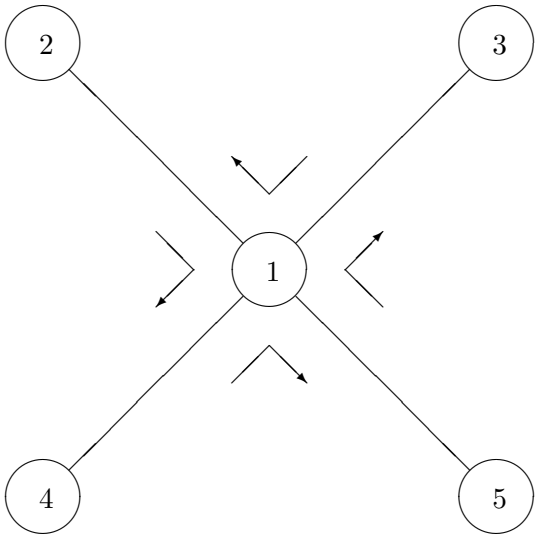


Abbildung 4.4 *Beispielgraph, bei dem nur rechts abgebogen werden kann*

Nach 4 Schritten ergibt sich durch vierfaches Splitten des Knoten v_1

$$V^{e_{41}} := \{v_1, ..., v_5, v_{1^2}, v_{1^3}, v_{1^4}, v_{1^5}\}$$

$$E^{e_{41}} := \{e_{12}, e_{13}, e_{14}, e_{15}, e_{21}, e_{31}, e_{41}, e_{51}, e_{31^2}, e_{21^4}, e_{41^5}, e_{51^3},$$

$$e_{1^2 2}, e_{1^2 3}, e_{1^4 2}, e_{1^4 4}, e_{1^5 4}, e_{1^5 5}, e_{1^3 5}, e_{1^3 3}\}$$

Die Inzidenzstruktur ist wieder durch die Gleichung (4.8) erklärt. Damit ergibt sich folgendes Bild:

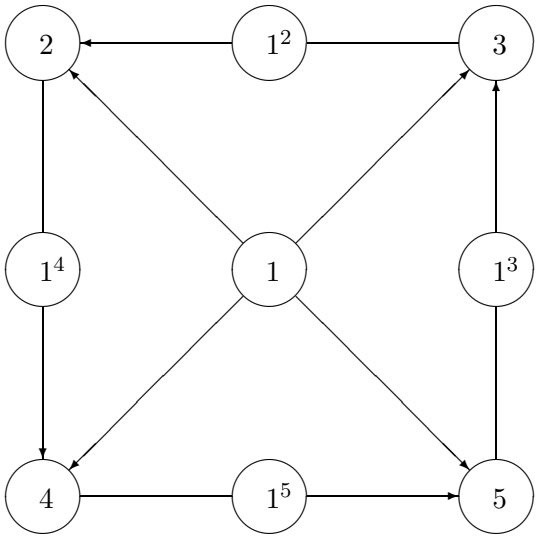


Abbildung 4.5 *Beispielgraph, bei dem nur rechts abgebogen werden kann, nach 4 Erweiterungsschritten*

Alle möglichen Wege in diesem Graphen sind Sequenzen der Form

$$..., v_2, v_1, v_4, v_1, v_5, v_1, v_3, v_1, v_2, ...$$

mit festem Anfangs- und Endpunkt und beliebig vielen inneren Schleifen der Form .., v_i, v_1, v_i, v_1, .. $i = 2, ..., 5$. Für einen Weg, der im Ausgangsgraphen beim Knoten v_1 beginnt, kann ohne die Bedingung $\alpha^f(e_1^{w^f}) \in V$ kein eindeutiger Startknoten gefunden werden. Mit dieser Bedingung wird eindeutig v_1 (im erweiterten Graphen) als Startknoten festgelegt, von welchem aus alle Knoten v_2, ..., v_5 erreicht werden können. Von allen anderen Knoten v_{1^i} $i = 2, ..., 5$ aus können nur jeweils zwei andere Knoten erreicht werden.

4.5.6 Fazit

Mit den Sätzen 4.4 und 4.5 wurde gezeigt, daß im erweiterten Graphen kürzeste Wege unter Berücksichtigung der Abbiegeverbote berechnet werden können. Im Satz 4.3 wird gezeigt, daß die Abbiegeverbotsklassen ohne eine feste Reihenfolge bearbeitet werden können und eine manchmal aufwendige Anordnung nicht erforderlich ist.

Der Aufwand in der Preprocessingphase liegt vor allem in der Berechnung von E_0, sowie in der Berechnung der Mengen N^{f_i}. Da in beiden Berechnungen im schlimmsten Fall alle Kanten untersucht werden müssen, liegt der Aufwand bei

$$O(|E|) \text{ (Berechnung von } E_0) * O(|E|) \text{ (Berechnung von } N^{f_i}) = O(|E|^2).$$

Seien $\tilde{V}$, $\tilde{E}$ die Mengen aller Knoten bzw. Kanten, die durch das Preprocessing neu zum erweiterten Graphen hinzugekommen sind, so gilt:

$$|\tilde{V}| = |E_0| \text{ und}$$

$$|\tilde{E}| = |E_0| * \text{Maximalzahl der Kanten, die von einem Knoten ausgehen.}$$

Im schlechtesten Fall bedeutet dies mit $n = |V|$ und $|E_0| = |E| = O(n^2)$:

$$|\tilde{V}| = O(n^2) \quad \text{und} \quad |\tilde{E}| = O(n^4).$$

4.6 Knotenorientierte Netzwerke

Neben der Methode der Kantenaufnahme befindet sich in [SB77] noch ein Hinweis auf ein graphveränderndes Verfahren, mit dem man kürzeste Wege in Straßennetzen mit Abbiegeverboten berechnen kann. Dazu wird jede Kreuzung so oft gesplittet, wie Straßen dort beginnen und enden. Eine Straße, die in zwei Richtungen befahren werden kann, wird also doppelt gezählt. Die alte Kreuzung verschwindet. Jede dieser gesplitteten Kreuzungen ist Anfangs- oder Endknoten der jeweils zugeordneten Ein- bzw. Ausfallstraße. Von jedem Einfallstraßenknoten wird eine Kante zu jedem Ausfallstraßenknoten gezogen. Sie stellt das Abbiegen von der Einfallstraße in die Ausfallstraße dar. Ein Abbiegeverbot wird durch einfaches Löschen der diesen Abbiegevorgang repräsentierenden Kante bewirkt. Gegeben sei eine Kreuzung von 4 Straßen.

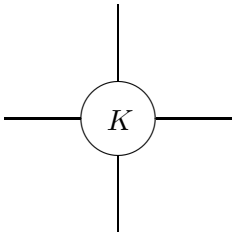


Abbildung 4.6 *Ausgangskreuzung*

Die folgende Abbildung zeigt die Erweiterung dieser Kreuzung mit allen Abbiegemöglichkeiten.

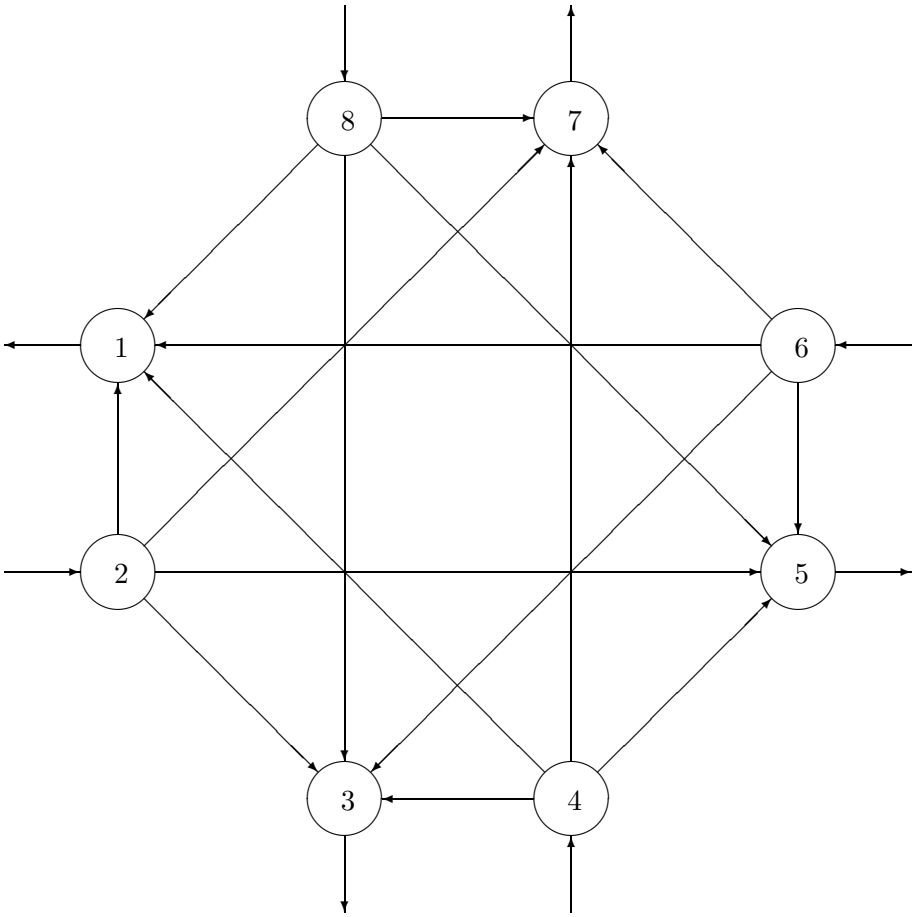


Abbildung 4.7 *Kreuzungsteilung*

Betrachten wir exemplarisch den Einfallstraßenknoten v_2. Die Kanten e_{23}, e_{25}, e_{27} und e_{21} haben v_2 als Anfangspunkt. Dabei stellt e_{23} das Rechtsabbiegen, e_{25} das geradeaus Weiterfahren, e_{27} das Linksabbiegen und e_{21} das Wenden dar. Wollen wir nun als Beispiel das Linksabbiegen verbieten, so löschen wir einfach die Kante e_{27}. Auf dieser Kreuzung ist der Knoten v_7 von v_2 aus dann nicht mehr erreichbar.

Die Ein- und Ausfallstraßen dürfen nicht gemeinsam betrachtet werden. Würde man die Knotenpaare (v_1, v_2), (v_3, v_4), (v_5, v_6) und (v_7, v_8) zusammenfassen, so entstünde folgende Kreuzung.

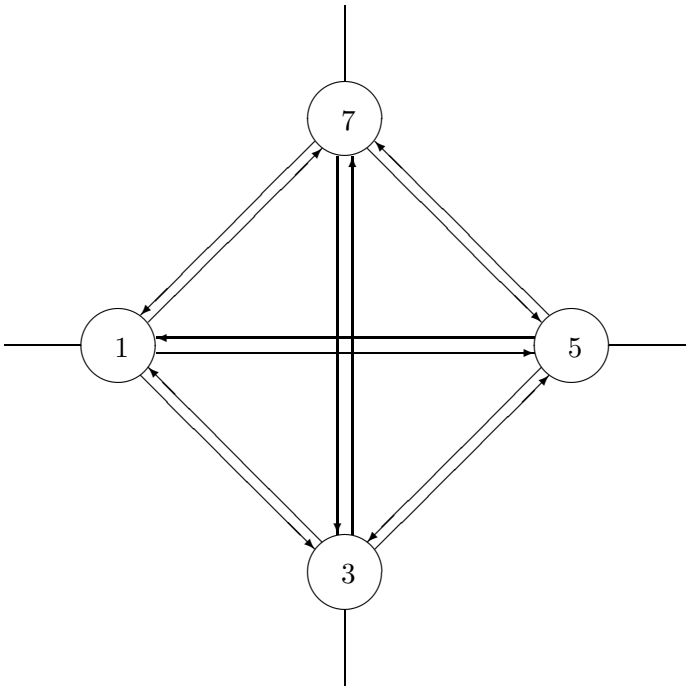


Abbildung 4.8 *Kreuzungsteilung mit Zusammenfassung der Ein- und Ausfallstraßen*

Bei einem Linksabbiegeverbot von v_1 aus würden wir zunächst die Kante v_{17} löschen. Das Linksabbiegen an dieser Kreuzung wäre aber weiterhin denkbar, denn die Fahrtrouten (e_{13}, e_{37}), (e_{15}, e_{57}) oder (e_{13}, e_{35}, e_{57}) bleiben weiterhin möglich. Um das Linksabbiegen generell zu verbieten, müssen also noch mehr Kanten gelöscht werden, was eventuell andere Fahrtrouten verbietet, die aber eigentlich erlaubt sind.

Das Aufteilen jeder Kreuzung vergrößert den Graphen wesentlich. In unserem Beispiel wurden aus einem Knoten 16 Kanten und 8 Knoten erzeugt. Allgemein sind dies für n Eingangskanten und m Ausgangskanten $n + m$ neue Knoten und $n \cdot m$ neue Kanten. Wenn an der beschriebenen Kreuzung unser Weg beginnen soll, ist auch nicht klar, an welchem der Knoten dies sein soll. Dieses Problem kann aber durch Definition eines weiteren Knotens behoben werden (Startknoten dieser Kreuzung). Von diesem Knoten muß dann zu jedem Ausfallstraßenknoten eine Kante gezogen werden.

Kapitel 5

Berechnung kürzester Wege in Graphen mit Wegeverboten

Bei der Berechnung von kürzesten Wegen in Straßennetzen sollen Wegeverbote einbezogen werden. Ein Wegeverbot ist ein verallgemeinertes Abbiegeverbot. Ein Abbiegeverbot verbietet aus der Straße A kommend die Weiterfahrt in Straße B. Ein Wegeverbot untersagt die Weiterfahrt in Straße C, wenn zuerst die Straße A und dann die Straße B befahren wurde. Befährt man die Straße B nicht von Straße A her kommend, so ist die Weiterfahrt in Straße C möglicherweise erlaubt. Ein Wegeverbot gibt es zum Beispiel in Stuttgart auf der B 14 vom Schattenring in Richtung A 831. Hier ist die Ausfahrt an der Anschlußstelle Stuttgart Vaihingen untersagt, wenn man die B 14 von der Anschlußstelle Universität aus befährt (siehe Abbildung 5.1).

5.1 Motivation

Es gibt mehrere Gründe, Wegeverbote zu betrachten.

1. Auf mehrspurigen Straßen mit Ein- und Ausfahrten auf der rechten und linken Seite ist es möglich, daß nach der Einfahrt die sofortige Ausfahrt, die ein gefährliches Queren der Straße zur Folge hätte, verboten ist. Um diesen Sachverhalt mit einem Abbiegeverbot zu beschreiben, kann z.B. die Ein- und Ausfahrt zu einer Kreuzung zusammengefaßt werden, was Ungenauigkeiten zur Folge hat. Die Definition eines Wegeverbotes spiegelt den Sachverhalt besser wider.
2. Bei einer Routenplanung sollen bewußt gewisse Wege nicht berücksichtigt werden.
3. Bei einer Reise sollen nicht zu viele Orte, an welchen etwa keine Wartung des Fahrzeuges möglich ist, hintereinander besucht werden.
4. Das Problem der k-kürzesten Wege kann ebenfalls mit Wegeverboten beschrieben werden, indem man im ersten Schritt den kürzesten Weg berechnet, diesen dann verbietet und dies $k-1$ mal wiederholt (siehe dazu auch Abschnitt 7.2).

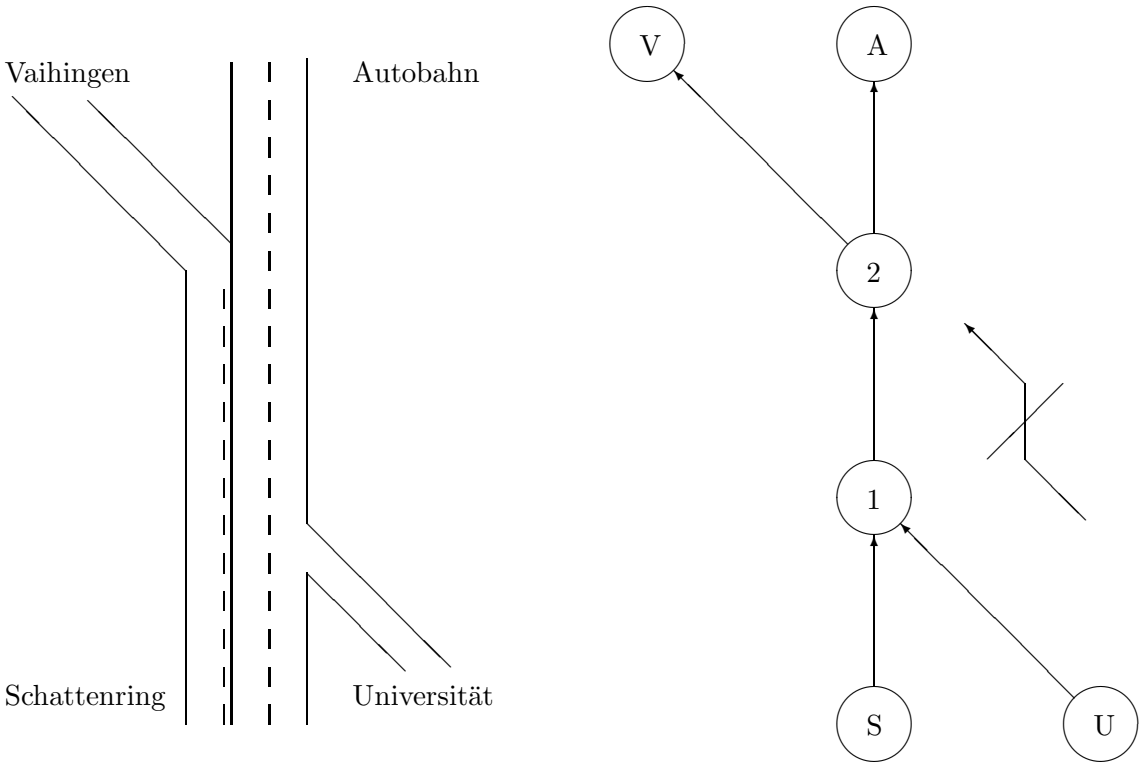


Abbildung 5.1 *Verkehrslage der B 14*

5.2 Der Lösungsansatz

Die Idee des Algorithmus besteht darin, die erste Kante eines Wegeverbotes umzuleiten, um dann den letzten Schritt zu verbieten. Als erstes werden die Kanten und Knoten eines Wegeverbotes ohne erstes und letztes Element gesplittet. Der verdoppelte Weg wird in den Graphen eingebunden. Am Ende werden die gesplitteten Wege mit dem Originalgraph und untereinander verbunden.

Im Folgenden sei $G = (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, welche analog zur Gleichung 2.3 in Äquivalenzklassen zusammengefaßt sind. Wenn zu einem Wegeverbot p ein Wegeverbot p' existiert und p' ist ein Teilweg von p, so soll auf die Betrachtung von p verzichtet werden. Der Einfachheit halber möchte ich auf die Betrachtung einer Kantengewichtsfunktion verzichten. Eine Erweiterung auf gewichtete Graphen ist jederzeit möglich und intuitiv klar.

Nach der Definition 2.9 ist ein Wegeverbot ein verallgemeinertes Abbiegeverbot. Deshalb ist es sinnvoll, einen Ansatz zur Berechnung kürzester Wege in Graphen mit Abbiegeverboten zu verallgemeinern. Der von mir vorgestellte Algorithmus baut auf dem Ansatz des verbotsorientierten Knotensplitting aus 4.5 auf.

Der Graph G soll zu einem Graphen G' ohne Wegeverbote erweitert werden, der folgende Eigenschaften hat:

1. Es existiert ein Epimorphismus von G' nach G, das heißt, G geht aus G' durch Zusammenfassen einiger Kanten und Knoten hervor. Dieser Epimorphismus heißt *kanonische Projektion*.

2. Jeder erlaubte Weg in G besitzt mindestens einen ihn repräsentierenden Weg in G'.

3. Die kanonische Projektion jedes Weges aus G' ist ein Weg in G und darf keinen verbotenen Teilweg enthalten.

4. G' erfüllt die gleiche Minimalitätseigenschaft wie die Grapherweiterung aus Abschnitt 4.5.

G' ist also ein von G erzeugter Wegegraph. Bei der Grapherweiterung werden zuerst die Kanten und Knoten des Wegeverbotes ohne die jeweils ersten und letzten Kanten und Knoten verdoppelt und in den Graphen eingebunden. Danach werden die verdoppelten Wege mit dem Ausgangsgraphen und untereinander verbunden.

Im Gegensatz zu den Abbiegeverboten gibt es bei den Wegeverboten drei zu unterscheidende Lagebeziehungen zwischen zwei Wegeverboten. Seien in einem Graphen zwei Wegeverbote $p = (e_1, ..., e_m)$ und $p' = (e'_1, ..., e'_m)$ definiert, dann hängt deren Lagebeziehung und damit die Bedeutung für den Algorithmus im Wesentlichen von den Anfangskanten e_1 und e'_1 ab.

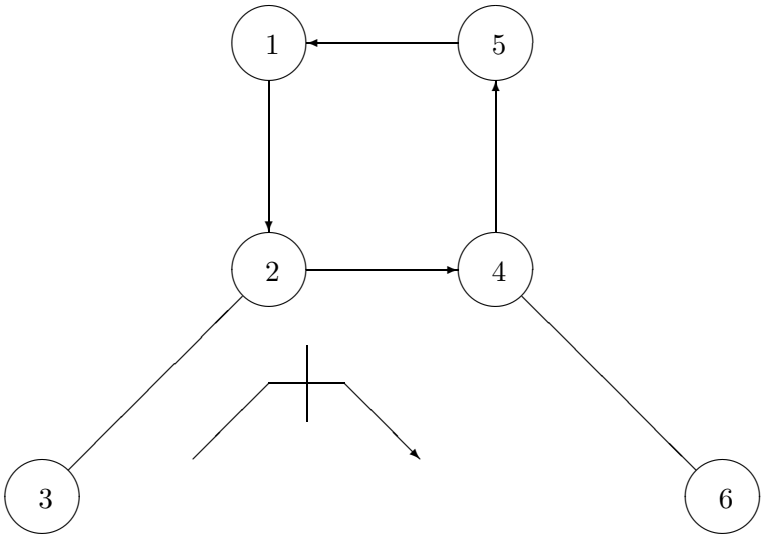
1. Fall: $e_1 \notin p'$ und $e'_1 \notin p$. In diesem Fall können die Wegeverbote getrennt voneinander betrachtet werden.

2. Fall: $e_1 = e'_1$. In diesem Fall sind die Wegeverbote in derselben Klasse. Für $p \neq p'$ können gewisse Kanten gemeinsam betrachtet werden.

3. Fall: $e'_1 \in p$ und $e_1 \neq e'_1$ (oder analog $e_1 \in p'$): Hier müssen die Wegeverbote gemeinsam betrachtet werden.

Bemerkung: Der Fall $e'_1 \in p$ und $e_1 \in p'$ kann vorkommen und ist in dieser Fallunterscheidung ebenfalls berücksichtigt. Wegeverbote können sich auch selbst kreuzen. Die dafür notwendigen Kanten werden ebenfalls in der Rubrik 5.5 definiert. Ein einzelnes Betrachten jedes Wegeverbotes für sich und das Erstellen von Zwischengraphen funktioniert mit diesem Algorithmus nicht (siehe Abschnitt 7.1).

In einem Graphen mit Wegeverboten ist es im Gegensatz zu Graphen mit Abbiegeverboten möglich, kürzeste Wege zu finden, die eine unidirektionale Kante mehrfach durchlaufen. Dies zeigt folgendes Beispiel:



Beispiel 5.1

Abbildung 5.2 *Beispielgraph mit einem Wegeverbot, bei dem kürzeste Wege eine Kante mehrmals beinhalten.*

Gegeben ist ein Graph (V, E, α, ω) mit Wegeverboten P und der Knotenmenge

$$V = v_i, \quad i = 1..6,$$

der Kantenmenge

$$E = \{e_{12}, e_{23}, e_{24}, e_{32}, e_{45}, e_{46}, e_{51}, e_{64}\},$$

den Inzidenzabbildungen $\alpha(e_{ij}) = v_i \;\; \omega(e_{ij}) = v_j$, und dem Wegeverbot

$$P := \{(e_{32}, e_{24}, e_{46})\}.$$

Ein Weg von v_3 nach v_6 muß des Wegeverbotes wegen mindestens ein Mal den gesamten Kreis v_2, v_4, v_5, v_1 durchlaufen. Der kürzeste Weg von v_3 nach v_6 wäre hier

$$(e_{32}, \mathbf{e_{24}}, e_{45}, e_{51}, e_{12}, \mathbf{e_{24}}, e_{46})$$

Definition 5.1 *Sei $G := (V, E, \alpha, \omega)$ ein Graph, $w = (e_1, ..., e_n) \in W$, P eine Menge von Wegeverboten und sei $P' \subseteq P \cap [w]$. Für jedes $p \in P$ sei die Menge $[\![p]\!] = \{p_1, ..., p_m\}$ angeordnet.*

Sei $\tilde{P} := \{p \in P' \mid p \text{ ist maximaler Anfangsweg von } w\}$, und sei $\tilde{p}$ aus $\tilde{P}$, dann gilt $\tilde{P} \subseteq [\![\tilde{p}]\!]$. Das Element $p_k \in \tilde{P}$ mit dem kleinsten Index k heißt **der maximale Anfangsweg** *von w in P'.*

Sei $\hat{P} := \{p \in P' \mid p \text{ ist Endweg von } w\}$, und sei

$$Eka(\hat{P}, w) := \max\{eka(\hat{p}, w) \mid \hat{p} \in \hat{P}\}.$$

Falls $eka(\hat{p}, w) = Eka(\hat{P}, w)$ *gilt, so nennt man* $\hat{p}$ **einen maximalen Endweg** *von* w *in* $\tilde{P}$. *Alle maximalen Endwege von* w *in* $\hat{P}$ *liegen in der Klasse* $[\![\hat{p}]\!]$. *Das Wegeverbot mit minimalem Index heißt* **der maximale Endweg** *von* w *in* $\hat{P}$. *Existiert kein Weg dieser Art, so sei* $\hat{p} := \lambda$ *und* $Eka(\hat{P}, w) := 0$.

Analog zur Gleichung 4.5 definieren wir wieder die Menge E_0 *durch*

$$E_0 := \{e \in E \mid \exists (e, e_2, ..., e_n) \in P\}$$

als Menge aller Kanten, von welchen aus ein Wegeverbot beginnt.

5.3 Die Wegeverdopplung

Die Grundidee meines Algorithmus besteht darin, den ersten Schritt eines verbotenen Weges umzuleiten, um dann den letzten Schritt zu verbieten. Bei einem verbotenen Weg bestehend aus n Kanten ist das Befahren der ersten $n-1$ Kanten erlaubt – erst der n-te Schritt ist dann verboten.

Erster Schritt der Wegeverdopplung

Sei $[\![p]\!]$ eine wie in Definition 5.1 angeordnete Wegeverbotsklasse. Wir betrachten im ersten Schritt nur das erste Wegeverbot $p_1 = p = (e_1, ..., e_n)$. Wir werden die Kanten und Knoten des Wegeverbotes p bis auf die ersten und letzten Elemente verdoppeln. Hierzu definieren wir eine Menge von neuen Knoten $\tilde{V}^p = \{v_1^p, ..., v_{n-1}^p\}$ Diese sind die gesplitteten Endknoten der Kanten aus p. Zusatzlich definieren wir eine Bijektion $h_V^p : \tilde{V}^p \to (e_1, ..., e_{n-1})$ (jede Komponente wird als ein Bildelement gesehen – im Folgenden soll eine Bijektion in einen Vektor immer komponentenweise erklärt sein) durch

$$h_V^p(v_i^p) := e_i \quad i = 1, ..., n-1.$$

Der Knoten Nr. n wird nicht benötigt, da der Weg in diese Richtung verboten ist. Da die Abbildung h_V^p in einen Vektor abbildet, wird sichergestellt, daß ein eventuell mehrfach vorkommender Knoten im Wegeverbot auch mehrfach gesplittet wird.

Falls $n > 2$ definieren wir weiterhin eine Menge von neuen Kanten $\tilde{E}_1^p = \{e_2^p, ..., e_{n-1}^p\}$ und eine Bijektion $h_1^p : \tilde{E}_1^p \to (e_2, ..., e_{n-1})$ durch

$$h_1^p(e_i^p) := e_i \quad i = 2, ..., n-1.$$

$\tilde{E}_1^p$ stellt die gesplitteten Kanten ohne erste und letzte Kante des Weges p dar. e_1^p ist keine neue Kante, sondern $e_1^p := e_1$. e_1 erhält den neuen Endpunkt v_1^p (= Umleitung).

$p := (e_1, ..., e_5) = (e_{12}, e_{23}, e_{34}, e_{45}, e_{56})$

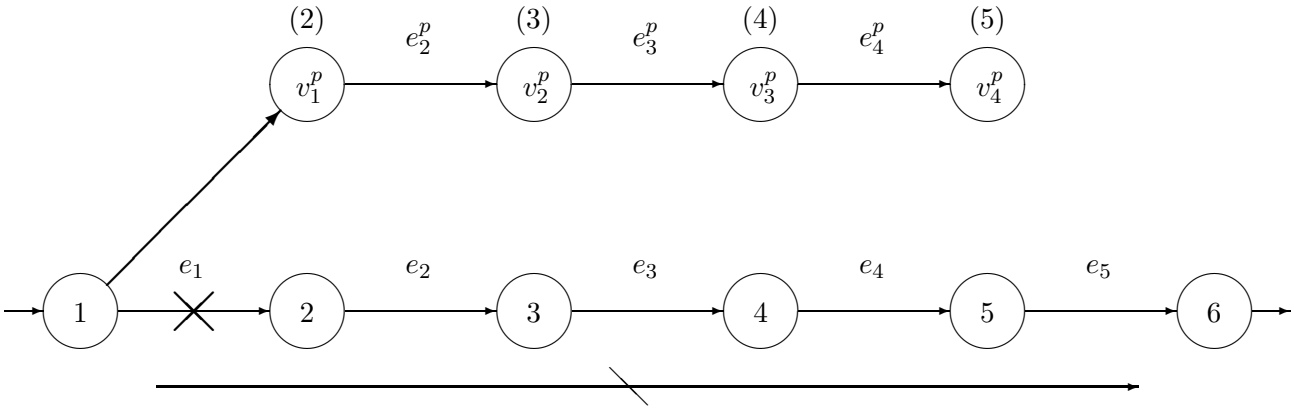


Abbildung 5.3 *Eine Wegeverdopplung*

k-ter Schritt der Wegeverdopplung

Siehe zu diesem Schritt auch Beispiel B.9.
Sei $\tilde{P} := \{p_1, ..., p_{k-1} \in [\![p]\!] \mid p_i \; i = 1, ..., k-1$ wurden bereits bearbeitet $\}$ – wir betrachten das Wegeverbot $p_k = (e_1, ..., e_n)$ welches noch nicht bearbeitet wurde. Es sei $p' = (e'_1, ..., e'_m)$ der maximale Anfangsweg von p_k in $\tilde{P}$ mit q gemeinsamen Kanten. Ab der Kante e_{q+1} nimmt das Wegeverbot p_k einen anderen Verlauf als alle zuvor behandelten Wegeverbote. Gilt $q = n-1$, so ist nichts weiter zu tun, da die Kante e_n nicht gesplittet wird. Gilt sogar $q \geq n$, so ist p_k Teilweg von p', und damit würde das Wegeverbot p' sowieso nicht betrachtet werden. Im Folgenden ist also $q < n$.

Die Kanten und deren Endknoten bis zum Index q können aus den vorigen Schritten übernommen werden und müssen nicht erneut gesplittet werden. Da $q < n$ ist, definieren wir eine Menge von neuen Kanten $\tilde{E}_1^{p_k} := \{e_{q+1}^{p_k}, ..., e_{n-1}^{p_k}\}$, und eine Bijektion $h_1^{p_k} : \tilde{E}_1^{p_k} \to (e_{q+1}, ..., e_{n-1})$ durch $h_1^{p_k}(e_i^{p_k}) := e_i \; i = q+1, ..., n-1$. Dies sind die Kanten des verdoppelten Restweges.

Zusätzlich definieren wir eine Menge von neuen Knoten $\tilde{V}^{p_k} := \{v_{q+1}^{p_k}, ..., v_{n-1}^{p_k}\}$, und eine Bijektion $h_V^{p_k} : \tilde{V}^{p_k} \to (e_{q+1}, ..., e_{n-1})$ durch $h_V^{p_k}(v_i^{p_k}) := e_i$, $i = q+1, ..., n-1$. Diese sollen die gesplitteten Endknoten der Kanten aus p_k darstellen.

Nachdem alle Wegeverbote betrachtet wurden fassen wir die neu definierten Elemente zusammen:

Definition von $G_1 := (V^1, E^1, \alpha^1, \omega^1, P)$:

$$\tilde{V}^1 := \bigcup_{p \in P} \tilde{V}^p \quad \tilde{E}^1 := \bigcup_{p \in P} \tilde{E}_1^p \quad V^1 := V \cup \tilde{V}^1 \quad E^1 := E \cup \tilde{E}^1$$

G_1 ist eine Erweiterung des Ausgangsgraphen mit gesplitteten Wegeverboten.

Definition der Inzidenzabbildungen:

Definition von $\omega^1 : E^1 \to V^1$ für jedes $p \in P$:

$$\omega^1(e) := \begin{cases} \omega(e) & \text{für alle } e \in E \backslash E_0 \\ (h_V^p)^{-1}(h_1^p(e)) & \text{falls } e = e_j^p \in \tilde{E}_1 \\ (h_V^p)^{-1}(e) & \text{falls } e = e_1 \text{ die erste Kante von } p \in E_0 \text{ ist.} \end{cases}$$

G ist kein Teilgraph von G^1, da für die Kanten e_0 aus E_0 gilt: $\omega^1(e_1) \notin V$ (formal müßte eigentlich eine Umbenennung erfolgen). Sonst bleibt die Inzidenz aber erhalten.

Definition von $\alpha^1 : E^1 \to V^1$: Für $e \in E$ definieren wir $\alpha^1(e) := \alpha(e)$. Dies gilt insbesondere für die nicht gesplitteten Kanten aus E_0. Sei nun das Wegeverbot $p := p_k = (e_1, ..., e_n)$, und sei $p' = (e'_1, ..., e'_m)$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten, dann definieren wir

$$\alpha^1(e_j^p) := \begin{cases} \omega^1(e_{j-1}^p) & \text{falls } j > q+1 \\ \omega^1(e_q^{p'}) & \text{falls } j = q+1 \end{cases}$$

Die Kanten e_j mit $j \leq q$ von p sind nicht gesplittet worden. Der verdoppelte Restweg von p wird also an den maximalen Anfangsweg p' angeheftet.

Mit diesen Definitionen gilt außerdem für $j = q, ..., n-1$:

$$\begin{aligned} \omega^1(e_j^p) &= (h_V^p)^{-1}(h_1^p(e_j^p)) && \text{Definition von } \omega^1 \\ &= (h_V^p)^{-1}(e_j) && \text{Definition von } e_j^p \\ &= v_j^p && \text{Definition von } v_j^p. \end{aligned} \tag{5.1}$$

Für e_j^p mit $q < j+1 < n$ gilt

$$\begin{aligned} \alpha^1(e_j^p) &= \omega^1(e_{j-1}^p) && \text{Definition von } \alpha^1 \\ &= v_{j-1}^p && \text{siehe oben,} \end{aligned}$$

und falls zusätzlich $p \neq p_1$ oder $q+1 \geq j$ ist, gilt

$$\begin{aligned} \alpha^1(e_{q+1}^p) &= \omega^1(e_q^{p'}) && \text{Definition von } \alpha^1 \\ &= v_q^{p'} && \text{siehe oben.} \end{aligned}$$

Für die erste Kante e_1 des ersten Wegeverbotes p_1 jeder Klasse gilt analog $\omega^1(e_1) = v_1^{p_1}$.

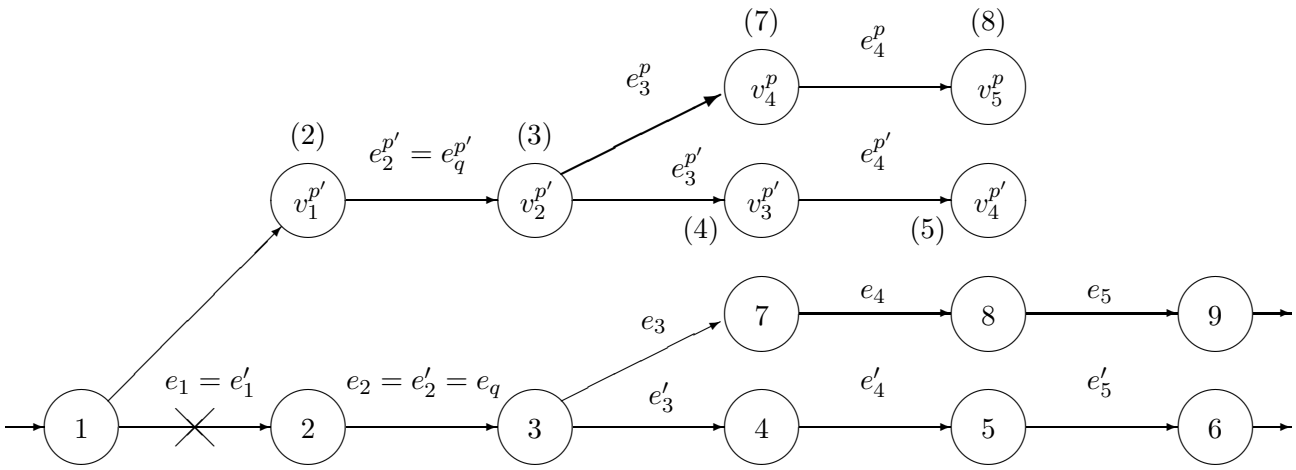


Abbildung 5.4 *Zweiter Schritt einer Wegeverdopplung*

Definition der kanonischen Projektionen Π

Wir definieren die Abbildung $\Pi^1_V : V^1 \to V$:

$$\Pi^1_V(v) := \begin{cases} v & \text{für alle } v \in V \\ \omega(h^p_V(v)) & \text{falls } v \in \tilde{V}^p \end{cases}$$

und die Abbildung $\Pi^1_E : E^1 \to E$:

$$\Pi^1_E(e) := \begin{cases} e & \text{für alle } e \in E \\ h^p_1(e) & \text{falls } e \in \tilde{E}^p_1. \end{cases}$$

Diese heißen kanonische Projektionen von G^1 nach G.

Mit dieser Definition gilt für $p := (e_1, ..., e_n)$ und dessen gesplitteten Weg $(e^p_{q+1}, ..., e^p_{n-1})$:

$$\Pi^1_E(e^p_i) \;\; = h^p_1(e^p_i) \;\; = e_i \tag{5.2}$$

Satz 5.1 *Seien alle Definitionen aus dem Abschnitt Wegeverdopplung gültig, dann gilt: Die kanonischen Projektionen sind Epimorphismen, das heißt surjektiv und inzidenzerhaltend. Es gilt also für alle* $e \in E^1$*:*

$$\omega(\Pi^1_E(e)) = \Pi^1_V(\omega^1(e)) \quad \textit{und} \quad \alpha(\Pi^1_E(e)) = \Pi^1_V(\alpha^1(e)).$$

Beweis:

$\omega^1(\Pi^1_E(e)) = \Pi^1_V(\omega^1(e))$:

Für $e \in E\backslash E_0$ gilt $\omega(e) \in V$ und damit gilt:

$$\begin{array}{lll} \omega(\Pi^1_E(e)) & = \omega(e) & \text{Definition von } \Pi^1_E \\ & = \Pi^1_V(\omega(e)) & \omega^1(e) \in V \\ & = \Pi^1_V(\omega^1(e)) & \text{Definition von } \omega^1 \text{ für } e \in E\backslash E_0. \end{array} \tag{5.3}$$

Sei $p = p_k := (e_1, ..., e_n) \in P$, $p' := (e'_1, ..., e'_m)$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten und sei $(e^p_{q+1}, ..., e^p_{n-1})$ der gesplittete Weg von p. Für $e^p_j \in \tilde{E}^p_1$ gilt $\omega(e^p_j) \in \tilde{V}^1$ und damit:

$$\begin{array}{lll} \Pi^1_V(\omega^1(e^p_j)) & = \Pi^1_V(v^p_j) & \text{nach Gleichung (5.1)} \\ & = \omega(h^p_V(v^p_j)) & \text{Def. von } \Pi^1_V \\ & = \omega(e_j) & \text{Def. von } h^p_V \\ & = \omega(\Pi^1_E(e^p_j)) & \text{wegen Gleichung (5.2).} \end{array}$$

Analog zeigt man die Beziehung für Kanten aus E_0.

$\alpha(\Pi^1_E(e)) = \Pi^1_V(\alpha^1(e))$:

Der Beweis $\alpha^1(\Pi^1_E(e)) = \Pi^1_V(\alpha^1(e))$ für $e \in E$ (und $e \in E_0$) verläuft analog zu Gleichung (5.3). Für $e^p_j \in \tilde{E}^p_1$ und $q + 1 < j < n$ gilt

$$\begin{array}{lll} \Pi^1_V(\alpha^1(e^p_j)) & = \Pi^1_V(v^p_{j-1}) & \text{nach Gleichung hinter (5.1)} \\ & = \omega(h^p_V(v^p_{j-1})) & \text{Def. von } \Pi^1_V \\ & = \omega(e_{j-1}) & \text{Def. von } h^p_V \\ & = \alpha(e_j) & p \text{ ist ein Weg} \\ & = \alpha(\Pi^1_E(e^p_j)) & \text{wegen Gleichung (5.2).} \end{array}$$

und für die Verbindungskante $e^p_{q+1} \in \tilde{E}^p_1$ gilt

$$\begin{array}{lll} \Pi^1_V(\alpha^1(e^p_{q+1})) & = \Pi^1_V(v^{p'}_q) & \text{nach Gleichung zwei hinter (5.1)} \\ & = \omega(h^{p'}_V(v^{p'}_q)) & \text{Def. von } \Pi^1_V \\ & = \omega(e'_q) & \text{Def. von } h^{p'}_V \\ & = \omega(e_q) & e_q = e'_q \text{ (Def. des maximalen Anfangsweges)} \\ & = \alpha(e_{q+1}) & p \text{ ist ein Weg} \\ & = \alpha(\Pi^1_E(e^p_{q+1})) & \text{wegen Gleichung (5.2).} \end{array}$$

■

Die Abbildungen

$$\iota^1_V : \quad V \to V^1 \qquad \iota^1_V(v) := v \in V^1$$

und

$$\iota^1_E : \quad E \to E^1 \qquad \iota^1_E(e) := e \in E^1$$

heißen *kanonische Injektionen* von G nach G^1. Wenn durch das Argument klar ist, ob ι_V^1 oder ι_E^1 gemeint ist, so schreiben wir stattdessen auch $\iota^1(v)$ bzw. $\iota^1(e)$.
Die Abbildungen ι_V^1 und ι_E^1 sind wohldefiniert, injektiv, aber nicht surjektiv. Sie sind auch nicht inzidenzerhaltend, da für e_0 aus E_0 $\omega^1(\iota_E^1(e_0)) \notin V$, aber $\iota_V^1(\omega(e_0)) \in V$ ist. Sonst bleibt die Inzidenz aber erhalten.

Für $e \in E$ gilt:

$$\begin{array}{lll} \Pi_E^1(\iota_E^1(e)) & = \Pi_E^1(e) & \text{Definition von } \iota_E^1 \\ & = e & \text{Definition von } \Pi_E^1 \end{array} \tag{5.4}$$

Sei $p := p_k = (e_1, ..., e_n) \in P$, $(e_{q+1}^p, ..., e_{n-1}^p)$ dessen gesplitteter Weg, und sei $p' = (e_1', ..., e_m') \in P$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten. Der Verlauf dieses Wegeverbotes im erweiterten Graphen G^1 wird durch die folgende Abbildung $\psi^p : (e_1, ..., e_{n-1}) \to E^1$ dargestellt:

$$\psi^p(e_i) := \begin{cases} \psi^{p'}(e_i') & \text{für } 1 \le i \le q \\ (h_1^p)^{-1}(e_i) = e_i^p & \text{für } q+1 \le i < n. \end{cases}$$

Für das erste Wegeverbot p_1 jeder Klasse gelten diese Definitionen ebenfalls, wenn man gleichzeitig $q := 0$ und $\psi^{p_1}(e_1) := \iota_E^1(e_1)$ definiert. Insbesondere folgt aus der Definition des maximalen Anfangsweges

$$\psi^p(e_q) \;=\; \psi^{p'}(e_q') \;=\; e_q^{p'}. \tag{5.5}$$

Insbesondere gilt

$$\omega^1(\psi^p(e_i)) = v_j^{p''} \Longrightarrow i = j \text{ und } p = p'' \text{ nach Definition von } \psi^p.$$

Satz 5.2 *Sei $p := p_k = (e_1, ..., e_n) \in P$, $(e_{q+1}^p, ..., e_{n-1}^p)$ dessen gesplitteter Weg und sei $p' = (e_1', ..., e_m') \in P$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten, dann gilt: $(\psi^p(e_1), , ..., \psi^p(e_n))$ ist ein Weg und*

$$\Pi_E^1(\psi^p(e_j)) = e_j \text{ für } 1 \le j < n \tag{5.6}$$

Beweis:

Für die Wegeigenschaft müssen wir $\omega^1(\psi^p(e_j)) = \alpha^1(\psi^p(e_{j+1}^p))$ zeigen. Diesen Beweis führen wir mit vollständiger Induktion über die Wegeverbote jeder Klasse.

Induktionsanfang: Sei $k = 1$ damit betrachten wir das erste Wegeverbot p_1, und sei e_j fest gewählt, dann gilt:

$$\begin{array}{lll} \omega^1(\psi^{p_1}(e_j)) & = \omega^1(\iota_E^1(e_j)) & \\ & = \omega^1(e_j^{p_1}) & \text{Def. von } \psi^{p_1} \\ & = v_j^{p_1} & \text{nach Gleichung (5.1)} \\ & = \alpha^1(e_{j+1}^{p_1}) & \text{nach Gleichung (5.1)} \\ & = \alpha^1(\psi^{p_1}(e_{j+1})) & \text{Def. von } \psi^{p_1}. \end{array}$$

Induktionsschritt: Sei die Eigenschaft bereits für p_i $1 \leq i \leq k-1$ gezeigt worden, dann gilt sie auch für p_k:

Zu zeigen ist nur noch $\omega^1(\psi^p(e_q)) = \alpha^1(\psi^p(e_{q+1}))$, denn die Wegeigenschaft der davorliegenden Kanten folgt aus der Induktionsvoraussetzung; und die Wegeigenschaft der nachfolgenden Kanten wird analog zum Induktionsanfang gezeigt.

$$\begin{array}{lll} \omega^1(\psi^p(e_q)) & = \omega^1(e_q^{p'}) & \text{nach Gleichung (5.5)} \\ & = \alpha^1(e_{q+1}^p) & \text{Def. von } \alpha^1 \\ & = \alpha^1(\psi^p(e_{q+1})) & \text{Def. von } \psi^p. \end{array}$$

Die Gleichung (5.6) beweisen wir ebenfalls mit vollständiger Induktion über die Wegeverbote jeder Klasse:

Induktionsanfang: Sei $k = 1$ und sei e_j fest gewählt, dann gilt

$$\begin{array}{lll} \Pi_E^1(\psi^p(e_j)) & = \Pi_E^1((h_1^p)^{-1}(e_j)) & \text{Def. von } \psi^p \\ & = h_1^p((h_1^p)^{-1}(e_j)) & \text{Def. von } \Pi_E^1 \\ & = e_j & h_1^p \text{ ist Bijektion} \end{array}$$

Induktionsschritt: Sei die Gleichung (5.6) bereits für p_i $1 \leq i \leq k-1$ gezeigt worden, dann gilt sie auch für p:

Wie im vorigen Beweis ist die Eigenschaft nur noch für $j = q$ zu zeigen, denn für Kanten mit kleinerem Index folgt sie aus der Induktionsvoraussetzung, und für Kanten mit größerem Index beweist man sie analog zum Induktionsanfang.

$$\begin{array}{lll} \Pi_E^1(\psi^p(e_q)) & = \Pi_E^1(e_q^{p'}) & \text{nach Gleichung (5.5)} \\ & = \Pi_E^1((h_1^{p'})^{-1}(e'_j)) & \text{Definition von } h_1^{p'} \ \ (j = q) \\ & = h_1^{p'}((h_1^{p'})^{-1}(e'_j)) & \text{Def. von } \Pi_E^1 \\ & = e'_j & h_1^{p'} \text{ ist Bijektion} \\ & = e_j & \text{Definition des maximalen Anfangsweges} \end{array}$$

■

Satz 5.3 *Sei $p := p_k = (e_1, ..., e_n) \in P$, $(e_{q+1}^p, ..., e_{n-1}^p)$ dessen gesplitteter Weg. Zu festem $1 \leq j < n$ sei $\hat{p} := p_{\hat{k}} = (\hat{e}_1, ..., \hat{e}_m)$ ein Wegeverbot mit der Eigenschaft $\hat{e}_i = e_i$ für $1 \leq i \leq j$ und $k \leq \hat{k}$, dann gilt:*

$$\psi^p(e_i) = \psi^{\hat{p}}(\hat{e}_i) \text{ für } 1 \leq i \leq j.$$

Beweis:

Um diesen Satz zu beweisen, benötigen wir eine Induktion über den Index j.

Induktionsanfang: Sei $j = 1$, dann gilt: $p_1 \in [\![p]\!]$ ist maximaler Anfangsweg der einkantigen Wege e_1 und $\hat{e}_1$. Nach Definition von ψ gilt für $i = j = 1$:

$$\psi^{\hat{p}}(\hat{e}_1) = e_1^{p_1} = \psi^p(e_1).$$

Induktionsschritt: $j \to j+1$:

Sei $\psi^p(e_i) = \psi^{\hat{p}}(e_i)$ für $1 \leq i \leq j$. Sei $p' = (e'_1, ..., e'_m)$ der maximale Anfangsweg von $(e_1, ..., e_{j+1})$, dann gilt: p' ist auch maximaler Anfangsweg von $e_1^{\hat{p}}, ..., e_{j+1}^{\hat{p}}$. Damit gilt:

$$\begin{aligned} \psi^{\hat{p}}(\hat{e}_{j+1}) &= e_{j+1}^{p'} && \text{da } p' \text{ maximaler Anfangsweg} \\ &= \psi^p(e_{j+1}) && \text{Definition von } \psi^p. \end{aligned}$$

Für $i \leq j$ folgt die Behauptung aus der Induktionsvoraussetzung, da der maximale Anfangsweg von $(e_1, ..., e_i)$ unabhängig von j ist. ∎

Definition 5.2 *Sei $p := (e_1, ..., e_n) \in P$ und die Wegeverdopplung sei bereits durchgeführt. Wir definieren*

$$\tilde{V}^{[p]} := \bigcup_{p \in [p]} \tilde{V}^p \quad \textit{und} \quad \tilde{E}^{[p]} := \bigcup_{p \in [p]} \tilde{E}^p,$$

dann ist $G^{[p]} = (\tilde{V}^{[p]}, \tilde{E}^{[p]}, \alpha^1, \omega^1)$ ein Baum mit Wurzel $\alpha^1(e_1)$, wenn der Definitionsbereich der Inzidenzabbildungen auf $\tilde{E}^{[p]}$ eingeschränkt wird. $G^{[p]}$ ist ein Teilgraph von G^1 und heißt der von der Klasse $[\![p]\!]$ erzeugte **Wegebaum**.

5.4 Die Verbindungen der gesplitteten Wege mit dem Ausgangsgraphen

In diesem Abschnitt werden alle Kanten definiert, über welche die Wegebäume ohne Beachtung weiterer Wegeverbote wieder verlassen werden können.

Sei $p := p_k = (e_1, ..., e_n) \in P$, $(e_{q+1}^p, ..., e_{n-1}^p)$ dessen gesplitteter Weg und sei $p' = (e'_1, ..., e'_m) \in P$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten. Wir definieren für jedes $q < j < n$ eine Menge N_j^p als Menge aller Kanten, in welche von der Kante e_j aus ohne Auswirkung von irgendeinem Wegeverbot abgebogen werden darf. e aus E wird in N_j^p aufgenommen genau dann, wenn gilt:

$$\begin{aligned} &(a) \quad \alpha(e) = \omega(e_j) \\ &(b) \quad \text{Es gibt kein Wegeverbot } \hat{p} \text{ mit } \hat{p} \text{ ist Endweg von } w := e_1 + ... + e_j + e \text{ in } P. \end{aligned} \tag{5.7}$$

Insbesondere gilt damit $e_{j+1} \notin N_j^p$, da das Wegeverbot p Endweg von $(e_1, ..., e_j, e_{j+1})$ ist. Es gilt $N_j^p \cap E_0 = \emptyset$, denn wenn für die Anfangskante $\hat{e}_1$ von $\hat{p}$ $\hat{e}_1 \in N_j^p$ gelten würde, dann wäre $\hat{p}$ ein in (5.7b) beschriebener Endweg. Zum Wegeverbot p definieren wir

$$N^p := \Big(\bigcup_{j=q+1}^{n-1} \Big(\bigcup_{e \in N_j^p} (e_j^p, e) \Big) \Big) \tag{5.8}$$

und insgesamt sei

$$N = \bigcup_{p \in P} N^p.$$

Die Elemente der Menge N sind Kantenpaare, die in der zweiten Komponente die Menge aller Kanten beinhalten, über welche der verbotene Weg p ohne Beachtung anderer Wegeverbote vorzeitig verlassen werden kann. In der ersten Komponente steht die (gesplittete) letzte Kante, von welcher aus der verbotene Weg verlassen wird. Diese Komponente ist z.B. notwendig, wenn sich das Wegeverbot selbst kreuzt.

Wir definieren eine Menge von neuen Kanten $\tilde{E}^2$ mit $\exists$ eine Bijektion $h_2 : \tilde{E}^2 \to N$. Sei (e_i^p, e) ein Element aus N, so können die Elemente aus $\tilde{E}^2$ als $e_{i,e}^p$ geschrieben werden, wobei die Bijektion h_2 folgendermaßen definiert ist:

$$h_2(e_{i,e}^p) := (e_i^p, e).$$

Wir erweitern G^1 zu G^2:

$$V^2 := V^1 \qquad E^2 := \tilde{E}^2 \cup E^1.$$

Definition der Inzidenzabbildungen:

Definition von $\omega^2 : E^2 \to V^2$:

$$\omega^2(e) := \begin{cases} \omega^1(e) & \text{für alle } e \in E^1 \\ \omega(\pi_2(h_2(e))) & \text{falls } e \in \tilde{E}^2 \end{cases} \tag{5.9}$$

Definition von $\alpha^2 : E^2 \to V^2$:

$$\alpha^2(e) := \begin{cases} \alpha^1(e) & \text{für alle } e \in E^1 \\ \omega^1(\pi_1(h_2(e))) & \text{falls } e \in \tilde{E}^2 \end{cases}$$

Damit gilt für $e_{i,e}^p \in \tilde{E}^2$:

$$\begin{array}{lll} \alpha^2(e_{i,e}^p) & = \omega^1(\pi_1(h_2(e_{i,e}^p))) & \text{Definition von } \alpha^2 \\ & = \omega^1(\pi_1(e_i^p, e)) & \text{Definition von } h_2 \\ & = v_i^p & \text{nach Gleichung (5.1).} \end{array} \tag{5.10}$$

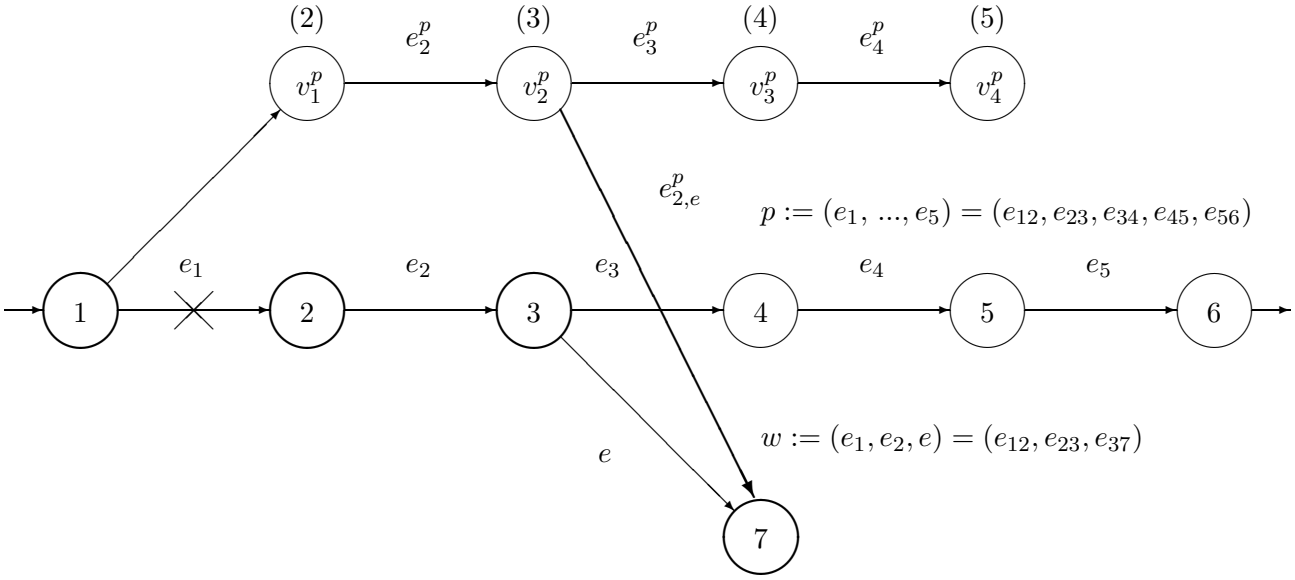


Abbildung 5.5 *Eine Verbindung eines gesplitteten Weges mit dem Ausgangsgraphen*

Die kanonische Projektion wird um die neuen Kanten erweitert: $\Pi_E^2 : E^2 \to E$:

$$\Pi_E^2(e) := \begin{cases} \Pi_E^1(e) & \text{falls } e \in E^1 \\ \pi_2(h_2(e)) & \text{falls } e \in \tilde{E}^2. \end{cases}$$

Die kanonische Projektion der Knoten bleibt unverändert, da keine neuen Knoten definiert wurden. Die Definition der kanonischen Injektionen ι_V^2, ι_E^2 und die Gleichung (5.4) werden aus dem ersten Schritt übertragen.

Damit gilt für $e_{i,e}^p \in \tilde{E}^2$:

$$\Pi_E^2(e_{i,e}^p) = \pi_2(h_2(e_{i,e}^p)) = \pi_2(e_i^p, e) = e \tag{5.11}$$

Mit diesen Definitionen bleibt die kanonische Projektion Π_E^2 inzidenzerhaltend - Satz 5.1 kann also auf Elemente von E^2 erweitert werden:

Für $e_{i,e}^p \in \tilde{E}^2$ gilt analog zur Gleichung (5.3):

$$\begin{array}{lll} \omega(\Pi_E^2(e_{i,e}^p)) & = \omega(e) & \text{nach Gleichung (5.11)} \\ & = \Pi_V^1(\omega(e)) & \omega(e) \in V, \text{ da } e \in E \backslash E_0 \\ & = \Pi_V^1(\omega(\pi_2(h_2(e_{i,e}^p)))) & \text{Definition von } h_2 \\ & = \Pi_V^1(\omega^2(e_{i,e}^p)) & \text{Definition von } \omega^2 \end{array}$$

und für deren Anfangsknoten gilt

$$\begin{aligned}
\Pi_V^1(\alpha^2(e_{i,e}^p)) &= \Pi_V^1(\omega^1(e_i^p)) && \text{nach Gleichung (5.10)}\\
&= \omega(\Pi_E^1(e_i^p)) && \text{nach Satz 5.1}\\
&= \omega(e_i) && \text{nach Gleichung (5.2)}\\
&= \alpha(e) && \text{nach Bedingung (5.7a)}\\
&= \alpha(\Pi_E^2(e_{i,e}^p)) && \text{nach Gleichung (5.11)}
\end{aligned}$$

5.5 Die Verbindungen der gesplitteten Wege untereinander

In diesem Abschnitt werden alle Kanten definiert, über welche die Wegebäume verlassen werden können, wenn zusätzlich noch andere Wegeverbote beachtet werden. Siehe zu diesem Abschnitt auch Beispiel B.10.

Sei $p := p_k = (e_1, ..., e_n) \in P$, $(e_{q+1}^p, ..., e_{n-1}^p)$ dessen gesplitteter Weg und sei $p' = (e_1', ..., e_m') \in P$ der maximale Anfangsweg von p in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten. Wir definieren für jedes $q < j < n$ eine Menge M_j^p als Menge aller Kanten, in welche von der Kante e_j aus bedingt (mit Auswirkung von Wegeverboten ($\notin [\![p]\!]$)) abgebogen werden darf. Wir betrachten alle Kanten e aus E, für die gilt:

$$\begin{aligned}
&(a) \quad \alpha(e) = \omega(e_j)\\
&(b) \quad \text{Es gibt Wegeverbote } p'' \notin [\![p]\!], \text{ die Endweg von } w := e_1 + ... + e_j + e \text{ in } P \text{ sind,}\\
&\qquad \text{und sei } \check{p} \text{ der maximale dieser Endwege mit } \check{e}_t = e.
\end{aligned} \tag{5.12}$$

Zu der Kante $e = \check{e}_t$ existiert eine im Abschnitt 5.3 definierte Kante $e_t^{\check{p}}$ mit $h_1^{\check{p}}(e_t^{\check{p}}) = \check{e}_t$. Diese Kante wird in M_j^p aufgenommen.

Bemerkung: Der Fall $\check{p} \in [\![p]\!]$ ist bereits im Abschnitt 5.3 betrachtet worden.

Zum Wegeverbot p definieren wir

$$M^p := \Big(\bigcup_{j=q+1}^{n-1} \Big(\bigcup_{e_t^{\check{p}} \in M_j^p} (e_j^p, e_t^{\check{p}}) \Big) \Big)$$

und insgesamt sei

$$M = \bigcup_{p \in P} M^p.$$

Ist $[\![p]\!] \neq [\![\check{p}]\!]$ oder $t < i + 1 < l$, dann muß die Kante $e = e_t^{\check{p}}$ so gesplittet werden, daß sie die Wegeverbote p und $\check{p}$ verbindet. Der Fall der Selbstkreuzung $p = \check{p}$ ist hier nicht ausgeschlossen. Das Verbinden des Weges p mit dem maximalen Endweg $\check{p}$ ist eine der zentralen Ideen des Algorithmus. Wir definieren eine Menge von neuen Kanten $\tilde{E}_3$ mit $\exists$ eine Bijektion $h_3 : \tilde{E}_3 \to M$ und

$$h_3(e_{i,t}^{p,\check{p}}) := (e_i^p, e_t^{\check{p}}).$$

Wir erweitern G^2 zu G^3:

$$V^3 := V^2 \qquad E^3 := \tilde{E}^3 \cup E^2.$$

Definition der Inzidenzabbildungen:

Definition von $\omega^3 : E^3 \to V^3$:

$$\omega^3(e) := \begin{cases} \omega^2(e) & \text{falls } e \in E^2 \\ \omega^2(\pi_2(h_3(e))) & \text{falls } e \in \tilde{E}^3 \end{cases}$$

Definition von $\alpha^3 : E^3 \to V^3$:

$$\alpha^3(e) := \begin{cases} \alpha^2(e) & \text{falls } e \in E^2 \\ \omega^2(\pi_1(h_3(e))) & \text{falls } e \in \tilde{E}^3. \end{cases}$$

Mit dieser Definition gilt

$$\alpha^3(e_{i,t}^{p,\check{p}}) = v_i^p \quad \text{und} \quad \omega^3(e_{i,t}^{p,\check{p}}) = v_t^{\check{p}}. \tag{5.13}$$

Die kanonische Projektion wird um die neuen Kanten erweitert: $\Pi_E^3 : E^3 \to E$:

$$\Pi_E^3(e) := \begin{cases} \Pi_E^2(e) & \text{falls } e \in E^2 \\ \Pi_E^1(\pi_2(h_3(e))) & \text{falls } e \in \tilde{E}^3. \end{cases}$$

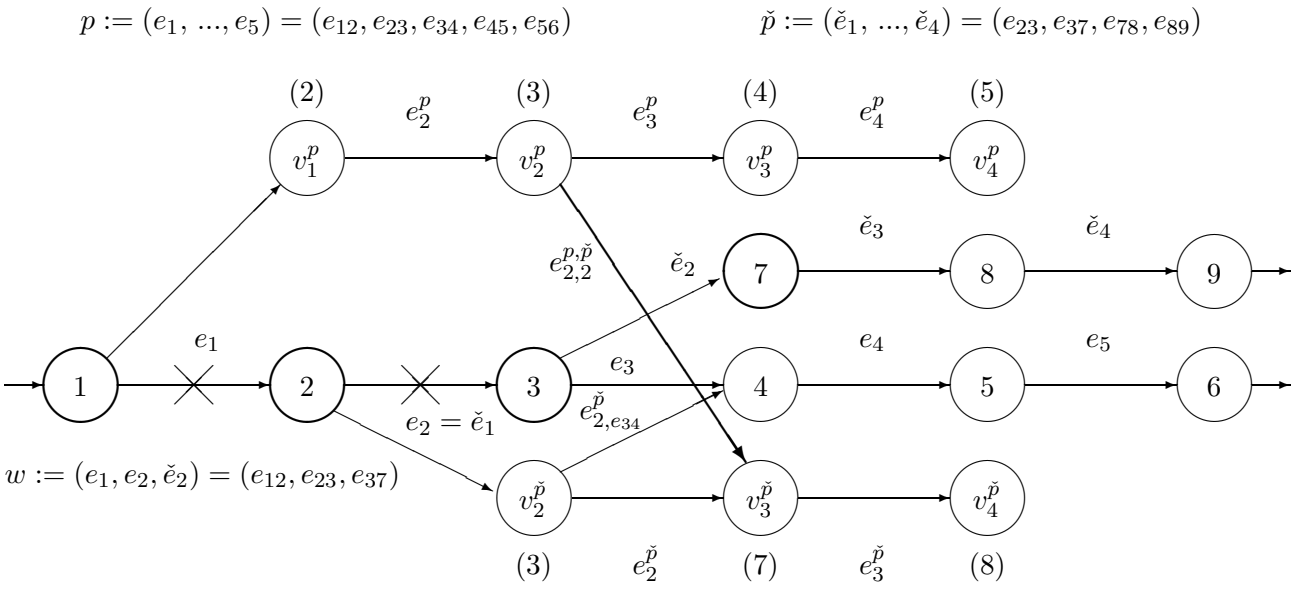


Abbildung 5.6 *Eine Verbindung zweier gesplitteter Wege*

Analog zum letzten Schritt bleibt die kanonische Projektion der Knoten unverändert, da keine neuen Knoten definiert wurden. Die Definition der kanonischen Injektionen ι_V^3, ι_E^3 und die Gleichung (5.4) können wörtlich aus den vorherigen Schritten übernommen werden. Damit gilt für $e_{i,t}^{p,\check{p}} \in \tilde{E}^3$:

$$\Pi_E^3(e_{i,t}^{p,\check{p}}) = \Pi_E^1(e_t^{\check{p}}) = \check{e}_t. \tag{5.14}$$

Also wurde tatsächlich die Kante $\check{e}_t$ gesplittet.

Wir erweitern Satz 5.1 auf Elemente von E^3: Sei $e_{i,t}^{p,\check{p}} \in \tilde{E}^3$, dann gilt:

$$\begin{aligned}
\Pi_V^1(\alpha^3(e_{i,t}^{p,\check{p}})) &= \Pi_V^1(\omega^1(e_i^p)) && \text{Nach Gleichung (5.13)}\\
&= \omega(\Pi_E^1(e_i^p)) && \text{Bew. der Inzidenztreue für}\\
& && \text{Elemente aus } \tilde{E}^1 \text{ Satz 5.1}\\
&= \omega(e_i) && \text{Nach Gleichung (5.2)}\\
&= \alpha(\check{e}_t) && \text{Def.von } M\text{(Bedingung 5.12}a\text{)}\\
&= \alpha(\Pi_E^3(e_{i,t}^{p,\check{p}})) && \text{Nach Gleichung (5.14)}
\end{aligned}$$

$$\begin{aligned}
\Pi_V^1(\omega^3(e_{i,t}^{p,\check{p}})) &= \Pi_V^1(\omega^1(e_t^{\check{p}})) && \text{Nach Gleichung (5.13)}\\
&= \omega(\Pi_E^1(e_t^{\check{p}})) && \text{Bew. der Inzidenztreue für}\\
& && \text{Elemente aus } \tilde{E}^1\\
&= \omega(\check{e}_t) && \text{Nach Gleichung (5.2)}\\
&= \omega(\Pi_E^3(e_{i,t}^{p,\check{p}})) && \text{Nach Gleichung (5.14)}
\end{aligned}$$

Die Abbildung Π_V^1 sowie die Zusammensetzung der Abbildungen Π_E^1, Π_E^2 und Π_E^3 sind die geforderten Abbildungen Φ und Ψ aus Definition 2.12.

Insgesamt ergeben sich also die folgenden neuen Mengen mit ihren Bijektionen:

$$\tilde{E}_1^p \xrightarrow{h_1^p} e_{q+1}, ..., e_{n-1} \qquad \tilde{E}^2 \xrightarrow{h_2} N \qquad \tilde{E}^3 \xrightarrow{h_3} M \qquad \tilde{V}^p \xrightarrow{h_V^p} e_{q+1}, ..., e_{n-1}.$$

Zusammengefaßt lauten die neuen Inzidenzstrukturen:

$$\alpha^3(e) := \begin{cases}
\alpha(e) & \text{falls} \quad e \in E & \text{Originalgraph}\\
v_{i-1}^p & \text{falls} \quad e = e_i^p & \text{erster Schritt der Wegeverdopplung}\\
v_q^{p'} & \text{falls} \quad e = e_{q+1}^p & k.\ \text{Schritt der Wegeverdopplung}\\
v_i^{\hat{p}} & \text{falls} \quad e = e_{i,\hat{e}}^{\hat{p}} & \text{Verbindung mit dem Ausgangsgraphen}\\
v_i^p & \text{falls} \quad e = e_{i,t}^{p,\check{p}} & \text{Verbindung der Wege}
\end{cases}$$

$$\omega^3(e) := \begin{cases}
\alpha(e) & \text{falls} \quad e \in E \backslash E_0 & \text{Originalgraph}\\
v_1^p & \text{falls} \quad e = e_1 \in E_0 & \text{erster Schritt der Wegeverdopplung}\\
v_i^p & \text{falls} \quad e = e_i^p & \text{erster Schritt der Wegeverdopplung}\\
\omega(\hat{e}) & \text{falls} \quad e = e_{i,\hat{e}}^{p} & \text{Verbindung mit dem Ausgangsgraphen}\\
v_t^{\check{p}} & \text{falls} \quad e = e_{i,t}^{p,\check{p}} & \text{Verbindung der Wege}
\end{cases}$$

Und die kanonischen Projektionen:

$$\Pi_E^3(e) := \begin{cases} e & \text{falls} \quad e \in E \\ e_i & \text{falls} \quad e = e_i^p \\ \hat{e} & \text{falls} \quad e = e_{i,\hat{e}}^p \\ \check{e}_t & \text{falls} \quad e = e_{i,t}^{p,\check{p}} \end{cases} \qquad \Pi_v(v) := \begin{cases} v & \text{falls} \quad v \in V \\ \omega(e_i) & \text{falls} \quad v = v_i^p \end{cases}$$

Bemerkung: Nach Konstruktion ist die Erweiterung G^3 unabhängig von der Reihenfolge der Wegeverbotsklassen. Wir werden im Satz 5.6 zeigen, daß die Erweiterung auch unabhängig von der Reihenfolge der einzelnen Wegeverbote einer Klasse ist.

5.6 Erläuterungen zum Algorithmus

In diesem Abschnitt soll die Grundidee des Algorithmus zur Wegegrapherweiterung erläutert werden. Dazu denken wir uns ein Fahrzeug, das sich in unserem Ausgangsgraphen mit Wegeverboten bewegt. In diesem Abschnitt werden vier Beispiele diskutiert. Weitere Beispiele befinden sich im Abschnitt B.2.

5.6.1 Beispiel mit einem Wegeverbot

Betrachten wir zunächst folgenden Beispielgraphen mit nur einem Wegeverbot: Gegeben sei ein Graph $G = (V, E, \alpha, \omega)$ mit $V = \{v_1, ..., v_5\}$, $E = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{34}, e_{43}, e_{45}, e_{54}\}$, mit dem Wegeverbot $p := (e_{12}, e_{23}, e_{34}, e_{45})$ und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$, $\omega(e_{ij}) = v_j$.

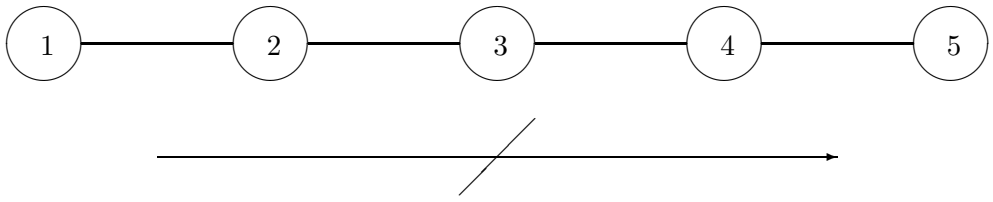


Abbildung 5.7 *Einfacher Beispielgraph mit nur einem Wegeverbot*

Es ist also darauf zu achten, daß e_{45} nicht durchfahren wird, wenn zuvor die Kanten e_{12}, e_{23} und e_{34} durchfahren wurden. Dazu konstruieren wir zusätzlich eine Art Umleitung am Knoten v_1 in Richtung $v_{2'}$(= Duplikat von v_2). Diese beinhaltet den gesamten Weg p außer der ersten und letzten Kante (siehe Abschnitt 5.3). Die erste Kante wird nicht benötigt, da der Kante e_{12} ein neuer Endknoten (Umleitung zum verdoppelten Weg p) zugewiesen wird. Das Weglassen der

letzten Kante (e_{45}) stellt das eigentliche Wegeverbot dar. Der verdoppelte Weg kann (zunächst) nur vom Knoten v_1 aus erreicht werden - er muß aber befahren werden, wenn v_1 in Richtung v_2 verlassen wird. e_{12} ist im Beispiel die einzige Kante, für die ab dem ersten Erweiterungsschritt $\omega(e_{ij}) \neq v_j$ gilt.

Folgende Mengen werden im Abschnitt 5.3 neu definiert:

$$\tilde{V}^p = \{v_1^p, v_2^p, v_3^p\} = \{v_{2'}, v_{3'}, v_{4'}\} \quad \text{und} \quad \tilde{E}_1^p = \{e_2^p, e_3^p\} = \{e_{2'3'}, e_{3'4'}\}$$

Wir betrachten nun die induzierten Abbildungen. Im Abschnitt 5.3 werden die Kanten von p mit $(e_1, e_2, e_3, e_4)[=(e_{12}, e_{23}, e_{34}, e_{45})]$ bezeichnet. Die Kanten (oder Wege) in eckigen Klammern sagen, welche Kanten des Originalgraphen gemeint sind. Da aber häufig die Indizierung wichtig ist, können die Kanten nicht direkt gleichgesetzt werden.

$$\begin{array}{ll}
h_V^p(v_{2'}) = h_V^p(v_1^p) = e_1[= e_{12}] & h_V^p(v_{3'}) = h_V^p(v_2^p) = e_2[= e_{23}] \\
h_V^p(v_{4'}) = h_V^p(v_3^p) = e_3[= e_{34}] & \\
h_1^p(e_{2'3'}) = h_1^p(e_2^p) = e_2[= e_{23}] & h_1^p(e_{3'4'}) = h_1^p(e_3^p) = e_3[= e_{34}] \\
 & \\
\Pi_V^1(v_{2'}) = \omega(h_V^p(v_1^p)) = \omega(e_{12}) = v_2 & \Pi_V^1(v_{3'}) = \omega(h_V^p(v_2^p)) = \omega(e_{23}) = v_3 \\
\Pi_V^1(v_{4'}) = \omega(h_V^p(v_3^p)) = \omega(e_{34}) = v_4 & \\
\Pi_E^1(e_{2'3'}) = h_1^p(e_2^p) = e_{23} & \Pi_E^1(e_{3'4'}) = h_1^p(e_3^p) = e_{34}
\end{array}$$

v_i^p ist der i-te gesplittete Knoten des Wegeverbotes p - e_i^p ist die i-te gesplittete Kante des Wegeverbotes p.

Die Umleitung muß auf jedem erlaubten Weg wieder verlassen werden können. Deshalb werden alle Kanten e, über die es erlaubt ist, den Weg p wieder zu verlassen (Menge N), benötigt (siehe Abschnitt 5.4). Die Elemente der Menge N sind Kantenpaare der Form (e_i^p, e). Die erste Komponente enthält die Kante e_i^p des erweiterten Wegeverbotes p ($e_i = i$-te Kante von p), über die die erlaubte Kante e erreicht wird. Diese ist in jedem Falle eindeutig, da in Abschnitt 5.3 ein Baum (im hier betrachteten Fall ohne Verzweigung) erzeugt wurde (siehe Definition 5.2). Wenn eine Kante in einem Wegeverbot mehrfach erscheint, so wird dies durch verschiedene Indizes (i) unterschieden. Die von (e_i^p, e) erzeugten neuen Kanten (Menge $\tilde{E}^2$) heißen $e_{i,e}^p$. $e_{i,e}^p$ ist die Kante, welche die i-te Kante e_i^p der Umleitung (Erweiterung) des Wegeverbotes p mit dem Ausgangsgraphen verbindet ($\omega^2(e_{i,e}^p) := \omega(e)$ siehe Formel 5.9).

$$\begin{array}{lll}
N = N^p & = \{(e_1, e_{21}), (e_2^p, e_{32}), (e_3^p, e_{43})\} & = \{(e_{12}, e_{21}), (e_{2'3'}, e_{32}), (e_{3'4'}, e_{43})\} \\
\tilde{E}^2 & = \{e_{1,e_{21}}^p, e_{2,e_{32}}^p, e_{3,e_{43}}^p\} & = \{e_{2'1}, e_{3'2}, e_{4'3}\} \\
 & & \\
h_2(e_{1,e_{21}}^p) & = (e_1^p, e_{21})[= (e_{12}, e_{21})] & h_2(e_{2,e_{32}}^p) = (e_2^p, e_{32})[= (e_{23}, e_{32})] \\
h_2(e_{3,e_{43}}^p) & = (e_3^p, e_{34})[= (e_{34}, e_{43})] & \\
 & & \\
\Pi_E^2(e_{2'1}) & = \Pi_E^2(e_{1,e_{21}}^p) = \pi_2(h_2(e_{1,e_{21}}^p)) & = \pi_2(e_1^p, e_{21}) = e_{21} \\
\Pi_E^2(e_{2,e_{32}}^p) & = e_{32} & \Pi_E^2(e_{3,e_{43}}^p) = e_{43}
\end{array}$$

Um vom Knoten v_1 zum Knoten v_5 zu gelangen, muß also an mindestens einem der Knoten $v_{2'}, v_{3'}$ oder $v_{4'}$ gewendet werden:

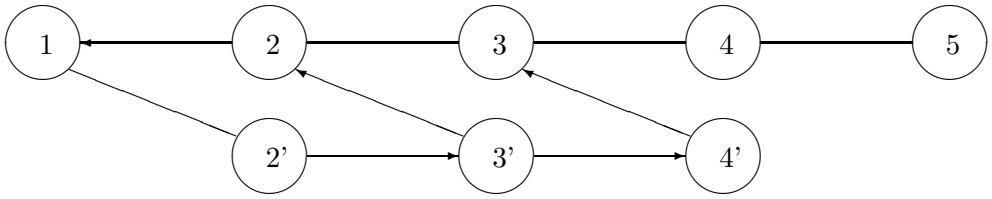


Abbildung 5.8 *Wegegraph des Beispielgraphen*

Die Abbildung macht deutlich, daß v_5 nur von v_1 aus nicht direkt erreichbar ist.

Die kanonischen Projektionen Π_v^i und Π_e^i werden benötigt, um den Verlauf eines Weges, der im erweiterten Graphen berechnet wurde, im Ausgangsgraphen verfolgen zu können. Die Projektionen sind wie in Definition 2.12 gefordert surjektiv und inzidenzerhaltend.

Weitere einfache Beispiele mit einem Wegeverbot sind die Beispiele B.7 und B.8.

5.6.2 Beispiel mit zwei Wegeverboten der gleichen Verbotsklasse

Sei nun beim Beispielgraph aus Abschnitt 5.6.1 noch ein weiteres Wegeverbot gegeben:

$$p' = (e_{12}, e_{23}, e_{32}, e_{21}) = (e'_1, e'_2, e'_3, e'_4) \quad \text{und } P := \{p, p'\}.$$

Hier gilt $p' \sim p$, weil p und p' die gleiche Anfangskante haben ($e_1^{p'} = e_{12} = e_1^p$ siehe Formel 2.3). Beide Wegeverbote verzweigen sich am Knoten v_3. Beide Wege könnten ab diesen Knoten getrennt betrachtet werden. Dabei ist zu beachten, daß $(e_2^p, e_{32}) \notin N^p$ und $(e_2^{p'}, e_{34}) \notin N^{p'}$ gilt, denn p' ist Anfangsweg von p (p und p' haben die gleiche Anfangskante, verlaufen ein Stück gleich und verzweigen sich dann - siehe Definition 2.10) und p ist Anfangsweg von p'. Damit ist Bedingung 5.7*b* nicht erfüllt.

Die beiden Anfänge der Wege können bis zum Knoten v_3 zusammengefaßt werden, da sie bis zum Knoten v_3 gleich verlaufen und bis zu diesem Knoten auf den gleichen Kanten verlassen werden dürfen. Dies geschieht im 2-ten ($k = 2$) Schritt der Wegeverdopplung aus Abschnitt 5.3. In der folgenden Abbildung sind die verdoppelten Kanten von p' fett dargestellt:

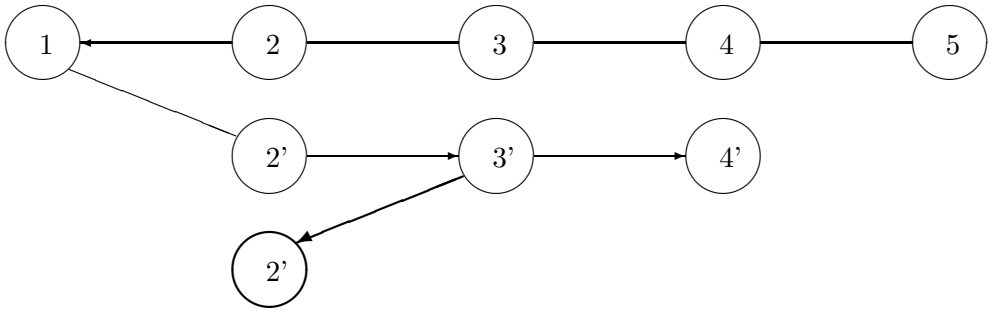


Abbildung 5.9 *Beispielgraph nach der Wegeverdopplung*

Es ergeben sich also folgende neue Elemente:

$$\begin{array}{lllll} p: & \tilde{E}_1^p = \{e_2^p, e_3^p\} & = \{e_{2'3'}, e_{3'4'}\} & \tilde{V}_1^p = \{v_2^p, v_3^p, v_4^p\} & = \{v_{2'}, v_{3'}, v_{4'}\} \\ p': & \tilde{E}_1^{p'} = \{e_3^{p'}\} & = \{e_{3'2''}\} & \tilde{V}_1^{p'} = \{v_4^{p'}\} & = \{v_{2''}\} \end{array}$$

Für die induzierten Abbildungen der gesplitteten Elemente von p' gilt:

$$\begin{array}{llll} h_V^{p'}(v_3^{p'}) & = h_V^{p'}(v_{2''}) = e_3'[= e_{32}] & \Pi_V^1(v_{2''}) & = \omega(h_V^{p'}(v_{2''})) = \omega(e_{32}) = v_2 \\ h_1^{p'}(e_3^{p'}) & = h_1^{p'}(e_{3'2''}) = e_3'[= e_{32}] & \Pi_E^1(e_{3'2''}) & = h_1^{p'}(e_{3'2''}) = e_3' = e_{32} \end{array}$$

Die Abbildung $\psi^{p'}$, die den Verlauf des Wegeverbotes im bisher erschaffenen Wegeverbotsbaum angibt, ist folgendermaßen definiert:

$$\begin{array}{llllll} \psi^{p'}(e_1') & = \psi^p(e_1') & = \psi^p(e_1) & = e_1^p & [= e_{12}] \\ \psi^{p'}(e_2') & = \psi^p(e_2') & = \psi^p(e_2) & = e_2^p & [= e_{2'3'}] \\ \psi^{p'}(e_3') & = (h_1^{p'})^{-1}(e_3') & = e_3^{p'} & & [= e_{3'2''}] \end{array}$$

In Abschnitt 5.4 werden folgende Mengen definiert:

$$N^p = \{(e_1^p, e_{21}), (e_3^p, e_{43})\} \quad N^{p'} = \{(e_3^p, e_{23})\} \quad \tilde{E}^2 = \{e_{1,e_{21}}^p, e_{3,e_{43}}^p, e_{3,e_{23}}^{p'}\} = \{e_{2'1}, e_{4'3}, e_{2''3}\}$$

mit den kanonischen Abbildungen

$$\begin{array}{lll} h_2(e_{1,e_{21}}^p) = (e_1^p, e_{21}) & h_2(e_{3,e_{43}}^p) = (e_3^p, e_{43}) & h_2(e_{3,e_{23}}^{p'}) = (e_3^{p'}, e_{23}) \\ \Pi_E^2(e_{1,e_{21}}^p) = e_{21} & \Pi_E^2(e_{3,e_{43}}^p) = e_{43} & \Pi_E^2(e_{3,e_{23}}^{p'}) = e_{23} \end{array}$$

Das Kantenpaar (e_2^p, e_{32}) wird nicht in N^p aufgenommen, weil das Wegeverbot p' Endweg von (e_{12}, e_{23}, e_{32}) ist und dies der Bedingung (5.7b) widerspricht. Die Kante e_{32} wurde bereits im Schritt zuvor verdoppelt ($e_{3'2''}$).

Der Abschnitt 5.5 bleibt ohne Bedeutung, da alle Wegeverbote aus einer Verbotsklasse stammen. Die folgende Abbildung zeigt den Wegegraphen von G, wobei die neuen Elemente fett dargestellt sind:

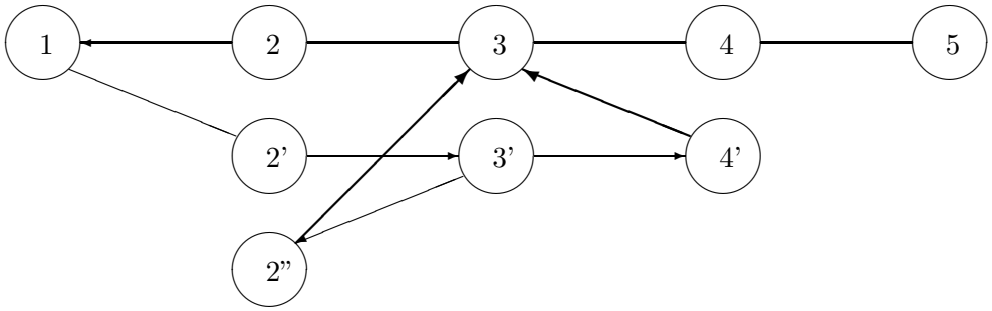


Abbildung 5.10 *Wegegraph des Beispielgraphen mit zwei Wegeverboten mit gleicher Anfangskante*

Weitere Beispiele mit zwei Wegeverboten der gleichen Klasse sind die Beispiele B.9 und B.11.

5.6.3 Beispiel mit drei verschachtelten Wegeverboten

Wenden wir uns nun einem Graphen mit verschachtelten Wegeverboten zu. Dazu definieren wir einen neuen Graph G mit $V := \{v_1, ..., v_{10}\}$, $E := \{e_{12}, e_{23}, e_{34}, e_{45}, e_{46}, e_{67}, e_{68}, e_{89}, e_{8\,10}\}$ mit den Wegeverboten $p_1 := (e_{12}, e_{23}, e_{34}, e_{45})$, $p_2 := (e_{23}, e_{34}, e_{46}, e_{67})$, $p_3 := (e_{34}, e_{46}, e_{68}, e_{89})$ und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$, $\omega(e_{ij}) = v_j$.

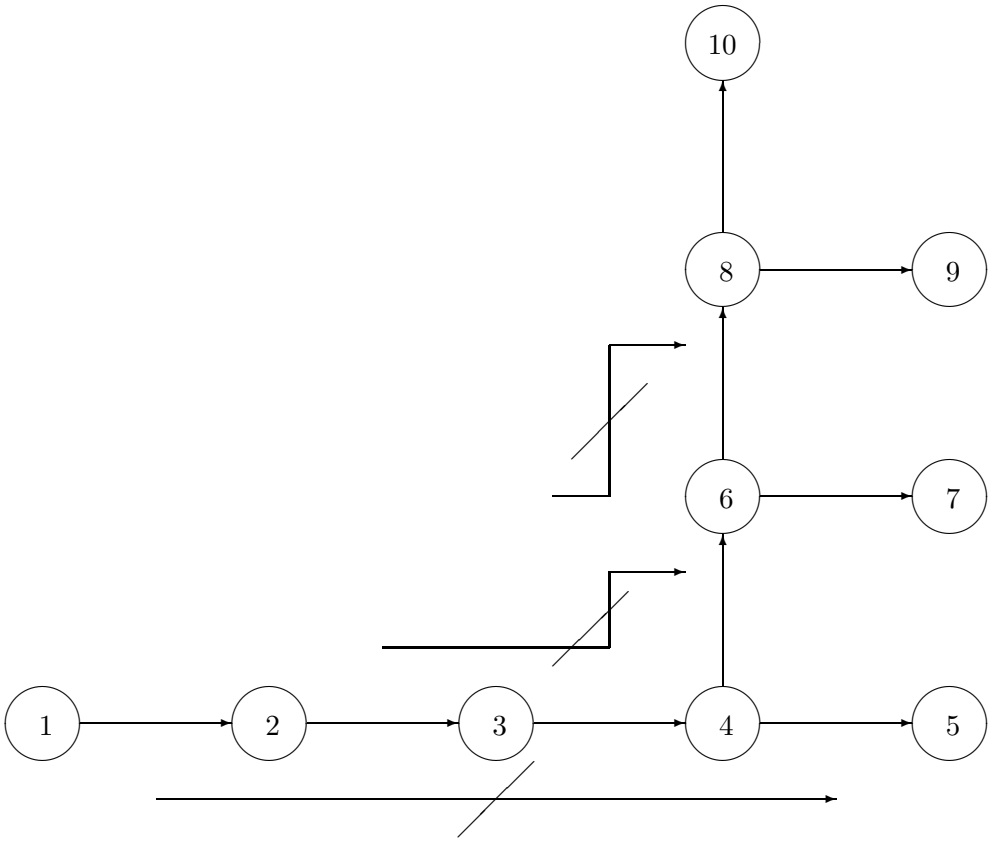


Abbildung 5.11 *Beispielgraph mit drei verschachtelten Wegeverboten*

Wir führen nun die Wegeverdopplung aus Abschnitt 5.3 aus. Dabei werden folgende neue Elemente erschaffen:

$$\begin{array}{lll}
p_1: & \tilde{V}^{p_1} = \{v_1^{p_1}, v_2^{p_1}, v_3^{p_1}\} =: \{v_{2'}, v_{3'}, v_{4'}\} & \tilde{E}_1^{p_1} = \{e_2^{p_1}, e_3^{p_1}\} =: \{e_{2'3'}, e_{3'4'}\} \\
p_2: & \tilde{V}^{p_2} = \{v_1^{p_2}, v_2^{p_2}, v_3^{p_2}\} =: \{v_{3''}, v_{4''}, v_{6''}\} & \tilde{E}_1^{p_2} = \{e_2^{p_2}, e_3^{p_2}\} =: \{e_{3''4''}, e_{4''6''}\} \\
p_3: & \tilde{V}^{p_3} = \{v_1^{p_3}, v_2^{p_3}, v_3^{p_3}\} =: \{v_{4'''}, v_{6'''}, v_{8'''}\} & \tilde{E}_1^{p_3} = \{e_2^{p_3}, e_3^{p_3}\} =: \{e_{4'''6'''}, e_{6'''8'''}\}
\end{array}$$

Exemplarisch betrachten wir nun die Bijektionen von p_1. Im Abschnitt 5.3 werden die Kanten von p_1 mit (e_1, e_2, e_3, e_4)=$(e_{12}, e_{23}, e_{34}, e_{45})$ bezeichnet.

$$\begin{array}{ll}
h_V^{p_1}(v_1^{p_1}) = h_V^{p_1}(v_{2'}) = e_1 = e_{12} & h_V^{p_1}(v_2^{p_1}) = h_V^{p_1}(v_{3'}) = e_2 = e_{23} \\
h_V^{p_1}(v_3^{p_1}) = h_V^{p_1}(v_{4'}) = e_3 = e_{34} & \\
h_1^{p_1}(e_1^{p_1}) = h_1^{p_1}(e_{2'3'}) = e_2 = e_{23} & h_1^{p_1}(e_2^{p_1}) = h_1^{p_1}(e_{3'4'}) = e_3 = e_{34}
\end{array}$$

Für die kanonischen Projektionen gilt

$$\begin{array}{lll}
\Pi_V^1(v_1^{p_1}) = \omega(e_{12}) = v_2 & \Pi_V^1(v_2^{p_1}) = \omega(e_{23}) = v_3 & \Pi_V^1(v_3^{p_1}) = \omega(e_{34}) = v_4 \\
\Pi_E^1(e_2^{p_1}) = h_1^{p_1}(e_2^{p_1}) = e_{23} & \Pi_E^1(e_3^{p_1}) = h_1^{p_1}(e_3^{p_1}) = e_{34} &
\end{array}$$

In der folgenden Abbildung sind die neuen Elemente fett dargestellt. Den jeweiligen Anfangskanten der Wegeverbote wurden neue Endpunkte zugewiesen:

$$\omega^1(e_{12}) := v_1^{p_1} = v_{2'} \quad \omega^1(e_{23}) := v_1^{p_2} = v_{3''} \quad \omega^1(e_{34}) := v_1^{p_3} = v_{4'''}$$

Der ursprünglicher Verlauf dieser Kanten ist gepunktet dargestellt.

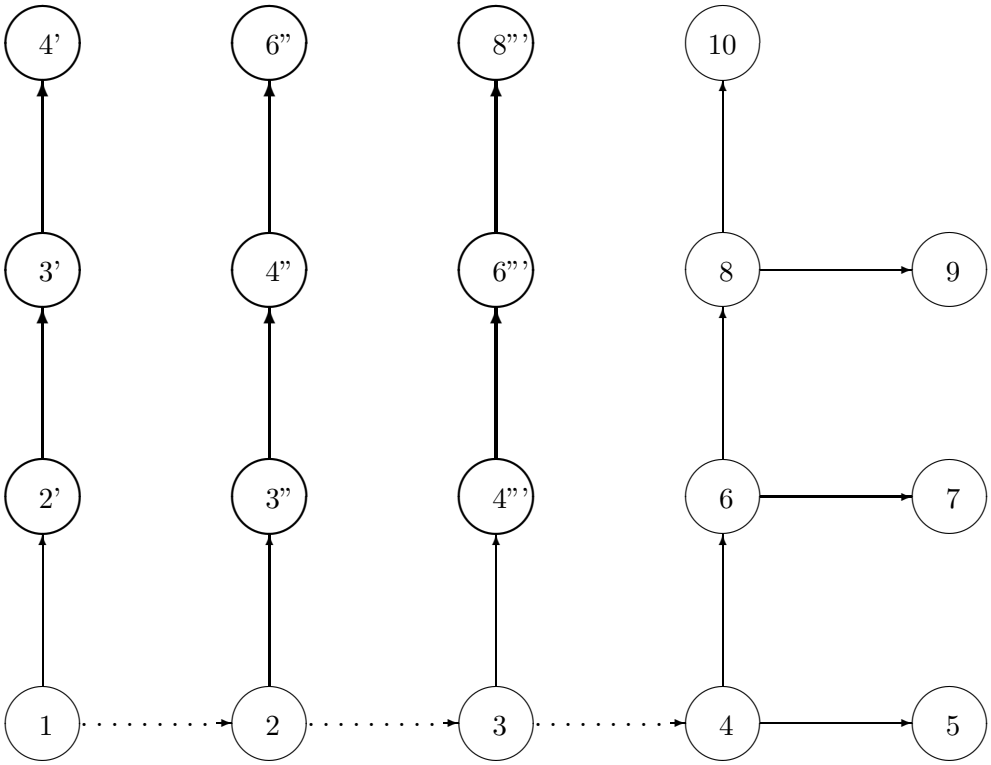


Abbildung 5.12 *Beispielgraph nach der Erweiterung aus Abschnitt 5.3*

Betrachten wir zunächst p_3: Der Weg p_3 kann auf den Kanten e_{45}, e_{67} und $e_{8\,10}$ wieder verlassen werden. Da alle drei Kanten der Bedingung 5.7 genügen, sie kommen also mit anderen Wegeverboten nicht in Konflikt, gilt

$$N^{p_3} = \{(e_1^{p_3}, e_{45}), (e_2^{p_3}, e_{67}), (e_3^{p_3}, e_{8\,10})\} \ [= \{(e_{34}, e_{45}), (e_{46}, e_{67}), (e_{68}, e_{8\,10})\}].$$

Analog zu Abschnitt 5.4 werden drei neue Kanten $e_{4'''5}$, $e_{6'''7}$ und $e_{8'''\,10}$ definiert und so der verdoppelte Weg mit dem Ausgangsgraphen verbunden. Auf diese Weise bleiben zum Beispiel die Wege (e_{34}, e_{45}), (e_{34}, e_{46}, e_{67}) und $(e_{34}, e_{46}, e_{68}, e_{8\,10})$ möglich.

Wenden wir uns nun p_2 zu. Der Weg p_2 kann auf den Kanten e_{45} und e_{68} wieder verlassen werden.

Die Menge N^{p_2} besteht aber nur aus dem Kantenpaar $(e_2^{p_2}, e_{45})[= (e_{34}, e_{45})]$, das die Bedingung aus 5.7 erfüllt, denn über die Kante e_{45} darf p_2 ohne weitere Wegeverbotsbeeinflussung verlassen werden. Deshalb wird im Abschnitt 5.4 der Graph um die Kante $e_{4''5}$ erweitert. Über die Kante e_{68} kann p_2 nicht ohne weiteres verlassen werden. Diese Kante wird im Abschnitt 5.5 behandelt und später mit dem Wegeverbot p_1 betrachtet.

Nach der Ausführung der Vorschriften des Abschnittes 5.4 sind folgende neue Kanten (incl. Abbildungen) entstanden:

$p_1:$		keine neuen Elemente	
$p_2:$	$e^{p_2}_{2,e_{45}} =: e_{4''5}$	$h_2(e_{4''5}) = (e_2^{p_2}, e_{45})[= (e_{34}, e_{45})]$	$\Pi^2_E(e_{4''5}) = e_{45}$
$p_3:$	$e^{p_3}_{1,e_{45}} =: e_{4'''5}$	$h_2(e_{4'''5}) = (e_1^{p_3}, e_{45})[= (e_{34}, e_{45})]$	$\Pi^2_E(e_{4'''5}) = e_{45}$
	$e^{p_3}_{2,e_{67}} =: e_{6'''7}$	$h_2(e_{6'''7}) = (e_2^{p_3}, e_{67})[= (e_{46}, e_{67})]$	$\Pi^2_E(e_{6'''7}) = e_{67}$
	$e^{p_3}_{3,e_{8\,10}} =: e_{8'''10}$	$h_2(e_{8'''10}) = (e_3^{p_3}, e_{8\,10})[= (e_{68}, e_{8\,10})]$	$\Pi^2_E(e_{8'''10}) = e_{8\,10}$

In der folgenden Abbildung sind diese neuen Kanten fett dargestellt:

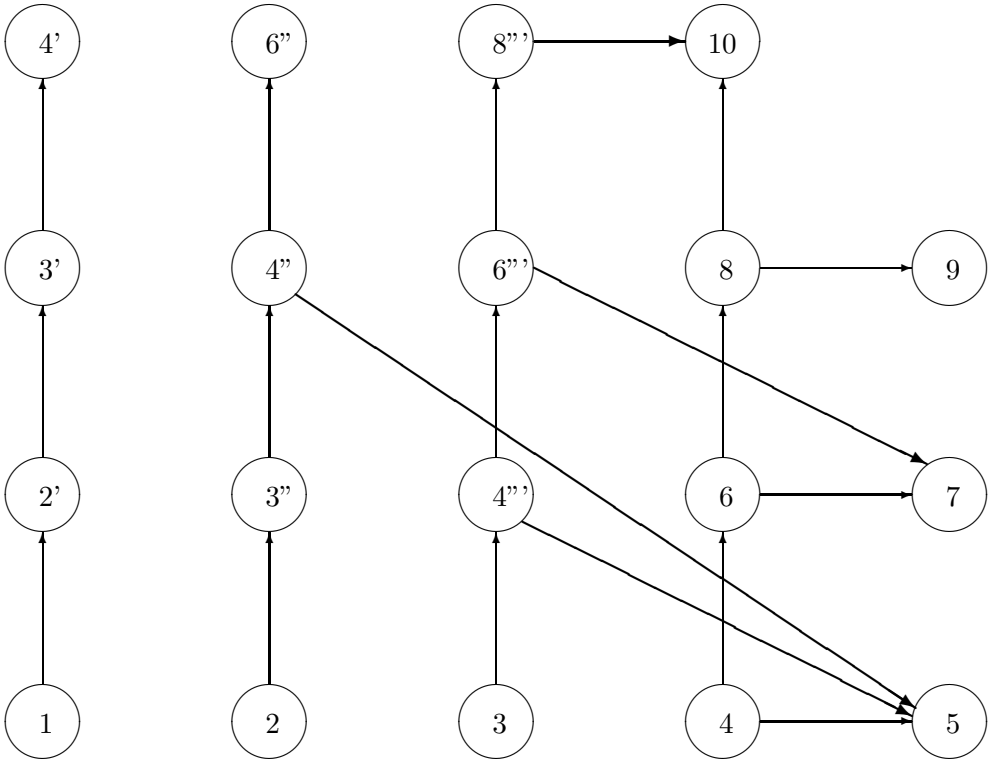


Abbildung 5.13 *Beispielgraph nach der Wegeverdopplung und der Verbindung mit dem Ausgangsgraphen*

Wir führen nun den dritten Teil des Algorithmus aus dem Abschnitt 5.5 durch. Beim Wegeverbot p_2 ist das Kantenpaar $(e_3^{p_2}, e_{68})$ $[= (e_{46}, e_{68})]$ noch nicht betrachtet worden. Dieses

Kantenpaar genügt nicht der Bedingung 5.7*b*, denn p_3 ist Endweg von $w := (e_1^{p_2}, e_2^{p_2}, e_3^{p_2}, e_{68})$ $[=(e_{23}, e_{34}, e_{46}, e_{68})]$ (p_3 beginnt innerhalb von w, w endet innerhalb von p_3 und dazwischen verlaufen beide Wege gleich). Dies bedeutet, daß das Wegeverbot p_3 im Verlauf des Wegeverbotes p_2 gültig geworden ist, und der Weg $(e_{23}, e_{34}, e_{46}, e_{68}, e_{89})$, der p_3 als Teilweg enthält, nicht möglich sein darf. Es ist also erlaubt, das Wegeverbot p_2 über die Kantenkombination (e_{46}, e_{68}) wieder zu verlassen (Menge M), wenn gleichzeitig das Wegeverbot p_3 berücksichtigt bleibt. Deshalb darf $v_{6''}$ nicht direkt mit v_8 im Ausgangsgraphen verbunden werden (Abschnitt 5.4), sondern mit $v_{8'''}$ aus der Wegeverbotserweiterung von p_3 (eine Umleitung geht in die nächste über). Dies wird sichergestellt, indem das Paar $(e_3^{p_2}, e_{68}) = (e_{46}, e_{68})$ nicht in N^{p_2}, sondern in M^{p_2} aufgenommen wird. Diese Kantenpaare werden im Abschitt 5.5 betrachtet.

Das Wegeverbot p_3 ist insbesondere maximaler Endweg von w. Nach der Algorithmusvorschrift aus Abschnitt 5.5 muß also das Wegeverbot p_2 über ein Duplikat der Kante e_{68} mit p_3 verbunden werden. Dies gelingt mittels Formel 5.13. Die neue Kante ist also $e_{3,3}^{p_2 p_3} = e_{6''8'''}$. Sie wird von dem Paar $(e_3^{p_2}, e_{68})$ aus M^{p_2} erzeugt. Damit ist das Wegeverbot p_3 berücksichtigt.

Innerhalb des Wegeverbotes p_1 beginnen p_2 und p_3. Die einzige Kante, über die p_1 verlassen werden könnte, ist e_{46} und zwar über die Kante $e_3^{p_1} = e_{34}$. Sowohl p_2 als auch p_3 sind Endwege von $(e_1^{p_1}, e_2^{p_1}, e_3^{p_1}, e_{46})$ $[= (e_{12}, e_{23}, e_{34}, e_{46})]$. Deshalb konnte p_1 im Abschnitt 5.4 nicht berücksichtigt werden, das heißt $N^{p_1} = \emptyset$.

Mit welchem Wegeverbot soll p_1 verbunden werden? Versuchen wir zunächst die Verbindung mit p_3 mit der neuen Kante $e_{4'6'''}$ $[= e_{3,2}^{p_1 p_3}]$: Da von $v_{6'''}$ eine Kante auf v_7 hinweist, wäre automatisch der Weg

$$(e_{12}, e_{23}, e_{34}, e_{46}, e_{67}) [= (e_1^{p_1}, e_2^{p_1}, e_3^{p_1}, e_{3,2}^{p_1 p_3}, e_{2e_{67}}^{p_3})]$$

möglich, der aber einen verbotenen Teilweg (p_2) enthält.

Verbinden wir aber p_1 mit p_2 über die Kante $e_{4'6''}[= e_{2,2}^{p_1 p_2}]$, so ist automatisch das Wegeverbot p_2 beachtet, denn von $v_{6''}$ gibt es laut Konstruktion keine Verbindung zu v_7 (der keine Duplikate besitzt). Auch p_3 wird beachtet, da von $v_{6''}$ aus nur eine Kante zu $v_{8'''}$ (und nicht etwa zu v_8) hin gezogen wurde. Von dort aus ist die Weiterfahrt nach v_9 nicht möglich.

Warum wählt der Algorithmus automatisch p_2 und nicht p_3? p_2 und p_3 sind beides Endwege von $w' := (e_1^{p_1}, e_2^{p_1}, e_3^{p_1}, e_{46})$ $[= (e_{12}, e_{23}, e_{34}, e_{46})]$, aber p_2 hat mit p_1 drei gemeinsame Kanten während p_3 mit p_1 nur zwei gemeinsame Kanten aufzuweisen hat.

$$\begin{array}{lccccc} w' = (& e_{12}, & e_{23}, & e_{34}, & e_{46}) & \\ p_2 = (& & e_{23}, & e_{34}, & e_{46}, & e_{68}) \\ p_3 = (& & & e_{34}, & e_{46}, & e_{68}, \quad e_{89}) \end{array}$$

Damit ist p_2 der maximale Endweg und wird zunächst mit der Erweiterung von p_1 im Abschnitt 5.5 verbunden.

Im erweiterten Graphen sind also genau die verbotenen Wege nicht möglich. Z. B. ist v_{10} von den Knoten v_1, v_2, v_3 trotzdem noch erreichbar.

Im Abschnitt 5.5 wurden folgende Kanten (incl. Abbildungen) definiert:

$p_1:$	$e_{3,3}^{p_1,p_2} =: e_{4'6''}$	$h_3(e_{4'6''}) = (e_3^{p_1}, e_3^{p_2}) = (e_{34}, e_{46})$	$\Pi_E^3(e_{4'6''}) = e_{46}$
$p_2:$	$e_{3,3}^{p_2,p_3} =: e_{6''8'''}$	$h_3(e_{6''8'''}) = (e_3^{p_2}, e_3^{p_3}) = (e_{46}, e_{68})$	$\Pi_E^3(e_{6''8'''}) = e_{68}$
$p_3:$		keine neuen Elemente	

In der folgenden Abbildung sind die Kanten, die im Abschnitt 5.5 definiert werden fett markiert.

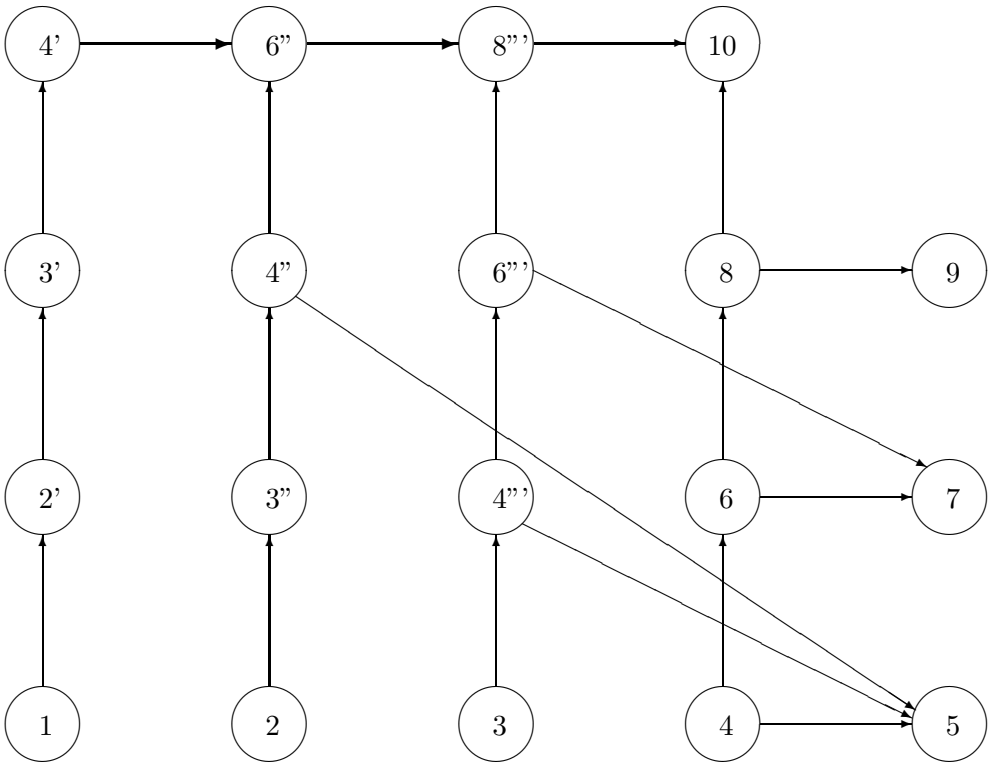


Abbildung 5.14 *Wegegraph des Beispielgraphen mit drei verschachtelten Wegeverboten*

In Beispiel B.10 wird ein weiterer Graph mit verschachtelten Wegeverboten diskutiert.

5.6.4 Beispiel mit einem selbstüberlappenden Wegeverbot

In diesem Beispiel soll ein Wegeverbot betrachtet werden, das seine Anfangskante mehrfach beinhaltet. Gegeben sei ein Graph $G = (V, E, \alpha, \omega)$ mit

$$V = \{v_1, ..., v_7\} \quad E = \{e_{12}, e_{23}, e_{25}, e_{34}, e_{36}, e_{41}, e_{63}, e_{71}\}$$

und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$, $\omega(e_{ij}) = v_j$. Weiter sei das Wegeverbot

$$p := (e_1, , ..., e_{10}) = (e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{25})$$

gegeben, das die Anfangskante e_{12} noch weitere zweimal enthält. Der 'Häuserblock' v_1, v_2, v_3, v_4 muß also (mindestens) zweimal umrundet werden, bevor in Richtung v_5 nicht abgebogen werden darf.

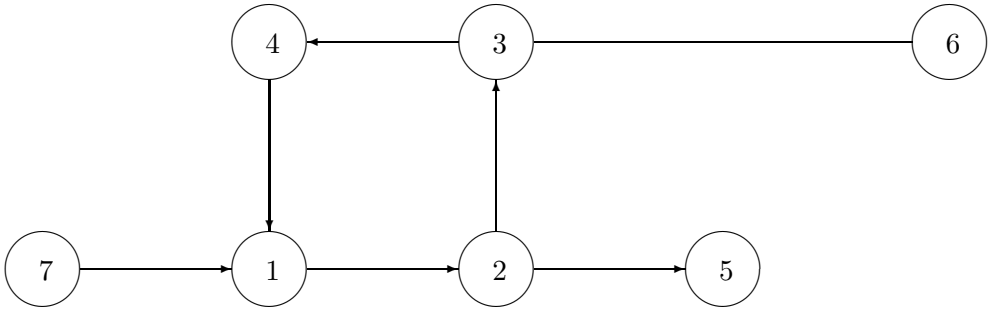


Abbildung 5.15 *Beispielgraph mit selbstüberlappendem Wegeverbot*

Im Abschnitt 5.3 wird zunächst das Wegeverbot p ohne erste und letzte Kante verdoppelt. Die neuen Elemente heißen:

$$\begin{array}{ll} \tilde{V}^p & = \{v_1^p, ..., v_9^p\} \quad = \{v_{2_1}, v_{3_1}, v_{4_1}, v_{1_1}, v_{2_2}, v_{3_2}, v_{4_2}, v_{1_2}, v_{2_3}\} \\ \tilde{E}_1^p & = \{e_2^p, ..., e_9^p\} \quad = \{e_{2_1 3_1}, e_{3_1 4_1}, e_{4_1 1_1}, e_{1_1 2_2}, e_{2_2 3_2}, e_{3_2 4_2}, e_{4_2 1_2}, e_{1_2 2_3}\}. \end{array}$$

Weil das Wegeverbot relativ lang ist, werden die induzierten Abbildungen nur exemplarisch beschrieben:

$$\begin{array}{llll} h_V^p(v_{2_1}) & = h_V^p(v_1^p) & = e_1[= e_{12}] & \\ h_V^p(v_{2_3}) & = h_V^p(v_9^p) & = e_9[= e_{12}] & \\ h_1^p(e_{3_2 4_2}) & = h_1^p(e_7^p) & = e_7[= e_{34}] & \end{array} \qquad \begin{array}{llll} h_V^p(v_{2_2}) & = h_V^p(v_5^p) & = e_5[= e_{12}] & \\ h_1^p(e_{2_1 3_1}) & = h_1^p(e_2^p) & = e_2[= e_{23}] & \\ h_1^p(e_{1_2 2_3}) & = h_1^p(e_9^p) & = e_9[= e_{12}] & \end{array}$$

$$\begin{array}{llll} \Pi_V^1(v_{2_1}) & = \omega(h_V^p(v_1^p)) & = \omega(e_1) = \omega(e_{12}) = v_2 & \\ \Pi_V^1(v_{3_2}) & = \omega(h_V^p(v_6^p)) & = \omega(e_6) = \omega(e_{23}) = v_3 & \end{array} \qquad \begin{array}{llll} \Pi_E^1(e_{4_1 1_1}) & = h_1^p(e_4^p) & = e_4 = e_{41} & \\ \Pi_E^1(e_{1_2 2_3}) & = h_1^p(e_9^p) & = e_9 = e_{12} & \end{array}$$

In der folgenden Abbildung die Wegeverdopplung aus Abschnitt 5.3 bereits ausgeführt. Die neuen Elemente sind dabei fett, der alte Verlauf der Kante e_{12} gepunktet dargestellt.

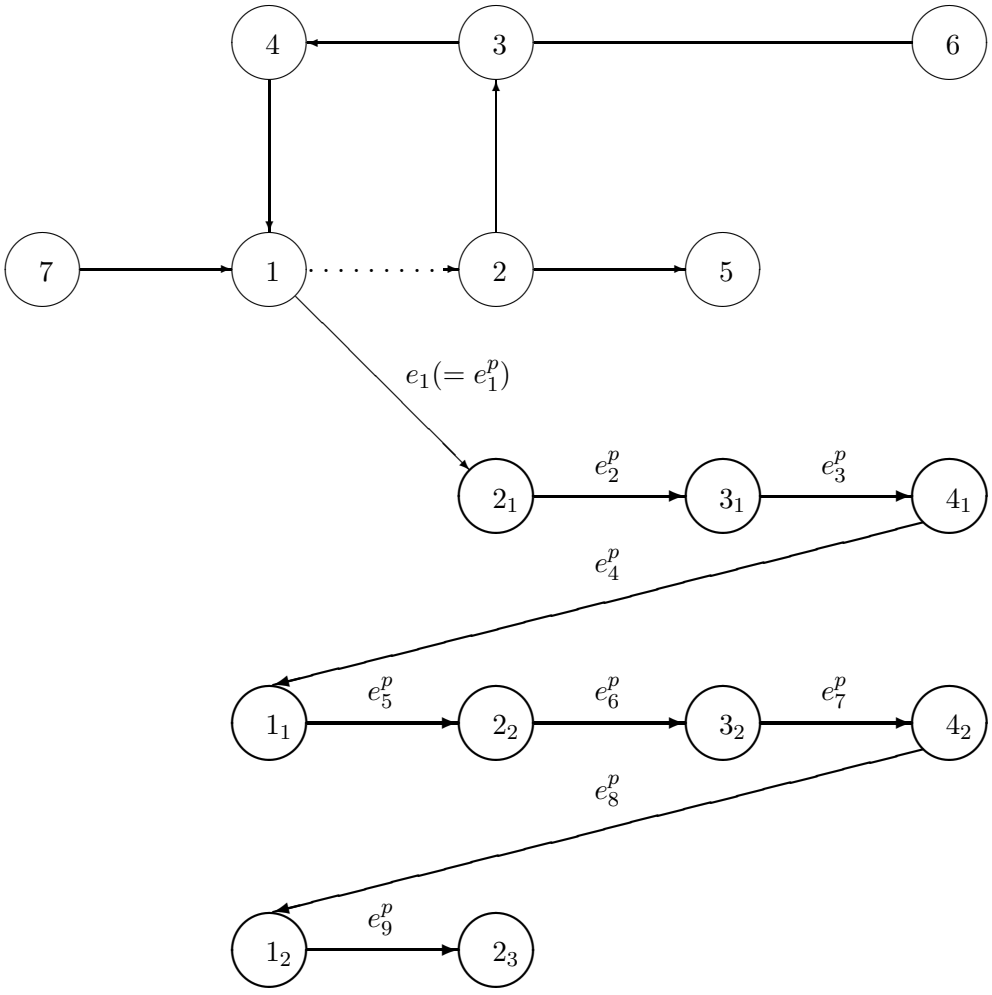


Abbildung 5.16 *Beispielgraph mit selbstüberlappendem Wegeverbot nach der Wegeverdopplung*

Wir wollen nun die Wegeverbotserweiterung wieder mit dem Ausgangsgraphen verbinden. p kann ohne Einschränkung immer von v_3 aus in Richtung v_6 verlassen werden. Damit gilt $(e_2^p, e_{36})[= (e_{23}, e_{36})] \in N^p$ und $(e_6^p, e_{36})[= (e_{23}, e_{36})] \in N^p$. Um diese Elemente unterscheiden zu können, benötigen wir die Indizierung in der ersten Komponente. Wenn der Häuserblock noch nicht zweimal umfahren wurde, kann das Wegeverbot vom Knoten v_2 aus in Richtung v_5 verlassen werden. Dies bedeutet $(e_1^p, e_{25})[= (e_{12}, e_{25})] \in N^p$ und $(e_5^p, e_{25})[= (e_{12}, e_{25})] \in N^p$. Wir betrachten nun die neu erzeugten Mengen. Dabei stehen jeweils zugeordnete Elemente untereinander.

$$\begin{array}{lllll} N = N^p & = \{(e_1^p, e_{25}), & (e_2^p, e_{36}), & (e_5^p, e_{25}), & (e_6^p, e_{36})\} \\ \tilde{E}^2 & = \{e^p_{1,e_{25}}, & e^p_{2,e_{36}}, & e^p_{5,e_{25}}, & e^p_{6,e_{36}}\} \\ & = \{e_{2_1 5}, & e_{3_1 6}, & e_{2_2 5}, & e_{3_2 6}\} \end{array}$$

Für die induzierten Abbildungen gilt:

$h_2(e^p_{1,e_{25}}) = (e^p_1, e_{25})[= (e_{12}, e_{25})]$ $\quad$ $h_2(e^p_{2,e_{36}}) = (e^p_2, e_{36})[= (e_{23}, e_{36})]$
$h_2(e^p_{5,e_{25}}) = (e^p_5, e_{25})[= (e_{12}, e_{25})]$ $\quad$ $h_2(e^p_{6,e_{36}}) = (e^p_6, e_{36})[= (e_{23}, e_{36})]$

$\Pi^2_E(e^p_{1,e_{25}}) = \pi_2(h_2(e^p_{1,e_{25}})) = \pi_2(e^p_1, e_{25}) = e_{25}$ $\quad$ $\Pi^2_E(e^p_{2,e_{36}}) = \pi_2(h_2(e^p_{2,e_{36}})) = \pi_2(e^p_2, e_{36}) = e_{36}$
$\Pi^2_E(e^p_{5,e_{25}}) = \pi_2(h_2(e^p_{5,e_{25}})) = \pi_2(e^p_5, e_{25}) = e_{25}$ $\quad$ $\Pi^2_E(e^p_{6,e_{36}}) = \pi_2(h_2(e^p_{6,e_{36}})) = \pi_2(e^p_6, e_{36}) = e_{36}$.

In der folgenden Abbildung sind die im Abschnitt 5.4 neu definierten Kanten (aus $\tilde{E}^2$) fett dargestellt:

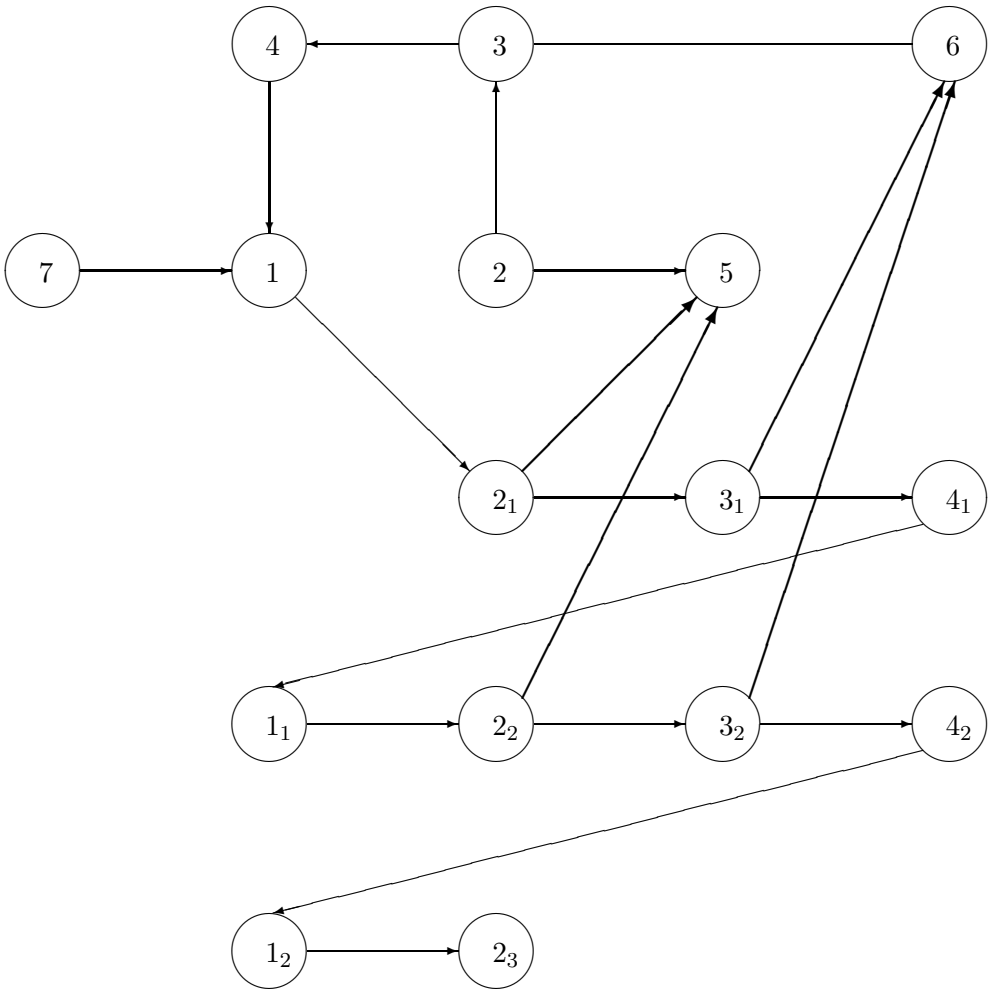


Abbildung 5.17 *Beispielgraph mit selbstüberlappendem Wegeverbot nach den Schritten aus Abschnitt 5.4*

Betrachten wir nun den Knoten v_{2_3}. Von diesem Knoten ist die Weiterfahrt in Richtung v_5 untersagt, nicht aber in Richtung v_3. Das Kantenpaar $(e_9^p, e_{23})[= (e_{12}, e_{23})]$ liegt nicht in N^p, denn p ist Endweg von $w := (e_1^p, ..., e_9^p, e_{23})$:

$$\begin{array}{lllll} w = & (e_{12}, e_{23}, e_{34}, e_{41}, & e_{12}, e_{23}, e_{34}, e_{41}, & e_{12}, e_{23}) \\ p = & & (e_{12}, e_{23}, e_{34}, e_{41}, & e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{25}) \\ p = & & & (e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{25}) \end{array}$$

Mit welchem Knoten aus der Menge $(\Pi_V^1)^{-1}(v_3) = \{v_3, v_{3_1}, v_{3_2}\}$ muß jetzt v_{2_3} verbunden werden?

1. Fall: Wir verbinden v_{2_3} mit v_3. Dies würde bedeuten, daß nach zweimaligem Umrunden des Häuserblocks eine weitere Runde genügen würde, um v_5 zu erreichen. Zwar würde man beim Erreichen von v_1 automatisch nach v_{2_1} umgeleitet, aber da im Abschnitt 5.4 die Kante $e_{2_1 5}$ erschaffen wurde, wäre der Weg

$$w' := (e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{23}, e_{34}, e_{41}, e_{12}, e_{25})$$

möglich, der offensichtlich den verbotenen Weg p enthält.

2. Fall: Nun verbinden wir v_{2_3} mit v_{3_1}. In diesem Fall tritt das gleiche Problem auf wie im 1. Fall, da wieder w' beschritten werden kann, der p als Teilweg enthält.

3. Fall: v_{2_3} wird mit v_{3_2} verbunden. Hier ist das Befahren von w' nicht möglich, da man von v_{3_2} wieder zu v_{2_3} geleitet wird, von dem aus das Weiterfahren nach v_5 nicht möglich ist.

Wie entscheidet dies der Algorithmus? p erfüllt zweimal die Endwegeigenschaft von w und zwar einmal mit sechs und einmal mit zwei gemeinsamen Kanten. Nach Definition 2.11 soll als Endweg immer der Weg gelten, der mehr gemeinsame Kanten hat. v_{2_3} wäre analog zu Fall 2 verbunden worden, wenn p Endweg von w mit nur zwei gemeinsamen Kanten gewesen wäre und analog zu Fall 1, wenn p gar nicht Endweg von w gewesen wäre.

So wird das Kantenpaar $(e_9^p, e_6^p)[= (e_{12}, e_{23})]$ in die Menge M^p aufgenommen und im Abschnitt 5.5 eine neue Kante $e_{9,6}^{p,p} = e_{2_3 3_2}$, mit $\Pi_E^3(e_{2_3 3_2}) = e_{23}$ erschaffen. So kann der Knoten v_5 erst wieder erreicht werden, wenn mindestens einmal der Knoten v_6 besucht wurde. In der abschließenden Abbildung des Wegegraphen ist die neue Kante wieder fett dargestellt:

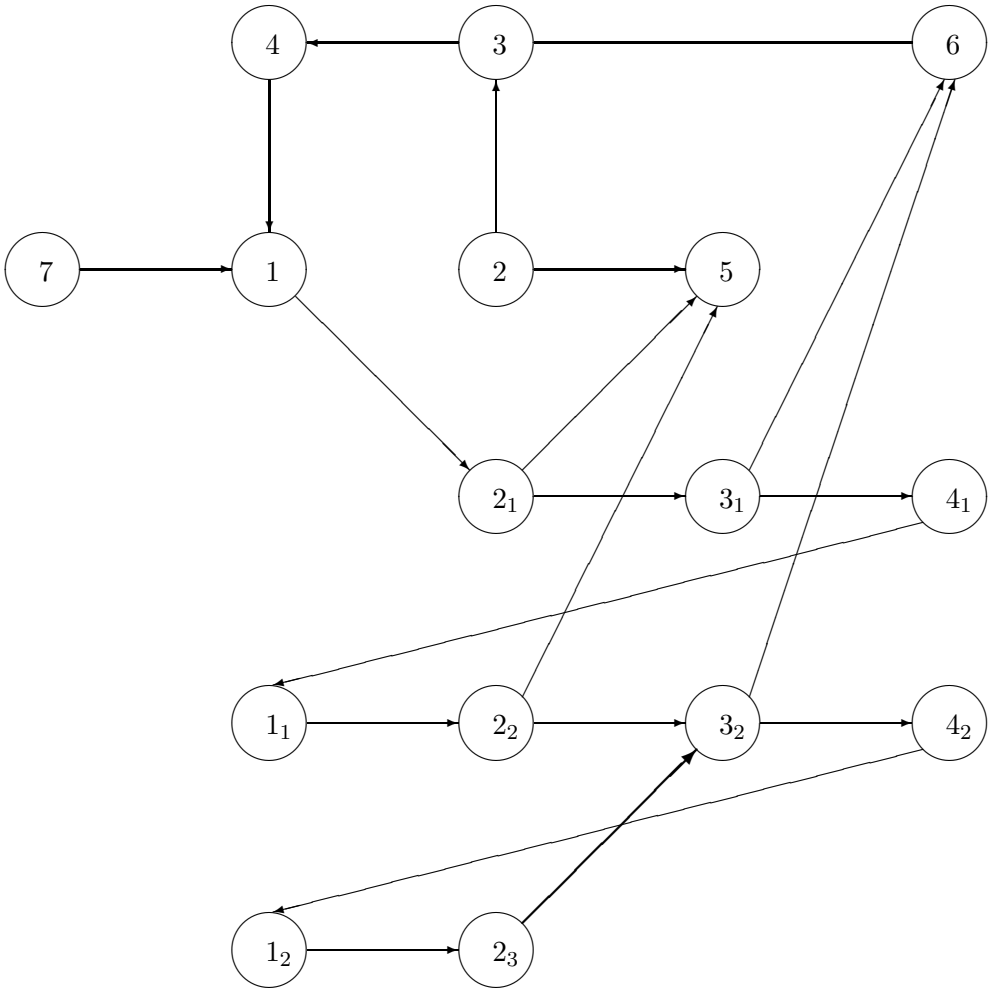


Abbildung 5.18 *Wegegraph des Beispielgraphen mit selbstüberlappendem Wegeverbot*

5.7 Beweis des Algorithmus

Der Beweis verläuft analog zum Beweis bei den Abbiegeverboten. Zuerst beweisen wir, daß man alle Originalkanten an ihrem Anfangspunkt erkennt. Mit diesem Resultat zeigen wir, daß die Ergebnisse der Grapherweiterung unabhängig von der Reihenfolge der Wegeverbote sind. Dies gilt sowohl innerhalb der gewählten Äquivalenzklassen, als auch bei der Wahl der Klassen selbst. Danach wird die Wegeäquivalenz gezeigt.

Satz 5.4 *Sei $G = (V, E, \alpha, \beta)$ ein Graph mit Wegeverboten P, $G^3 = (V^3, E^3, \alpha^3, \omega^3)$ dessen Wegegrapherweiterung, dann gilt für $e \in E^3$*

$$e \in E \iff \alpha^3(e) \in V$$

Insbesondere gilt für $e_1 \in \tilde{E}^1$:

$$\alpha^3(e_1) \in V^1 \backslash V \quad \textit{und} \quad \omega^3(e_1) \in V^1 \backslash V$$

für $e_2 \in \tilde{E}^2$:

$$\alpha^3(e_2) \in V^1 \backslash V \quad \textit{und} \quad \omega^3(e_2) \in V$$

und für $e_3 \in \tilde{E}^3$:

$$\alpha^3(e_3) \in V^1 \backslash V \quad \textit{und} \quad \omega^3(e_3) \in V^1 \backslash V.$$

Beweis:

$\Longrightarrow$: Für $e \in E$ gilt

$$\begin{array}{lll} \alpha^3(e) & = \alpha^2(e) & \text{da } e \in E \subseteq E^2 \\ & = \alpha^1(e) & \text{da } e \in E \subseteq E^1 \\ & = \alpha(e) \in V & \text{da } e \in E \end{array}$$

$\Longleftarrow$: Diese Richtung zeigen wir indirekt, das heißt für $e \notin E$ gilt $\alpha(e) \notin V$. Wir unterscheiden drei Fälle:

Sei $e \in \tilde{E}^1$, dann gibt es ein Wegeverbot $p := p_k = (e_1, ..., e_n)$ mit $h_1^p(e) = e_j$. Sei $(e_{q+1}^p, ..., e_{n-1}^p)$ der gesplittete Weg von p und sei $p' = (e'_1, ..., e'_m) \in P$ der maximale Anfangsweg von $(e_1, ..., e_j)$ in $\{p_1, ..., p_{k-1}\}$ mit q gemeinsamen Kanten, dann gilt nach Konstruktion $q < j$. Nach Gleichung (5.1) gilt für $q = j - 1$

$$\alpha(e_j^p) = v_{j-1}^{p'} \in \tilde{V}^1, \notin V$$

und für $q < j - 1$ gilt

$$\alpha(e_j^p) = v_{j-1}^{p} \in \tilde{V}^1, \notin V.$$

Sei $e \in \tilde{E}^2$, dann gibt es ein Wegeverbot p, eine Kante $e \in E$ und einen Index i mit $e = e_{i,e}^p \in \tilde{E}^2$, dann gilt mit Gleichung (5.10):

$$\alpha(e) = \tilde{v}_i^p \in \tilde{V}^1, \notin V.$$

Sei $e \in \tilde{E}^3$, dann gibt es zwei Wegeverbote $p, \hat{p}$ und zwei Indices i und j, so daß $e = e_{i,j}^{p,\hat{p}}$ ist. Es gilt mit Gleichung (5.13):

$$\alpha(e) = \tilde{v}_i^p \in \tilde{V}^1, \notin V.$$

Die restlichen Ergebnisse ergeben sich analog aus den Gleichungen (5.1), (5.10) und (5.13) sowie aus der Definition der Inzidenzabbildungen. ∎

Die Ergebnisse dieses Satzes werden in fast allen folgenden Sätzen verwendet.

5.7.1 Die Unabhängigkeit der Reihenfolge

Mit diesem Satz werden wir zeigen, daß die Grapherweiterungen unabhängig von der gewählten Reihenfolge der Wegeverbote innerhalb einer Klasse sind.

Satz 5.5 *Sei $G := (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, p_l=(e_1 , ..., e_n), p_{l+1}=(f_1 , ..., f_m) $\in [\![p_l]\!]$ und sei p_l der maximale Anfangsweg von p_{l+1} in $\{p_1, ..., p_l\}$ mit q gemeinsamen Anfangskanten, dann gilt p_{l+1} ist maximaler Anfangsweg von p_l in $\{p_1, ..., p_{l-1}, p_{l+1}\}$ mit q gemeinsamen Anfangskanten.*

Beweis:

Die Anzahl der gemeinsamen Anfangskanten ist eine symmetrische Eigenschaft, deshalb ist nur die Symmetrie des maximalen Anfangsweges zu zeigen. Diese beweisen wir indirekt. Annahme: Sei $p_k = (e'_1, ..., e'_{n'}) \in P$ mit $k \leq l-1$ maximaler Anfangsweg von p_l in $\{p_1, ..., p_{l-1}, p_{l+1}\}$, dann gilt $e'_i = f_i$ für $1 \leq i \leq q$, denn p_l und p_{l+1} haben q Kanten gemeinsam (und sonst wäre p_l maximaler Anfangsweg) und damit gilt $e'_i = e_i$ für $1 \leq i \leq q$, das heißt da $k < l$ gilt, ist p_l nicht maximaler Anfangsweg von p_{l+1} in $\{p_1, ..., p_l\}$. ∎

Satz 5.6 *Sei $G := (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, und seien $p := p_l$=(e_1 , ..., e_n), $\hat{p} := p_{l+1}$=($\hat{e}_1$, ..., $\hat{e}_m$) $\in [\![p_l]\!]$. Sei G^3_a der erweiterte Graph mit der gegebenen Reihenfolge der Wegeverbote und G^3_b der erweiterte Graph mit vertauschter Reihenfolge p_{l+1} ist Vorgänger von p_l, dann gilt, daß es zwei bijektive Abbildungen $g : E^3_a \to E^3_b$ und $g_v : V^3_a \to V^3_b$ gibt, für welche*

$$\alpha^3_b(g(e)) = g_v(\alpha^3_a(e)) \quad \textit{und} \quad \omega^3_b(g(e)) = g_v(\omega^3_a(e)) \textit{ für alle } e \in E^3_a$$

gilt. Die Grapherweiterung ist also unabhängig von der gewählten Anordnung in den Klassen.

Beweis:

Im Fall a verwenden wir zuerst p_l, dann p_{l+1}. Im Fall b vertauschen wir diese Reihenfolge. Die erschaffenen Elemente aus den verschiedenen Fällen erhalten zur Unterscheidung immer die Indizes a bzw. b.

1. Fall: $p := p_l = (e_1, ..., e_n)$ ist maximaler Anfangsweg von $\hat{p} := p_{l+1} = (\hat{e}_1, ..., \hat{e}_{\hat{n}})$ in $\{p_1, ..., p_l\}$ mit q gemeinsamen Anfangskanten. Sei $p' := p_k = (e'_1, ..., e'_{n'})$ der maximale Anfangsweg von p_l in $\{p_1, ..., p_{l-1}\}$ mit t ($t \leq q$) gemeinsamen Anfangskanten, dann definieren wir die gemeinsam verlaufenden Wege

$$w_a := (e_{t+1}, ..., e_q) \text{ und } w_b := (\hat{e}_{t+1}, ..., \hat{e}_q).$$

Eine Bijektion g^w_1 zwischen diesen Wegen kann intuitiv definiert werden:

$$g^w_1(e_i) := \hat{e}_i \quad \text{für } i = t+1, ..., q.$$

Die Menge der gesplitteten Kanten $\{e^p_{t+1}, ..., e^p_q\}$ aus w_a sei $\tilde{E}^1_{w,a}$, die Menge der gesplitteten Kanten aus w_b sei $\tilde{E}^1_{w,b}$. Die im Schritt 5.3 erschaffenen Endknoten der Kanten aus $\tilde{E}^1_{w,a}$ bzw. $\tilde{E}^1_{w,b}$ heißen $\tilde{V}^1_{w,a}$ bzw. $\tilde{V}^1_{w,b}$.

2.Fall: $p_k = (e'_1, ..., e'_{n'})$ mit $k < l$ sei maximaler Anfangsweg von p_{l+1} in $\{p_1, ..., p_l\}$. Hier definieren wir $w_a = w_b := \lambda$. In diesem Fall muß natürlich keine Bijektion betrachtet werden.

Die in Fall a in den Abschnitten 5.4 und 5.5 erschaffenen Kanten, von welchen aus w_a verlassen werden kann, sind

$$\tilde{E}^2_{w,a} := \{e \in E^2 \mid \pi_1(h_{2,a}(e)) \in \tilde{E}^1_{w,a}\} \text{ und } \tilde{E}^3_{w,a} := \{e \in E^3 \mid \pi_1(h_{3,a}(e)) \in \tilde{E}^1_{w,b}\}.$$

Analoges definieren wir für den Fall b. Sei $N^p_{w,a}$ die Menge aller Kantenpaare (e^p_i, e), für die $e^p_i \in \tilde{E}^1_{w,a}$ ist, ferner sei $N^{\hat{p}}_{w,b}$ die Menge aller Kantenpaare $(e^{\hat{p}}_i, e)$, für die $e^{\hat{p}}_i \in \tilde{E}^1_{w,b}$ gilt. Sei $g^w_2 : N^p_{w,a} \to N^{\hat{p}}_{w,b}$ eine Abbildung mit

$$g^w_2((e^p_{i,a}, e)) := (g^w_1(e^p_{i,a}), e) = (e^{\hat{p}}_{i,b}, e),$$

dann ist g^2_w eine Bijektion, da die zweite Komponente der Elemente aus N unabhängig von der Reihenfolge der Wegeverbote ist. Sei $M^p_{w,a}$ die Menge aller Kantenpaare $(e^p_i, e^{\check{p}}_j)$, für die $e^p_i \in \tilde{E}^1_{w,a}$ ist, ferner sei $M^{\hat{p}}_{w,b}$ die Menge aller Kantenpaare $(e^{\hat{p}}_i, e^{\check{p}}_j)$, für die $e^{\hat{p}}_i \in \tilde{E}^1_{w,b}$ gilt. Weiter sei $g^w_3 : M^p_{w,a} \to M^{\hat{p}}_{w,b}$

$$g^w_3((e^p_i, e^{\check{p}}_j)) := (g^w_1(e^p_i), e^{\check{p}}_j) = (e^{\hat{p}}_i, e^{\check{p}}_j).$$

Auch diese Abbildung ist eine Bijektion. Für $w_a = \lambda$ (Fall 2) entfallen diese Definitionen. Wir definieren die Kantenbijektion $g : E^3_a \to E^3_b$ zu

$$g(e) := \begin{cases} e & \text{für alle } e \in E \\ \\ (h^{\check{p}}_{1,b})^{-1}(h^{\check{p}}_{1,a}(e)) & \text{falls } e = e^{\check{p}}_i \in \tilde{E}^1_a \backslash \tilde{E}^1_{w,a} \\ (h^{\hat{p}}_{1,b})^{-1}(g^w_1(h^p_{1,a}(e))) & \text{falls } e \in \tilde{E}^1_{w,a} \\ \\ (h_{2,b})^{-1}(h_{2,a}(e)) & \text{falls } e \in \tilde{E}^2_a \backslash \tilde{E}^2_{w,a} \\ (h_{2,b})^{-1}(g^w_2(h_{2,a}(e))) & \text{falls } e \in \tilde{E}^2_{w,a} \\ \\ (h_{3,b})^{-1}(h_{3,a}(e)) & \text{falls } e \in \tilde{E}^3_a \backslash \tilde{E}^3_{w,a} \\ (h_{3,b})^{-1}(g^w_3(h_{3,a}(e))) & \text{falls } e \in \tilde{E}^3_{w,a} \end{cases}$$

und die Knotenbijektion $g_v : V^1_a \to V^1_b$:

$$g_v(v') := \begin{cases} v' & & \text{für alle } v' \in V \\ \omega^1_b((h^{\check{p}}_{1,b})^{-1}(h^{\check{p}}_{V,a}(v'))) & = v^{\check{p}}_{i,b} & \text{falls } v' = v^{\check{p}}_{i,a} \in \tilde{V}^1_a \backslash \tilde{V}^1_{w,a} \\ \omega^1_b((h^{\hat{p}}_{1,b})^{-1}(g^w_1(h^p_{V,a}(v')))) & = v^{\hat{p}}_{i,b} & \text{falls } v' = v^p_{i,a} \in \tilde{V}^1_{w,a}. \end{cases}$$

Weil die Abbildungen g und g_v Verkettungen von Bijektionen sind, folgt, daß g und g_v selbst Bijektionen sind (ω^1_b bildet bijektiv nach $\tilde{V}^1_{w,a}$ ab). Mit dieser Definition der Kantenbijektion gilt für $e^p_{i,a} \in \tilde{E}^1_{w,a}$ (i=t+1 , ..., q):

$$\begin{aligned} g(e^p_{i,a}) &= (h^{\hat{p}}_{1,b})^{-1}(g^w_1(h^p_{1,a}(e^p_{i,a}))) && \text{Definition von } g \\ &= (h^{\hat{p}}_{1,b})^{-1}(g^w_1(e_i)) && \text{Definition von } h^p_{1,a} \\ &= (h^{\hat{p}}_{1,b})^{-1}(\hat{e}_i) && \text{Definition von } g^w_1 \\ &= e^{\hat{p}}_{i,b} && \text{Definition von } h^{\hat{p}}_{1,b}. \end{aligned} \tag{5.15}$$

Analog zeigt man die Beziehungen

$$g(e^p_{i,e,a}) = e^{\hat{p}}_{i,e,b} \quad \text{für } e^p_{i,e,a} \in \tilde{E}^2_{w,a} \quad \text{und}$$

$$g(e^{p,\check{p}}_{i,j,a}) = e^{\hat{p},\check{p}}_{i,j,b} \quad \text{für } e^{p,\check{p}}_{i,j,a} \in \tilde{E}^3_{w,a}.$$

Für $e^{\check{p}}_{i,e,a} \in \tilde{E}^2_{w,a} \quad i = t+1, ..., q$ gilt:

$$\begin{aligned} g(e^{\check{p}}_{i,a}) &= (h_{2,b})^{-1}(h_{2,a}(e^{\check{p}}_{i,e,a})) && \text{Definition von } g \\ &= (h_{2,b})^{-1}((e^{\check{p}}_{i,a}, e)) && \text{Definition von } h_{2,a} \\ &= (h_{2,b})^{-1}((e^{\check{p}}_{i,b}, e)) && e^{\check{p}}_{i,a} \notin \tilde{E}^1_{w,a} \\ &= e^{\check{p}}_{i,b} && \text{Definition von } h_{2,b}. \end{aligned}$$

Analog zeigt man $g(e^{\check{p}}_{i,a}) = e^{\check{p}}_{i,b}$ für $e^{\check{p}}_{i,a} \in \tilde{E}^1_a \backslash \tilde{E}^1_{w,a}$ und $g(e^{\check{p},\bar{p}}_{i,j,a}) = e^{\check{p},\bar{p}}_{i,j,b}$ für $e^{\check{p},\bar{p}}_{i,j,a} \in \tilde{E}^3_a \backslash \tilde{E}^3_{w,a}$. Ebenso gilt bei den Knoten für $v^p_{i,a} \in \tilde{V}^1_{w,a}$ $(i = t+1, ..., q)$:

$$\begin{aligned} g_v(v^p_{i,a}) &= \omega^1_b(h^{\hat{p}}_{1,b})^{-1}(g^w_1(h^p_{V,a}(v^p_{i,a})))) && \text{Definition von } g_v \\ &= \omega^1_b(h^{\hat{p}}_{1,b})^{-1}(g^w_1(e_i))) && \text{Definition von } h^p_{V,a} \\ &= \omega^1_b(h^{\hat{p}}_{1,b})^{-1}(\hat{e}_i)) && \text{Definition von } g^w_1 \\ &= \omega^1_b(e^{\hat{p}}_{i,b}) && \text{Definition von } h^{\hat{p}}_{1,b} \\ &= v^{\hat{p}}_{i,b} && \text{Nach Gleichung (5.1).} \end{aligned} \tag{5.16}$$

Analog gilt $g_v(v^{\check{p}}_{i,a}) = v^{\check{p}}_{i,b}$ für $v^{\check{p}}_{i,a} \in \tilde{V}^1_a \backslash \tilde{V}^1_{w,a}$.

Die Verträglichkeit der Abbildungen α und ω mit g und g_v

$\alpha^3_b(g(e)) = g_v(\alpha^3_a(e))$:

Für die Kanten $e \in E$ gilt die Behauptung trivialerweise, denn die Anfangspunkte der Originalkanten werden nach Satz 5.4 in den einzelnen Schritten nicht verändert.

Sei $e = e^p_{i,a} \in \tilde{E}^1_{w,a}$, dann gilt

$$\begin{aligned} \alpha^3_b(g(e^p_{i,a})) &= \alpha^3_b(e^{\hat{p}}_{i,b}) && \text{nach Gleichung (5.15)} \\ &= v^{\hat{p}}_{i-1,b} && \text{nach Gleichung (5.1)} \\ &= g_v(v^p_{i-1,a}) && \text{nach Gleichung (5.16)} \\ &= g_v(\alpha^3_a(e^p_{i,a})) && \text{nach Gleichung (5.1).} \end{aligned}$$

Analog zeigt man die Gleichheit für die Kanten aus $\tilde{E}^2_{w,a}$ und $\tilde{E}^3_{w,a}$ mit den Gleichungen (5.10) und (5.13). Für $e^{\check{p}}_{i,a} \in \tilde{E}^1_a \backslash \tilde{E}^1_{w,a}$ gilt:

$$\begin{aligned} \alpha^3_b(g(e^{\check{p}}_{i,a})) &= \alpha^3_b(e^{\check{p}}_{i,b}) && \text{nach Gleichung (5.15)} \\ &= v^{\check{p}}_{i-1,b} && \text{nach Gleichung (5.1)} \\ &= g_v(v^{\check{p}}_{i-1,a}) && \text{nach Gleichung (5.16)} \\ &= g_v(\alpha^3_a(e^{\check{p}}_{i,a})) && \text{nach Gleichung (5.1).} \end{aligned}$$

Analog zeigt man die Gleichheit für die Kanten aus $\tilde{E}^2_a \backslash \tilde{E}^2_{w,a}$ und $\tilde{E}^3_a \backslash \tilde{E}^3_{w,a}$ mit den Gleichungen (5.10) und (5.13).

$\omega^3_b(g(e)) = g_v(\omega^3_a(e))$:

Für die Kanten $e \in E \backslash E_0$ gilt wiederum trivialerweise die Behauptung, da die Endknoten dieser Kanten in keinem Schritt verändert werden. Sei $e^p_{i,a} \in \tilde{E}^1_{w,a}$, dann gilt

$$\begin{array}{lll} \omega^3_b(g(e^p_{i,a})) & = \omega^3_b(e^{\hat{p}}_{i,b}) & \text{nach Gleichung (5.15)} \\ & = v^{\hat{p}}_{i,b} & \text{nach Gleichung (5.1)} \\ & = g_v(v^p_{i,a}) & \text{nach Gleichung (5.16)} \\ & = g_v(\omega^3_a(e^p_{i,a})) & \text{nach Gleichung (5.1)} \end{array}$$

Analog zeigt man die Gleichheit für die Kanten aus $\tilde{E}^1_a, \tilde{E}^2_a$ und $\tilde{E}^3_a$. Für $e_0 \in E_0$ (wobei e_0 die erste Kante des Wegeverbotes $\check{p}$ ist) gilt:

$$\begin{array}{lll} \omega^3_b(g(e_0)) & = \omega^3_b(e_0) & \text{da } e_0 \in E \\ & = (h^{\check{p}}_{V,b})^{-1}(e_0) & \text{Definition von } \omega^3_b \\ & = v^{\check{p}}_{1,b} & \text{Definition von } v^{\check{p}}_{1,b} \\ & = g_v(v^{\check{p}}_{1,a}) & \text{nach Gleichung 5.16} \\ & = g_v((h^{\check{p}}_{V,a})^{-1}(e_0)) & \text{Definition von } v^{\check{p}}_{1,a} \\ & = g_v(\omega^3_a(e_0)) & \text{Definition von } \omega^3_a. \end{array}$$

Die verschiedenen Abbildungen und Definitionen werden im Beispiel B.11 erläutert.

5.7.2 Die Wegeäquivalenz

Der nächste Satz zeigt, daß jeder erlaubte Weg w in G ein Urbild w' in G^3 besitzt.

Satz 5.7 *Sei $G := (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, $G^3 = (V^3, E^3, \alpha^3, \omega^3)$ dessen Wegegrapherweiterung, dann gilt:*

Zu jedem Weg $w = (e_1, ..., e_n)$ im Ausgangsgraphen G unter Berücksichtigung der Wegeverbote existiert ein Weg $w' = (e'_1, ..., e'_n)$ in G^3 mit:

$$\begin{array}{lll} \Pi^3_E(e'_i) & = e_i & \text{für } i = 1, ..., n \\ \Pi^1_V(\omega^3(e'_i)) & = \Pi^1_V(\alpha^3(e'_{i+1})) & \text{für } i = 1, ..., n-1. \end{array} \tag{5.17}$$

Beweis:

Idee der Konstruktion von w': Der Weg w' verläuft bis zur ersten Kante $e_i \in E_0$ identisch mit w im nicht wegeverbotsbeeinflußten Teil des Graphen (Fall a). Ab der Kante e_i verläuft der Weg w'

innerhalb der Wegeverbotserweiterungen, bis er entweder endet oder diese wieder verläßt (Fall c). Wir definieren die erste Kante des neuen Weges:

$$e'_1 := \iota^3_E(e_1).$$

Sei $e_1, ..., e_j$ bereits abgearbeitet, $\tilde{p} = (\tilde{e}_1, ..., \tilde{e}_{\tilde{n}})$ sei maximaler Endweg von $e_1 + ... + e_j$ in P mit $\tilde{q}$ gemeinsamen Kanten und $\hat{p} = (\hat{e}_1, ..., \hat{e}_{\hat{n}})$ sei maximaler Endweg von $e_1 + ... + e_{j+1}$ in P mit t gemeinsamen Kanten.

Fall a: $\tilde{p} = \lambda$, dann definieren wir

$$e'_{j+1} := \iota^3_E(e_{j+1}).$$

Fall b: $\tilde{p} \neq \lambda$, $\hat{p} \neq \lambda$, $\tilde{q} + 1 = t$ und $[\![\tilde{p}]\!] = [\![\hat{p}]\!]$. Hier bleibt der Weg w' in dem von der Klasse $[\![\tilde{p}]\!]$ erzeugten Wegebaum. Also definieren wir

$$e'_{j+1} := (h^{\hat{p}}_1)^{-1}(e_t) = e^{\hat{p}}_t.$$

Fall c: $\tilde{p} \neq \lambda$, $\hat{p} = \lambda$ und sei $e'_j = e^{\tilde{p}}_i$ aus $\tilde{E}^{\tilde{p}}_1$: Mit der Kante e_{j+1} verläßt w den von Wegeverboten beeinflußten Teil des Graphen (Abschnitt 5.4). Damit definieren wir

$$e'_{j+1} := (h^{\tilde{p}}_2)^{-1}((e'_j, e_{j+1})) = e^{\tilde{p}}_{i,e_{j+1}}.$$

Fall d: $\tilde{p} \neq \lambda$, $\hat{p} \neq \lambda$, ($\tilde{q} + 1 \neq t$ oder $[\![\tilde{p}]\!] \neq [\![\hat{p}]\!]$) und sei $e_j = \hat{e}_t = \tilde{e}_i$: Hier verläßt der Weg w den von der Klasse $[\![\tilde{p}]\!]$ erzeugten Wegebaum, bleibt aber in dem von Wegeverboten beeinflußten Teil des Graphen (Abschnitt 5.5). Wir definieren:

$$e'_{j+1} := (h_3)^{-1}((e^{\tilde{p}}_i, e^{\hat{p}}_t)) = e^{\tilde{p},\hat{p}}_{i,t}.$$

w' ist ein Weg:

Zu zeigen ist $\omega^3(e'_j) = \alpha^3(e'_{j+1})$ für $1 \leq j \leq |w| - 1$. Dazu unterscheiden wir jede mögliche Zweierkombination der obigen Fälle a-d, also insgesamt 16 Kombinationen.

Sei $e'_j = \iota^3_E(e_j)$ im Fall a erschaffen worden.

1. $e_j \notin E_0$. Damit ist $\tilde{p} = \lambda$, also gilt nach Fall a $e'_{j+1} = \iota^3_E(e_{j+1})$, und mit der Wegeigenschaft von w folgt die Behauptung.

2. $e_j = \tilde{e}_1$ und $e_{j+1} = \tilde{e}_2$ mit $\tilde{p}$ ist maximaler Endweg von $e_j + e_{j+1}$ in P. Trivialerweise gilt, daß das erste Wegeverbot $\tilde{p}_1$ der Klasse $[\![\tilde{p}]\!]$ maximaler Endweg des Weges e_j in P ist. Damit sind die Voraussetzungen von Fall b erfüllt und

$$\begin{array}{lll} \omega^3(e'_j) & = \omega^3(\tilde{e}_1) & \text{Voraussetzung} \\ & = v^{\tilde{p}_1}_1 & \text{nach Gleichung (5.1)} \\ & = \alpha^3(e^{\tilde{p}}_2) & \tilde{p} \text{ ist maximaler Endweg} \\ & = \alpha^3(e'_{j+1}) & \text{Def. von Fall b} \end{array} \tag{5.18}$$

3. $e_j = \tilde{e}_1$ und es existiert kein Wegeverbot p', welches Endweg von $(e_1, ..., e_{j+1})$ in P ist. Damit ist $(\tilde{e}_1, e_{j+1}) \in N$ und $(h_2)^{-1}((\tilde{e}_1^{\tilde{p}}, e_{j+1})) = e_{1,e_{j+1}}^{\tilde{p}}$. Hier sind die Voraussetzungen von Fall c gegeben und

$$\begin{array}{rll} \omega^3(e'_j) & = v_1^{\tilde{p}} & \text{siehe oben} \\ & = \alpha^3(e_{1,e_{j+1}}^{\tilde{p}}) & \text{nach Gleichung (5.10)} \\ & = \alpha^3(e'_{j+1}) & \text{Def. von Fall c} \end{array} \tag{5.19}$$

4. $e_j = \tilde{e}_1$ und es existiert kein Wegeverbot $p \in [\![\tilde{p}]\!]$ mit p ist Endweg von $(e_1, ..., e_{j+1})$ in P, aber $e_{j+1} \in E_0$, das heißt, es existieren Wegeverbote die mit der Kante e_{j+1} beginnen. Sei $\check{p} = (\check{e}_1, ..., \check{e}_{\check{n}})$ mit $e_{j+1} = \check{e}_1$ das erste Wegeverbot dieser Klasse. Dann ist $\check{p}$ der maximale Endweg von $e_1 + ... + e_{j+1}$. Dies sind die Voraussetzungen von Fall d. Es ist $(e_1^{\tilde{p}}, e_1^{\check{p}}) \in M$ und $(h_3)^{-1}((e_1^{\tilde{p}}, e_1^{\check{p}})) = e_{1,1}^{\tilde{p},\check{p}}$. Damit gilt:

$$\begin{array}{rll} \omega^3(e'_j) & = v_1^{\tilde{p}} & \text{siehe oben} \\ & = \alpha^3(e_{1,1}^{\tilde{p},\check{p}}) & \text{nach Gleichung (5.13)} \\ & = \alpha^3(e'_{j+1}) & \text{Def. von Fall d.} \end{array} \tag{5.20}$$

Sei $e'_j = e_i^{\tilde{p}}$ im Fall b erschaffen worden.

1. Die Voraussetzung von Fall a kann nicht erfüllt sein.
2. $e_j = \tilde{e}_i$ und $\check{p} = (\check{e}_1, ..., \check{e}_{\check{n}}) \in [\![\tilde{p}]\!]$ ist maximaler Endweg von $e_1 + ... + e_{j+1}$ in P. damit gilt $e_{j+1} = \check{e}_{i+1}$, und die Behauptung folgt mit der Definition von α^1 analog zu (5.18).
3. $e_j = \tilde{e}_i$ und es existiert kein Wegeverbot p mit p ist Endweg von $e_1 + ... + e_{j+1}$ in P, und damit ist $(\tilde{e}_i, e_{j+1}) \in N$ und die Behauptung folgt mit der Definition von α^2 analog zu (5.19).
4. $e_j = \tilde{e}_i$ und $\check{p} = (\check{e}_1, ..., \check{e}_{\check{n}})$ ist maximaler Endweg von $e_1 + ... + e_{j+1}$ in P mit t gemeinsamen Kanten und $t < i + 1$ oder $[\![\tilde{p}]\!] \neq [\![\check{p}]\!]$. Dies sind die Voraussetzungen von Fall d, und mit der Definition von α^3 folgt die Behauptung analog zur Gleichung (5.20).

Die restlichen Kombinationen zeigt man analog.

Zu zeigen sind weiter die Gleichungen aus (5.17):

$\Pi_E^3(e'_{j+1}) = e_{j+1}$:

Fall a: Die Behauptung folgt aus Gleichung (5.4).

Fall b: Hier gilt nach der Definition des Endweges $e_{j+1} = \hat{e}_t$ und damit gilt

$$\begin{array}{rll} \Pi_E^3(e'_{j+1}) & = \Pi_E^3(e_t^{\hat{p}}) & \text{Definition von } e'_{j+1} \\ & = \Pi_E^1(e_t^{\hat{p}}) & \text{da } e_t^{\hat{p}} \in \tilde{E}_1 \\ & = \hat{e}_t & \text{nach Gleichung (5.2)} \\ & = e_{j+1} & \text{siehe oben} \end{array}$$

Fall c: Hier wurde $e'_{j+1} := e^{\tilde{p}}_{i,e_{j+1}}$ definiert.

$$\begin{array}{lll} \Pi^3_E(e'_{j+1}) & = \Pi^3_E(e^{\tilde{p}}_{i,e_{j+1}}) & \text{Definition von } e'_{j+1} \\ & = \Pi^2_E(e^{\tilde{p}}_{i,e_{j+1}}) & \text{da } e^{\tilde{p}}_{i,e_{j+1}} \in \hat{E}_2 \\ & = e_{j+1} & \text{nach Gleichung (5.11)} \end{array}$$

Fall d:

$$\begin{array}{lll} \Pi^3_E(e'_{j+1}) & = \Pi^3_E(e^{\tilde{p},\hat{p}}_{i,t}) & \text{Definition von } e'_{j+1} \\ & = \hat{e}_t & \text{nach Gleichung (5.14)} \\ & = e_{j+1} & \text{nach der Definition des Endweges} \end{array}$$

$\Pi^1_V(\omega^3(e'_j)) = \Pi^1_V(\alpha^3(e'_{j+1}))$:

$$\begin{array}{lll} \Pi^1_V(\omega^3(e'_j)) & = \omega(\Pi^3_E(e'_j)) & \text{nach Satz 5.1} \\ & = \omega(e_j) & \text{siehe oben} \\ & = \alpha(e_{j+1}) & w \text{ ist ein Weg} \\ & = \alpha(\Pi^3_E(e'_{j+1})) & \text{siehe oben} \\ & = \alpha(\Pi^3_E(e'_{j+1})) & \text{nach Satz 5.1} \end{array}$$

■

Mit dem folgenden Satz zeigen wir, daß die Projektion jedes Weges in G^3 ein erlaubter Weg in G ist.

Satz 5.8 *Sei $G = (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, $G^3 = (V^3, E^3, \alpha^3, \omega^3)$ dessen Wegegrapherweiterung und sei $w' := (e'_1, ..., e'_n)$ ein beliebiger Weg in G^3, dann gilt:*

$(\Pi^3_E(e'_1), ..., \Pi^3_E(e'_n))$ *ist ein Weg in G, der keinen verbotenen Teilweg enthält.*

Beweis:

Weil nach Satz 5.1 Π^3_E die Inzidenz erhält, ist der Beweis für die Wegeigenschaft trivial. Den zweiten Teil des Satzes zeigen wir indirekt:

Sei e'_i $i = 1, ..., n$ ein Weg in E^3, dessen Projektion in G einen verbotenen Teilweg p enthält. Wir können oBdA annehmen, daß $\Pi^3_E(e'_i)$ $1 \leq i \leq n$ der verbotene Teilweg selbst ist und damit gilt $p = (e_1, ..., e_n) = (\Pi^3_E(e'_1), ..., \Pi^3_E(e'_n))$ und p ist der maximale Endweg von $(\Pi^3_E(e'_1) + ... + \Pi^3_E(e'_n))$ in P.

Schritt 1: Sei $e'_1 = \iota^3_E(e_1)$, dann gilt $e'_2 = \psi^p(e_2)$. Ferner sei p_1 das erste Wegeverbot der Klasse $[\![p]\!]$ (das kann man wegen Satz 5.6 annehmen).

Beweis: $\omega^3(e'_1) = v_1^{p_1}$ und mit der Wegeigenschaft folgt $\alpha^3(e'_2) = v_1^{p_1}$. Da p Endweg von $(e'_1 + e'_2)$ ist, kann e'_2 keine Kante sein, die in den Abschnitten 5.4 oder 5.5 erschaffen wurde. Nach Satz 5.4 ist $e'_2 \in \tilde{E}_1$. Sei $\hat{p} = (\hat{e}_1, ..., \hat{e}_{\hat{n}})$ der maximale Anfangsweg von $(e'_1 + e'_2)$ in $[\![p]\!]$, dann gilt nach Gleichung 5.5: $\psi^p(e_2) = e_2^{\hat{p}}$.

Analog zeigt man $e'_i = \psi^p(e_i)$. Die Kante e^p_n aber wird nicht erschaffen. Damit existiert die letzte Kante e_n dieses Weges nicht. Die Konstruktion des erweiterten Graphen will genau dies erreichen.

$e'_1 \notin \tilde{E}^2$: Wir führen diesen Beweis indirekt. Sei $e'_1 \in \tilde{E}^2$ und $\hat{p} = (\hat{e}_1, ..., \hat{e}_{\hat{n}})$ das Wegeverbot, von dessen Kante $\hat{e}^{\hat{p}}_t$ die Wegeverbotserweiterung mit dem Ausgangsgraphen über die Kante $e^{w'}_1$ verbunden wird (Abschnitt 5.4). Es gilt dann, daß $e_1 + ... + e_j + \Pi^3(e'_1)$ ein Endweg von p mit einer gemeinsamen Kante ($e_1 = \Pi^3(e'_1)$) ist. Dies steht im Widerspruch zur Bedingung 5.7.

Schritt 2: Für $2 \leq i \leq n-1$ gilt $e'_i \in \tilde{E}^1 \cup \tilde{E}^3$.

Zuerst zeigen wir, daß aus $e'_i \in \tilde{E}^1 \cup \tilde{E}^3$ folgt $e'_{i+1} \notin E$:

$$\begin{array}{lll} \alpha^3(e'_{i+1}) & = \omega(e'_i) & \text{da } w' \text{ ein Weg ist} \\ & \in \tilde{V}^1 (\notin V) & \text{nach Satz 5.4} \end{array}$$

und mit Satz 5.4 folgt $e^{w'}_{i+1} \notin E$.

Den Beweis, daß e'_i nicht in $\tilde{E}^2$ ist, führen wir indirekt: Annahme $\exists i$ mit $e'_i \in \tilde{E}^2$, und falls es mehrere solcher i gibt, sei i minimal gewählt. Aus der Definition des maximalen Endweges folgt für jedes Wegeverbot $\check{p}$, dessen Wegeverbotserweiterung w' durchläuft, gilt $\check{p}$ ist Endweg von p. Wegen der Bedingung 5.7 wird also von $\check{p}$ aus keine Kante aus $\tilde{E}^2$ erschaffen, die in Richtung p verläuft.

Damit kann von der Kante e'_{n-1} nicht in eine Kante f abgebogen werden, für die $\Pi^3_E(f) = (e_n)$ gilt, da nach Konstruktion der Mengen N und M in Gleichung (5.8) keine derartige Kante in $\tilde{E}^1 \cup \tilde{E}^3$ liegen kann. ∎

Aus den Sätzen 5.7 und 5.8 folgt die Wegeäquivalenz.

5.7.3 Die Minimalität

Satz 5.9 *Sei $G = (V, E, \alpha, \omega)$ ein Graph mit Wegeverboten P, $G^3 = (V^3, E^3, \alpha^3, \omega^3)$ dessen Wegegrapherweiterung, dann gilt:*

Zu jeder Kante $e' \in E^3$ existiert ein (erlaubter) Weg $w = (e_1, ..., e_n)$ in G mit der Eigenschaft, daß alle Wege $w' = (e'_1, ..., e'_n)$ aus G^3 mit $\Pi^3_E(e'_i) = e_i$, $1 \leq i \leq n$ und

$$(*) \qquad \alpha^3(e'_1) \in V$$

die Kante e' enthalten. Würde man also e' aus G^3 entfernen, so wäre der erlaubte Weg w in G nicht mehr möglich (dies steht im Widerspruch zu (VE 2) aus Definition 2.12). Aus G^3 darf also keine Kante weggelassen werden. Damit liegt eine lokale Minimalität vor (vergleiche auch die Minimalität von Abschnitt 4.5).

Beweis : Sei $e' \in E^3$ fest vorgegeben:

1. Fall: Sei $e' \in E$, dann definieren wir den Weg $w := e'$. Es gilt nach Satz 5.4 $\alpha^3(\iota_E^3(e')) \in \iota(V)$ und für jede andere Kante e'' aus $(\Pi_E^3)^{-1}(e')$ gilt $\alpha^3(e'') \notin \iota(V)$. Damit ist w einer der gesuchten Wege.

2. Fall: Sei $e' \in \tilde{E}^1$, dann gilt, es existiert ein Wegeverbot $\tilde{p} = (\tilde{e}_1, ..., \tilde{e}_{\tilde{n}})$ mit $e' = (h_1^{\tilde{p}})^{-1}(e_i) = e_i^{\tilde{p}}$ $i < n$. Wir definieren den Weg $w := (\tilde{e}_1, ..., \tilde{e}_i)$. Sei $w' := (e'_1, ..., e'_i)$ ein Weg, dessen komponentenweise Projektion $= w$ ergibt, dann gilt nach Voraussetzung $\alpha^3(e'_1) \in V$ und damit gilt nach Satz 5.4 $e'_1 = \iota^3(\tilde{e}_1)$. Damit ist analog zum Beweis des Satzes 5.8 (erster Schritt) $e'_j = \psi^{\tilde{p}}(\tilde{e}_j)$ $1 \leq j \leq i$. Damit ist der Verlauf von w' eindeutig in G^3 festgelegt, und das Weglassen der Kante e'_i hätte zur Folge, daß kein Weg w' mit der Eigenschaft $(*)$ Urbild vom erlaubten Weg w ist.

3. Fall: Sei $e' \in \tilde{E}^2$ und $e := \Pi_E^2(e')$, dann existiert ein Wegeverbot $\tilde{p} = (\tilde{e}_1, ..., \tilde{e}_{\tilde{n}})$ und ein Index i mit $e' = (h_2)^{-1}((\tilde{e}_i^{\tilde{p}}, e)) = (e_{i,e}^{\tilde{p}})$. Wir definieren den Weg $w := (\tilde{e}_1, ..., \tilde{e}_i, e)$. Sei $w' := (e'_1, ..., e'_i, e'_{i+1})$ wie im Fall 2 ein Weg, dessen komponentenweise Projektion $= w$ ergibt, dann gilt analog zum Fall 2, daß der Weg $(e'_1, ..., e'_i)$ eindeutig bestimmt ist. Nach dem Satz 5.4 ist aber auch die Kante e'_{i+1} eindeutig, und analog zum Fall 2 folgt die Behauptung, daß e' nicht weggelassen werden darf.

4. Fall: Sei $e' \in \tilde{E}^3$, dann gilt, es existieren zwei Wegeverbote $\tilde{p} = (\tilde{e}_1, ..., \tilde{e}_{\tilde{n}})$ und $\hat{p} = (\hat{e}_1, ..., \hat{e}_{\hat{n}})$ und zwei Indices i und t mit $e' = (h_3)^{-1}((e_i^{\tilde{p}}, e_t^{\hat{p}})) = e_{i,t}^{\tilde{p},\hat{p}}$. Wir definieren den Weg $w := (\tilde{e}_1, ..., \tilde{e}_i) + \hat{e}_t$. Analog zum Fall 3 folgt die Behauptung. ∎

5.7.4 Fazit

In den Abschnitten 5.3, 5.4 und 5.5 wurde aus einem beliebigen Graphen G mit Wegeverboten ein Wegegraph G^3 konstruiert. Es bleibt zu zeigen, daß die kanonischen Projektionen die Bedingungen der Abbildungen Φ und Ψ aus der Definition 2.12 erfüllen:

VE 1: Der Satz 5.1 zeigt, daß die kanonischen Projektionen Epimorphismen (surjektiv und inzidenzerhaltend) sind.

VE 2: Im Satz 5.7 zeigen wir, daß jeder erlaubte Weg im erweiterten Graphen ein Urbild besitzt.

VE 3: Satz 5.8 besagt, daß das Bild jedes Weges in G^3 in G nicht verboten ist.

Damit ist G^3 ein Wegegraph von G mit den Epimorphismen Π_v und Π_e.

Der Satz 5.6 zeigt, daß der Wegegraph G^3 von G unabhängig von der Reihenfolge der Wegeverbote ist. Das Suchen einer bestimmten Reihenfolge könnte bis zu $|P|!$ Untersuchungen notwendig machen.

Sei m die Maximalzahl der Kanten eines Wegeverbotes und k die Maximalzahl der Kanten, die von einem beliebigen Knoten wegweisen (Ausgangsgrad des Graphen). Mit diesen Definitionen gilt $|\tilde{V}^p| \leq m$ und $|\tilde{V}^1| \leq m \cdot |P|$. Der Wegegraph besitzt also maximal $|V| + m \cdot |P|$ Knoten.

Die Anzahl der neu gezogenen Kanten ist schwieriger zu beurteilen: $|\tilde{E}_1^p| (= |\tilde{V}^p| - 1) \leq m - 1$ und damit ist $|\tilde{E}_1| \leq (m-1) \cdot |P|$. Jeder Knoten kann über maximal k Kanten verlassen werden.

Damit ist $|N^p| + |M^p| \leq m \cdot k$ und $|\tilde{E}^2| + |\tilde{E}^3| \leq m \cdot k \cdot |P|$. Damit kommen maximal

$$|\tilde{E}^1| + |\tilde{E}^2| + |\tilde{E}^3| \leq |P| \cdot m \cdot (k+1)$$

Kanten neu hinzu. Für fast vollständige Graphen ist $k \approx |E|$. Bei Straßennetzen gilt normalerweise $k \leq 5$ (maximal 5 Straßen pro Kreuzung) und $m \leq 4$ (maximal 4 Kanten pro Wegeverbot). Hier kommen pro Wegeverbot also höchstens 4 Knoten und 24 Kanten neu hinzu.

Bemerkung: Der Algorithmus betrachtet nur Wegeverbote der Länge $n \geq 2$ (so sind Wegeverbote definiert). Der Algorithmus ist auf 'Wegeverbote der Länge 1' (= Kantenverbote) erweiterbar, indem man einfach vor Beginn des Algorithmus alle verbotenen Kanten wegläßt.

Kapitel 6

Implementationsaspekte

In diesem Kapitel erfolgt eine programmtechnische Ausarbeitung. Dabei wird aus Straßenverkehrsdaten ein repräsentierender Graph erstellt. Mittels der vorher beschriebenen Methoden werden die Grundlagen eines Navigationssystems gelegt, das Abbiegeverbote berücksichtigt.

Die Lösung dieser Aufgabe vollzieht sich in 5 unabhängigen Schritten:

1. : Die Berechnung, das Erkennen oder die Eingabe der Straßenmitten
2. : Die Erzeugung eines repräsentierenden Graphen
3. : Die Erzeugung eines Wegegraphen ohne Abbiege und Wegeverbote
4. : Die Berechnung des kürzesten Weges zwischen zwei beliebigen Knoten
5. : Die graphische Darstellung des Ergebnisses

Wir werden zwei Datenformate kennenlernen, bei welchen auf verschiedene Weise die Straßenmitten bestimmt werden. Weitere Datenmodelle wie ATKIS (Amtliches Topographisches Kartographisches Informationssystem) befinden sich zum Beispiel in [Wal97], [Wal93] und [Fin90]. Zur allgemeinen Theorie der Geoinformationssysteme (GIS) siehe auch [Bar88], [Bar95] und [DeM97]. Wir werden GIS-Daten einlesen, überflüssige Information ignorieren und den Rest, auf das Wesentliche zusammengefaßt, wieder speichern. Für die Darstellung von Straßen und die Berechnung des erzeugten Graphen benötigen wir nur den Verlauf der Straßenmitten, Abbiegeverbote und Attribute.

6.1 Interne Datenformate

Um Graphen und deren kürzeste Wege darstellen und berechnen zu können, verwenden wir folgende drei Datenformate:

6.1.1 Das Datenformat der Straßenränder und Straßenmitten (Format 1)

Da Straßenmitten und Straßenränder ähnliche Struktur besitzen, werden sie im gleichen Datenformat gespeichert. Im Folgenden sprechen wir also allgemein nur noch von Straßen.

Jede Straße besitzt drei Nummern, die im record rId=(Nummer,Name,AI) zusammengefaßt sind. Dabei ist 'Nummer' die Nummer (Position) der Straße im Straßennetz, 'Name' ist die eindeutige Identifikationsnummer der Strasse und 'AI' gibt den Arrayindex im Array 'Strasse' der Unit KlDisk an, wenn die Straße in den Hauptspeicher geladen wurde.

Straßen werden durch folgenden Datenverbund (record rStrasse) dargestellt:

Id	=	Variable vom Typ rId
Stat	=	gibt an, ob die Straße verändert oder gelöscht wurde
FP	=	Position der Straße auf der Festplatte
StartX, StartY	=	Koordinaten des Startpunktes der Straße
EndeX, EndeY	=	Koordinaten des Endpunktes der Straße
Von, Nach	=	Nummern des Start- bzw. Endknotens
Typ	=	Quelle der Straße
pStart, pEnde	=	Start und Ende der Zwischenpunktliste
MaxX,MaxY,MinX,MinY	=	Ausdehnung der Straße (überdeckendes achsenparalleles Rechteck)

Zusatzinformationen wie Abbiegeverbote oder Einbahnstraßen werden in separaten verketteten Listen gespeichert und verwaltet und beim Speichern auf Festplatte hinter den Straßen an das Ende der Datei angehängt.

6.1.2 Das Datenformat eines erzeugten Graphen (Format 2)

Da beim erzeugten Graphen nicht mehr der genaue Straßenverlauf, sondern nur noch eine Gewichtung der Straße relevant ist, kann beim Graph auf die Zwischenpunkte sowie auf die genaue Lokalität der Straße verzichtet werden. Gespeichert wird eine Adjazenzliste, bei der zu jedem Knoten die Menge aller von ihm wegweisenden Kanten incl. deren Endpunkte und Gewicht aufgeführt sind, also das Quadrupel:

(Startknoten, Endknoten, Kantenname, Gewicht)

Abbiegeverbote werden auch hier wieder in separaten verketteten Listen verwaltet und auf der Festplatte hinter dem eigentlichen Graphen an das Ende der Datei angehängt. Einbahnstraßen erkennt man daran, daß die Kante, die die verbotene Richtung darstellen soll, nicht gezogen wurde. Nach Durchführung des Programms Knotensplitting (KnSplit) entfallen auch die Abbiegeverbote.

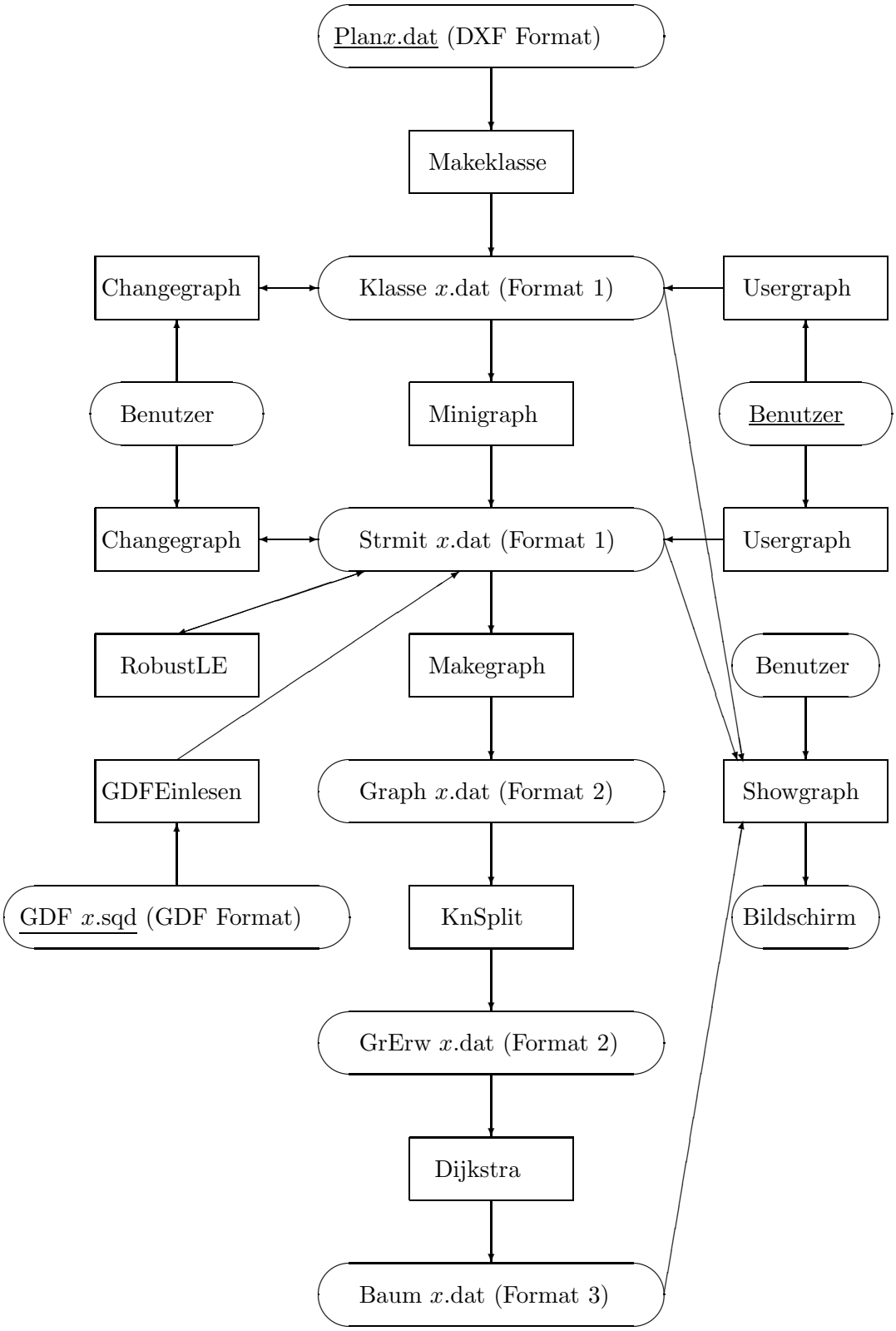
6.1.3 Das Datenformat eines kürzeste-Wege-Baumes (Format 3)

Beim kürzeste-Wege-Baum interessiert der Kantenverlauf eines Weges sowie dessen Gesamtlänge, nicht aber die Länge der Einzelkanten. Zu jedem Knoten s des Graphen wird ein kürzester-Wege-Baum in der Datenstruktur record rBaum

(Knotennummer v, Vorgängerkante, Kosten von s nach v)

gespeichert. Ist der Knoten v selbst der Startknoten oder kann der Knoten v von s aus nicht erreicht werden, so wird dies durch spezielle Wahl der Nummer der Vorgängerkante deutlich.

Im folgenden Diagramm werden die von mir entwickelten Programme mir deren Quell- und Zieldaten dargestellt. Die Anfangsdatenbestände sind unterstrichen.


Plan$x$.dat (DXF Format)
Makeklasse
Changegraph
Klasse x.dat (Format 1)
Usergraph
Benutzer
Minigraph
Benutzer
Changegraph
Strmit x.dat (Format 1)
Usergraph
RobustLE
Makegraph
Benutzer
GDFEinlesen
Graph x.dat (Format 2)
Showgraph
GDF x.sqd (GDF Format)
KnSplit
Bildschirm
GrErw x.dat (Format 2)
Dijkstra
Baum x.dat (Format 3)

6.2 Daten im AutoCAD - Austauschformat DXF

Die Stadtgrundkarte von Stuttgart, welche mir zur Verfügung gestellt wurde, ist im DXF-Format (Austauschformat von Auto-CAD siehe [Born95]) gespeichert. Die Stadtgrundkarte ist in verschiedene Schichten (Layer) eingeteilt. Eine Schicht beinhaltet die Straßenumrisse. Um diese Schicht von den anderen (zum Beispiel Bauwerke ...) zu trennen, verwende man das Programm 'LayerSor'. Zu weiteren Informationen über Datenseparation mit Schichten siehe auch [Bur86].

Meine Aufgabe war es nun, aus den Straßenrändern der Stadtgrundkarte die Straßenmitten zu errechnen. Um aus den Straßenrändern die Straßenmitten zu errechnen, fassen wir zunächst die Straßenrandstücke in Zusammenhangskomponenten zusammen (Programm Makeklasse). Danach berechnen wir punktweise die Mitte zwischen zwei benachbarten Zusammenhangskomponenten (Programm Minigraph).

6.2.1 Das Programm Makeklasse

Das Programm 'Makeklasse' faßt die Straßenrandstücke in Zusammenhangskomponenten (= Klassen) zusammen. Der Einfachheit halber interpretiere ich Kreissektoren und Scheitel als Linienelemente (wie 'Line'). Hier wäre eine Verfeinerung denkbar. Die Linienelemente werden sukzessive abgearbeitet. Ein Linienelement gehört zu einer Komponente, wenn sein Anfangs- oder Endpunkt von ihrem Anfang bzw. Ende weniger weit entfernt ist als 'epsNbar' (hier = 1m). Ist dies der Fall, wird das Linienelement am Anfang bzw. Ende der Komponente angegliedert (Prozeduren Startwechseln bzw. Endeangliedern). Gehört ein Linienelement zu keiner Komponente, so wird für dieses eine Neue erschaffen (Prozedur Neueklasse). Nachdem alle Daten eingelesen worden sind, oder wenn zu viele Komponenten benötigt werden, werden die bisher gespeicherten Komponenten untereinander auf Zusammenhang überprüft und gegebenenfalls verkettet (Prozedur Klpruef) und, falls die Klassen geschlossen (Prozedur PruefKlasseVoll) sind, abgespeichert.

Um die Straßenmitten besser berechnen zu können, werden bei langen Linienelementen zusätzlich Hilfszwischenpunkte (spätestens alle 4m) gesetzt.

Das Programm 'Makeklasse' liest das File 'PlanJ.Dat' ein, welches dem Straßenlayer des DXF Files entspricht. Das Ergebnis wird in 'KlasseJ.Dat' abgespeichert.

Probleme: In der Stadtgrundkarte sind neben den Straßenrändern auch funktionsfremde Linien wie zum Beispiel Treppen auf dem gleichen Layer abgespeichert. Im Programm werden deshalb schon diverse kleine Klassen gelöscht. Leider sind aber brauchbare und unbrauchbare Klassen nicht immer zu unterscheiden. Um nicht zu viele Klassen zu löschen, werden einige der unbrauchbaren Klassen mit abgespeichert. Diese können mit dem Programm 'Changegraph' nachträglich entfernt werden.

6.2.2 Das Programm Minigraph

Das Programm 'Minigraph' berechnet die Mittellinie (Straßenmitte) zwischen zwei benachbarten Zusammenhangskomponenten, bis deren Abstand eine vorgegebene Größe übersteigt (Option

/mul=??). Zunächst wird entschieden, ob zwei Klassen benachbart sind (in der Prozedur Makeminigraphen). Nachbarschaft erkennt man daran, daß die zwei Klassen nicht weit voneinander entfernt sind, und sich zwischen ihnen keine weitere Klasse befindet.

Seien X^1 und X^2 die Polygonzüge zweier benachbarter Klassen. Wir berechnen die zwei bestapproximierenden Punkte beider Klassen X_p^1 und X_q^2 (Prozedur Proxima), deren Abstand minimal ist. Von dort aus vergleichen wir die Orientierung der Klassen. Ist diese gleich, so wird die Orientierung einer Klasse geändert (Prozedur Invertliste). Der erste Punkt der Straßenmitte ist das arithmetische Mittel von X_p^1 und X_q^2. Die weiteren Punkte der Straßenmitte berechnen sich wie folgt:

Sei die Straßenmitte zwischen X_i^1 und X_j^2 als letztes berechnet worden. Bestimme das Proximum X_k^2 von X_{i+1}^1 in der Klasse X^2 (Prozedur Proximum). Gilt $X_k^2 = X_j^2$ oder $X_k^2 = X_{j+1}^2$, so speichern wir deren arithmetisches Mittel als Straßenmitte und betrachten den nächsten Punkt. Gilt aber $k > j + 1$, so wurden bei der Straßenmittenberechnung einige Punkte von X^2 nicht berücksichtigt. Wir bestimmen das Proximum X_l^1 von X_{j+1}^2. Gilt $X_l^1 = X_i^1$ oder $X_l^1 = X_{i+1}^1$, so speichern wir wieder deren arithmetisches Mittel als Straßenmitte und betrachten den nächsten Punkt. Gilt nichts von alledem, so besitzen die Häuserblocks eine entartete Form, und somit kann die Straßenmitte sowieso nicht genau bestimmt werden. In diesem Fall wird kein arithmetisches Mittel berechnet und mit dem Punkt X_{i+1}^1 weiter verfahren.

Dies wird so lange fortgeführt, bis der Abstand der Punkte einen gewissen Wert übersteigt. Dann gilt die Straße als beendet, die Orientierung beider Klassen wird gewechselt, und das Ganze wiederholt sich von den Proxima aus in die andere Richtung.

Das Programm 'Minigraph' liest das File 'KlasseJ.Dat' (welches das Ergebnis von 'Makeklasse' ist) ein, und das Ergebnis wird in 'StrMitJ.Dat' abgespeichert.

Probleme: Sackgassen werden bei dieser Methode nicht erkannt, und deren Existenz führt zu Fehlern. Ein Lösungsvorschlag für das Auffinden von Sackgassen ist das Zerlegen der Häuserblocks in Richtungskomponenten. Dieser Ansatz ist sehr zeitaufwendig und funktioniert nur bedingt bei Häuserblocks mit klaren Richtungsstrukturen. Bei Kreuzungen mit kleinem Schnittwinkel oder bei schmalen Komponenten können falsche Ministraßen entstehen. Diese können mit dem Programm 'Changegraph' (Option /DS) gelöscht werden. Diese Straßenmittenberechnung führt zu Ungenauigkeiten am Straßenende.

Abbiegeverbote und Attribute werden in diesem System nicht erfaßt und müssen nachträglich eingegeben werden (Programm Changegraph).

6.3 Das GDF-Datenmodell

Die erste Version des Prä-Standards Geographic Data File (GDF, Release 1.0) wurde 1988 aus dem EUREKA-Projekt DEMETER entwickelt. EUREKA ist eine Initiative der Europäischen Industrie zur Unterstützung der Wettbewerbsfähigkeit durch Zusammenarbeit. DEMETER (**D**igital **E**lektronic **M**apping of **E**uropean **TER**ritory) ist ein EUREKA unterstütztes Projekt, welches

von den Firmen Bosch und Philips initiiert wurde, um einen Datenstandard für digitale Straßennetze und Fahrzeugnavigation zu erhalten. Hieraus entwickelte die europäische Elektronik- und Automobilindustrie in der Task Force European Digital Road Map den GDF 2.0 und 3.0 ([Her91]).

'In jedem Vektordatenmodell ist der Punkt der Träger der geometrischen Information ([Fra83])'. Alle höheren Strukturen (Linien, Flächen ...) bauen auf Punkten auf. Im Zentrum des GDF-Daten-Modells steht das (geographische) Objekt ('feature'). Je nach Ausprägung ordnen wir jedes Objekt einer der folgenden 4 Objektklassen zu:

1. punktförmiges Objekt (Knoten)
2. linienförmiges Objekt (Polygon)
3. flächenförmiges Objekt (Innenfläche eines geschlossenen Polygons)
4. komplexes Objekt (aus den obigen Objekten zusammengesetztes Objekt)

Die GDF (Release 2.0)-Gesamtdokumentation von [Her91] gliedert sich in 9 Kapitel:

Das erste Kapitel enthält eine Einführung in das GDF-Datenformat und eine Einführung in das konzeptionelle Datenmodell.

Im zweiten Kapitel werden die zu beschreibenden Objekte (Features) dargestellt. Features sind Objekte der realen Welt, welche in Form und Ort zeitbeständig sind. Damit sind z.B. Straßen und Fährlinien Objekte, aber Personen oder Fahrzeuge sind keine Objekte. Objekte können zu Objektgruppen (featurethemes) zusammengefaßt werden. Zwei Objekte der gleichen Objektklasse müssen disjunkt sein. Für die KFZ-Navigation wichtige Objekte sind: Straßenelement, Kreuzung, eventuell Gebäude und Verkehrszeichen.

Das dritte Kapitel beinhaltet den Attributkatalog. Attribute sind Eigenschaften der Objekte; sind aber selbst keine Objekte. Attribute definieren auch keine Beziehung zwischen den Objekten (siehe Kapitel 4). Attribute für Straßen sind:

1. Verkehrsattribute: Durchschnittsgeschwindigkeit für PKW und LKW,Verkehrsdichte, Stauanfälligkeit, Höchstgeschwindigkeit
2. Geometrische Attribute: Breite, Länge, Passhöhe, Steigung
3. Straßenzustand: Geplante Öffnung/ Schließung der Straße, Öffnungszeiten, Fahrtrichtung, Straßenform, Baustellen, Straßentyp
4. Landschaftliche Attribute: Hausnummernstruktur, landschaftlich ansprechende Straße, Paß oder Zollstraße

Die jeweiligen Attribute haben häufig einen sehr individuellen Einfluß.

Das vierte Kapitel beschreibt, welche Zusammenhänge (Relationen) zwischen den verschiedenen Objekten bestehen. In Bezug auf Straßenelemente werden folgende Fragen geklärt:

1. In welchem Verwaltungsbezirk oder in welcher Stadt befindet sich die Straße
2. Welche Schilder bzw. Wegweiser befinden sich an der Straße
3. Welche Bauwerke (Tankstellen) befinden sich entlang der Straße
4. Wie ist die Vorfahrt geregelt
5. An welcher Kreuzung ist das Abbiegen in welche Straße verboten

Um die Abbiegeverbote zu definieren, benötigt das GDF Format drei Informationen. Das Abbiegen von der Straße e_1 aus ist an der Kreuzung v in Richtung der Straße e_2 verboten.

Das fünfte Kapitel umfaßt das Objektrepräsentationsschema (FRS). Es gibt an, mit welchen Grundbausteinen (Primitiven) reale Objekte erfaßt werden. Objekte der gleichen Klasse können in verschiedenen Objektkategorien dargestellt werden. Zum Beispiel kann eine Straße als Fläche, Linie oder als Punkt als Teil eines Dorfes interpretiert werden.

Das Objektrepräsentationsschema ist in drei Ebenen aufgeteilt: Die Ebene 0 beschreibt die Geometrie einer zwei dimensionalen Karte, wobei krumme Linien durch Polygonzüge interpoliert werden. Die Ebene 1 beschreibt die Karte in Grundbausteinen. Ein Straßenstück ist hierbei ein linienförmiges Objekt, eine Bahnstation ein punktförmiges. Dem GDF 2.0 liegt ein nicht planarer Graph zu Grunde. Eine dreidimensionale Darstellung ist allerdings nicht vorgesehen. Die Ebene 2 beschreibt zusammengesetzte Objekte.

Das sechste Kapitel beschreibt die Anforderungen an den Dateninhalt und die Datenqualität für Navigationssysteme.

Das siebte Kapitel ist der 'Globale Datenkatalog'. Er enthält zu den geometrisch-topologischen Daten Informationen wie Datenqualität oder Zeitpunkt der Datenerfassung.

Das achte Kapitel ist noch nicht veröffentlicht. Es soll die logische Datenstruktur spezifizieren.

Die Daten im GDF-Format werden in Records (Gruppe sequentieller Datenfelder, die zusammengehören) gespeichert. Das neunte Kapitel beinhaltet die Record-Struktur, in der die GDF-Daten dargestellt werden.

6.3.1 Die Record-Struktur der GDF-Daten

Jeder Daten-Record-Typ besitzt eine eigene Identifikationsnummer, den 'Record-Code'. Dafür sind die ersten beiden Zeichen im Code vorgesehen. Die Länge der einzelnen Datenrecords kann innerhalb eines Daten-Record-Typs variieren. Jedes Daten-Record besitzt eine im Daten-Record-Typ eindeutige zehnstellige Identifikationsnummer (Zeichen 3-12). Jedes Record wird in 80 Zeichenblöcken übertragen. Ist das Record kürzer als 80 Zeichen, so wird es mit Blancs aufgefüllt und das Zeichen Nr. 80 bekommt den Wert '0'. Ist das Record mindestens 80 Zeichen groß, so wird ein weiterer 80 Zeichenblock angehängt, und das Zeichen Nr. 80 bekommt den Wert '1'. Ich möchte kurz alle für die Straßenverkehrsdaten wichtigen Daten-Record-Typen beschreiben.

1. Das XY-Koordinaten-Record (Nr. 22). Dieses bildet den Grundbaustein für jede geometrische Information im GDF-Format. Es gibt zwei Typen von Koordinaten: Die *Punkte*, bei welchen nur eine $x-$ und y- Koordinate gespeichert ist, und die *Polygonzüge*, bei welchen mehrere Koordinatenpaare in einem Daten-Record abgespeichert sein können. Die Anzahl der Koordinatenpaare befindet sich in den Zeichen 21-25. Jede Koordinate besitzt genau zehn Stellen.

2. Das Knoten-Record (Nr. 25). Dieses sagt, welche Koordinaten aus dem Koordinaten-Record (Nr. 22) einen Knoten mit Eigenschaften darstellen und weist diesen eine neue Identifikationsnummer zu. Die Zeichen Nr. 13 bis 22 verweisen dabei auf den Koordinaten-Code und die Zeichen Nr. 33 und 34 beschreiben den Knotenstatus: 1 = Grenzknoten, 2 = normaler Knoten und 3 = Grenzknoten der Datenmenge.

3. Das Kanten-Record (Nr. 28). Dieses beschreibt, welche zwei Knoten (Nr. 25) mit welchen Zwischenpunkten (Nr. 22) zusammen eine Kante bilden. Dabei bilden die Zeichen Nr. 13 bis 22 den (Koordinaten-)Code des Polygonzuges, der die Zwischenpunkte darstellt, die Zeichen Nr. 23-32 den (Knoten-)Code des Startknotens und die Zeichen 33 bis 42 den (Knoten-)Code des Endknotens.

4. Das Punktobjekt-Record (Nr. 37). Dieses beinhaltet, welcher Knoten ein punktförmiges Objekt darstellt und klassifiziert dieses. Die Zeichen Nr. 18 bis 21 enthalten den Objektklassen-Code. Dabei bedeutet Code Nr. 4120 Kreuzung, Code Nr. 4220 Bahnübergang und Code Nr. 7500 Brücke/Tunnel. Die Zeichen Nr. 27 bis 36 bilden den Code des Knotens (Nr. 25), welcher das punktförmige Objekt darstellt, und zeigen damit indirekt auf dessen Koordinaten.

5. Das Linienobjekt-Record (Nr. 38). Dieses setzt Kanten (Nr. 28) zu einem linienförmigen Objekt zusammen und klassifiziert es. Dabei enthalten die Zeichen Nr. 18 bis 21 den Objektklassen-Code. Für uns interessant ist nur der Code 4110 = Straßenelement - 4210 bedeutet Eisenbahnelement und 4310 ist ein Wasserwegelement. Die Zeichen Nr. 22 bis 26 geben an, wieviele Kanten zu einem linienförmigen Objekt zusammengefaßt werden. Die Information wird in Blöcken zu zwölf Zeichen ab dem Zeichen Nr. 27 übermittelt. Die ersten zehn Zeichen beinhalten den Kantencode, die letzten zwei Zeichen die Richtung des linienförmigen Objekts. Zusätzlich werden noch Attribute (Klasse Nr. 43), ein Objektname-Code (Klasse Nr. 41), sowie die Objektcodes der punktförmigen Objekte 'Startpunkt' und 'Zielpunkt' abgespeichert.

6. Das Namen-Record (Nr. 41). Es enthält die Namen der beschriebenen Objekte (zum Beispiel Straßennamen). Die Zeichen Nr. 18 bis 20 enthalten die Sprache, in der der Name gespeichert wird (GER = deutsch). Ab dem Zeichen 21 beginnt der Text.

7. Das Segmented-Attribut-Record (Nr. 43). Ein segmentiertes Attribut kann ein linienförmiges Objekt in verschiedene Segmente aufteilen und diesen verschiedene Werte zuweisen. Das Record definiert den Ort und die Art des Attributes. Dabei enthalten die Zeichen Nr. 13 bis 17 den Startpunkt und die Zeichen Nr. 18 bis 22 den Endpunkt der Attributsgültigkeit - also die Aufteilung. Die Zeichen Nr. 23 bis 27 besagen, wieviele Attribute

dem Attributcode zugeordnet werden sollen. Das Attribut selbst besteht dann aus dem Attributtyp (Zeichen Nr. 28 bis 29) und dem Attributwert (Zeichen Nr. 35 bis 44). Häufig vorkommende Attribute sind: 'EA' eingeschlossener Verkehrsraumtyp und 'TS' Verkehrsschild.

8. Das Relationen-Record (Nr. 49). Es enthält die Zusammenhänge zwischen den Objekten. Hier enthalten die Zeichen Nr. 13 bis 16 Relationstyp. '2101' heißt zum Beispiel Abbiegeverbot und '2111' heißt Vorfahrt. Die Zeichen Nr. 22 und 23 sagen uns, wieviele Objekte in Relation stehen. Die Zeichen Nr. 24 und 25 bestimmen deren Objekttyp (1=Punkt, 2=Linie usw.), die nächsten 10 Zeichen enthalten ihren Objektcode. Das Relationen-Record beinhaltet noch zugeordnete Attributcodes und Namencodes.

Für die Errechnung eines Straßenverkehrsgraphen sind punktförmige und linienförmige, nicht aber flächenförmige oder komplexe Objekte relevant. Für die Datenstruktur ergibt sich folgendes Bild, wobei $'\rightarrow'$ 'verweist auf' bedeutet (Objekte sind unterstrichen):

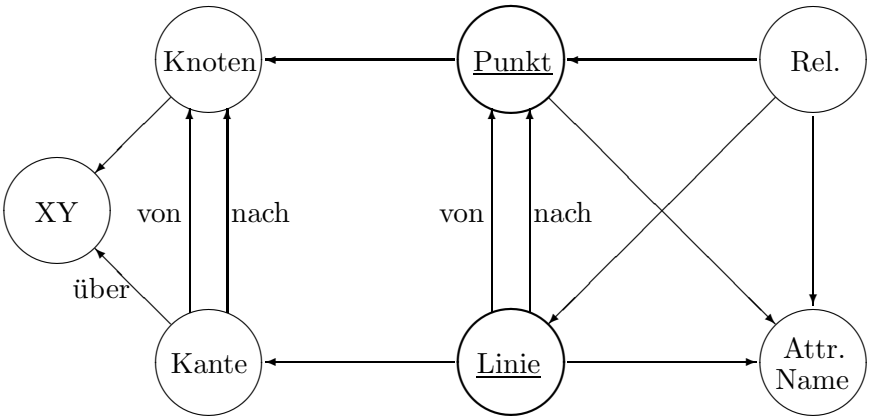


Abbildung 6.1 *Datenstruktur im GDF-Format*

Umwandlung der GDF-Daten in einen Graph:

reales Objekt	**GDF**	repräsentiert durch	**Graph**
Straße	Linefeature	Kanten Knoten Koordinaten	zwei Kanten KantenNr_1=–KantenNr_2
Kreuzung	Pointfeature	Knoten Koordinaten	Knoten
Abbiegeverbot	Relation	Straße_1 Kreuzung Straße_2	von Kante_1 nach Kante_2

6.3.2 Programm GDF-Auswerten

Das Programm 'GDFAuswerten' liest die GDF-Records ein (procedure GDF Einlesen). Die Koordinaten der einzelnen Punkte, die Knoten und die punktförmigen Objekte werden zusammengefaßt. Genauso verfahren wir mit jedem linienförmigen Objekt und den zugehörigen Kanten mit deren Zwischen- und Endpunkten. Wenn ein linienförmiges Objekt kein Straßenelement darstellt, so wird dessen Information verworfen, anderenfalls wird es als Straßenmitte inklusive Name abgespeichert. Am Ende der Straßenmittendatei speichern wir nach dem Signalwort 'Attribute' die relevanten Attribute in folgendem Format:

(orientierte Kante, Attributtyp, Attributwert) (6.1)

und nach dem Signalwort 'Verbote' die Abbiegeverbote im Format:

(von orientierter Kante, nach orientierte Kante).

Ist eine Straße eine Einbahnstraße, so wird dies als Attribut vermerkt.

6.4 Das Programm Usergraph

Um meine Programme zu testen, und um eigene Beispielstraßennetze erschaffen zu können, habe ich das Programm 'Usergraph' entwickelt, mit welchem man Straßenränder und Straßenmitten graphisch erstellen kann. Das Programm besitzt einen Graphikcursor, den man mit den normalen Cursortasten über den Bildschirm bewegen kann. '+' markiert den Anfang eines Polygonzuges bzw. Zwischenpunkte, '-' löscht den zuletzt gesetzten Polygonpunkt und 'INS' schließt den Polygonzug ab.

Das Programm erstellt eine Datei StrmitX.dat genau wie das Programm Minigraph. Hierbei sollte besonders beachtet werden, daß das Programm bisher keine Kreuzungen automatisch setzen kann. Eine Kreuzung kann mit 'Enter' nachträglich eingefügt werden. Nach jedem Setzen einer Kreuzung werden die Kreuzungen neu berechnet und angezeigt. Diesen Mechanismus kann man mit der Option /-K abschalten. Bei den nachfolgenden Programmen sollte die Option /-B wegen fehlender Straßenränder gewählt werden.

Sollen Straßenränder erzeugt werden, muß die Option /-S verwendet werden. Das Programm erstellt eine Datei im AutoCAD-Austauschformat DXF namens PlanX.dxf und eine Datei KlasseX.dat wie das Programm Makeklasse.

6.5 Umwandlung der Verkehrsdaten zu einem Graphen

Wenn wir von einem Weg sprechen, meinen wir damit ein eindimensionales Gebilde, welches auch als Parameterdarstellung einer Kurve gedeutet werden kann. Digitalisieren wir die Parameterdarstellung eines Weges, so erhalten wir einen Polygonzug, wobei gewisse Eigenschaften des

Weges zwischen den Stützpunkten des Polygonzuges als konstant angenommen werden. Dadurch verlieren Berechnungen wie zum Beispiel 'Länge des Weges' an Genauigkeit.

Der Vorteil dieser Darstellung besteht darin, daß man zu der Beschreibung eines endlichen Graphen sicher mit endlich vielen Punkten auskommt, und gewisse entartete Parameterdarstellungen von Kurven von vorneherein ausscheiden. Außerdem werden gewisse Rechenvorgänge verkürzt. Fast alle Straßenmodelle wie zum Beispiel das GDF-Datenmodell (siehe Abschnitt 6.3) basieren auf Polygonzügen.

Definition 6.1 *In einer Koordinatenebene* $\mathbb{R}^2$ *sei eine endliche Menge von Polygonzügen* $Z := \{z_1, ..., z_n\}$ *gegeben, wobei jedes* z_i *durch seine Stützpunkte* $z_i := (X_1^i, ..., X_{n_i}^i)$ *definiert ist. Dabei heißt* X_1^i **Anfangspunkt** *und* $X_{n_i}^i$ **Endpunkt** *des Polygonzuges* z_i*.*

Bei der Umwandlung von Verkehrsdaten in einen Graphen sollen die Polygonzüge zu Kanten und deren Endpunkte zu Knoten werden. Sei

$$V := \{X \in \mathbb{R}^2 \mid X \text{ ist Anfangs- oder Endpunkt eines Polygonzuges aus } Z\} \tag{6.2}$$

die Menge aller Anfangs- und Endpunkte von Z. Zu jedem $z_i \in Z$ definieren wir eine Kante e^{z_i} mit den Inzidenzstrukturen

$$\alpha(e^{z_i}) = X_1^i \quad \text{und} \quad \omega(e^{z_i}) = X_{n_i}^i.$$

Sei $E := \cup_{i=1}^{n}\{e^{z_i}\}$, dann ist $G = (V, E, \alpha, \omega)$ ein (gerichteter) Graph, der von den Polygonzügen Z *erzeugt* wird. Z heißt *Darstellung* von G.

Sei $G = (V, E, \alpha, \omega)$ ein Graph, der von den Polygonzügen Z erzeugt wird, $X, Y \in \mathbb{R}^2$, $d_e(X, Y)$ der euklidische Abstand zwischen X und Y (Luftlinie) und e^{z_j} beliebig aus E. Hier kann auf zwei Weisen intuitiv eine Gewichtsfunktion definiert werden:

$$1. \quad \beta(e^{z_j}) := \sum_{i=2}^{n_j} d_e(X_i^j, X_{i-1}^j)$$

$$2. \quad \beta_e(e^{z_j}) := d_e(X_1^j, X_{n_j}^j).$$

Haben die Polygonzüge eine gewisse Mindestanzahl von Punkten, so sind noch weitere Definitionen dieser Art denkbar. Beide Definitionen haben unterschiedliche kürzeste Wege. Dabei ist für uns die erste Definition interessant, da sie eine Fahrtroute repräsentieren kann. Ist ein Polygonzug z in beide Richtungen befahrbar, dann wird zu der bereits definierten Kante e^z eine Kante e^{-z} mit der Inzidenzstruktur $\alpha(e^{-z}) = \omega(e^z)$ und $\omega(e^{-z}) = \alpha(e^z)$ erzeugt. Gilt dies für alle Polygonzüge, dann ist der Graph ungerichtet. Die Gewichtsfunktionen können entsprechend erweitert werden. Zu jedem z_i definieren wir die Menge

$$\mathbf{z_i} := \bigcup_{l=1}^{n_i-1} \bigcup_{t\in[0,1]} X_l^i + t * (X_{l+1}^i - X_l^i).$$

Dies ist eine Kurve, welche die Menge aller Punkte des Polygonzuges enthält. Sie heißt die von z_i *erzeugte* Kurve. Sei $\varepsilon > 0$ fest, $X_j^i \in z_i$, dann definieren wir die Indizes $p \leq j$ und $q \geq j$ mit $d_e(X_k^i, X_j^i) \leq \varepsilon$ für $p \leq k \leq j$ und $d(X_{p-1}^i, X_j^i) > \varepsilon$ und $d(X_j^i, X_k^i) \leq \varepsilon$ für $j \leq k \leq q$ und $d(X_{q+1}^i, X_j^i) > \varepsilon$. Ab diesen Indizes verläßt der Polygonzug die ε-Umgebung von X_j^i. Sei K die vom Polygonzug X_{p-1}^i, ..., X_{q+1}^i erzeugte Kurve. Die Menge aller Punkte $X \in K$, für die $d(X, X_k^i) \leq \varepsilon$ gilt, heißt ε-*Umgebung* von X_k^i oder $U_\varepsilon(X_k^i)$.

Seien z_i, z_j zwei Polygonzüge, dann definieren wir eine Abstandsfunktion d:

$$d(z_i, z_j) := \min_{X \in \mathbf{z_i}} \min_{Y \in \mathbf{z_j}} d(X, Y).$$

Der Punkt $X \in \mathbf{z_i}$, für den das Minimum angenommen wird, heißt *Proximum* von z_j in $\mathbf{z_i}$.

Definition 6.2 *Sei Z eine Menge von Polygonzügen in $\mathbb{R}^2$. Für alle z_i, z_j $(i \neq j)$ soll gelten*

1. *Für $\mathbf{z_i}$ darf $X_1^i = X_{n_i}^i$ gelten, sonst muß $\mathbf{z_i}$ doppelpunktfrei sein*
2. *Für $Y \in \mathbf{z_i} \cap \bar{z_j}$ gilt ($Y = X_1^i$ oder $Y = X_{n_i}^i$) und ($Y = X_1^j$ oder $Y = X_{n_j}^j$)*

und sei $G = (V, E, \alpha, \omega)$ ein Graph, der von diesen Polygonzügen erzeugt wurde, dann heißt der Graph eben *oder* planar.

Ein von Z erzeugter Graph kann intuitiv zu einem gewichteten Graph erweitert werden: $\beta(e^{z_i}) :=$ Länge von z_i oder $\beta(e^{z_i}) :=$ Zeit, die benötigt wird, um z_i zu durchfahren. Die Länge eines Polygonzuges ist eine objektive und zeitlich beständige Größe. Deshalb ist hier eine Gewichtsfunktion leicht zu definieren. Soll die Gewichtung die Durchfahrtszeit darstellen, so besitzt diese Funktion individuellen Charakter und hängt in der Regel von vielen Parametern wie zum Beispiel Fahrzeug und Straßenbeschaffenheit ab. Der von einem Straßennetz aufgespannte Graph ist näherungsweise planar und kann auch als parallelenfrei aufgefaßt werden. Für planare parallelenfreie Graphen gilt (nach dem Eulerschen Polyedersatz) die Abschätzung

$$|E| \leq 3 * |V| - 6$$

und damit ist $|E| = 0(|V|)$.

6.6 Robustheit eines Straßennetzes

Sei ein Straßennetz durch eine Polygonzugmenge dargestellt. Die gegebene Polygonzugmenge, die einen ebenen Graphen erzeugen soll, besitzt häufig Ungenauigkeiten, die der Definition eines ebenen Graphen widersprechen. Um aus den Straßenverkehrsdaten dennoch einen ebenen Graphen erzeugen zu können, bedienen wir uns folgender im Programm 'RobustLE' (Robustheit eines linienelementweise gegebenen Graphen) realisierten Methode:

Es sei $\delta, \varepsilon \geq 0$ und eine Menge von Polygonzügen in $\mathbf{R}^2$ gegeben. δ stellt die Größe einer Kreuzung dar und soll der Weg sein, der ohne Polygonzugverbindung gerade noch 'übersprungen' werden kann. ε stellt die minimale Entfernung dar, die zwei verschiedene Straßen außerhalb eines Kreuzungsbereiches haben müssen - denkbar wäre hier zum Beispiel $\varepsilon = \frac{\delta}{3}$. Sei z_k ein Polygonzug und $\mathbf{z_k}$ dessen erzeugte Kurve.

1. Begradigung von Kreuzungen: Sei z_k ein Polygonzug und j der Index, für den gilt $d(X_1^k, X_i^k) \leq \delta$ für $i = 1, ..., j-1$ und $d(X_1^k, X_j^k) > \delta$, dann werden die Zwischenpunkte X_i^k für $i = 1, ..., j-1$ aus dem Polygonzug entfernt. Gibt es keinen Index dieser Art, so wird die Straße als Schleife mit der Länge 0 gedeutet. Analoges geschieht am Endpunkt $X_{n_k}^k$. Die Numerierung des Polygonzuges wird entsprechend geändert. Der Polygonzug, der entsteht, wenn man die δ-Umgebungen der Anfangs– und Endpunkte wegläßt, heißt $\tilde{z}_k$.

2. Aufheben von Selbstkreuzungen: Zu jedem X_j^k betrachten wir die Menge M aller Punkte Y aus $\mathbf{z_k}$, für die gilt $Y \notin U_\varepsilon(X_j^k)$ und $d(Y, X_k^i) \leq \varepsilon$, und sei $M_{\min}$ die Menge aller Punkte, für die dieser Abstand minimal wird, dann fügen wir an jedem Punkt $Y \in M_{\min}$ einen weiteren Knoten ein und trennen den Polygonzug an diesen Punkten und an Punkt X_k^i so, daß der jeweilige Trennungspunkt gleichzeitig Endpunkt des einen und Anfangspunkt des anderen neuen Polygonzuges wird. Diese neu erschaffenen Kreuzungen müssen ebenfalls begradigt werden. Dies geschieht in der Procedure 'MakeKreuzungen'.

3. Aufheben von Kreuzungen mit anderen Polygonzügen: Seien z_k und z_l $k \neq l$ zwei Polygonzüge, Q das Proximum von z_l in $\tilde{\mathbf{z}}_\mathbf{k}$ mit Abstand d. Wenn $d \leq \varepsilon$ ist, so teilen wir den Polygonzug z_k am Punkt Q.

4. Definition der Präknoten: Der Anfangs- und der Endpunkt jedes Polygonzuges heiße Präknoten. Seien Y_1, Y_2 Präknoten, dann definieren wir folgende Relation $\sim$:

   $$Y_1 \sim Y_2 \iff d(Y_1, Y_2) \leq \delta.$$

   Die Relation $\sim$ ist reflexiv, symmetrisch, aber im allgemeinen nicht transitiv. Um ein Überwechseln von allen Polygonzügen an einer Kreuzung zu gewährleisten, fordern wir für die Umwandelbarkeit des Straßennetzes in einen ebenen Graphen die Transitivität der Relation $\sim$. Mit der geforderten Transitivität ist $\sim$ eine Äquivalenzrelation.

5. Definition des Graphen: Wir definieren die Knoten des Graphen als Äquivalenzklassen der Relation $\sim$. Die Koordinaten eines Knoten berechnen sich als arithmetisches Mittel der an der Erzeugung beteiligten Präknoten. Im Polygonzug werden die Präknoten durch die entsprechenden Knoten ersetzt. Der Graph wird analog zur Definition 6.1 definiert.

Wenn bei einer Menge von Polygonzügen die Transitivität der Relation $\sim$ nicht gewährleistet ist, so ist diese ohne Ortskenntnis nicht in einen beschreibenden Graphen zu verwandeln, da die Kreuzungen nicht eindeutig zugeordnet werden können. Der Ausgangsgraph ist natürlich nicht mehr mit dem reparierten Graph äquivalent - er ist nur ansatzweise ein Untergraph desselben.

Um den Straßencharakter zu bewahren, dürfen die Winkel innerhalb der Polygonzüge nicht 'zu spitz' sein. Die Procedure 'EntschaerfeKurven' glättet optional die Polygonzüge.

6.7 Das Programm Makegraph

Die Aufgabe dieses Programmes besteht darin, aus einem durch eine Polygonzugmenge definierten Straßennetz einen Graph zu erzeugen. Dabei werden die Straßenkreuzungen errechnet, welche die Knoten darstellen. Zu jeder durch einen Polygonzug dargestellten Straße definieren wir eine Kante und deren Inzidenzstruktur. Alle hier beschriebenen Unterprogramme befinden sich in der Unit Grapherstellen. Ein Listing befindet sich in Abschnitt C.3.

6.7.1 Das Erstellen der Straßenkreuzungen

Straßenkreuzungen (Knoten) werden in der Procedure Makeknoten errechnet. Eine Straße z_i beginnt oder endet genau dann an einer Kreuzung v, wenn deren Anfangs- oder Endpunkt um weniger als δ von v entfernt ist. Ist dies bei keiner bisher definierten Kreuzung der Fall, so definiert dieser Anfangs- oder Endpunkt eine neue Kreuzung, sonst wird die Kreuzung nach dem Prinzip des arithmetischen Mittels neu berechnet.

6.7.2 Das Erstellen eines Graphen

Nach der Berechnung der Straßenkreuzungen kann nun der Graph, der das Straßennetz repräsentiert, berechnet werden. Graphen werden in der Procedure Graphbauen erstellt. Gespeichert werden von jeder Kreuzung aus alle Straßen, die von ihr wegführen, deren Endpunkt, Orientierung und Gewichtung. Diese Technik ermöglicht auch das Speichern von Graphen, bei welchen zwischen den einzelnen Knoten mehrere Straßen existieren (Graph mit Multikanten), und bei welchen Anfangs- und Endpunkt einer Straße identisch sind (wie bei Wendeplatten und ähnlichem). Das Datenformat ist also von folgender Form:

Anfangsknoten Endknoten orientierte Identifikationsnummer Gewichtung.

Die Orientierung der Straße ist in Polygonzugrichtung positiv und gegen diese Richtung negativ. Der Graph wird zunächst als verkettete Liste gespeichert und dann knotenweise auf ein Speichermedium übertragen.

Die Gewichtung einer Kante kann zum Beispiel die Länge des erzeugenden Polygonzuges sein, sie kann auch von von der Attributierung der Straße abhängen. Ist zum Beispiel die Durchschnittsgeschwindigkeit aller Straßen bekannt, kann als Gewichtung auch die benötigte Zeit verwendet werden. Die Gewichtung muß aber für alle Kanten nach einheitlichen Regeln erfolgen.

6.7.3 Die Berechnung der Fehlerellipsen

Der folgende Abschnitt ist in der Procedure MakeFehlerEllipsen realisiert. Die Straßen enden in der Regel nicht direkt am gemittelten Kreuzungspunkt. Dies hat zur Folge, daß die Kreuzungen nicht genau bestimmt werden können. Die tatsächliche Kreuzung befindet sich höchstwahrscheinlich innerhalb einer Ellipse um die errechnete Kreuzung herum. Seien (x_i, y_i) die Endpunkte der

Straßen, die an der Kreuzung K enden, m deren Anzahl und A die Darstellungsmatrix der Fehlerellipse, so berechnen sich die Koeffizienten a_{ij} wie folgt:

$$a_{11} = \sum_{i=1}^{m} (x_i - K_x)^2$$

$$a_{12} = a_{21} = \sum_{i=1}^{m} (x_i - K_x) * (x_i - K_y)$$

$$a_{22} = \sum_{i=1}^{m} (y_i - K_y)^2$$

Das Programm enthält noch die Knoten- und Fehlerellipsenausgaberoutinen sowie Funktionen, welche von einer Straße die Endknotennummern bestimmen.

Das Programm 'Makegraph' liest das File 'StrMitJ.Dat' ein, und das Ergebnis wird in 'GraphJ.Dat' abgespeichert. Der Graph wird kantenweise gespeichert. Jede Straßenmitte erzeugt zwei Kanten, mit der Ausnahme, wenn die Straße das Attribut 'Einbahnstraße' besitzt. Hier wird nur die gültige Richtung gespeichert. Am Ende der Datei stehen wieder die Attribute und Wegeverbote analog zu 6.1. Die Attribute 'Einbahnstraßen' brauchen nicht mehr gespeichert zu werden.

6.8 Das Programm Knotensplitting (KnSplit)

Es soll der Algorithmus aus Abschnitt 4.5 programmiert werden. Dazu wird zuerst ein Graph mit Abbiegeverboten eingelesen. Dieser Graph wird dann durch den Algorithmus so modifiziert, daß die Abbiegeverbote überflüssig werden und danach in der gleichen Datenstruktur ohne Abbiegeverbote wieder gespeichert. Um dem Algorithmus mehr Geschwindigkeit und Transparenz zu geben, habe ich ihn modifiziert:

Der beschriebene Algorithmus verwandelt einen gegebenen Graph $G = (V, E, \alpha, \omega)$ mit Abbiegeverboten T schrittweise in einen Graphen $G' = (V', E', \alpha', \omega')$ ohne Abbiegeverbote, wobei jede Abbiegeverbotsklasse [siehe Formel (4.4)] einen eigenen Schritt erzwingt. Nach jedem dieser Schritte wird ein Zwischenergebnisgraph G^{f_i} definiert, bei welchem (kürzeste) Wege unter Beachtung aller bereits betrachteter Abbiegeverbote bestimmt werden können. Bei jedem weiteren Schritt muß aber auf alles vorangegangene zurückgegriffen werden [siehe Definition von $\omega^{f_{i-1}}(e)$ (4.7)].

Bei der Implementation verwende ich stattdessen den Algorithmus aus Kapitel 5, der natürlich auch für Abbiegeverbote gilt, die ja spezielle Wegeverbote sind. Der gesplittete Weg aus dem Abschnitt 5.3 besteht nur aus einem einzigen Knoten. Die k-ten Schritte der Wegeverdopplung entfallen für $k > 1$. Bei der Angabe der Referenzformeln werde ich zuerst auf die Formel für die Abbiegeverbote und dann die entsprechende Formel bei den Wegeverboten verweisen.

6.8.1 Die Datenstruktur von KnSplit

Den Graphen G speichere ich in Form einer Adjazenzliste (record rKante) mit folgendem Inhalt:

VonKn	=	Nummer des Anfangsknotens der Kante
UeberKa	=	Nummer der Kante
NachKn	=	Nummer des Endknotens der Kante
KaIn_E0	=	Flag, das angibt, ob diese Kante in der Menge E_0 enthalten ist
OldVonKn	=	Nummer des Original-Anfangsknotens der Kante
OldNachKn	=	Nummer des Original-Endknotens der Kante
Entf	=	Gewicht der Kante

Stimmen die Werte von VonKn und OldVonKn sowie von NachKn und OldNachKn nicht überein, so handelt es sich um eine gesplittete Kante. Eine gesplittete Kante und ihr Original haben aber aus Darstellungsgründen immer die gleiche Identifikationsnummer - sie unterscheiden sich am Anfangs- bzw. Endknoten. KaIn_E0 ist bei gesplitteten Kanten immer falsch.

Die Abbiegeverbote werden in einer weiteren Liste (record rVerbot) mit folgendem Inhalt gespeichert:

VonKa	=	Nummer der Anfangskante des Abbiegeverbotes
UeberKn	=	Nummer des Zwischenknotens
NachKa	=	Nummer der Endkante des Abbiegeverbotes

6.8.2 Die Prozeduren von KnSplit

In der Prozedur 'Create_Menge_E0_und_splitte_Knoten' wird die Menge E_0 [siehe Formel (4.5) und Definition 2.9] erschaffen. Im gleichen Schritt wird zu jedem $e \in E_0$ der Endknoten $\omega(e)$ gesplittet und der Kante e als neuer Endknoten zugewiesen [siehe Definition von $\omega^{f_{i-1}}(f_i)$ in der Formel (4.7) und im Abschnitt 5.3 den ersten Schritt - hier gilt $n = 2$ - es wird also keine Kante gesplittet].

Bei der Prozedur 'splitte_Kanten' werden zu jeder Kante e aus E_0 alle Kanten e' gesplittet, die über die Kante e erreichbar sind [siehe Definitionen von N bzw. $\tilde{E}^e$ bei den Formeln (4.6) und (5.7)]. Danach wird diesen neuen Kanten ein Endknoten zugewiesen [siehe Definition der Abbildung $\omega^{f_{i-1}}$ in der Formel (4.7) sowie die Abschnitte 5.4 falls $e' \notin E_0$ und 5.5 falls $e' \in E_0$].

Das Programmlisting von Knsplit befindet sich im Abschnitt C.1.

6.9 Weitere Programme

Das Programm Dijkstra wendet den Dijkstra-Algorithmus (siehe Abschnitt A.1) auf den Graphen an, der von Makegraph und Knsplit erschaffen wurde, und berechnet dort die kürzesten Wege. Abbiegeverbote werden genau dann berücksichtigt, wenn vor dem Programm Dijkstra

das Programm Knsplit ausgeführt wurde. Der Ergebnisbaum wird in BaumJ.Dat abgespeichert. Eine Implementation befindet sich in Abschnitt C.2

Das Programm Layersor sortiert die Rubrik 'Entities' eines DXF-Files nach Layernummern.

Mit dem Programm Changegraph kann man Komponenten, die das Programm 'Makeklasse' nicht als falsch erkennt (zum Beispiel breite Treppen), und (eventuell) falsch berechnete Straßenmitten (Option /DS) löschen. Außerdem kann man Abbiegeverbote eingeben.

Das Programm Showgraph macht alle Ergebnisse der vorigen Programme sichtbar. Es können ein Start– und ein Zielknoten eingegeben werden (F3), und der kürzeste Weg zwischen diesen Knoten wird angezeigt. Mit F4 wird die Distanz angezeigt.

Im Anhang C befinden sich nur die Listings der beschriebenen Programme, die speziell mit Graphentheorie zu tun haben.

Kapitel 7

Ausblick

Es ist uns gelungen, vier Verfahren für Routenplaner zu entwerfen, welche die kürzesten Wege in Straßennetzen mit Wegeverboten berechnen. Diese Verfahren verändern entweder den Graphen oder den Algorithmus. Die Methode des verbotsorientierten Knotensplittings, welche den Graphen verändert, wurde auf Wegeverbote erweitert. Außerdem wurde beschrieben, wie aus Straßenverkehrsdaten ein Graph konstruiert werden kann. Dazu wurde ein Programmpaket entwickelt, welches die Ergebnisse der Arbeit anwendet.

Der vorgestellte Algorithmus ist so genau beschrieben, um mit existierenden Methoden zur Routenplanung kombiniert werden zu können. Ein weiterer interessanter Ansatz wäre die Übertragung des Verfahrens auf andere Bereiche, wie zum Beispiel die Logistik siehe dazu auch Abschnitt 5.1.

Künftige Untersuchungen könnten das Verfahren der Mehrknotenaufnahme aus Abschnitt 4.3 auf Wegeverbote erweitern. Dabei ist zu berücksichtigen, daß der Endknoten einer bestimmten Kante auch dann in den kürzeste-Wege-Baum aufgenommen werden kann, wenn dieser bereits ohne Beeinflussung von Wegeverboten von dieser Kante aus aufgenommen wurde.

Es wäre auch eine andere mathematische Vorgehensweise denkbar. Zuerst zeigt man, daß zu jedem Graphen G mit Wegeverboten Wegegraphen ohne Wegeverbote existieren. Dann definiert man Graphen G' mit Homomorphismen $\Phi : V' \to V$ und $\Psi : E' \to E$, welche die Eigenschaft 'kürzester Weg' erhalten (etwa wie bei einem topologischen Filter) und zeigt, daß diese gegen einen Wegegraphen von G konvergieren.

7.1 Schrittweises Vorgehen

Die beschriebene Lösung des Problems, (kürzeste) Wege in einem Graphen $G = (V, E, \alpha, \omega)$ mit Wegeverboten (oder auch Abbiegeverboten) P zu berechnen, betrachtet immer die Gesamtmenge aller Wegeverbote. Schön wäre es, einen Algorithmus zu finden, der ein n-Tupel von Graphen $G^i = (V^i, E^i, \alpha^i, \omega^i)$ mit Wegeverboten P^i $i = 1, ..., n$ erzeugt, wobei

1. $G^1 = G$ und $P^1 = P$
2. $P^n = \emptyset$
3. Der Algorithmus führt G^{i-1} nach G^i und P^{i-1} nach P^i über, ohne die vorherigen Schritte zu kennen.

Bei dem oben gegebenen Graph wäre folgender Algorithmus denkbar:

Sei $p = (e_1, ..., e_m) \in P$ beliebig. Wir führen den Algorithmus aus Kapitel 5 so durch, als ob nur dieses eine Wegeverbot existieren würde. Das heißt, wir verdoppeln den Weg analog zum Abschnitt 5.3 ohne die Kanten e_1 und e_n, und wir ziehen alle nicht verbotenen Kanten vom gesplitteten Weg hin zum Ausgangsgraphen (Abschnitt 5.4). Der Abschnitt 5.5 der eventuell vorher behandelte Wegeverbote einbezieht, soll ohne Bedeutung bleiben.

Wenn nach diesem Schritt nur p weggelassen, also die Wegeverbotsmenge $P \backslash \{p\}$ betrachtet wird, so wäre der Algorithmus nicht korrekt, wie folgendes Beispiel zeigt:

Gegeben sei ein Graph $G = (V, E, \alpha, \omega)$ mit $V := \{v_1, ..., v_4\}$, $E := \{e_{21}, e_{23}, e_{32}, e_{34}\}$ und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i \;\; \omega(e_{ij}) = v_j$. Zusätzlich seien folgende Wegeverbote definiert:

$$P := \{p_1, p_2\} = \{(e_{32}, e_{21}), (e_{23}, e_{34})\}$$

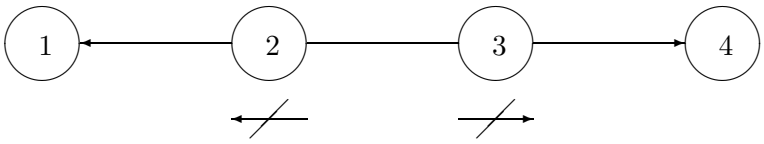


Abbildung 7.1 *Beispielgraph*

Wir betrachten die Wegeverbote in der kanonischen Reihenfolge. Zuerst wird der Knoten v_2 gesplittet (dieser heiße $v_{2'}$) und eine neue Kante $e_{2'3}$ eingefügt. Das Gleiche geschieht mit dem Knoten v_3. Für den entstandenen erweiterten Graph gilt: $V := \{v_1, ..., v_4, v_{2'}, v_{3'}\}$, $E := \{e_{21}, e_{23}, e_{32}, e_{34}, e_{2'3}, e_{3'2}\}$ und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i \;\; \omega(e_{ij}) = v_j$, außer $\omega(e_{23}) = v_{3'}$ und $\omega(e_{32}) = v_{2'}$.

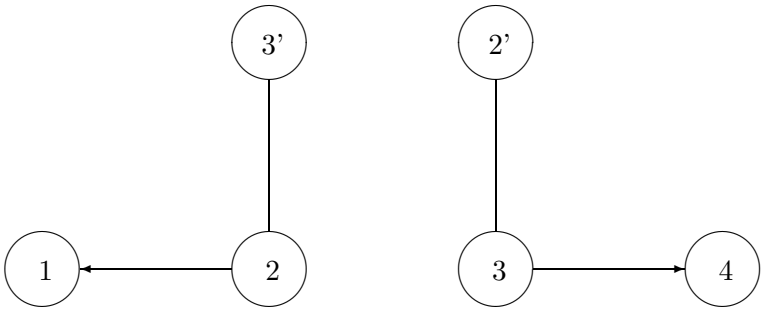


Abbildung 7.2 *Erweiterter Graph*

Der erzeugte Wegegraph ist unkorrekt, denn der Weg $(v_2, v_{3'}, v_2, v_1)$ ist möglich und im Ausgangsgraphen verboten.

Offensichtlich liegt die Unkorrektheit des Algorithmus daran, daß Kanten aus E_0 (im Beispiel sind dies die Kanten e_{23} und e_{32}) gesplittet, und deren Abbiegeverbote nicht mehr berücksichtigt wurden. Der Algorithmus muß also um folgenden Schritt erweitert werden:

Wenn eine Kante aus E_0 gesplittet wird, so werden dieser auch die zugehörigen Wegeverbote zugewiesen. In unserem Beispiel würde dies bedeuten, daß nach jedem Schritt ein weiteres Wegeverbot entstanden wäre: $p_{1'} := (e_{3'2}, e_{21})$ und $p_{2'} := (e_{2'3}, e_{34})$.

Die erneute mehrfache Anwendung des Algorithmus auf den erweiterten Graphen mit diesen Wegeverboten ergibt dann folgenden Wegegraphen ohne Wegeverbote:

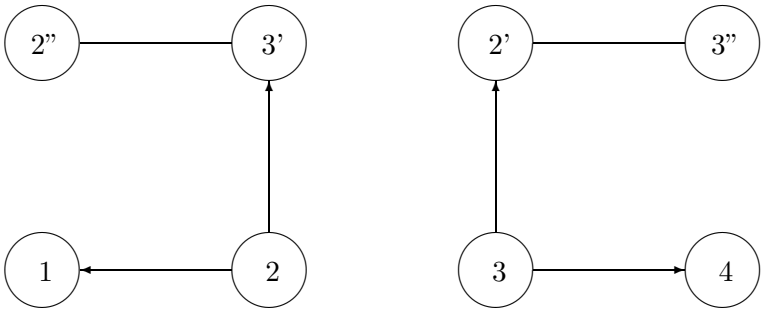


Abbildung 7.3 *Erweiterter Graph mit übertragenen Wegeverboten*

Dieser erweiterte Graph ist tatsächlich ein Wegegraph des Ausgangsgraphen. Hätte man auf den Ausgangsgraphen den Algorithmus aus Kapitel 5 angewandt, so wäre folgender Wegegraph entstanden:

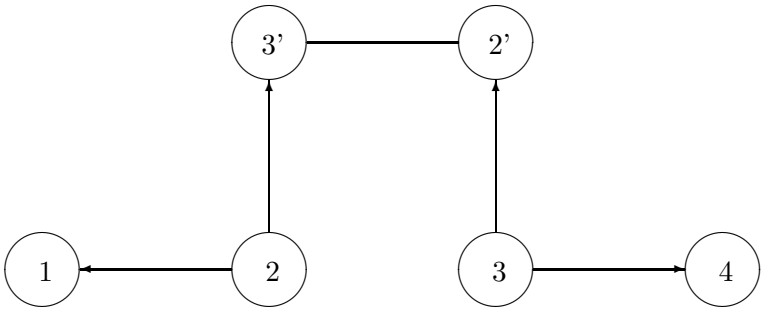


Abbildung 7.4 *Erweiterter Graph mit dem Algorithmus aus Kapitel 5*

Dieser Wegegraph ist also offensichtlich kleiner als der schrittweise entstandene Wegegraph [siehe auch Minimalitätseigenschaft in Satz 5.9]. Betrachtet man nur Graphen mit Abbiegeverboten, so ist eine schrittweise Konstruktion analog zur Methode des Ziehens neuer Kanten in Abschnitt 4.4 denkbar, die den Korrektheitsbeweis stark vereinfacht.

Für Wegeverbote hingegen ist dies nicht möglich, da der Algorithmus hier nicht determiniert: Das Wegeverbot in Beispiel 5.6.4 würde bei Durchführung des Algorithmus sich selbst nocheinmal erzeugen. Damit ist hier eine Reduktion der Wegeverbote unmöglich. Der kritische Unterschied zwischen Abbiege- und Wegeverboten ist, daß Abbiegeverbote nur an einem Knoten (Kreuzung) wirken, Wegeverbote hingegen an vielen.

Interessant wäre es ferner, einen Algorithmus zu finden, der auch in unabhängigen Schritten funktioniert und einen Wegegraphen erzeugt, der äquivalent zum minimalen Wegegraphen ist. Zur Äquivalenz von Graphen siehe auch [Reu80].

7.2 Die Verwandtschaft von Wegeverboten mit k-kürzesten Wegen

Die Probleme 'Berechnung von (kürzesten) Wegen in Graphen mit Wegeverboten' und 'Berechnung von k-kürzesten Wegen' sind verwandt. In einem Graphen können k-kürzeste Wege berechnet werden, indem man den kürzesten Weg berechnet, diesen verbietet und dies $k-1$ mal wiederholt. Die Menge aller kürzesten Wege läßt sich in zwei Typen aufteilen:

1. k- kürzeste Wege, die andere k'-kürzeste Wege ($k' < k$) enthalten,
2. k- kürzeste Wege, die keinen anderen k'-kürzeste Weg ($k' < k$) enthalten.

Der in Kapitel 5 beschriebene Algorithmus kann aber nur k- kürzeste Wege vom Typ 2 berechnen (siehe dazu Abschnitt 5.2). Um auch k- kürzeste Wege vom Typ 1 berechnen zu können müssen wir den Algorithmus verändern. Gesucht seien die k kürzesten Wege (auch die vom Typ 1) zwischen den Knoten s und z. Vor Beginn des Algorithmus definieren wir zwei neue Knoten s', z' und zwei neue Kanten e'_s, e'_z mit Kantengewicht 0 und erweitern den Ausgangsgraphen auf folgende Weise:

$$\alpha(e'_s) := s' \quad \omega(e'_s) := s \quad \alpha(e'_z) := z \quad \alpha(e'_z) := z'.$$

Wenden wir den oben beschriebenen Algorithmus auf den so erweiterten Graphen an so erhalten wir alle k- kürzesten Wege zwischen s' und z' und daraus ableitbar die k- kürzesten Wege zwischen s und z.

Bemerkung: Damit sind alle k- kürzesten Wege von Typ 1 von der Form:

Schleifen am Knoten s – k'- kürzester Weg von s nach z – Schleifen am Knoten z

In [AMMP90] wird ein Algorithmus angegeben, der die k-kürzesten Wege in einem Graphen mit einem ähnlichen Ansatz ('Path deletion') berechnet. Die Vorgehensweise des Algorithmus soll hier kurz veranschaulicht werden:

Gegeben sei ein Graph $G = (V, E, \alpha, \omega)$ und zwei Knoten s = Startknoten und z = Zielknoten. Zwischen s und z sollen nun k-kürzeste Wege bestimmt werden. Azevedo erweitert seinen Graphen um die oben beschreibenen Kanten und Knoten. Dann wird der kürzeste Weg zwischen s' und z' berechnet - dieser sei $(e_1, ..., e_m)$, wobei $\alpha(e_1) = s'$ und $\omega(e_m) = z'$ gilt. Dieser Weg wird analog zum Abschnitt 5.3 dieser Arbeit verdoppelt. Die Kanten des gesplitteten Weges seien $(e'_2, ..., e'_{m-1})$. Der Knoten z' wird mitsamt der Kante e_m an den verdoppelten Weg angehängt: $\alpha(e_m) := \omega(e'_{m-1})$. Danach wird jede Kante e gesplittet, die folgende Eigenschaften besitzt:

1. $e \neq e_i \; i = 1, ..., m$
2. Es existiert ein j mit $2 \leq j \leq m$ und $\omega(e) = \alpha(e_j)$.

Die verdoppelte Kante heiße e'. Diese wird an den entsprechenden Knoten des verdoppelten Weges angehängt: $\omega(e') := \alpha(e'_j)$. Es werden also Kanten verdoppelt, die auf die Knoten des verdoppelten Weges weisen. Dieser Schritt entspricht etwa dem Abschnitt 5.4. Der Unterschied zu meinem Ansatz besteht darin, daß ich den gesplitteten Weg mit dem Ausgangsgraphen verbinde und der Kante e_1 ein neuer Endpunkt zugewiesen wird. Wie einfache Beispiele zeigen, sind die Grapherweiterungen nicht äquivalent.

Offensichtlich ist diese Vorgehensweise ähnlich zu der Vorgehensweise in Kapitel 5. Künftige Untersuchungen könnten vesuchen, diesen Ansatz oder andere Lösungsverfahren zur Berechnung k kürzester Wege auf Wegeverbote zu verallgemeinern.

Literaturverzeichnis

[AHU83] A. V. Aho, J. E. Hopcroft, J. D. Ullman: The Design and Analysis of Computer Algorithms. Addison-Wesley Publishing Company, 1983

[ASIN91] Ausiello, G. F. Italiano, A.M. Spaccamela, U. Nanni: Incremental Algorithms for minimal length paths. *Journal of Algorithms 12* (S. 615 - 638), 1991

[AMMP90] J.A. de Azevedo, J.J. Madeira, E.Q. Martins, F. M. Pires: A shortest path ranking algorithm. AIRO '90 - Models and Methods for Decision Support, Sorrent (Italien), Proceedings of the Annual Conference, Assoziazione Italiana di Ricera Operativa (S. 1001 - 1011), Oktober 1990

[Bar95] N. Bartelme: Geoinformatik. Modelle, Strukturen, Funktionen. Springer Verlag Berlin, 1995

[Bar88] N. Bartelme: GIS Technologie. Geoinformationssysteme, Landinformationssysteme und ihre Grundlagen. Springer Verlag Berlin, 1988

[BL74] M. S. Bazarra, R. W. Langeley: A dual shortest path algorithm. *SIAM J. Appl. Math. 26* (S. 496 - 501), 1974

[Born95] G. Born: Referenzhandbuch Dateiformate. Grafik, Text, Datenbanken, Tabellenkalkulation. Addison-Wesley Deutschland GmbH, 1995

[Bra94] A. Brandstädt: Graphen und Algorithmen. B.G. Teubner Verlag, 1994

[BLW76] N. L. Briggs, E. K. Lloyd, R. J. Wilson: Graphtheory 1736-1936. Clarendon Press, Oxford, 1976

[Buck90] C. Buckley: Path-Planning Methods for Robot Motion, in Rembold, U. Robot Technology and Applications, Marcel Dekker, 1990

[BC96] F. Buchholz, V. Claus: Fahrgemeinschaften Abschlußbericht. Unveröffentlichtes Manuskript, Institut für Informatik der Technischen Universität Stuttgart, 1996

[Buc00] F. Buchholz, Hierarchische Graphen zur kürzesten Wegesuche. Dissertation an der Fakultät für Informatik der Universität Stuttgart, 2000

[Bur86] P. A. Burrough: Principles of Geographical Informationssystems for Land Resources Assessment. Oxford University Press, 1986

[Cla94] Clark John: Graphentheorie: Grundlagen und Anwendungen. Spektrum Akademischer Verlag, 1994

[CRP93] V. Claus, J, Risau, M. Papenbrock: Projekt Fahrgemeinschaften. Oldenburger Forschungs und Entwicklungsinstitut für Informatik Werkzeuge und Systeme, 1993

[Cla99] V. Claus: Vorlesung Verkehrsmodellierung. Universität Stuttgart, SS 1999

[DeP85] R. Dechter, J. Pearl: Generalized Best-First Search Strategies and the Optimality of $A*$. *Journal of the Association for Computing Machinery, 15 (3)* (S. 505 - 536), 1985

[DeM97] M. N. DeMers: Fundamentals of Geographic Informationssystems. John Wiley and Sons Inc, New York, 1997

[DeP84] N. Deo, C. Pang: Shortest-Path Algorithms: Taxonomy and Annotation. *Networks, Vol. 14* (S. 275-323), 1984

[Dij59] E. W. Dijkstra: A note on two problems in connection with graphs. Numerical Mathematics 1, Seiten 269 - 271, 1959

[Dial69] Dial, R.B.: Algorithm 360: Shortest Path Forest with Topological Ordering. *Comm. ACM 12* (S. 632 und 633), 1969

[Dor73] W. Dörfler: Graphentheorie für Informatiker. Walter de Gruyter Verlag, Berlin, 1973

[Dom72] W. Domschke: Kürzeste Wege in Graphen: Algorithmen, Verfahrensvergleiche. Dissertation an der Uni Karlsruhe, 1971

[Dre69] S. E. Dreyfus: An appraisal of some shortest-path algorithms. *Operations Res. 17* (S. 395 - 412), 1969

[Fin90] D. Findeisen: Datenstrukturen und Abfragesprachen für raumbezogene Informationen. Kirschbaum Verlag, Bonn, 1990

[Flo62] R. W. Floyd: Algorithm 97: Shortest Path. *Communications of the ACM*, 5, 1962

[Fra83] A. Frank: Datenstrukturen für Landinformationssysteme. Semantische, topologische und räumliche Beziehungen in Daten der Geowissenschaften. Dissertation am Institut für Geodäsie und Photogrammetrie an der Eidg. technischen Hochschule Zürich, 1983

[FJ89] Greg Frederickson, Ravi Janardan: Efficient Message routing in planar Networks. *SIAM J. of Comput., 18(4)* (S. 843 - 857), 1989

[Fre87] Greg Frederickson: Fast Algorithms for shortest Paths in planar Graphs, with Applications. *SIAM J. of Comput., 16(6)* (S. 1004 - 1022), 1987

[Fre76] M. L. Fredman: New bounds on the complexity of the shortest path problem. *SIAM J. Comput. 5* (S. 83-89), 1976

[FrT87] M. L. Fredman, R. E. Tarjan: Fibonacci Heaps and their Uses in improved Network Optimation Algorithms. *Journal of the ACM Vol. 34, No. 3* (S. 596 - 625), 1987

[GMO76] Gabow, H.N., Maheshwari, S. N., Osterweil, L. J.: On two Problems in the generation of program test paths. *IEEE Trans. on Softw. Eng. SE-2, Nr. 3* (S. 227 - 231), 1976

[Gib91] A. Gibbons: Algorithmic graph theory. Cambridge Univ. Press, 1991

[Hal89] R. Halin: Graphentheorie. Akademie Verlag Berlin, 1989

[Har74] F. Harary: Graphentheorie. Oldenburg Verlag München, 1974

[Har73] F. Harary: New Directions in the Theorie of Graphs. Academic Press New York, London, 1973

[Hec84] H. Heck: Anwendung von Optimierungsverfahren bei Entwurf und bei der Gestaltung von städtischen Straßennetzen unter Berücksichtigung des Betriebs. Forschung Straßenbau und Straßenverkehrstechnik Heft 492, (Bundesministerium für Verkehr Abt. Straßenbau in Bonn-Bad Godesberg) Typo Verlagsgesellschaft mbH, Bonn, 1984

[Her91] L. Heres und andere: GDF-Documentation Vol.1 - Vol.8. Task Force EDRM, 1991.

[Huc97] U. Huckenbeck: External Paths in Graphs. Foundations, Search Strategies and Related Topics. Akademie Verlag GmbH, Berlin, 1997

[Huc96] U. Huckenbeck: Searching for the k best paths. Universität Greifswald, 1996.

[Joh73] D. B. Johnson: Algorithms for shortest paths. Cornell University Ph. D. University Microfilms International, Ann Arbor, Michigan, USA, 1973

[Jun90] D. Jungnickel: Graphen, Netzwerke und Algorithmen. B.I. Wissenschaftsverlag, Mannheim, 1990.

[Kad95] Firoz Kaderali, Werner Poguntke: Graphen Algorithmen Netze Grundlagen und ANwendungen in der Nachrichtentechnik. Verlag Viehweg, Wiesbaden, 1995.

[Kann94] V. Kann: Polynamially bounded Minimization Problems that are hard to approximate. *Nordic Journal of Computing*, 1994.

[Kna91] W. Knapp: Geometrie der Landkartenentwürfe mit computergraphischer Darstellung. Diplomarbeit bei der Fakultät Mathematik Universität Stuttgart, 1991

[KRRS94] Faster shortest path algorithms for planar Graphs. Z.Fülöp, F. Gcses (ed) - proceedings of the 22 nd International Colloquium on Automata, Languages and Programming (ICALP), Szeged, 1995

[KPSKPC95] H. Keller, G. Ploss, R. Sykora, D. Konienko, P. Philipps, M. Cremer: Systemdynamische Schätzung der Matrix der Verkehrsbeziehungen als Grundlage für die Steuerung von Verkehrsleitsystemen. Forschung Straßenbau und Straßenverkehrstechnik Heft 702, (Bundesministerium für Verkehr Abt. Straßenbau in Bonn-Bad Godesberg) Typo Verlagsgesellschaft mbH, Bonn, 1995

[Kno69] W. Knödel: Graphentheoretische Methoden und ihre Anwendungen. Springer Verlag Berlin, Heidelberg 1969

[Knu97] D. Knuth: The Art of Computerprogramming. Addison Wesley, 1997

[Koe86] D. König: Theorie der endlichen und unendlichen Graphen. B.G.Teubner Verlagsgesellschaft Leipzig, 1986

[KSG73] K. W. Krause, R. W. Smith, M. A. Goodwin: Optimal software test planning through automated network analysis. *Proc 1973 IEEE Symp. Computer Software Reliability New York* (S. 18-22), 1973

[Lan97] S. Lange: Separation von planaren Graphen: Implementation und Anwendung eines Algorithmus von Lipton und Tarjan. Universität Stuttgart, Fakultät Informatik, Studienarbeit 19054, 1997

[Lau91] Peter Läuchli: Algorithmische Graphentheorie. Birkhäuser Berlin, 1991

[Lee90] van Leeuwen: Graph Algorithms, Handbook of theoretical Computer Science, 1990

[LHPS86] W. Leutzbach, M. Haas, V. Papavasilion, T. Schwerdtfeger: Dynamische Umlegung in Verkehrsnetzen. Forschung Straßenbau und Straßenverkehrstechnik Heft 469, (Bundesministerium für Verkehr Abt. Straßenbau in Bonn-Bad Godesberg) Typo Verlagsgesellschaft mbH, Bonn, 1986

[Lew97] S. Lewandowski: Anwendung für Separatortheoreme auf planaren Graphen, Diplomarbeit Nr. 19120, Fakultät Informatik, Universität Stuttgart, 1997

[LiT79] R. Lipton und R. Tarjan: A separator theorem for planar Graphs. *SIAM J. of Appl. Math., 36*, 1979

[Mack96] M. Mack: Untersuchung von effizienten Algorithmen zur Bestimmung der k-kürzesten Wege innerhalb von ÖPNV-Verkehrsnetzen, Diplomarbeit Nr. 1374, Fakultät Informatik, Universität Stuttgart, 1996

[Meh77] K. Mehlhorn: Effiziente Algorithmen, Teubner Studienbücher : Informatik, 1977

[Mei94] M. Meier: Borland C++ bis Version 4.5. Entwicklung und Design von DOS-Applikationen. tewi Verlag GmbH, München, 1994

[NiC88] T. Nishizeki, N. Chiba: Planar Graphs. Theory und Algorithms. Elseiier Science Publishers B.V., 1988

[Nol76] Hartmut Noltemeier: Graphentheorie. de Gruytner Lehrbuch Berlin, 1976

[OW93] T. Ottmann und P. Widmayer, Algorithmen und Datenstrukturen, Dr. Alfred Hüthig Verlag Heidelberg, 1985

[Pol61] M. Pollack, W. Wiebenson, Solutions of the shortest-route problem - a review. *Operations Res. 8* (S. 224-230), 1960

[PB94] L.C. Polymenakos, D.P. Bertsekas: Parallel shortest path auction algorithmus. *Parallel Computing 20* (S. 1221-1247), 1994

[PBH86] Posthoff, Bochmann, Haubold: Diskrete Mathematik. Teubner Verlag Leipzig 1986,

[Reu80] W. Reuß: Systematische Erzeugung von matrixförmigen Datenstrukturen als Repräsentanten für Klassen isomorpher Graphen. Universität Stuttgart, Institut für Informatik, Bericht 09626, 1980

[Sac71] H. Sachs: Einführung in die Theorie der endlichen Graphen. B.G.Teubner Verlagsgesellschaft Leipzig, 1971

[Ser94] D. Serwill: DRUM - Modellkonzept zur dynamischen Routensuche und Umlegung. Institut für Stadtbauwesen RWTH Aachen, Mies-van-der-Rohe-Straße 1 52056 Aachen, 1994

[Schl97] J. Schließer: Erstellung eines Systems zur Modellierung und Analyse hierarchischer Graphen. Universität Stuttgart, Fakultät Informatik, Studienarbeit 19350, 1997

[Schn85] W. Schneeweiss: Grundbegriffe der Graphentheorie für praktische Anwendungen. Dr. Alfred Hüthig Verlag Heidelberg, 1985

[SHF94] E. Schöneburg, F. Heinzmann, S. Feddersen: Genetische Algorithmen und Evolutionsstrategien. Addison-Wesley Deutschland GmbH, 1994

[SB77] K. Schaechterle, J. Braun: Vergleichende Untersuchungen vorhandener Verfahren für Verkehrsumlegungen unter Verwendung elektronischer Rechenanlagen. Forschung Straßenbau und Straßenverkehrstechnik Heft 222, (Bundesministerium für Verkehr Abt. Straßenbau in Bonn-Bad Godesberg) Typo Verlagsgesellschaft mbH, Bonn, 1977

[The95] D. Theune: Robuste und effiziente Methoden zur Lösung von Wegeproblemen. B.G.Teubner Verlagsgesellschaft Stuttgart, 1995

[Tur96] V. Turau: Algorithmische Graphentheorie. Addison-Wesley Deutschland, 1996

[Uph95] M. Uphoff: VGA-und Super-VGA-Programmierung. Addison-Wesley Deutschland GmbH, 1995

[Vol91] L. Volkmann: Graphen und Digraphen. Springer Verlag Wien, 1991

[Wag90] K. Wagner: Graphentheorie I-III. B. I. Hochschultaschenbücher Mannheim, 1990

[Wal97] V. Walter: Zuordnung von raumbezogenen Daten - am Beispiel der Datenmodelle ATKIS und GDF. Verlag der Bayerischen Akademie der Wissenschaften, München, 1997

[Wal93] V. Walter: Datenmodelle und Datenschnittstellen zur Speicherung raumbezogener Daten. Interner Bericht, Universität Stuttgart, Institut für Photogrammetrie, 1993

[Wir75] N. Wirth: Algorithmen und Datenstrukturen. Teubner Studienbücher, 1975

[Zell90] A. Zell: Entwurf und Analyse von Algorithmen und Datenstrukturen. Vorlesungsskript Universität Stuttgart, 1990

Anhang A

Standardverfahren zur Berechnung kürzester Wege

Erste Ansätze zur Graphentheorie gehen bis auf Leonhard Euler (Königsberger Brückenproblem, 1736) zurück. Zur genaueren historischen Betrachtung siehe [BLW76], [Koe86] und [Har73]. Graphentheoretische Verfahren zur Berechnung kürzester Wege wurden erst in den Fünfziger Jahren dieses Jahrhunderts entwickelt.

In diesem Kapitel werden die Algorithmen von Dijkstra [Dij59] und Floyd [Flo62] beschrieben, die vielen späteren Lösungsansätzen zu Grunde liegen. Weitreichendere Verfahrensvergleiche und Überblicke befinden sich zum Beispiel in [Dom72], [DeP84] und in [Joh73].

A.1 Der Dijkstra-Algorithmus

Der Dijkstra-Algorithmus wird z. B. in [AHU83] beschrieben. Dijkstra's Idee zur Lösung des SSSP ist, daß die Menge aller kürzesten Wege in einem Graphen von einem Punkt aus Baumstruktur hat, wenn man sich bei mehreren kürzesten Wegen zwischen zwei Knoten einen beliebigen Repräsentanten wählt.

Sei $G = (V, E, \alpha, \omega, \beta)$ ein positiver Graph und ein Startknoten $s \in V$ fest vorgegeben. Wir werden schrittweise einen kürzeste-Wege-Baum aufbauen. Sei $B(i) := \{v \in V \mid d(s,v)$ ist im Schritt i bereits errechnet$\}$. Im i+1-ten Schritt wird derjenige Knoten v in $B(i{+}1)$ aufgenommen, welcher den $i+1$-kürzesten Abstand zu s hat.

Algorithmus:

Start: Wir setzen $B(0) := \{s\}$ (Procedure Setstartknoten), $d(s,s) := 0$ und sei $d(s,v) := \infty$ für alle $v \in V \backslash \{s\}$.

Konstruktion des Baums $B(i+1)$ aus dem Baum $B(i)$: Sei

$$E_{i+1} := \{e \in E \mid \alpha(e) \in B(i), \omega(e) \notin B(i)\}.$$

Bestimme eine Kante e aus E_{i+1}, für die gilt

$$d(s, \alpha(e)) + \beta(e) \leq d(s, \alpha(e')) + \beta(e') \quad \text{für alle } e' \in E_i.$$

Wir setzen $B(i+1) := B(i) \cup \{\omega(e)\}$ und $d(s, \omega(e)) := d(s, \alpha(e)) + \beta(e)$; e ist eine Kante des kürzesten-Wege-Baumes.

Satz A.1 *Der Dijkstra-Algorithmus berechnet $d(s,z)$ für alle $z \in V$, das heißt für alle $v \in B(i)$ gilt $d(s,v)$ ist die Länge eines kürzesten Weges von s nach v. Ist kein Weg zwischen den Knoten s und z vorhanden, so ist $d(s,z) = \infty$.*

Beweis:

Der Satz wird durch vollständige Induktion bewiesen. Für $B(0)$ gilt die Aussage trivialerweise.

$i \to i+1$: Nach Induktionsvoraussetzung gilt für alle $v \in B(i)$, daß das oben bestimmte $d(s,v)$ der kürzeste Weg von s nach v ist.

Annahme: Der durch den Algorithmus hinzugewonnene Knoten $v = \omega(e)$ hat einen kürzeren Abstand zu s als der vom Algorithmus angegebene $d(s,v)$. Sei w' einer der Wege von s nach v, für den $\delta(w') < d(s,v)$ gilt. Sei e' die erste Kante des kürzeren Weges w', für die gilt $e' \in E_{i+1}$. Existiert keine Kante dieser Art, so ist v bereits in einem früheren Schritt in den Baum aufgenommen worden. Mit diesen Vereinbarungen gilt:

$$\begin{array}{lll} \delta(w') & \geq d(s, \alpha(e')) + \beta(e') & \text{Kantenlänge ist } \geq 0 \\ & \geq d(s, \alpha(e)) + \beta(e) & \text{Minimumbildung im Algorithmus} \\ & = d(s, \omega(e)) & \text{Definition von } d \\ & > \delta(w') & \text{Annahme.} \end{array}$$

Dies ist ein Widerspruch. ∎

Bewertung:

Der Zeitbedarf dieses Algorithmus ist $O(n^2)$. Sucht man allerdings alle kürzesten Wege aller Knoten untereinander, so muß der Algorithmus n mal mit wechselnden Startknoten wiederholt werden, der Zeitbedarf steigt also auf $O(n^3)$. Möchte man nur den kürzesten Weg von s nach z errechnen, so kann der Algorithmus abgebrochen werden, wenn $z \in B(i)$ ist. Der Algorithmus benötigt gerade bei dünn besetzten Nachbarschaftsmatrizen wenig Speicherplatz.

Der Dijkstra-Algorithmus gehört zu der Gruppe der greedy (gierig - lokal optimal) Algorithmen. Er kann beidseitig (von Start- und Zielknoten aus) angewandt werden. Die Menge aller Endknoten der Menge E_i kann auch in einer Nachbarschaftsliste $N(i)$ gespeichert werden. Bei geeigneter Wahl der Datenstruktur sinkt der Zeitbedarf.

Ein wesentlicher Nachteil besteht in der Voraussetzung, daß die Gewichtsfunktion immer positive Werte annehmen muß. Bazaraa und Langeley geben in [BL74] einen Algorithmus an, einen Graphen mit negativen Kantenlängen in einen Graphen mit positiven Kantenlängen umzuwandeln, sofern dieser keine Kreise mit negativer Länge besitzt. In dem umgewandelten Graphen kann nun der Dijkstra-Algorithmus angewendet werden und aus dessen Ergebnis die Länge des kürzesten Weges sowie dessen Verlauf abgeleitet werden.

A.2 Organisation der Nachbarschaftsliste in einem Heap

Eine geeignete Verwaltung dieser Nachbarschaftsliste ist für die Effizienz des Dijkstra-Algorithmus relevant. Ist die Menge $N(i)$ als Liste gespeichert, muß beim Erweiterungsschritt des Algorithmus die gesamte Liste durchsucht werden. Bei Graphen mit dicht besetzten Adjazenzmatrizen muß also fast jeder Knoten untersucht werden. Eine linear verwaltete Nachbarschaftsliste verbessert die Zeitkomplexität des Algorithmus im Vergleich zum Dijkstra-Algorithmus ohne Nachbarschaftsliste nicht.

Die Zeitkomplexität des Dijkstra-Algorithmus wird wesentlich verbessert, wenn man die Nachbarschaftsliste in Form eines *Heaps* anlegt.

Definition A.1 *Ein Binärbaum $B = (V, E, \alpha, \omega)$ mit einer Funktion $f : V \to \mathbb{R}$ heißt* Heap der Höhe h, *wenn* (1) *jedes Blatt die Höhe h oder $h-1$ hat und* (2) *für jeden Knoten v mit Söhnen v_1 und v_2 gilt: $f(v) \leq f(v_1)$ und $f(v) \leq f(v_2)$.*

Bedingung (2) aus Definition A.1 impliziert insbesondere, daß ein kleinstes Element an der Wurzel steht. Eine äquivalente Definition würde sich ergeben, wenn man in (2) jedes '$\leq$' durch ein '$\geq$' ersetzen würde. Dann würde ein größtes Element an der Wurzel stehen.

Beispiel A.1 Als Beispiel wählen wir einen Heap aus 9 Elementen, wobei die Ordnungszahl der Knoten gleichzeitig ihre Knotenfunktion darstellt. In einem Heap wäre es auch möglich, 2 oder mehr Elemente mit dem gleichen Knotenwert zu haben.

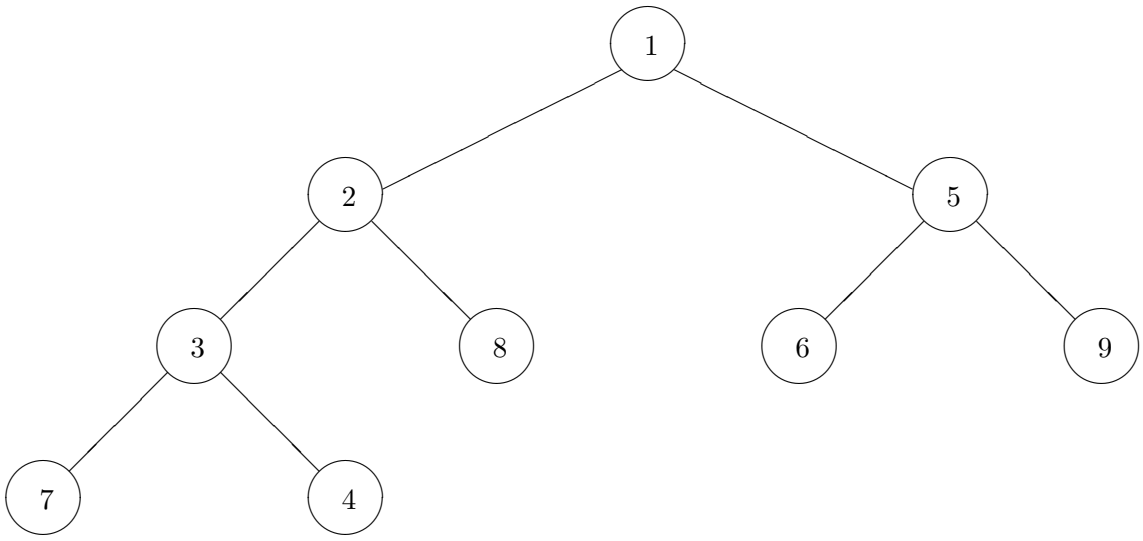


Abbildung A.1 *Beispielheap, bei welchem die Knotennummern gleichzeitig die Knotenfunktion darstellen*

Algorithmus zur Einfügung eines Elements (Hochwandern):

Der Heap der Höhe h soll um ein Element erweitert werden, ohne daß er die Heapstruktur verliert. Falls möglich, fügen wir das neue Element k als Blatt auf der Höhe h ein (dies ist genau dann möglich, wenn der Heap auf der Höhe h weniger als doppelt so viele Knoten besitzt als auf der Höhe $h-1$), andernfalls fügen wir k als Knoten der Höhe $h+1$ ein. In jedem Fall erhalten wir einen neuen Binärbaum, der zwar (1) aber nicht notwendig (2) erfüllt. Ist (2) nicht erfüllt, so vertausche k mit dessen Vaterelement und wiederhole dies so lange, bis k die Wurzel oder der Wert des Vaterelements kleiner ist als der Wert von k. Die Heapstruktur bleibt so erhalten. k kann maximal bis zur Wurzel aufsteigen, die Kosten dafür sind also $O(\log_2(|V|))$.

Beispiel A.2 Wir betrachten den Heap aus Beispiel A.1 und fügen ein neues Element $1'$ ein, dessen Knotenfunktionswert 1 sein soll. $1'$ wird als linker Sohn des Elementes 8 in den Heap aufgenommen. Da $8 > 1$ ist (2) verletzt, und Vater und Sohn müssen vertauscht werden. Da auch $2 > 1$, wird der Vorgang wiederholt. Da $1 \leq 1$ ist nach diesem Schritt die Heapstruktur wiederhergestellt.

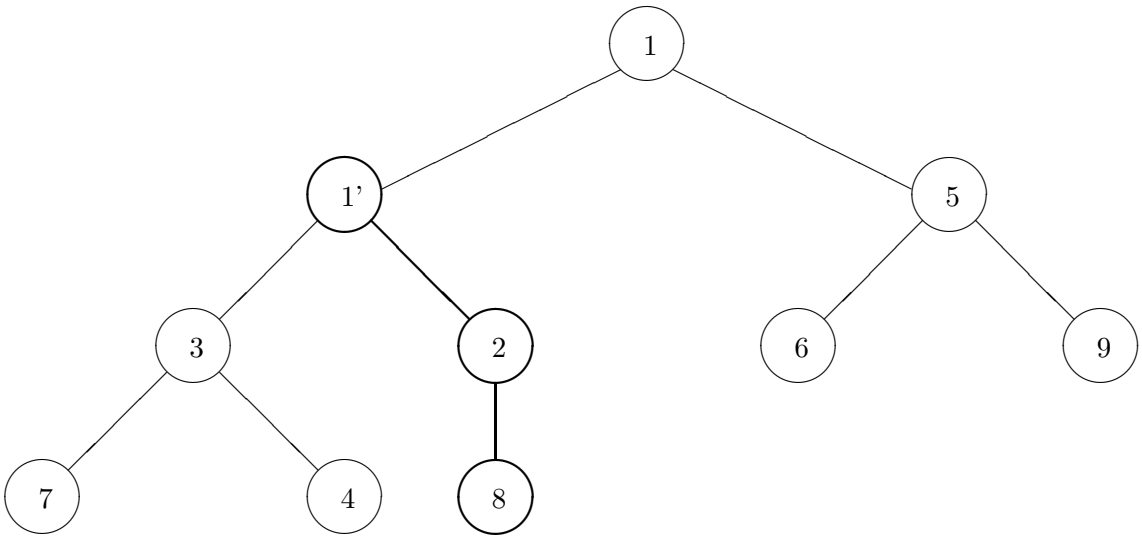


Abbildung A.2 *Beispielheap, bei welchem das Element* $1'$ *eingefügt wurde und hochgewandert ist.*

Algorithmus zum Löschen der Wurzel (Absinken):

Das minimale Element des Heaps wird entfernt. Es entstehen zwei Bäume mit Heapstruktur. Wir nehmen das Blatt l, das in der untersten Reihe am weitesten rechts steht und ersetzen damit die

Wurzel. Falls der Wert von l größer ist als der Wert eines oder beider Söhne, so vertausche l mit dem kleineren von ihnen und wiederhole dies so lange, bis l am Ende des Baumes angekommen ist, oder der Wert von l kleiner oder gleich dem Wert beider Söhne ist. Auch hierfür belaufen sich die Kosten auf $O(\log_2(|V|))$. Die Heapstruktur bleibt auch hier erhalten, da man immer mit dem kleineren Sohn vertauscht.

Beispiel A.3 Wir betrachten den erweiterten Heap aus Beispiel A.2 und löschen das Element 1. Das im Heap in der untersten Reihe am weitesten rechts stehende Element ist der Knoten 8. Der Knoten wird an die Wurzel gesetzt. Hier ist die Heapeigenschaft 3 verletzt. 8 wird mit seinem kleineren Sohn, der $1'$, vertauscht. Da immer noch die Heapeigenschaft (2) verletzt ist, muß dieser Vorgang wiederholt werden. Diesmal werden 8 und 2 vertauscht. Der Knoten 8 ist bis auf Höhe $h-1$ abgesunken. Dies ist in der Abbildung fett markiert.

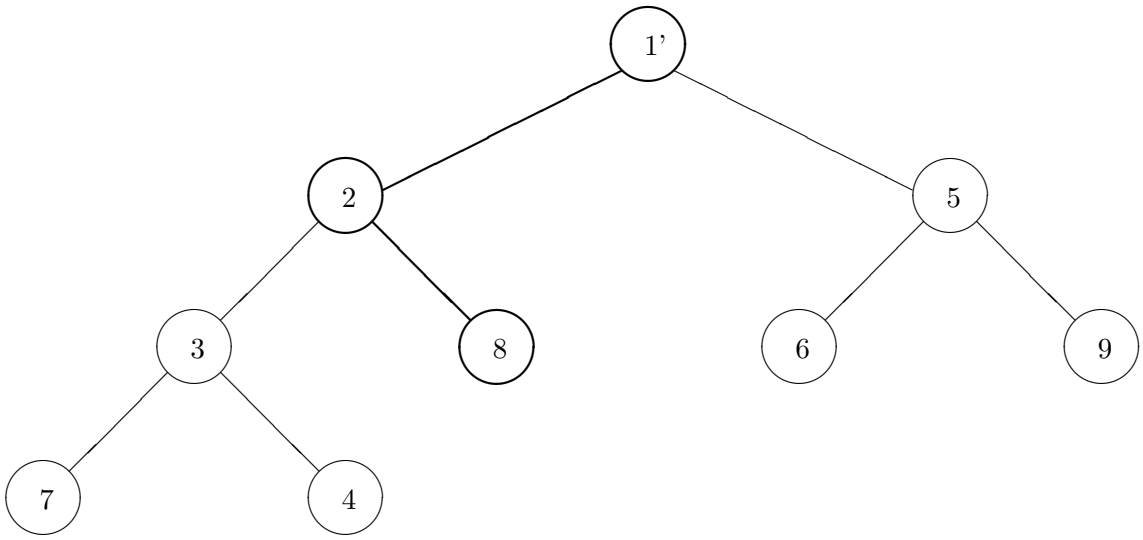


Abbildung A.3 *Beispielheap, bei welchem das Element* 1 *gelöscht wurde und* 8 *abgesunken ist.*

Speichert man die Nachbarschaftsliste in Form eines Heaps ab, und ordnet man die Knoten nach ihrem Abstand zum Startknoten, so steht automatisch das Element an der Wurzel, welches beim Erweiterungsschritt des Dijkstra-Algorithmus in den minimalaufspannenden Baum aufgenommen werden soll. Jeder Knoten, der nach dem Erweiterungsschritt in die Nachbarschaftsliste aufgenommen werden soll, wird in der untersten Stufe im Heap angesetzt und muß so lange hochwandern, bis die Heapstruktur wieder hergestellt ist.

Problem

Soll ein Knoten v in den Nachbarschaftsheap aufgenommen werden, so könnte dieser schon einmal aufgenommen worden sein. Den vormals aufgenommenen Knoten v' zu finden und eventuell zu löschen, ist aufwendiger, als den Knoten einfach ein weiteres Mal mit einem eventuell anderen Knotenfunktionswert in den Heap aufzunehmen. Bei der Entnahme der Knoten muß darauf geachtet werden, daß der in den Minimalbaum aufzunehmende Knoten das erste Mal aufgenommen wird. Ist dies nicht der Fall, so wird der gerade entnommene Knoten verworfen. Jeder Knoten kann maximal so oft im Nachbarschaftsheap aufgenommen werden, wie Kanten auf ihn weisen.

Bei dieser Implementation werden in den Nachbarschaftsheap nicht mehr Knoten aufgenommen als Kanten in dem Graphen vorhanden sind. Er hat also maximal die Höhe $\log_2 |E|$, und da G parallelenfrei ist, gilt $\log_2 |E| \leq \log_2 |V^2| = 2 \log_2 |V|$.

Fazit:

Durch die Verwendung der Heapstruktur kann die Ordnung des Algorithmus von Dijkstra auf $O(n \log n)$ gesenkt werden, wenn die Kantenmenge des Graphen $O(n)$ nicht übersteigt. Im allgemeinen Fall liegt die Anzahl der Operationen bei $O((m+n) \log n)$. Bei der Verwendung von Fibonacci-Heaps wurde in [FrT87] gezeigt, daß eine Lösung für das SSSP schon in der Zeit von $O(n \log n + m)$ möglich ist. Bei Straßennetzen liegen aber Graphen vor, bei welchen $m = O(n)$ gilt, deshalb ist die Ordnung des Dijkstra-Algorithmus unabhängig von der Wahl von Heaps oder Fibonacci-Heaps.

A.3 Der Tripelalgorithmus

Sei $G = (V, E, \alpha, \omega, \beta)$ ein parallelenfreier, gewichteter gerichteter Graph und (β'_{uv}) dessen Adjazenzmatrix. Der Tripelalgorithmus löst nur das APSP Problem und kann in Gegensatz zum Dijkstra-Algorithmus auch in Graphen mit negativen Kantenlängen verwendet werden, solange diese keine Kreise negativer Länge beinhalten.

Methode:
Es wird die Länge aller Kanten der Graphen mit den Längen aller möglichen Umwege verglichen und die Gewichtung der Kante gesenkt, falls der Umweg kürzer ist.

Algorithmus:
For $i := 1$ to n do
  for $j := 1$ to n do
    for $k := 1$ to n do

$$\beta'(v_j, v_k) := \min\{\beta'(v_j, v_k), \beta'(v_j, v_i) + \beta'(v_i, v_k)\}$$

Satz A.2 *Der Tripelalgorithmus berechnet alle kürzesten Wege im Graphen.*

Beweisskizze:
1.) Durch Induktion wird bewiesen, daß das Resultat des Algorithmus unabhängig von der Numerierung der Knoten v_i ist.
2.) Nach Durchführung des Algorithmus ist $\beta'(u,v) = d(u,v)$ für alle u und v.

Bewertung:

Der Zeitbedarf dieses Algorithmus ist $O(n^3)$. Selbst wenn man nur zwischen zwei Punkten den kürzesten Weg sucht, muß der Algorithmus alle Knotentripel bearbeiten. Die Zahl der unnötigen Vergleiche wächst sehr schnell mit n. Die Implementierung des Algorithmus ist sehr einfach (siehe dazu auch [Dor73]). Der Algorithmus benötigt gerade bei dünn besetzten Adjazenz-Matrizen unverhältnismäßig viel Speicherplatz.

Der Algorithmus von Dantzig:

Der Algorithmus von Dantzig ist ein modifizierter Tripelalgorithmus:

Algorithmus:
For $i := 1$ to n do
  for $j := 1$ to $i - 1$ do
    for $k := 1$ to $j - 1$ do

$$\beta(v_j, v_k) := \min\{\beta(v_j, v_k), \beta(v_j, v_i) + \beta(v_i, v_k)\}$$

Bewertung:
Der Zeitbedarf dieses Algorithmus sinkt gegenüber dem Tripelalgorithmus um etwa die Hälfte ab, ist aber immer noch $O(n^3)$.

Der Vorteil der Tripelalgorithmen ist, daß sie auch bei negativen Kantenlängen funktionieren, falls keine Kreise negativer Länge existieren. Ein weiterer Vorteil liegt in der Einfachheit der Handhabung.

Ein Straßennetz wird in der Regel durch einen Graphen repräsentiert, bei dem die Anzahl der Kanten $|E|$ nach dem Eulerschen Polyedersatz höchstens dreimal so groß ist wie die Anzahl der Knoten $|V|$. Kürzeste Wege zwischen zwei Knoten u und v können zum Beispiel mit dem Algorithmus von Dijksta berechnet werden, der mit der Zeitkomplexität $O(|V| * \log |V|)$ arbeitet. Ziel ist es, Algorithmen zu finden, die die Zeitkomplexität auf $O(|V|)$ senken. Dies gelingt mit dem Algorithmus von D'Esopo (siehe Abschnitt 3.1) und mit dem $A*$–Algorithmus (siehe Abschnitt 3.2).

In [Fre76] wird bewiesen, daß $O(n^{\frac{5}{2}})$ Vergleiche ausreichen, um das APSP zu lösen. Er gibt einen Algorithmus an, der das APSP asymptotisch schneller löst als die hier beschriebenen Tripelalgorithmen. Trotzdem sind für alle praktischen Anwendungen die Algorithmen von Floyd und Dantzig überlegen, da die Proportionalitätskonstanten bei Fredmann sehr groß sind.

Der Dijkstra-Algorithmus ist von der Zeitkomplexität her schneller und benötigt weniger Speicherplatz. Außerdem kann beim Dijkstra-Algorithmus nicht nur der Wert des kürzesten Weges,

sondern auch dessen Verlauf auf einfache Weise bestimmt werden. Bei den Tripelalgorithmen ist dies wesentlich schwieriger.

Es gibt noch weitere Ansätze, kürzeste Wege zu berechnen. In [LiT79] wird ein Separatortheorem vorgestellt, welches Graphen in Teilgraphen zerlegt. Fredman und Tarjan zeigen in [FrT87], wie dieses Separatortheorem, nach dem 'Teile und Beherrsche' Prinzip (devide and conquer) zu einem Algorithmus der kürzeste Wege berechnet, ausgeweitet werden kann. Dieser Algorithmus arbeitet in der Zeit $O(n \cdot \sqrt{\log n})$. Der Algorithmus findet aber kaum praktische Anwendung, weil die Konstanten in dieser Ordnung sehr hoch sind [Lew97]. Eine Implementation befindet sich in [Lan97]

Anhang B

Beispiele

Beispiel B.1 Um die Begriffe der Anfangs und Endwege zu verdeutlichen, bedienen wir uns eines Beispiels. Gegeben ist ein Graph (V, E, α, ω) mit Wegeverboten P und der Knotenmenge

$$V = v_i, \quad i = 1..9,$$

der Kantenmenge

$$E = \{e_{12}, e_{14}, e_{21}, e_{23}, e_{25}, e_{32}, e_{36}, e_{41}, e_{45}, e_{47}, e_{52}, e_{54},$$

$$e_{56}, e_{58}, e_{63}, e_{65}, e_{69}, e_{74}, e_{78}, e_{85}, e_{89}, e_{96}, e_{98}\},$$

den Wegeverboten

$$\begin{array}{ll} p_1 := (e_{12}, e_{23}, e_{36}, e_{69}, e_{98}) & p_2 := (e_{12}, e_{23}, e_{36}, e_{65}, e_{52}) \\ p_3 := (e_{12}, e_{23}, e_{36}, e_{65}, e_{58}) & p_4 := (e_{12}, e_{25}, e_{58}) \\ p_5 := (e_{74}, e_{41}, e_{12}, e_{23}, e_{36}, e_{69}) & p_6 := (e_{36}, e_{69}, e_{98}, e_{87}, e_{74}, e_{41}) \end{array}$$

und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i \;\; \omega(e_{ij}) = v_j$. Die Kanten des Wegeverbotes p_j heißen $f_i^j \;\; i = 1, ..., |p_j|$.

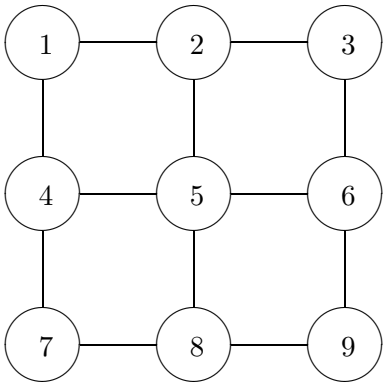


Abbildung B.1 *Beispielgraph für Anfangs– und Endwege*

Damit gilt $E_0 = \{e_{12}, e_{36}, e_{74}\}$ und die Wegeverbote p_1, p_2, p_3 und p_4 bilden eine Wegeverbotsklasse. $\{p_5\}$ und $\{p_6\}$ bilden zwei andere Klassen. Nach der Klassenbildung gelten folgende Bezeichnungen:

$$p_i^1 := p_i \quad i = 1, ..., 4 \quad p_1^2 := p_5 \quad p_1^3 := p_6.$$

Gesucht ist der maximale Anfangsweg von p_1^1 in $\tilde{P} := \{p_2^1, p_3^1, p_4^1\}$. Damit folgt für die Abbildung $aga : \tilde{P} \to \mathbb{N}$

$$aga(p_2^1) = 3 \quad aga(p_3^1) = 3 \quad aga(p_4^1) = 1.$$

und damit ist $Aga(\tilde{P}, p_1^1) := aga(p_2^1) = 3$ und p_2^1 ist dank des kleineren Indexes maximaler Anfangsweg von vw_1.

Betrachten wir nun das Wegeverbot p_5 und sei $\tilde{P} := P \backslash \{p_5\}$. Für p_1 gilt mit $q = 2$ und $r = |p_5| = 6$:

$$f_{q+i}^5 \; = \; f_i^1 \text{ für } 1 \leq i \leq r - q$$

und für p_6 gilt mit $q = 4$ und $r = 6$:

$$f_{q+i}^5 \; = \; f_i^6 \text{ für } 1 \leq i \leq 2.$$

Für die anderen Wegeverbote gilt keine derartige Beziehung. Das Wegeverbot p_1 ist maximaler Endweg von p_5 in $\tilde{P}$. Die Anzahl der gemeinsamen Kanten ist $eka(p_5, p_1) = r - q = 4$.

Die Wegeverbote p_5 und p_6 sind gegenseitig Endwege voneinander ohne identisch zu sein.

Beispiel B.2 Um das Rathausmodell und die Methode der Überbrückung von Regionen zu verdeutlichen, betrachten wir folgenden Graphen $G = (V, E, \alpha, \omega, \beta)$ mit $V := \{v_1, ..., v_9, v_a, v_b, v_c\}$, $E := \{e_{12}, e_{14}, e_{21}, e_{23}, e_{25}, e_{2b}, e_{32}, e_{36}, e_{3c}, e_{41}, e_{45}, e_{47}, e_{52}, e_{54}, e_{56}, e_{58}, e_{63}, e_{65}, e_{69}, e_{74}, e_{78}, e_{7a}, e_{85}, e_{87}, e_{89}, e_{96}, e_{98}, e_{9c}, e_{a7}, e_{b2}, e_{c3}, e_{c9}\}$, den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$ $\omega(e_{ij}) = v_j$ und der Gewichtsfunktion $\beta(e) = 1$ für alle $e \in E$. Damit kann der Graph auch als ungerichtet aufgefaßt werden.

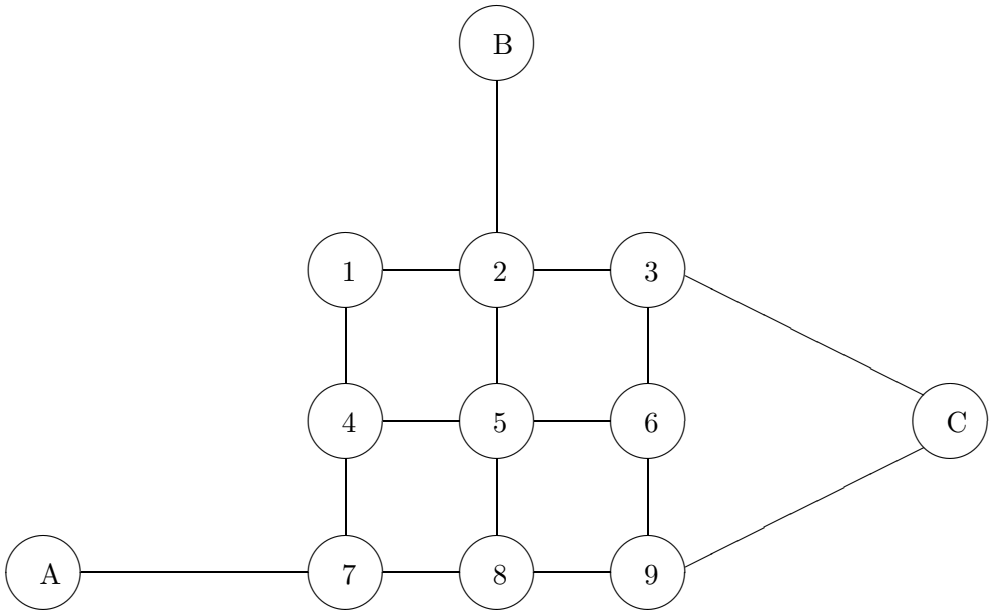


Abbildung B.2 *Beispielgraph für das Rathausmodell und die Methode der Überbrückung von Regionen*

Sei $\tilde{E} := \{e_{ij} \in E \mid i, j = 1, ..., 9\}$. $\tilde{E}$ ist wegweise zusammenhängend. Mit diesen Definitionen gilt $V_i = \{v_1, v_4, v_5, v_6, v_8\}$, $V_r = \{v_2, v_3, v_7, v_9\}$ und $V_a = \emptyset$, $E_1 = \tilde{E}$, $E_2 = E \backslash \tilde{E}$ und $E_3 = \emptyset$. Wir definieren einen neuen Knoten $\tilde{v}$. Der neue Graph $G' = (V', E', \alpha', \omega')$ ist dann definiert durch

$$V' = (V \backslash \tilde{V}) \cup \{\tilde{v}\} = \{v_a, v_b, v_c, \tilde{v}\}$$

$$E' = (E \backslash \tilde{E}) = \{e_{2b}, e_{3c}, e_{7a}, e_{9c}, e_{a7}, e_{b2}, e_{c3}, e_{c9}\}$$

mit den Inzidenzabbildungen:

$$\begin{array}{llllll} \alpha'(e_{2b}) = \tilde{v} & \omega'(e_{2b}) = v_b & \alpha'(e_{3c}) = \tilde{v} & \omega'(e_{3c}) = v_c & \alpha'(e_{7a}) = \tilde{v} & \omega'(e_{7a}) = v_a \\ \alpha'(e_{9c}) = \tilde{v} & \omega'(e_{9c}) = v_c & \alpha'(e_{a7}) = v_a & \omega'(e_{a7}) = \tilde{v} & \alpha'(e_{b2}) = v_b & \omega'(e_{b2}) = \tilde{v} \\ \alpha'(e_{c3}) = v_c & \omega'(e_{c3}) = \tilde{v} & \alpha'(e_{c9}) = v_c & \omega'(e_{c9}) = \tilde{v} & & \end{array}$$

Sei $v_r = v_8$ das Rathaus von $\tilde{V}$, dann ist die Gewichtsfunktion β' ist wie folgt definiert: $\beta'(e_{b2}) = d(\omega(e_{2b}), v_8) = d(v_b, v_8) = 3 = \beta'(e_{2b})$, $\beta'(e_{a7}) = \beta'(e_{7a}) = 2$, $\beta'(e_{c3}) = \beta'(e_{3c}) = 4$ und $\beta'(e_{c9}) = \beta'(e_{9c}) = 2$.

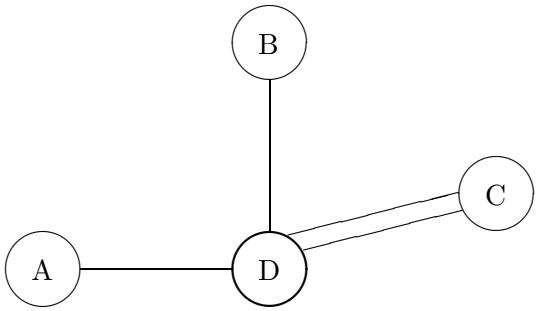


Abbildung B.3 *Beispielgraph nach dem Rathausmodell*

Mit diesem Modell wird der kürzeste Weg zwischen A und C und zwischen A und B richtig berechnet. Der kürzeste Weg von C nach B ist allerdings fehlerbehaftet. Der tatsächliche kürzeste Weg e_{c3}, e_{32}, e_{2b} mit Länge 3 ist kürzer als der errechnete Weg $e_{c9}, e_{98}, e_{85}, e_{52}, e_{2b}$ der die Länge 5 hat. Für den Fehler gilt folgende Abschätzung: Sei $u, v \in v\backslash\tilde{V}$ dann ist

$$d'(u,v) - d(u,v) \leq \max_{v\in\tilde{V}} d(v_r, v) + \max_{v\in\tilde{V}} d(v, v_r) = 2 + 2 = 4.$$

Bei diesem Ansatz werden 8 Knoten und 24 Kanten eingespart.

Wir wollen nun auf diesen Graphen den Ansatz der Überbrückung von Regionen anwenden. Dazu bestimmen wir die Randknotenmenge $V_r = \{v_2, v_3, v_7, v_9\}$ und definieren zwischen diesen die folgenden 12 neuen Kanten:

$$E_3 := \{e_{v_2,v_3}, e_{v_2,v_7}, e_{v_2,v_9}, e_{v_3,v_2}, e_{v_3,v_7}, e_{v_3,v_9}, e_{v_7,v_2}, e_{v_7,v_3}, e_{v_7,v_9}, e_{v_9,v_2}, e_{v_9,v_3}, e_{v_9,v_7}\}.$$

Für die Inzidenz gilt $\alpha'(e_{v_i,v_j}) = v_i$ und $\omega'(e_{v_i,v_j}) = v_j$. Für die Gewichtsfunktion ergibt sich $\beta'(e_{v_1,v_2}) := d(v_1, v_2)$ also z. B. $\beta'(e_{v_7,v_2}) = 3$. Der veränderte Graph hat folgende Form: $G' := (V', E', \alpha', \omega', \beta')$, wobei

$$V' = \{v_2, v_3, v_7, v_9, v_a, v_b, v_c\}$$

und

$$E' = E\backslash\tilde{E} \cup E_3$$

ist. Durch die Überbrückung von Regionen werden hier 5 Knoten und 12 Kanten eingespart.

Bei entsprechend umfangreicher gewählten Gebieten ist die Ersparnis wesentlich größer.

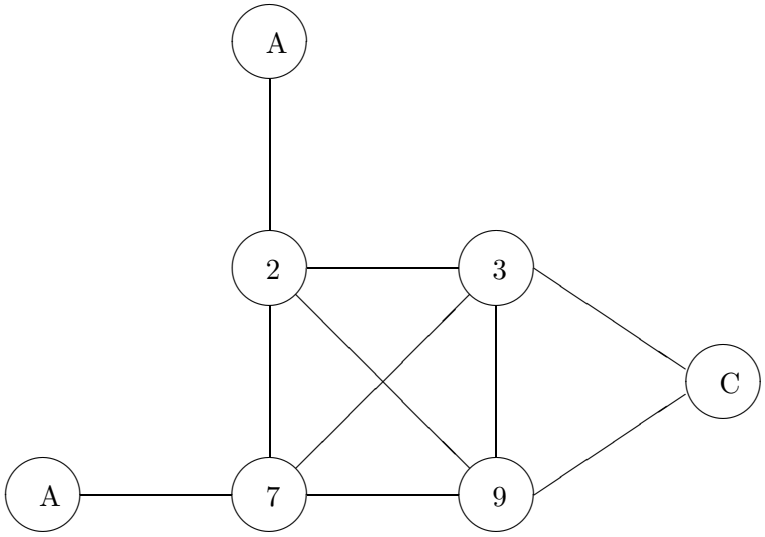


Abbildung B.4 *Beispielgraph nach Anwendung der Methode der Überbrückung von Regionen*

B.1 Beispiele zur Berechnung kürzester Wege in Graphen mit Abbiegeverboten

Beispiel B.3 In diesem Beispiel sollen die kürzesten Wege von einem Punkt aus in einem Graphen mit Abbiegeverboten mit der Methode der Kantenaufnahme berechnet werden.

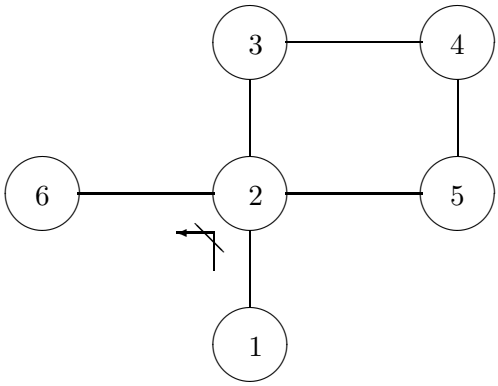


Abbildung B.5 *Graph mit Abbiegeverbot, in welchem die kürzesten Wege mit der Methode der Kantenaufnahme berechnet werden.*

Sei ein Graph $G = (V, E, \alpha, \omega, \beta)$ mit Abbiegeverboten T gegeben mit

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{e_{12}, e_{21}, e_{23}, e_{25}, e_{26}, e_{32}, e_{34}, e_{43}, e_{45}, e_{52}, e_{54}, e_{62}\}$$

$$\beta(e) = 1$$

$$T = \{(e_{12}, e_{26})\}$$

und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$ $\omega(e_{ij}) = v_j$.

Als Startknoten soll der Knoten v_1 gewählt werden, da von ihm aus das Abbiegeverbot eine Bedeutung hat. Die Bezeichnungen $B(i)$, $N(i)$ und $R(i)$ seinen analog zum Beispiel 4.3 definiert.

1.Schritt:
$B(1) = \{e_{12}\}$ damit gilt, daß alle Kanten welche den Anfangspunkt v_2 besitzen, und in deren Richtung von e_{12} aus kein Abbiegevebot besteht, in $N(1)$ aufgenommen werden: $N(1) = \{e_{21}, e_{23}, e_{25}\}$ und damit folgt für $R(1) = \{e_{26}, e_{32}, e_{34}, e_{43}, e_{45}, e_{52}, e_{54}, e_{62}\}$. e_{26} wird erst nach 5 Schritten in die Nachbarschaftsliste und nach 11 Schritten in den Kürzeste-Wege-Baum aufgenommen.

2.Schritt:
Da die Kantengewichte alle gleich sind können beide Elemente aus $N(1)$ nach $B(1)$ aufgenommen werden. Diese Auswahl ist bei der Berechnung der Länge der kürzesten Wege nicht relevant, wohl aber bei der Streckenführung. Im Zweifelsfall wähle ich die Kante mit dem niedrigeren Index, hier e_{21}. $B(2) = \{e_{12}, e_{21}\}$, $N(2) = \{e_{23}, e_{25}\}$ und $R(2) = R(1)$. Würden wir den Algorithmus knotenorientiert betrachten, so würde in diesem Schritt der Knoten v_1 zum zweiten und letzten Mal in den Kürzeste-Wege-Baum aufgenommen werden.

Die weiteren Schritte laufen analog ab. Nach 12 Schritten ergibt sich folgender Kürzeste-Wege-Baum, wobei hier die Kanten wieder durch deren Endknoten repräsentiert werden:

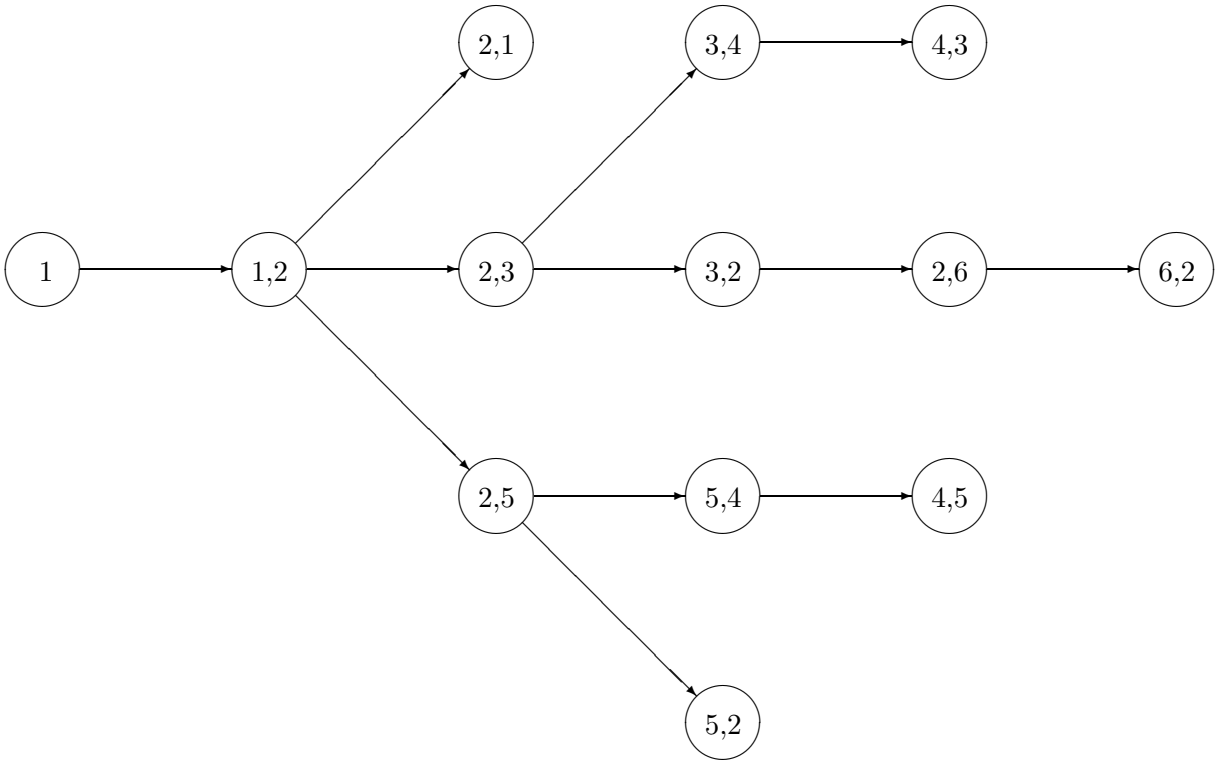


Abbildung B.6 *Der Kürzeste-Wege-Baum des Graphen aus Beispiel B.5 vom Knoten v_1 aus.*

Der kürzeste Wegebaum, der mit der Methode der mehrfachen Knotenaufnahme erzeugt wurde, hat nur 7 Knoten.

Beispiel B.4 Wir werden nun bei einem Graphen mit einem Abbiegeverbot alle kürzesten Wege von einem Punkt aus berechnen. Damit das Abbiegeverbot eine Wirkung hat, wählen wir bei folgendem Graphen v_1 als Startknoten.

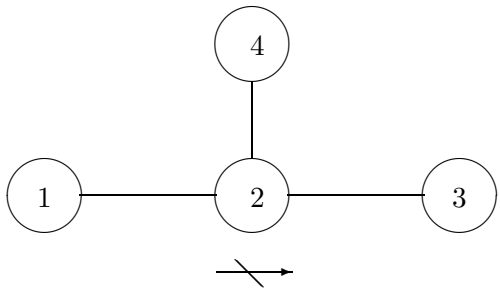


Abbildung B.7 *Beispielgraph*

Sei ein Graph $G = (V, E, \alpha, \omega, \beta)$ mit Abbiegeverboten T gegeben mit

$$V = \{v_1, v_2, v_3, v_4\}$$

$$E = \{e_{12}, e_{21}, e_{23}, e_{24}, e_{32}, e_{42}\}$$

$$\beta(e) = 1$$

$$T = \{(e_{12}, e_{23})\}$$

und den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$ $\omega(e_{ij}) = v_j$.

Sei $B(i)$ der Kürzeste-Wege-Baum, $N(i)$ die Nachbarschaftsliste und $R(i)$ die noch aufzunehmenden Knoten nach dem Schritt i.

1.Schritt:
$B(1) := \{v_1\}$ damit gilt $N(1) = \{v_2\}$, $R(1) = \{v_2, v_3, v_4\}$.

2.Schritt:
Der einzige Knoten, der in $B(2)$ aufgenommen werden kann, ist v_2. Da dies über die Kante e_{12} geschieht, aus der ein Abbiegeverbot besteht, kann v_3 nicht in $N(2)$ aufgenommen werden und v_2 muß noch ein weiteres Mal in $B(i)$ aufgenommen werden. $B(2) := \{v_1, v_2^{12}\}$, $N(2) = \{v_4\}$, $R(2) = \{v_2, v_3, v_4\}$.

3.Schritt:
Wie im zweiten Schritt gibt es nur einen Knoten, der in $B(3)$ aufgenommen werden kann. Dies ist v_4. v_2 wird ein zweites Mal in die Nachbarschaftsliste aufgenommen. $B(3) := \{v_1, v_2^{12}, v_4\}$, $N(3) = \{v_2\}$, $R(3) = \{v_2, v_3\}$.

4.Schritt:
Jetzt wird v_2 ein zweites Mal in den Kürzeste-Wege-Baum aufgenommen und zwar über die Kante e_{42}. $B(4) := \{v_1, v_2^{12}, v_4, v_2^{42}\}$, $N(4) = \{v_3\}$, $R(4) = \{v_3\}$.

5.Schritt:
Nur noch v_3 ist übrig, um in den Kürzeste-Wege-Baum aufgenommen zu werden. $B(5) := \{v_1, v_2^{12}, v_4, v_2^{42}, v_3\}$, $N(5) = \emptyset$, $R(5) = \emptyset$.

Der Kürzeste-Wege-Baum vom Knoten v_1 aus besitzt also folgende Struktur:

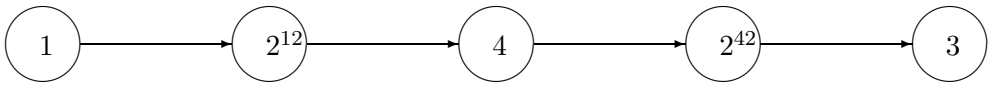


Abbildung B.8 *Kürzester-Wege-Baum*

Der kürzeste Weg von v_1 nach v_3 beinhaltet also ein Wenden an Knoten v_4.

Würde man die Bedingung, daß ein Knoten von ein und derselben Kante nicht zwei Mal in den Kürzeste-Wege-Baum aufgenommen werden kann, ignorieren so würde im Schritt 2 der Knoten v_2 ebenfalls in die Nachbarschaftsliste aufgenommen werden, und der dritte Schritt würde folgendermaßen aussehen:

3'.Schritt:
Zwei Knoten können in den Kürzeste-Wege-Baum aufgenommen werden. Da gilt $\beta(e_{12}) < d(v_1, v_2) + \beta(e_{23})$, wird dies v_2 sein. Da dies von der Kante e_{12} geschieht, aus deren Richtung ein Abbiegeverbot existiert, muß v_2 ein weiteres Mal in den Kürzeste-Wege-Baum aufgenommen werden können. Damit ergeben sich die einzelnen Mengen zu $B(3) := \{v_1, v_2^{12}, v_2^{12}\}$, $N(3) = \{v_2, v_3\}$, $R(3) = \{v_2, v_3, v_4\}$.

An dieser Stelle befindet sich der Algorithmus in einer Endlosschleife.

Beispiel B.5 Bei diesem Beispiel wird die Methode des Ziehens Neuer Kanten erläutert.

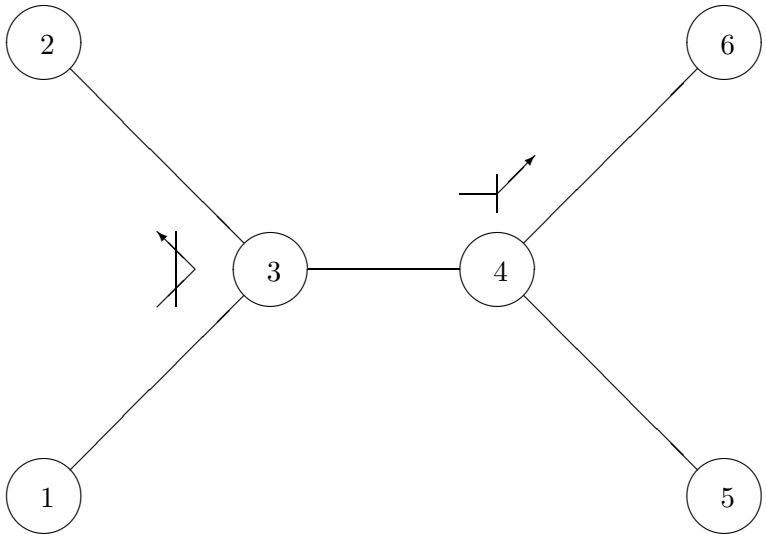


Abbildung B.9 *Beispielgraph an dem die Methode des Ziehens neuer Kanten erläutert wird*

Sei $G = V, E, \alpha, \omega, \beta$ ein Graph mit

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\},$$

$$E = \{e_{13}, e_{23}, e_{31}, e_{32}, e_{34}, e_{43}, e_{45}, e_{46}, e_{54}, e_{64}\},$$

$$T = \{(e_{13}, e_{32}), (e_{34}, e_{46})\},$$

$\beta(e_{ij}) = 1$, $\alpha(e_{ij}) = v_i$ und $\omega(e_{ij}) = v_j$.

Analog zum Algorithmus werden die Kanten aus E mit $\{f_1, ..., f_{10}\}$ bezeichnet ($f_1 = e_{13} usw.$). Da an den Knoten v_1 und v_2 keine Abbiegeverbote existieren gilt $G^0 = G^1 = G^2$. Da von der Kante $e_{13} = f_1$ aus ein Abbiegeverbot existiert, sind die Graphen H^0 und H^1 verschieden. Die Menge $F_j = \{e_{34}, e_{31}\}$ wird elementweise verdoppelt. Es entstehen die neuen Kanten $f_{11} := \tilde{e}_{11}$ und $f_{12} := \tilde{e}_{14}$ mit den üblichen Inzidenzstrukturen und $\tilde{\beta}^1(e_{11}) = \beta(e_{13}) + \beta(e_{31}) = 2$ $\tilde{\beta}^1(e_{14}) = \beta(e_{13}) + \beta(e_{34}) = 2$ nach Gleichung 4.1. Die Abbiegeverbotsmengen sind $\tilde{T}_1^1 = \{(e_{34}, e_{46})\}$ und $\tilde{T}_2^1 = \{(\tilde{e}_{14}, e_{46})\}$, da $(e_{34}, e_{46}) \in T$ und $\tilde{e}_{14}$ das verdoppelte Element von e_{34} ist. Die Kante e_{13} wird aus dem Graphen entfernt. Da keine weiteren Abbiegeverbote an v_3 existieren, ist dieser

Schritt die einzige Veränderung und mit $E^3 = E \backslash e_{13} \cup \{\tilde{e}_{11}, \tilde{e}_{14}\}$ kann der neue Graph G^3 dargestellt werden als:

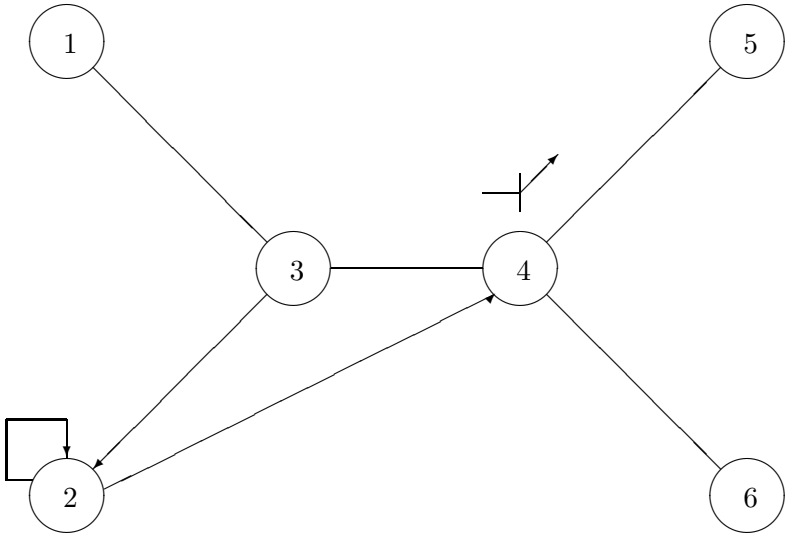


Abbildung B.10 *Beispielgraph nach Betrachtung des Knoten v_3*

Am Knoten v_4 ist das Abbiegen von der Kante $e_{34} = f_5$ und von der Kante $e_{14} = f_{12}$ aus verboten. Bei der Konstruktion von H^5 erhält man $F_5 = \{e_{43}, e_{45}\}$ und die verdoppelten Elemente $f_{13} := \tilde{e}_{33}$ und $f_{14} := \tilde{e}_{35}$ mit der üblichen Inzidenz. Für die Gewichtung gilt analog zum obigen Schritt $\tilde{\beta}^5(e_{35}) = \tilde{\beta}^5(e_{33}) = 2$ und für die verbleibenden Wegeverbote sind $\tilde{T}^5 = \{(e_{14}, e_{46})\}$.

Am Knoten v_4 ist aber auch das Abbiegen von $\tilde{e}_{14} = f_{12}$ nach e_{46} verboten. Beim zwölften Schritt erhält man analog zum obigen Schritt $F_{12} = \{e_{43}, e_{45}\}$ und damit die gesplitteten Kanten $\tilde{e}_{13}$ und $\tilde{e}_{15}$ mit den Gewichten $\beta^{12}(\tilde{e}_{13}) = \beta^{12}(\tilde{e}_{15}) = 3$ und den üblichen Inzidenzstrukturen. Nach diesem Schritt besitzt H^{12} keine Abbiegeverbote mehr und G^6 kann folgendermaßen dargestellt werden:

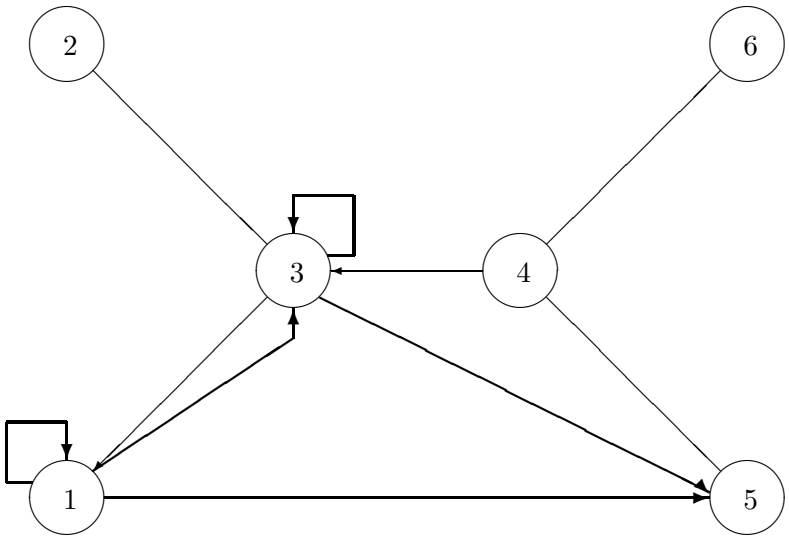
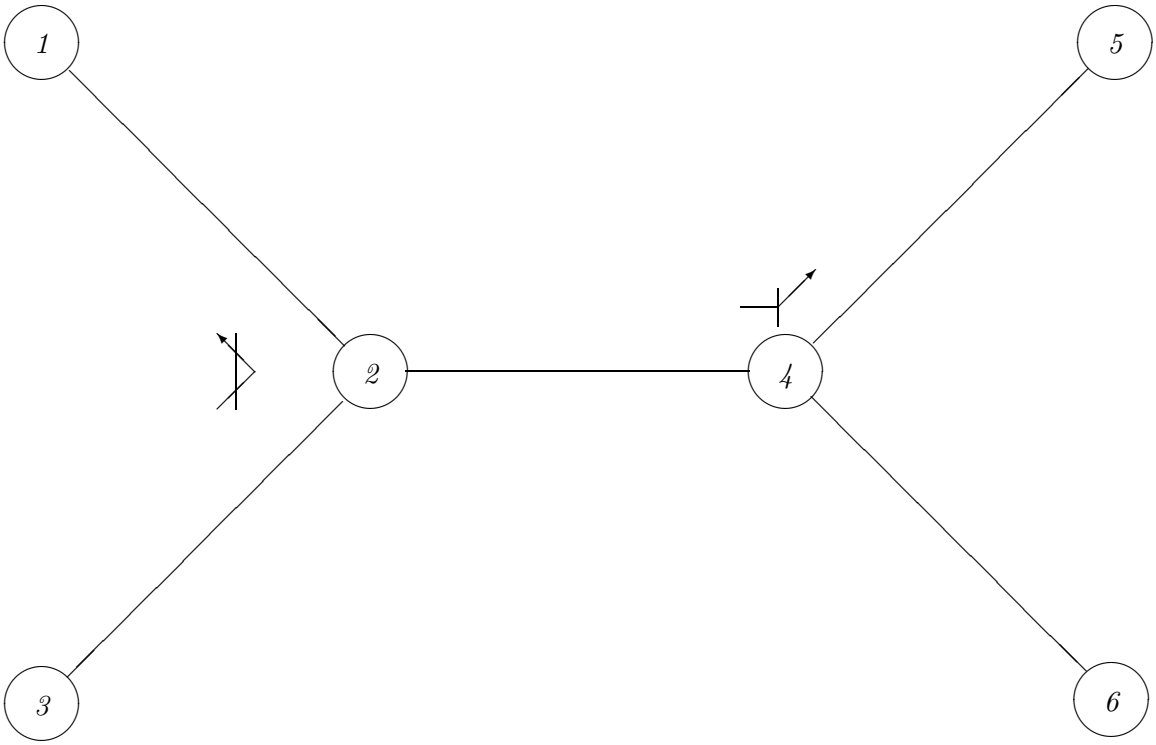


Abbildung B.11 *Beispielgraph G^6 nach allen Erweiterungen*

In diesem Graphen können nun vom Knoten v_1 aus alle kürzesten Wege unter Berücksichtigung der Abbiegeverbote mit einem beliebigen kürzeste–Wege–Algorithmus berechnet werden.

Beispiel B.6 *Um den Satz 4.3 zu veranschaulichen, betrachten wir folgenden Graphen mit 2 Abbiegeverboten:*



Gegeben ist ein Graph (V, E, α, ω) *mit Abbiegeverboten* T *und der Knotenmenge*

$$V = v_i, \quad i = 1..6,$$

der Kantenmenge

$$E = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{24}, e_{42}, e_{45}, e_{54}, e_{46}, e_{64}\},$$

den Inzidenzabbildungen $\alpha(e_{ij}) = v_i$, $\omega(e_{ij}) = v_j$ *und den Abbiegeverboten*

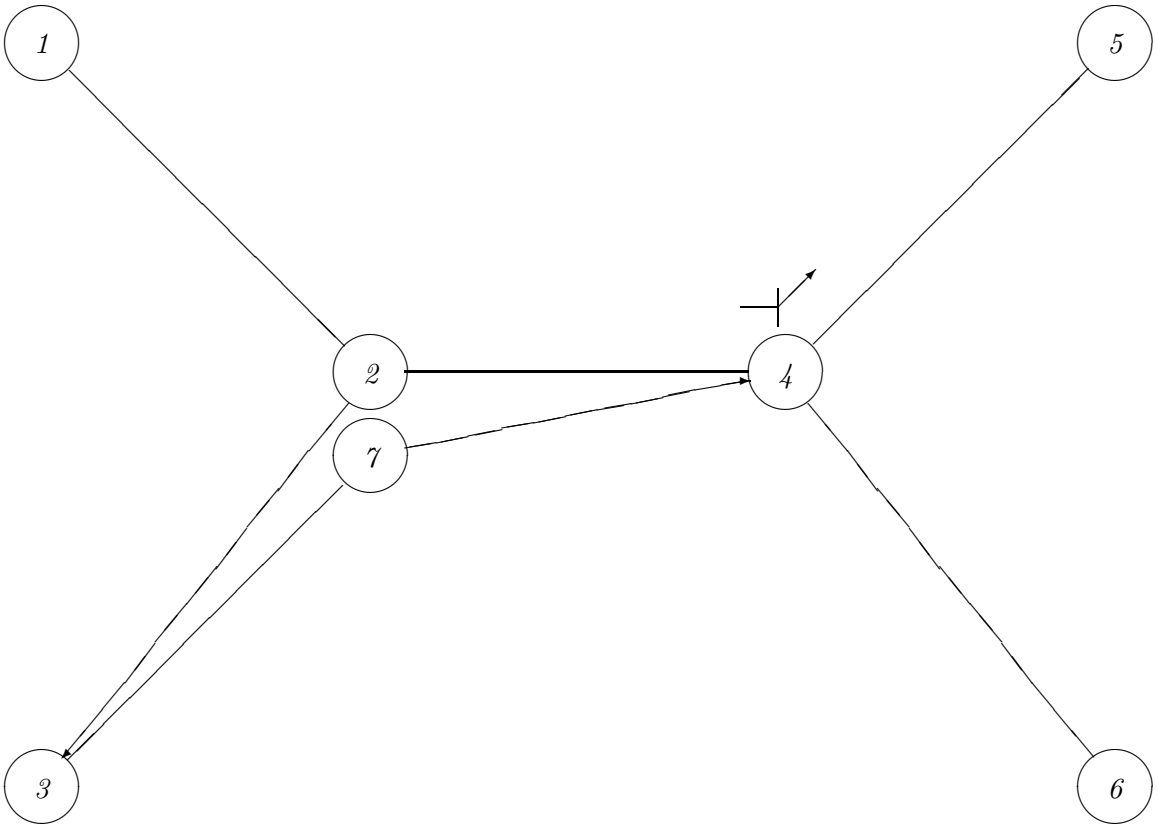
$$T := \{(e_{32}, e_{21}), (e_{24}, e_{45})\}.$$

E_0 *ergibt sich zu*

$$E_0 = \{e_{32}, e_{24}\}$$

Im Fall a wird bei der Erweiterung zuerst die Kante e_{32}*, im Fall b zuerst die Kante* e_{24} *verwendet.*

Schritt 1a *mit* $f_1 = e_{32}$*:*



Der aus v_2 erschaffene Knoten $v^{e_{32}}$ heiße v_7. Da (e_{32}, e_{12}) verboten ist, gilt

$$N^{e_{32}} = \{e_{24}, e_{23}\}.$$

Die neuen Kanten seien

$$\tilde{E}^{e_{32}} = \{e_{74}, e_{73}\}$$

mit den Vaterbeziehungen

$$h^{e_{32}}(e_{74}) = e_{24} \quad h^{e_{32}}(e_{73}) = e_{23}.$$

Da $E_0 \cap N^{e_{32}} \neq \emptyset$ ist, gilt

$$T^1 = \{(e_{24}, e_{45})\} \text{ und damit } \tilde{T}^1 = \{(e_{74}, e_{45})\}$$

Der erweiterte Graph $G^{e_{32}}$ wird also folgendermaßen definiert:

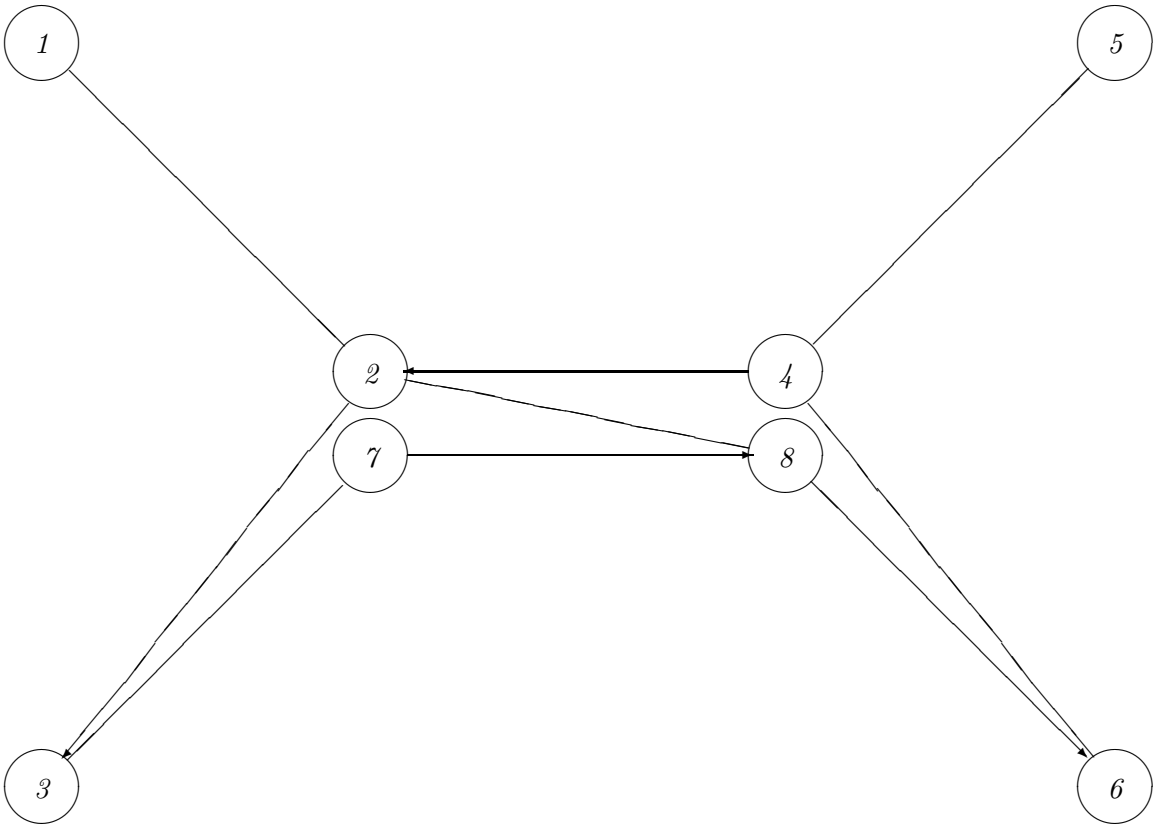
$$V^{e_{32}} = v_i, \quad i = 1..7$$

$$E^{e_{32}} = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{24}, e_{42}, e_{45}, e_{54}, e_{46}, e_{64}, e_{73}, e_{74}\}$$

$$\alpha^{e_{32}}(e_{ij}) = v_i$$

$$\omega^{e_{32}}(e_{ij}) = v_j \; \textit{falls} \; (i,j) \neq (3,2)$$
$$\omega^{e_{32}}(e_{32}) = v_7$$
$$T^{e_{32}} := \{(e_{24}, e_{45}), (e_{74}, e_{45})\}$$

Schritt 2a *mit* $f_2 = e_{24}$*:*



Der aus v_4 *erschaffene Knoten* $v^{e_{24}}$ *heiße* v_8*. Da* (e_{24}, e_{45}) *verboten ist, gilt*

$$N^{e_{24}} = \{e_{42}, e_{46}\}$$

Die neuen Kanten sind

$$\tilde{E}^{e_{24}} = \{e_{86}, e_{82}\}$$

mit den Vaterbeziehungen

$$h^{e_{24}}(e_{86}) = e_{46} \quad h^{e_{24}}(e_{82}) = e_{42}$$

Da $E_0 \cap N^{e_{24}} = \emptyset$ *ist, gilt* $T^1 = \emptyset$

Somit ergibt sich der erweiterte Graph:

$$V^{e_{24}} = v_i, \quad i = 1..8$$

$$E^{e_{24}} = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{24}, e_{42}, e_{45}, e_{54}, e_{46}, e_{64}, e_{73}, e_{74}, e_{86}, e_{82}\}$$

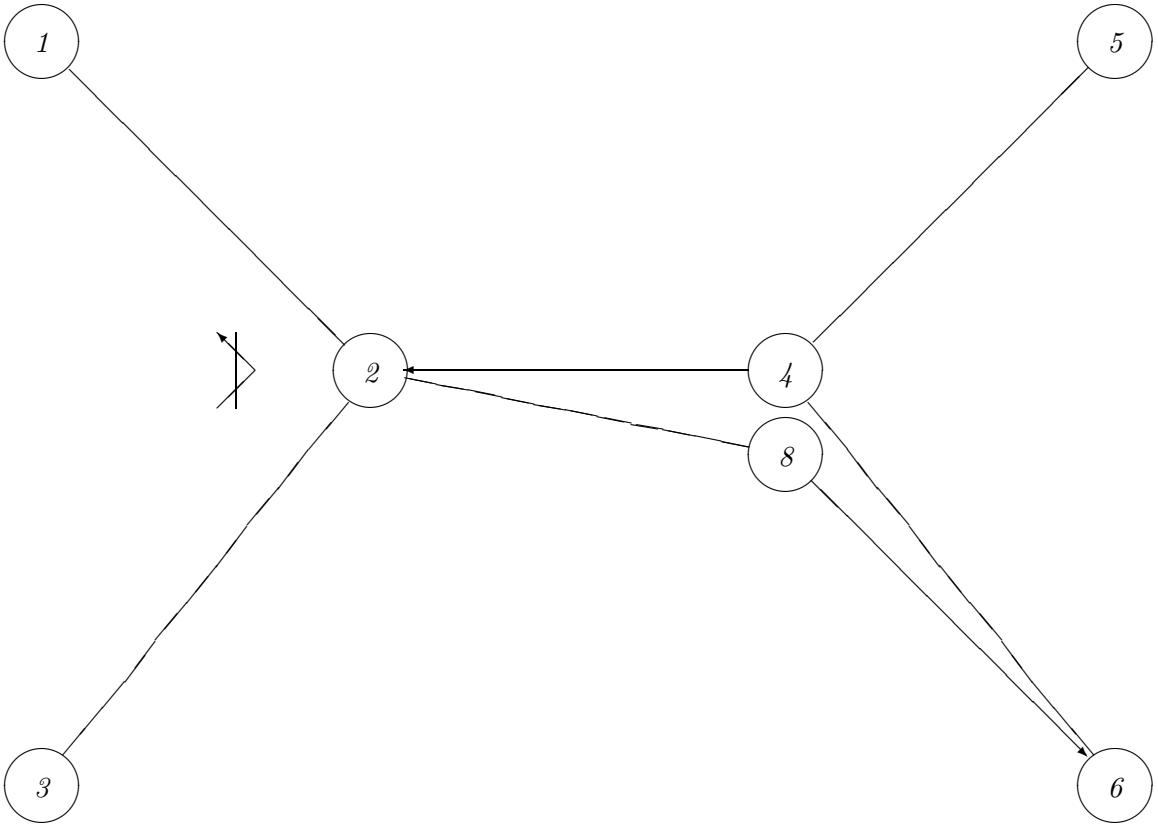
$$\alpha^{e_{24}}(e_{ij}) = v_i$$

$$\omega^{e_{24}}(e_{ij}) = v_j \text{ falls } (i,j) \neq (3,2), (2,4), (7,4)$$

$$\omega^{e_{24}}(e_{32}) = v_7 \;\; \omega^{e_{24}}(e_{24}) = v_8$$

$$\omega^{e_{24}}(e_{74}) = v_8 \text{ da } e_{74} = h^{e_{32}{}^{-1}}(e_{24})$$

Schritt 1b *mit* $f_1 = e_{24}$*:*



Der aus v_4 *erschaffene Knoten* $v^{e_{24}}$ *heiße analog zum Schritt 2a* v_8*. Da* (e_{24}, e_{45}) *verboten ist, gilt*

$$N^{e_{24}} = \{e_{42}, e_{46}\}$$

Die neuen Kanten sind

$$\tilde{E}^{e_{24}} = \{e_{86}, e_{82}\}$$

mit den Vaterbeziehungen

$$h^{e_{24}}(e_{86}) = e_{46} \quad h^{e_{24}}(e_{82}) = e_{42}.$$

Da $E_0 \cap N^{e_{24}} = \emptyset$ *ist gilt* $T^1 = \emptyset$*.*

Für den erweiterten Graphen gilt also:

$$V^{e_{24}} = v_i, \quad i = 1..6, 8$$

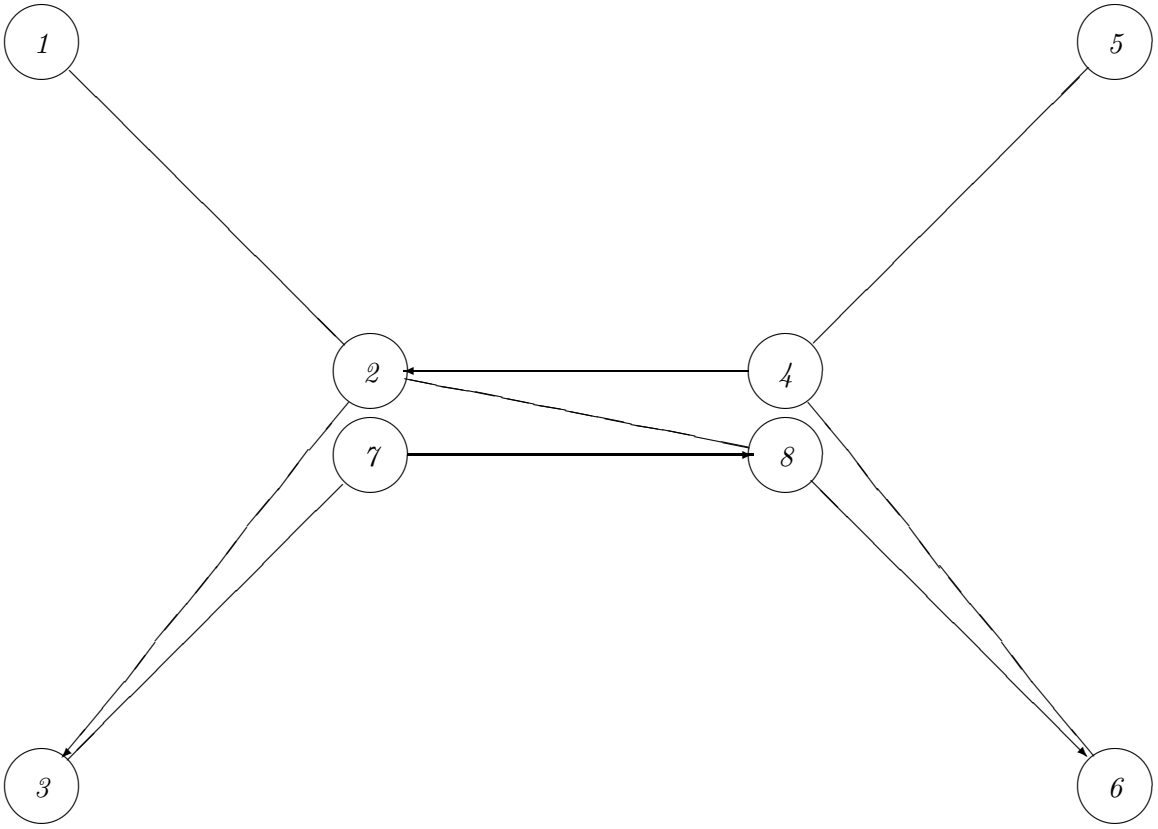
$$E^{e_{24}} = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{24}, e_{42}, e_{45}, e_{54}, e_{46}, e_{64}, e_{86}, e_{82}\}$$

$$\alpha^{e_{24}}(e_{ij}) = v_i$$

$$\omega^{e_{24}}(e_{ij}) = v_j \textit{ falls } (i,j) \neq (2,4)$$

$$\omega^{e_{24}}(e_{24}) = v_8$$

Schritt 2b *mit* $f_2 = e_{32}$*:*



Der aus v_2 *erschaffene Knoten* $v^{e_{32}}$ *heiße analog zum Schritt 1a* v_7*. Da* (e_{32}, e_{12}) *verboten ist gilt*

$$N^{e_{32}} = \{e_{24}, e_{23}\} \textit{ mit } \omega^{e_{24}}(e_{24}) = v_8$$

Die neuen Kanten seien

$$\tilde{E}^{e_{32}} = \{e_{78}, e_{73}\}$$

mit den Vaterbeziehungen

$$h^{e_{32}}(e_{78}) = e_{24} \quad h^{e_{32}}(e_{73}) = e_{23}.$$

Für den erweiterten Graph ergibt sich:

$$V^{e_{32}} = v_i, \quad i = 1..8$$

$$E^{e_{32}} = \{e_{12}, e_{21}, e_{23}, e_{32}, e_{24}, e_{42}, e_{45}, e_{54}, e_{46}, e_{64}, e_{74}, e_{78}\}$$

$$\alpha^{e_{32}}(e_{ij}) = v_i$$

$$\omega^{e_{32}}(e_{ij}) = v_j \; \textit{falls} \; (i,j) \neq (3,2), (2,4)$$

$$\omega^{e_{32}}(e_{32}) = v_7 \;\; \omega^{e_{32}}(e_{24}) = v_8$$

Die erweiterten Graphen aus den Schritten 2a und 2b sind isomorph.

B.2 Beispiele zur Erstellung eines Wegegraphen in Graphen mit Wegeverboten

Beispiel B.7 Um die Methode der Wegeverdopplung im Abschnitt 5.3 betrachten wir folgendes Beispiel eines Graphen mit nur einem Wegeverbot.

Gegeben sei $G = (V, E, \alpha, \omega)$ ein Graph mit den erzeugenden Mengen

$$V := \{v_1, , ..., v_3\} \quad \text{und} \quad E := \{e_{12}, e_{23}, e_{31}\},$$

den Inzidenzabbildungen $\alpha(e_{ij}) = v_i \quad \omega(e_{ij}) = v_j$ und dem Wegeverbot

$$p := (e_{12}, e_{23}, e_{31}).$$

Wir werden G so erweitern, daß ein von G erzeugter Wegegraph entsteht. Die Verdopplung des Weges ohne die erste und letzte Kante ergibt

Abbildung B.12 *Beispielgraph mit einem Wegeverbot*

Da im Ausgangsgraphen nur ein Rundlauf möglich ist und endet jeder Weg spätesten beim zweiten Erreichen des Knotens $v_{3'}$.

Beispiel B.8 Um die Methode der Wegeverdopplung im Abschnitt 5.3 und die anschließende Verbindung mit dem Ausgangsgraphen (Abschnitt 5.4) zu erläutern betrachten wir folgendes Beispiel eines Graphen mit nur einem Wegeverbot.

Gegeben sei $G = (V, E, \alpha, \omega)$ ein Graph mit der Knotenmenge

$$V = \{v_1, , ..., v_8\},$$

der Kantenmenge

$$E := \{f_{12}, f_{21}, f_{23}, f_{32}, f_{34}, f_{37}, f_{43}, f_{45}, f_{48}, f_{54}, f_{67}, f_{73}, f_{76}, f_{78}, f_{84}, f_{87}\},$$

den Inzidenzabbildungen $\alpha(f_{ij}) = v_i \quad \omega(f_{ij}) = v_j$ und dem Wegeverbot

$$p := (e_1, ..., e_4) = (f_{12}, f_{23}, f_{34}, f_{48}).$$

Wir werden G in zwei Schritten so erweitern, daß ein von G erzeugter Wegegraph entsteht. Die Kanten des Wegeverbotes werden wie im Algorithmus e_i genannt.

Schritt 1: Die Wegeverdopplung

Wir werden nun die Kanten und Knoten des Wegeverbotes ohne die jeweils ersten und letzten Kanten und Knoten verdoppeln. Diese Kanten heißen $e_i^p \ i = 2, 3$ und werden in der Menge $\tilde{E}^1$ zusammengefaßt. Die Menge aller zum verdoppelten Weg gehörenden Knoten heißt $\tilde{V}^p$ und ihre Elemente heißen $v_i^p \ i = 1, ..., 3$. Damit definieren wir

$$e_2^p := f_{2'3'} \quad e_3^p := f_{3'4'} \quad v_1^p = v_{2'} \quad v_2^p = v_{3'} \quad \text{und} \quad v_3^p = v_{4'}.$$

Die kanonischen Projektionen bilden die jeweiligen verdoppelten Elemente auf Ihre Originale ab. Es gilt also

$$\Pi_e^1(f_{2'3'}) = f_{23} \quad \Pi_e^1(f_{3'4'}) = f_{34} \quad \Pi_v^1(v_{2'}) = v_2 \quad \Pi_v^1(v_{3'}) = v_3 \quad \Pi_v^1(v_{4'}) = v_4.$$

Der verdoppelte Weg ist in der folgenden Abbildung fett dargestellt.

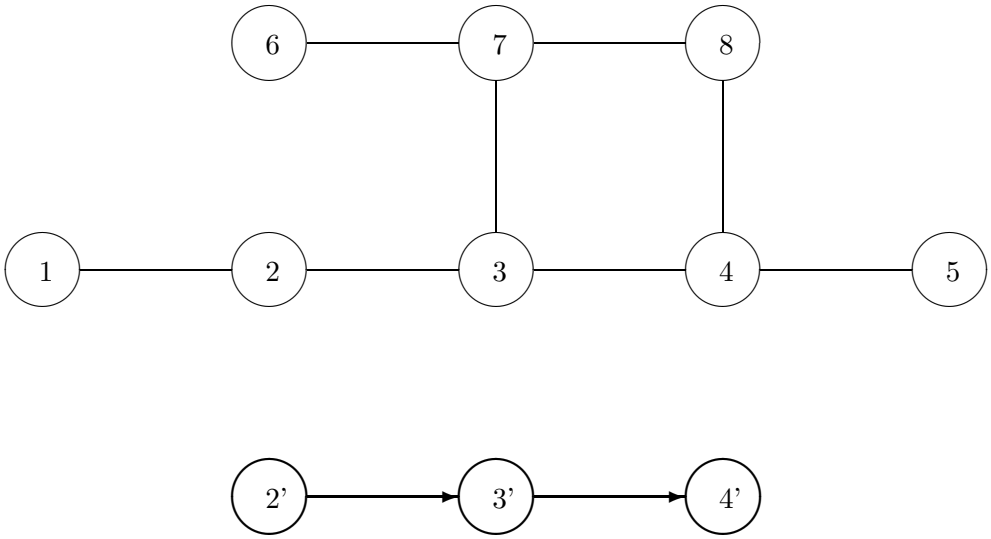


Abbildung B.13 *Graph der die Schritte Wegeverdopplung und Verbindung mit dem Ausgangsgraphen veranschaulicht.*

Die Abbildung ψ ist nur bei mehreren Wegeverboten einer Klasse relevant.

Schritt 2: Die Verbindung mit dem Ausgangsgraphen

Nun wird dieser neue Weg in den Graphen eingebunden. Zuerst wird der Kante f_{12} ein neuer Endpunkt - der Knoten $v_{2'}$ - zugewiesen. Damit kann man vom Knoten v_1 nicht mehr direkt zum Knoten v_2 gelangen. Der neue Weg kann nur über die Kante $e_1 = f_{12}$ erreicht werden.

Von der Kante $e_2 = f_{23}$ des Wegeverbotes aus betrachten wir nun die Menge aller Kanten N_2^p über die das Wegeverbot von $\omega(e_2) = v_3$ aus verlassen werden kann. Sei $f \in N_2^p$, so gelten für f die Bedingungen (5.7a): $\alpha(f) = \omega(e_2) = v_3$ und (5.7b): $f \neq e_3 = f_{34}$ (da nur ein Wegeverbot betrachtet wird. Damit ergibt sich

$$N_2^p = \{f_{32}, f_{37}\}.$$

Die Menge N_i^p definieren wir zu jedem e_i für $i = 1, ..., 3$ und definieren dann analog zur Gleichung (5.8) die Menge N

$$N := \{(e_1, f_{21}), (e_2, f_{32}), (e_2, f_{37}), (e_3, f_{43}), (e_3, f_{45})\}.$$

Wenn $(e_i, f) \in N$ ist, so bedeutet dies, daß das Wegeverbot p vom Endknoten von e_i aus über die Kante f verlassen werden kann. Jedem dieser Kantenpaare wird eine neue Kante zugeordnet. Diese Kanten werden den verdoppelten Weg mit dem Originalgraphen verbinden.

Diese neuen Kanten heißen $f_{2'1}, f_{3'2}, f_{3'7}, f_{4'3}, f_{4'5}$ mit der Inzidenzstruktur $\alpha(f_{ij}) = v_i, \omega(f_{ij}) = v_j$. Im Algorithmus werden diese Kanten in der Menge $\tilde{E}^2$ zusammengefaßt. Z.B. $f_{2'1}$ heißt dort

$e^p_{1,f_{21}}$, da diese Kante von $e_1(= f_{12})$ erreicht wird und in Richtung der Kante f_{21} verläuft. Die Abbildung h_2 ist folgendermaßen definiert:

$$h_2(f_{2'1}) = h_2(e^p_{1,f_{21}}) = f_{21} \quad h_2(f_{3'2}) = h_2(e^p_{2,f_{32}}) = f_{32} \quad h_2(f_{3'2}) = h_2(e^p_{2,f_{37}}) = f_{37}$$

$$h_2(f_{4'3}) = h_2(e^p_{3,f_{43}}) = f_{43} \quad h_2(f_{4'5}) = h_2(e^p_{3,f_{45}}) = f_{45}$$

Die neu erschaffenen Kanten werden in der folgenden Abbildung verstärkt dargestellt.

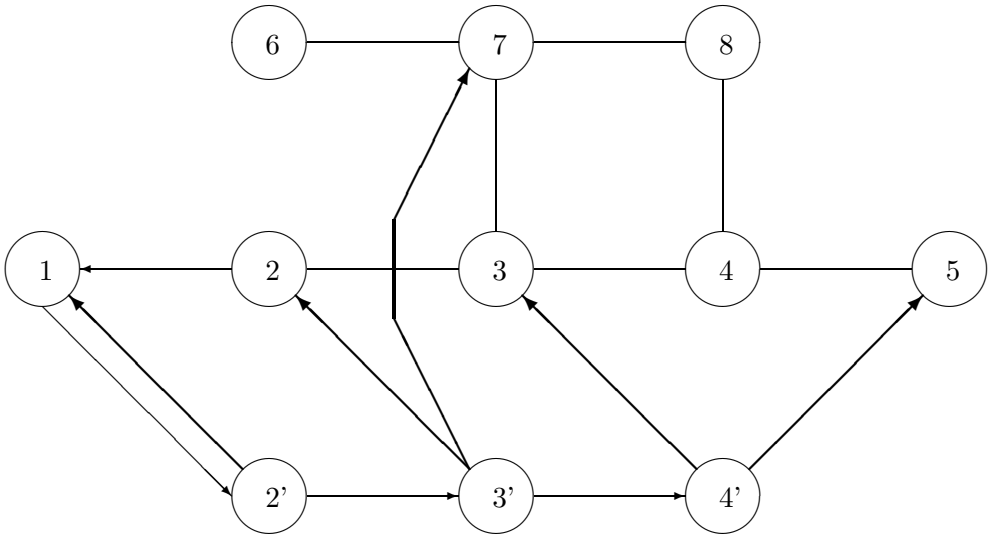


Abbildung B.14 *Wegegraph des Beispiels B.8*

Möchte man vom Knoten v_1 zum Knoten v_8 gelangen, so ist dies durch das Wegeverbot wv nicht auf direktem Wege über den Knoten v_4 möglich. Betrachten wir die Grapherweiterung, so kann v_1 nur in Richtung $v_{2'}$ verlassen werden. Gehen wir den Weg weiter, so gelangen wir nur zu neuen Knoten, und laut Konstruktion kann der Knoten v_8 nicht direkt vom Knoten $v_{4'}$ erreicht werden. Der Weg v_1, v_2, v_3, v_7, v_8 ist weiterhin möglich (siehe auch Satz 5.8).

Die einzige Kante, über die der neue Weg vom Ausgangsgraphen aus erreicht werden kann, ist $e_1 = f_{12}$. Er kann nur über eine Kante von $\tilde{E}^2$ wieder verlassen werden.

Beispiel B.9 Dieses Beispiel ist eine Erweiterung des Beispiels B.8 mit zwei Wegeverboten der gleichen Wegeverbotsklasse. Gegeben sei also der Graph (V, E, α, ω) aus Beispiel B.8, wobei zusätzlich zu dem Wegeverbot $p := (f_{12}, f_{23}, f_{34}, f_{48})$ das Wegeverbot

$$p' := (e'_1, ..., e'_4) = (f_{12}, f_{23}, f_{37}, f_{78})$$

gelten soll. Ihre Reihenfolge sei p, p'. Wir konstruieren wieder einen von G erzeugten Wegegraphen. Dazu führen wir zuerst den ersten Schritt des Abschnitts 5.3 analog zum Beispiel B.8 durch.

Da die Wegeverbote p und p' die gemeinsame Anfangskante f_{12} haben, gehören sie zu einer Wegeverbotsklasse. Weil $e_3 \neq e'_3$ und $e_i = e'_i$ für $i = 1, 2$ gilt, trennen sich die Wegeverbote nach der Kante f_{23}. Diesen gemeinsamen Anfangsverlauf der Wegeverbote wollen wir so nützen, daß die ersten gemeinsamen Kanten beider Wegeverbote nur ein Mal gesplittet werden und ein verdoppelter Restweg von p' an den verdoppelten Weg von p angehängt wird.

Trivialerweise gilt, daß $\tilde{P} = \{p\}$ und damit gilt, daß p maximaler Anfangsweg von p' ist mit $q = 2$ gemeinsamen Kanten. In diesem Schritt splitten wir also nur noch die Kante $e'_3 = f_{37}$ zu $e_3^{p'} = f_{3'7''}$ und deren Endknoten v_7 zu $v_{7''}$ und übernehmen $v_{2'}, v_{3'}$ und $f_{2'3'}$ aus dem Schritt 1. Es ergibt sich

$$\tilde{E}^{p'} = \{f_{3'7''}\} \quad V^{p'} = \{v_{7''}\}$$

Der Weg, der von p' erzeugt wurde, wird in der folgenden Abbildung verstärkt dargestellt.

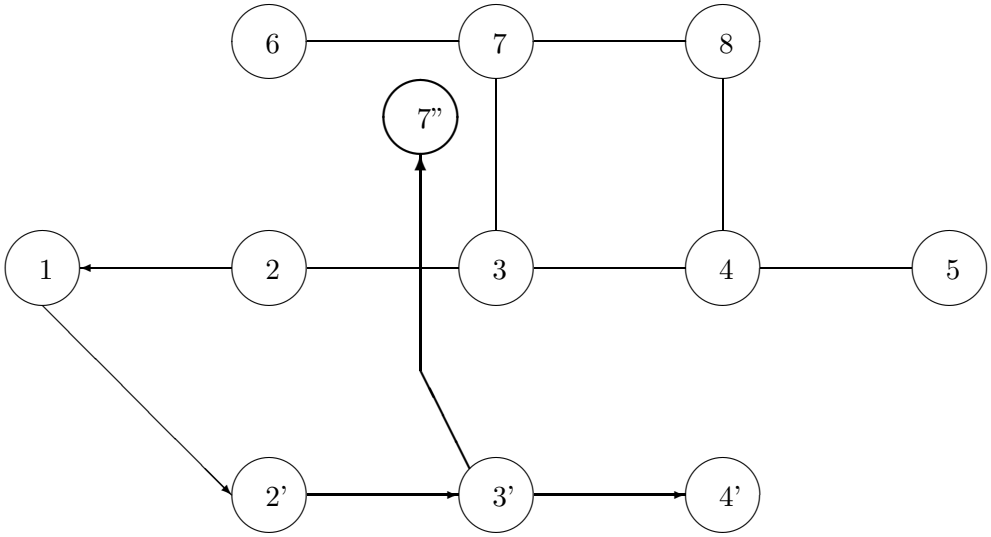


Abbildung B.15 *Erweiterter Graph nach der Wegeverdopplung von Abschnitt 5.3*

Die Verbindung des verdoppelten Weges (Abschnitt 5.4) von p verläuft fast analog zum Beispiel B.8. Nur die Kante f_{37} wird nicht verdoppelt, denn p' ist ein Endweg des Weges (f_{12}, f_{23}, f_{37}), und somit ist die Bedingung 5.7b nicht mehr erfüllt. Wiederum wird das Wegeverbot p' erst ab der Kante e'_3 betrachtet. Damit ergibt sich für die in 5.8 definierte Menge:

$$N = \{(e_1, f_{21}), (e_2, f_{32}), (e_3, f_{43}), (e_3, f_{45})(e'_3, f_{76}), (e'_3, f_{73})\}.$$

Insgesamt werden also 6 neue Kanten definiert:

$$\tilde{E}^2 = \{f_{2'1}, f_{3'2}, f_{4'3}, f_{4'5}, f_{7''6}, f_{7''3}\}.$$

Für Bijektion h_2 gilt: $h_2(f_{i'j}) = (*, f_{ij})$. Die neu erschaffenen Kanten sind in der folgenden Abbildung verstärkt dargestellt.

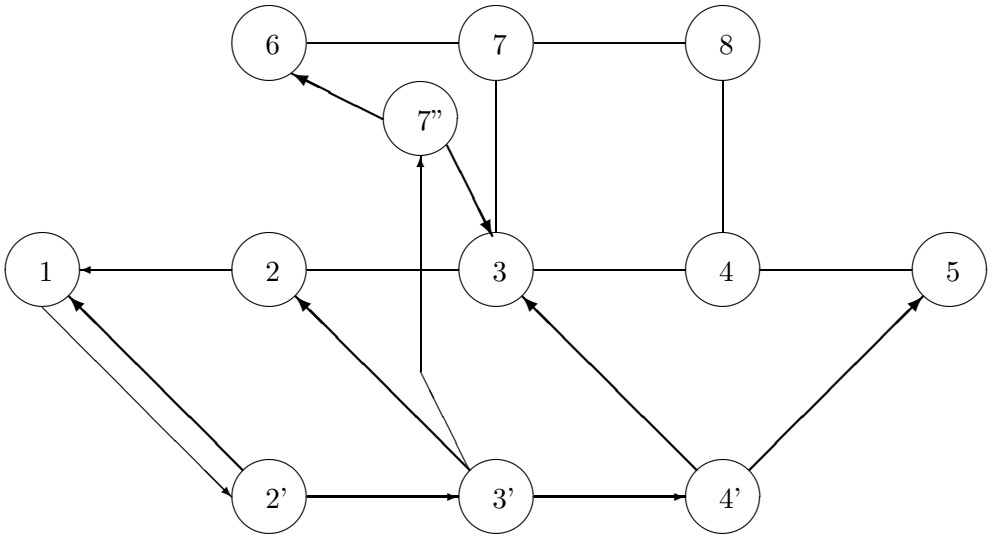


Abbildung B.16 *Erweiterter Graph nach der Verbindung mit dem Ausgangsgraphen vom Abschnitt 5.4*

Beispiel B.10 Um die Vorgehensweise im Abschnitt 5.5 zu erläutern, erweitern wir den Graph aus Beispiel B.8 um einen Knoten und um ein Wegeverbot. Gegeben ist also ein Graph $G = (V, E, \alpha, \omega)$ mit der Knotenmenge

$$V = v_i, \quad i = 1..9,$$

der Kantenmenge

$$E := \{f_{12}, f_{21}, f_{23}, f_{32}, f_{34}, f_{37}, f_{43}, f_{45}, f_{48}, f_{54}, f_{67}, f_{73}, f_{76}, f_{78}, f_{84}, f_{87}, f_{92}\},$$

den Inzidenzabbildungen $\alpha(f_{ij}) = v_i$, $\omega(f_{ij}) = v_j$ – und den verschachtelten Wegeverboten

$$p = p_1 = (e_1, ..., e_4) = (f_{12}, f_{23}, f_{34}, f_{48}) \text{ und}$$

$$p' = p_2 = (e'_1, ..., e'_4) = (f_{23}, f_{37}, f_{78}).$$

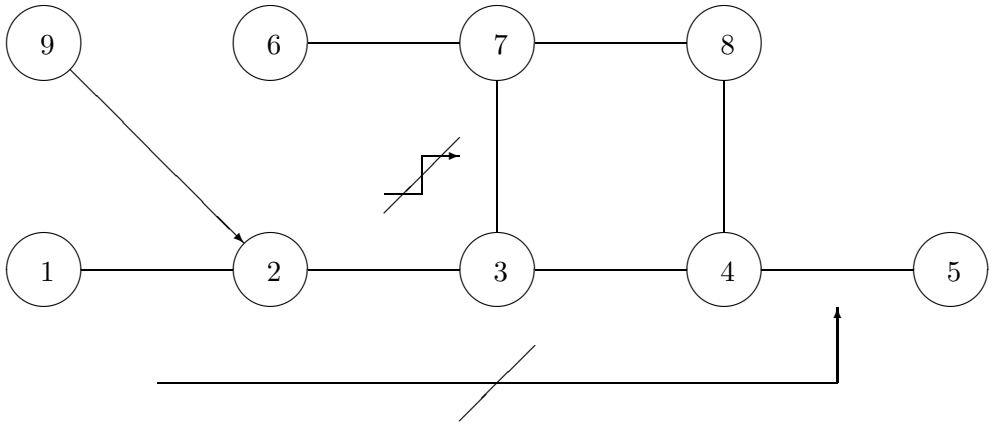


Abbildung B.17 *Beispielgraph mit zwei Wegeverboten, die ineinander beginnen*

Zuerst verdoppeln wir den Weg p analog zum Beispiel B.8. Beim Wegeverbot p' wird nur die Kante $e'_2 = f_{37}$ mit ihrem Anfangs- und Endknoten verdoppelt. Es sei

$$\tilde{E}_1^{p'} := \{f_{3''7''}\} \quad \tilde{V}_1^{p'} := \{v_{3''}, v_{7''}\}.$$

Das Wegeverbot p wird fast genau so wie im Beispiel B.8 mit dem Ausgangsgraphen verbunden. Die einzige Ausnahme besteht darin, daß die Kante f_{37} nicht verdoppelt wird, weil das Wegeverbot p' Endweg von $(f_{12} + f_{23} + f_{37})$ und dies im Widerspruch zur Bedingung 5.7b steht.

$$N^{p'} = \{(e'_1, f_{32}), (e'_1, f_{34}), (e'_2, f_{76}), (e'_2, f_{73})\}$$

und damit definieren wir die gesplitteten Kanten

$$\tilde{E}^2 := \{f_{3''2}, f_{3''4}, f_{7''6}, f_{7''3}\}$$

mit den zugehörigen Bijektionen

$$h_2(f_{i''j}) := (e_*^{p'}, f_{ij}).$$

Betrachten wir nun die Erweiterungen aus Abschnitt 5.5. Im Abschnitt 5.4 wurde die Menge $M = \{(e_2, f_{37})\}$ definiert, da p' (maximaler) Endweg von $(e_1, e_2) + f_{37}$ in P ist. Wir definieren eine neue Kante $e_{2,2}^{p,p'} := f_{3'7''}$, die beide Wegeverbote verbindet. Damit gilt

$$\tilde{E}^3 = \{f_{3'7''}\} \quad \text{und} \quad h_3(f_{3'7''}) = h_3(e_{2,2}^{p,p'}) = (e_2^p, e_2^{p'}).$$

Die im Abschnitt 5.5 definierte Kante wird in der folgenden Abbildung verstärkt dargestellt.

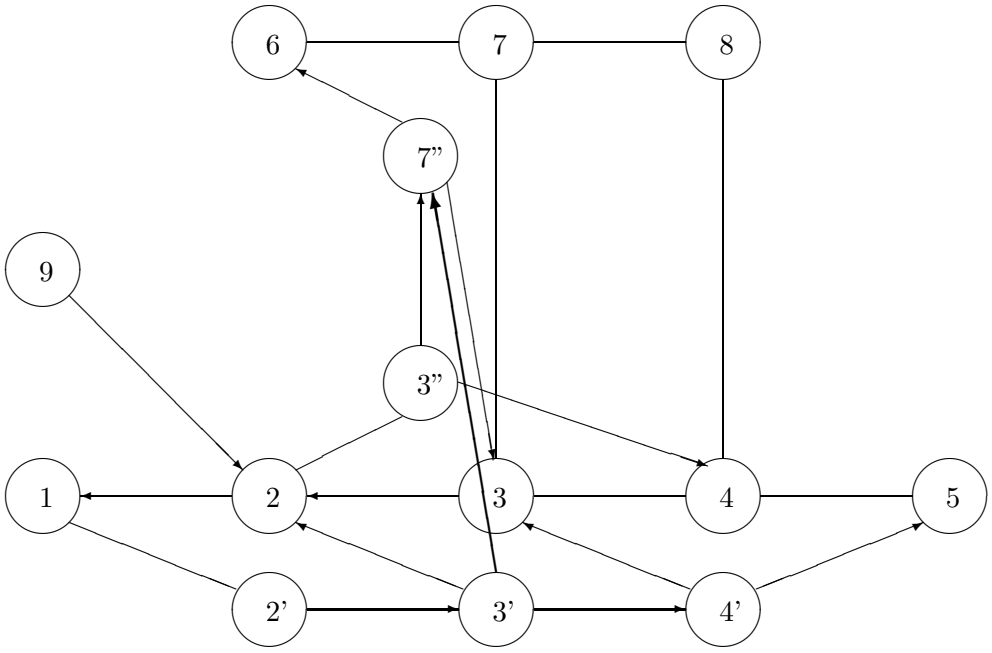


Abbildung B.18 *Wegegraph des Beispielgraphen mit zwei verschachtelten Wegeverboten*

Die kanonischen Projektionen sind folgendermaßen definiert:

$$\Pi_v^1(v_{i'}) = v_i \quad i = 2,3,4 \qquad \Pi_v^1(v_{i''}) = v_i \quad i = 3,7$$

$$\Pi_e^1(f_{2'3'}) = f_{23} \quad \Pi_e^1(f_{3'4'}) = f_{34} \quad \Pi_e^1(f_{3''7''}) = f_{37}$$

$$\Pi_e^2(f_{i'j}) = f_{ij} \quad \Pi_e^2(f_{i''j}) = f_{ij} \quad \Pi_e^3(f_{3'7''}) = f_{37}$$

Der Weg $(f_{92}, f_{23}, f_{34}, f_{48})$ besitzt im Wegegraphen ein Urbild - der Weg $(f_{12}, f_{23}, f_{37}, f_{78})$ dagegen nicht, obwohl zuerst die Wegeverbotserweiterung von p betreten wird.

Beispiel B.11 Um den Beweis des Satzes 5.6 besser zu veranschaulichen bedienen wir uns eines Beispiels. Sei $G = (V, E, \alpha, \omega)$ mit

$$V := \{v_1, ..., v_8\},$$

$$E := \{f_{12}, f_{16}, f_{21}, f_{23}, f_{32}, f_{45}, f_{54}, f_{56}, f_{61}, f_{65}, f_{67}, f_{76}, f_{78}, f_{87}\}$$

den Inzidenzabbildungen $\alpha(f_{ij}) = v_i$ $\omega(f_{ij}) = v_j$ und den Wegeverboten

$$p = (f_{45}, f_{56}, f_{61}, f_{12}, f_{23}) \quad p' = (f_{45}, f_{56}, f_{67}, f_{78})$$

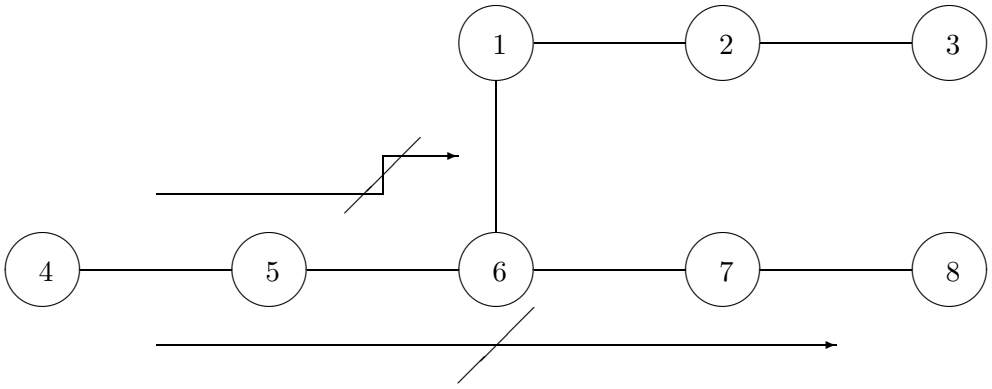


Abbildung B.19 *Beispielgraph mit zwei Wegeverboten, der selben Äquivalenzklasse*

Zuerst betrachten wir die Bearbeitung in der Reihenfolge p, p' - dies entspricht dem Fall a.

Zuerst verdoppeln wir analog zum Schritt 5.3 das Wegeverbot p ohne erste und letzte Kante. Die neu erschaffenen Knoten sind

$$\tilde{V}_a^p = \{v_{1,a}^p, v_{2,a}^p, v_{3,a}^p\} = \{v_{5',a}, v_{6',a}, v_{7',a}\}$$

und die neuen Kanten heißen

$$\tilde{E}_{1,a}^p = \{e_{2,a}^p, e_{3,a}^p\} = \{f_{5'6',a}, f_{6'7',a}\}.$$

Der maximale Anfangsweg von p' in $\tilde{P} = \{p\}$ ist p mit $q = 2$ gemeinsamen Kanten. Im zweiten Schritt der Wegeverdopplung müssen also nur die Kanten f_{61} und f_{12} gesplittet werden. Es werden folgende neuen Elemente erschaffen: $\tilde{V}_a^{p'} = \{v_{1'',a}, v_{2'',a}\}$ und $\tilde{E}_{1,a}^{p'} = \{f_{6'1'',a}, f_{1''2'',a}\}$. Jetzt werden die verdoppelten Wege in den Ausgangsgraphen eingebunden. Dazu definieren wir analog zur Gleichung 5.8 die Menge N_a:

$$N_a = \{(e_1^p, f_{54}), (e_2^p, f_{65}), (e_3^p, f_{76}), (e_3^{p'}, f_{16}), (e_4^{p'}, f_{21})\}.$$

Die Menge M bleibt in diesem Beispiel leer. Zu jeder Kante aus N_a definieren wir analog zum Abschnitt 5.4 eine neue Kante:

$$\tilde{E}_a^2 = \{f_{5'4,a}, f_{6'5,a}, f_{7'6,a}, f_{1''6,a}, f_{2''1,a}\}.$$

Für die Bijektion h_2 gilt $h_2(f_{i'j,a}) = (*, f_{ij})$. Der Abschnitt 5.5 entfällt, da $M = \emptyset$. Für alle neuen Kanten gilt die Inzidenzstruktur $\alpha(f_{ij}) = v_i$, $\omega(f_{ij}) = v_j$. Nur der Endknoten von $e_1 = f_{45}$ wird verändert: $\omega(f_{45}) = v_1^p = v_{5',a}$.

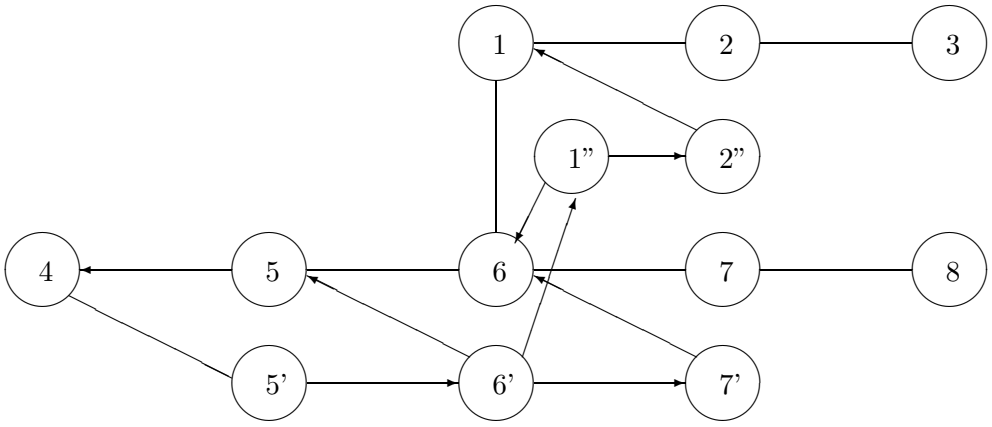


Abbildung B.20 *Wegegraph des Beispielgraphen mit zwei Wegeverboten, mit der Wegeverbotsreihenfolge p, p'*

Wir vertauschen nun die Reihenfolge der Wegeverbote und betrachten zuerst $p' = (f_{45}, f_{56}, f_{61}, f_{12})$ – dies entspricht dem Fall b. Zuerst verdoppeln wir wieder das erste Wegeverbot p' ohne erste und letzte Kante. Die neu erschaffenen Elemente sind

$$\tilde{V}_b^{p'} = \{v_{1,b}^{p'}, v_{2,b}^{p'}, v_{3,b}^{p'}, v_{4,b}^{p'}\} = \{v_{5',b}, v_{6',b}, v_{1',b}, v_{2',b}, \}$$

$$\tilde{E}_{1,b}^{p'} = \{e_{2,b}^{p'}, e_{3,b}^{p'}, e_{4,b}^{p'}, \} = \{f_{5'6',b}, f_{6'1',b}, f_{1'2',b}\}.$$

Der maximale Anfangsweg von p in $\tilde{P} = \{p'\}$ ist p' wieder mit $q = 2$ gemeinsamen Kanten. Im zweiten Schritt der Wegeverdopplung wird also nur noch die Kante f_{67} gesplittet. Dabei werden die Elemente der Mengen $\tilde{V}_b^p = \{v_{7'',b}\}$ und $\tilde{E}_{1,b}^p = \{f_{6'7'',b}\}$ erschaffen. Die Menge N ergibt sich zu

$$N_b = \{(e_1^{p'}, f_{54}), (e_2^{p'}, f_{65}), (e_3^{p'}, f_{16}), (e_4^{p'}, f_{21}), (e_3^p, f_{76})\},$$

die Menge M bleibt wie im Fall a leer. Die zu jedem Kantenpaar aus N_b definierten neuen Kanten heißen

$$\tilde{E}_b^2 = \{f_{5'4,b}, f_{6'5,b}, f_{1'6,b}, f_{2'1,b}, f_{7''6,b}, \}.$$

Wie im Fall a gilt für die Bijektion h_2: $h_2(f_{i'j,b}) = (*, f_{ij})$. Für f_{45} gilt $\omega(f_{45}) = v_1^{p'} = v_{5',b}$ sonst ist die Inzidenz $\alpha(f_{ij}) = v_i$, $\omega(f_{ij}) = v_j$ definiert.

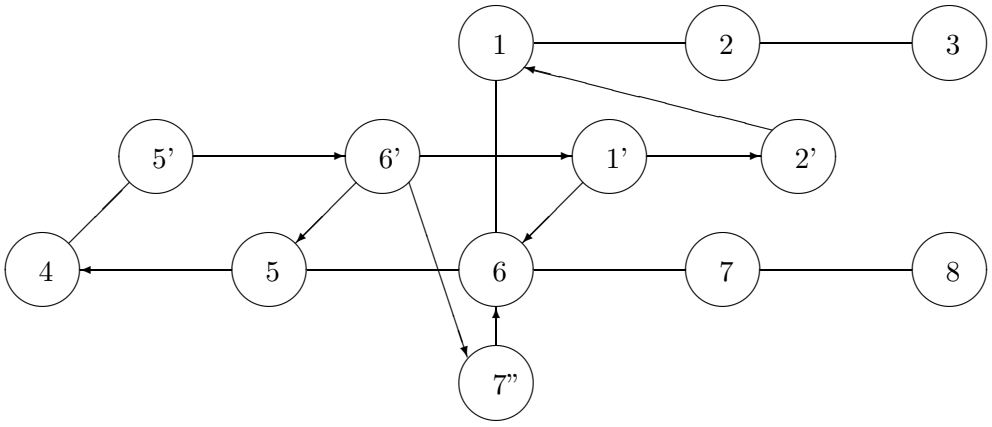


Abbildung B.21 *Wegegraph des Beispielgraphen mit zwei Wegeverboten mit der Wegeverbotsreihenfolge p', p*

Der gemeinsame Anfangsweg von p und p' ist $w = (f_{45}, f_{56})$. Die für die Definition von g und g_v gebrauchten Mengen sind:

$$\begin{array}{lll} \tilde{E}^1_{w,a} = \{f_{5'6',a}\} & \tilde{E}^2_{w,a} = \{f_{5'4,a}, f_{6'5,a}\} & \tilde{V}^1_{g,a} = \{v_{5',a}, v_{6',a}\} \\ \tilde{E}^1_{w,b} = \{f_{5'6',b}\} & \tilde{E}^2_{w,b} = \{f_{5'4,b}, f_{6'5,b}\} & \tilde{V}^1_{w,b} = \{v_{5',b}, v_{6',b}\} \end{array}$$

Um die Abbildung g zu veranschaulichen wählen wir exemplarisch zwei Kanten aus. $f_{1''6,a} \in \tilde{E}^1_a \backslash \tilde{E}^1_{w,a}$ und damit gilt:

$$\begin{array}{lll} g(f_{1''6,a}) & = (h_{2,b})^{-1}(h_{2,a}(f_{1''6,a})) & \text{Definition von } g \\ & = (h_{2,b})^{-1}((e^{p'}_{3,a}, f_{16})) & \text{Definition von } h_{2,a} \\ & = f_{1'6,b} & \text{Definition von } h_{2,b} \end{array}$$

$f_{5'6',a} \in \tilde{E}^1_{w,a}$:

$$\begin{array}{lll} g(f_{5'6',a}) & = (h^{p'}_{1,b})^{-1}(h^{p'}_{1,a}(f_{5'6',a})) & \text{Definition von } g \\ & = (h^{p'}_{1,b})^{-1}(f_{56}) & \text{nach Gleichung 5.1} \\ & = e^{p'}_2 = f_{5'6',b} & \text{nach Gleichung 5.1.} \end{array}$$

Anhang C

Programmlistings

C.1 Das Programmlisting von KnSplit

```
program Knotensplitting;

const unendlich = 30000;
      sVerbote  =’Verbote:’;
      sNeuerKnotenNr=’Neuer Knoten Nr.’;

type rVerbot  = record
                VonKa  :integer;
                UeberKn:word;
                NachKa :integer;
     end; (* record *)
     rVerbotsListe  = record
                Verbot:rVerbot;
                p:Pointer;
     end; (* record *)
     pVerbotsliste =^rVerbotsliste;
     rKante  = record
               VonKn  :word;
               UeberKa:integer;
               NachKn :word;
               KaIn_E0:boolean;
               Entf   :real;
               OldVonKn, OldNachKn:word;
     end; (* record *)
     rKantenListe  = record
               Kante:rKante;
               p:Pointer;
```

```
     end; (* record *)
     pKantenListe   =^rKantenListe;

var AnzKnoten: word;
    AnzKantenGesplittet:longint;
    DInt     : Integer;
    DStr     : String;
    pStartVerbot, pEndeVerbot: pVerbotsliste;
    pStartKante,  pEndeKante : pKantenliste;

(* Erweitertes Pascal *)

function BlancCut(S:String):String;
var i:byte;
    Erg:string;
begin
  Erg:='';
  for i:=1 to length(s) do if s[i]<>' ' then Erg:=Erg+s[i];
  BlancCut:=Erg;
end;

function WordCut(var S:String):String;
var i:byte;
    Erg:string;
begin
  Erg:=''; WordCut:='';
  if BlancCut(s)='' then exit;
  while s[1]=' 'do s:=copy(s,2,length(s)-1);
  if s='' then exit;
  s:=s+' ';
  for i:=1 to length(s) do
    if s[i]<>' ' then Erg:=Erg+s[i]
    else begin
      s:=copy(s,length(Erg)+1,Length(s)-Length(Erg));
      WordCut:=Erg;
      exit;
    end;
end;

function ValueNaechsteZahl(var S:String):longint;
var t:string; Fehler:integer; Erg:longint;
begin Erg:=0;
  t:=WordCut(s); val(t,Erg,Fehler);
  If Fehler>0 then writeln(#7+'Zahl erwartet');
```

```
  ValueNaechsteZahl:=Erg;
end;

(* Listentechnik *)

procedure ErschaffeNeuesVerbotsElement(var Pt:pVerbotsliste);
begin New(Pt);
  Pt^.Verbot.VonKa:=0;
  Pt^.Verbot.UeberKn:=0;
  Pt^.Verbot.NachKa:=0;
  Pt^.P:=Nil;
end;

procedure SetVerbot(Verbot:rVerbot);
var Pt:pVerbotsliste;
begin ErschaffeNeuesVerbotsElement(Pt);
  Pt^.Verbot:=Verbot;
  if pStartVerbot=Nil then pStartVerbot:=Pt else pEndeVerbot^.P:=Pt;
  pEndeVerbot:=Pt;
end;

function IsVerboten(VonKa,NachKa:integer):boolean;
var pAktVerbot: pVerbotsliste;
begin IsVerboten:=false;
  pAktVerbot:=pStartVerbot;
  while pAktVerbot<>Nil do begin
    if (pAktVerbot^.Verbot.VonKa =VonKa)  and
       (pAktVerbot^.Verbot.NachKa=NachKa) then begin
         IsVerboten:=true; exit; end;
    pAktVerbot:=pAktVerbot^.P;
  end;
end;

procedure ErschaffeNeueKante(var Pt:pKantenListe);
begin New(Pt);
  Pt^.Kante.VonKn:=0;
  Pt^.Kante.UeberKa:=0;
  Pt^.Kante.NachKn:=0;
  Pt^.Kante.KaIn_E0:=false;
  Pt^.Kante.Entf:=unendlich;
  Pt^.Kante.OldVonKn:=0;
  Pt^.Kante.OldNachKn:=0;
  Pt^.P:=Nil;
end;
```

```
procedure SetKante(Kante:rKante);
var Pt:pKantenliste;
begin ErschaffeNeueKante(Pt);
  Pt^.Kante:=Kante;
  if pStartKante=Nil then pStartKante:=Pt else pEndeKante^.P:=Pt;
  pEndeKante:=Pt;
end;

function GetOriginalNummer(KnNr:word):word;
var pAktKante:pKantenliste;
begin GetOriginalNummer:=KnNr; pAktKante:=pStartKante;
  while pAktKante<>Nil do begin
    if KnNr=pAktKante^.Kante.VonKn then begin
      if KnNr<>pAktKante^.Kante.OldVonKn then begin
        GetOriginalNummer:=pAktKante^.Kante.OldVonKn; exit;
      end;
    end;
    pAktKante:=pAktKante^.P;
  end;
end;

(* Datei-Operationen *)

procedure GraphMitVerbotenEinlesen(DateiName:string);
var Gelesen:String;
    DateiNr:text;
    bVerbote:boolean;
    Kante:rKante;
    Verbot:rVerbot;
begin
  assign(DateiNr,DateiName); reset(DateiNr); bVerbote:=false;
  Kante.KaIn_EO:=false;
  repeat
    readln(DateiNr,Gelesen);
    if copy(Gelesen,1,length(sNeuerKnotenNr)) =sNeuerKnotenNr then begin
      inc(AnzKnoten);
    end else if copy(Gelesen,1,length(sVerbote)) =sVerbote then begin
      bVerbote:=true;
    end else if not bVerbote then begin
      Kante.VonKn      := AnzKnoten;
      Kante.NachKn     :=ValueNaechsteZahl(Gelesen);
      Kante.UeberKa    :=ValueNaechsteZahl(Gelesen);
      Gelesen:=WordCut(Gelesen); Val(Gelesen,Kante.Entf,DInt);
```

```
      Kante.OldVonKn   :=Kante.VonKn;
      Kante.OldNachKn  :=Kante.NachKn;
      SetKante(Kante);
    end else if bVerbote then begin
      Verbot.VonKa  :=ValueNaechsteZahl(Gelesen);
      Verbot.UeberKn:=ValueNaechsteZahl(Gelesen);
      Verbot.NachKa :=ValueNaechsteZahl(Gelesen);
      SetVerbot(Verbot);
    end;
  until eof(DateiNr);
  Close(DateiNr);
end;

procedure GraphOhneVerboteSpeichern(DateiName:string);
var DateiNr:text;
    Schreibe:string;
    pAktKante:  pKantenliste;
    KnNrVon,OldKnNrVon:Longint;
begin
  assign(DateiNr,DateiName); rewrite(DateiNr);
  for KnNrVon:=1 to AnzKnoten do begin
    pAktKante:=pStartKante;
    Str(KnNrVon,Schreibe);
    OldKnNrVon:=GetOriginalNummer(KnNrVon);
    If OldKnNrVon<>KnNrVon then begin
      Schreibe:=Schreibe+' gesplittet von ';
      Str(OldKnNrVon,DStr); Schreibe:=Schreibe+DStr;
    end;
    writeln(DateiNr,sNeuerKnotenNr+Schreibe);
    while pAktKante<>Nil do begin
      if (KnNrVon=pAktKante^.Kante.VonKn) then begin
        Str(pAktKante^.Kante.NachKn, Schreibe);
        Str(pAktKante^.Kante.UeberKa,DStr);   Schreibe:=Schreibe+'  '+DStr;
        Str(pAktKante^.Kante.Entf:10:2,DStr); Schreibe:=Schreibe+'  '+DStr;
        writeln(DateiNr,Schreibe);
      end;
      pAktKante:=pAktKante^.P;
    end;
  end;
  Close(DateiNr);
end;

(* Algorithmus *)
```

```
procedure Create_Menge_E0_und_splitte_Knoten;
var pAktVerbot: pVerbotsliste; pAktKante:  pKantenliste;
begin
  pAktKante :=pStartKante;
  while pAktKante<>Nil do begin
    pAktVerbot:=pStartVerbot;
    while pAktVerbot<>Nil do begin
      if pAktKante^.Kante.UeberKa=pAktVerbot^.Verbot.VonKa then begin
        pAktKante^.Kante.KaIn_E0:=true;
        Inc(AnzKnoten);
        pAktKante^.Kante.NachKn:=AnzKnoten; (* Knotensplitting *)
        pAktVerbot:=pEndeVerbot;
      end;
      pAktVerbot:=pAktVerbot^.p;
    end;
    pAktKante:=pAktKante^.p;
  end;
end;

procedure splitte_Kanten;
var pAktVerbot: pVerbotsliste; pAktKanteVon,pAktKanteNach: pKantenliste;
    Kante:rKante;
begin
  pAktKanteVon:=pStartKante;
  while pAktKanteVon<>Nil do begin
    pAktKanteNach:=pStartKante;
    if pAktKanteVon^.Kante.KaIn_E0 then begin
      while pAktKanteNach<>Nil do begin
        if (pAktKanteVon^.Kante.OldNachKn=pAktKanteNach^.Kante.VonKn) then
          begin if not IsVerboten(pAktKanteVon ^.Kante.UeberKa,
                                  pAktKanteNach^.Kante.UeberKa) then begin
            Kante:=pAktKanteNach^.Kante;    (* Kantensplitting *)
            Kante.VonKn:=pAktKanteVon^.Kante.NachKn;
            Kante.KaIn_E0:=false;
            SetKante(Kante);
            inc(anzKantenGesplittet);
          end;
        end;
        pAktKanteNach:=pAktKanteNach^.p;
      end;
    end;
    pAktKanteVon:=pAktKanteVon^.p;
  end;
end;
```

```
(* Initialisierung *)

procedure init;
var i:word;
begin
  pStartVerbot:=Nil; pEndeVerbot:=Nil;
  pStartKante:=Nil;  pEndeKante:=Nil;
  AnzKnoten:=  0;    AnzKantenGesplittet:=0;
end;

procedure uninit;
begin
  pEndeVerbot:=pStartVerbot;
  while pStartVerbot<>Nil do begin
    pEndeVerbot:=pStartVerbot^.P; Dispose(pStartverbot);
    pStartVerbot:=pEndeVerbot;
  end;
  pEndeKante:=pStartKante;
  while pStartKante<>Nil do begin
    pEndeKante:=pStartKante^.P; Dispose(pStartKante);
    pStartKante:=pEndeKante;
  end;
end;

begin
  init;
  GraphMitVerbotenEinlesen('StadtDat\Graph7.dat');
  Create_Menge_E0_und_splitte_Knoten;
  splitte_Kanten;
  GraphOhneVerboteSpeichern('StadtDat\ErwGra7.dat');
  uninit;
end.
```

C.2 Das Programmlisting von Dijkstra

```
program DijkstraAlgorithmus;

uses declare;

type tKosten = real;
     tMatrix = array[1..MaxKnoten,1..MaxKnoten] of tKosten;
```

```
     rListe  = record
                 KnNr:word;
                 KaNr:word;
                 Entf:tKosten; (* Kosten *)
                 P :pointer;
               end; (* record *)
     pListe=^rListe;
     tListe = array[1..MaxKnoten] of pListe;
     rBaum   = record
                 Vg:word;
                 Kn:tKosten; (* Kosten *)
               end; (* record *)
     tBaum = array[1..MaxKnoten] of rBaum;

var Dist     : tListe;
    Baum     : tBaum;
    AnzKnoten: word;
    DateiNr  : text;

procedure NeueListe(var Pt:pListe);
begin New(Pt); Pt^.KnNr:=0; Pt^.KaNr:=0; Pt^.Entf:=unendlich; Pt^.P:=Nil; end;

procedure NeuNachbar(KnNr:word);
var Pt,PtNeu:pListe;
begin
  Pt:= Dist[KnNr];
  repeat
    if Baum[Pt^.KnNr].Vg =NoNbar then Baum[Pt^.KnNr].Vg:= Nbar;
    PtNeu:=Pt; Pt:=PtNeu^.P;
  until (Pt =nil);
end;

procedure BaumErweitern;
var Kandidat, Vorgaenger, i:word;
    Minimum :real;
Pt,PtNeu:pListe;
begin
  Minimum:=unendlich; Kandidat:=0; Vorgaenger:=0;
  for i:= 1 to AnzKnoten do begin
    if Baum[i].Vg = Nbar then begin
      Pt:=Dist[i];
      repeat
        if not(Baum[Pt^.KnNr].Vg =Nbar) and
           not(Baum[Pt^.KnNr].Vg =NoNbar) then begin
```

```
          if Pt^.Entf+Baum[Pt^.KnNr].Kn< Minimum then begin
            Kandidat:= i; Vorgaenger:= Pt^.KnNr;
            Minimum:= Pt^.Entf+Baum[Pt^.KnNr].Kn;
          end;
        end;
        PtNeu:= Pt; Pt:= PtNeu^.P;
      until (Pt =Nil);
    end;
  end;
  if Kandidat =0 then exit;
  Baum[Kandidat].Vg:= Vorgaenger;
  Baum[Kandidat].Kn:= Minimum;
  NeuNachbar(Kandidat);
end;

procedure ListeEinlesen(DateiName:string; var aListe:tListe; var AnzElem:word);
var i,KnotenNr,KantenNr:word;
    t,s:string;
    Entfernung:tKosten;
    Fehler:integer;
    DateiNr:text;
    PtAlt,PtNeu,Pt:pListe;
    bVerbote:boolean;
begin
  assign(DateiNr,DateiName); reset(DateiNr);
  i:=0; bVerbote:=false;
  repeat
    readln(DateiNr,s);
    if copy(s,1,length(sNeuerKnotenNr)) =sNeuerKnotenNr then begin
      inc(i); NeueListe(Pt); aListe[i]:=Pt;
      if i>0 then begin Dispose(PtAlt^.P); PtAlt^.P:=NIL; end;
    end else if copy(s,1,length(sVerbote)) =sVerbote then begin
      bVerbote:=true;
    end else if not bVerbote then begin
      t:=WordCut(s); val(t,KnotenNr,Fehler);
      if Fehler >0 then write(#7);
      t:=WordCut(s); val(t,KantenNr,Fehler);
      if Fehler >0 then write(#7);
      t:=WordCut(s); val(t,Entfernung,Fehler);
      if Fehler >0 then write(#7);
      If Entfernung < Unendlich then begin
        Pt^.KnNr:=KnotenNr;
        Pt^.KaNr:=KantenNr;
        Pt^.Entf:=Entfernung;
```

```
        NeueListe(PtNeu); Pt^.P:=PtNeu; PtAlt:=Pt; Pt:=PtNeu;
      end;
    end;
  until eof(DateiNr);
  Dispose(PtAlt^.P);
  PtAlt^.P:=NIL;
  Close(DateiNr);
  AnzElem:=i;
end;

procedure BaumSpeichern(StartKnotenNr:word);
var i,j:word;
    Erg,s:string;
begin
  str(StartKnotenNr,s);
  writeln(DateiNr,sNeuerKnotenNr+s);
  for i:=1 to AnzKnoten do begin
    str(i,Erg);
    if Baum[i].Vg = NoNbar then begin
      Erg:='Knoten Nr '+Erg+' im Baum nicht erreichbar';
    end else begin
      j:=i;
      while (Baum[j].Vg <>Wurzel) and (j <>0) do begin
        Str(Baum[j].Vg,s);
        Erg:=Erg+' '+s;
        j:=Baum[j].Vg;
      end;
      Str(Baum[i].Kn:10:2,s);
      Erg:=Erg+' '+sEntfernung+s;
      if j=0 then begin
        i:=i;
      end;
    end;
    writeln(DateiNr,Erg);
  end;
end;

procedure SetStartKnoten(Nummer:word);
var i,j:word;
begin
  Baum[Nummer].Vg:= Wurzel;
  Baum[Nummer].Kn:= 0;
  NeuNachbar(Nummer);
end;
```

```
procedure ClearBaum;
var i:word;
begin
  for i:=1 to MaxKnoten do begin
    Baum[i].Vg:= NoNbar; Baum[i].Kn:= unendlich;
  end;
end;

procedure init;
var i:word;
begin
  ClearBaum;
  for i:=1 to MaxKnoten do Dist[i]:=    Nil;
  AnzKnoten:=  0;
end;

var i,Wrzl:word;
begin
  init;
  ListeEinlesen(Pfad+fnGraph+getFileCodeNr+spDat,Dist,AnzKnoten);
  assign(DateiNr,Pfad+fnAlg+getFileCodeNr+'P'+spDat); rewrite(DateiNr);
  for Wrzl:=1 to AnzKnoten do begin
    ClearBaum;
    SetStartKnoten(Wrzl);
    for i:=1 to AnzKnoten-1 do begin
      BaumErweitern;
    end;
    BaumSpeichern(Wrzl);
  end;
  Close(DateiNr);
end.
```

C.3 Das Programmlisting von Grapherstellen

```
unit GraphErstellen; (* Zusammenschlieen der Kanten zu einem Graph *)

interface

uses Klasse, KlDisk, DatDisk, Bilder, Declare, crt, graph, Verbot, Ellipsen,
     VGAMode;
```

```
procedure MakeKnoten;
procedure GraphBauen;

implementation

function GetKnotenNr(X,Y:real; var ErgKn:rKnoten):word;
var KnNr, Erg:word;
    Abstand, Minimum:real;
    s1,s2:String;
var Knoten:rKnoten;
begin GetKnotenNr:=1; Minimum:=10E37; Erg:=0;
  for KnNr:=1 to GetAnzKnoten do begin
    GetKnDisk(KnNr,Knoten);
    Abstand:=Abst(Knoten.X,Knoten.Y,X,Y);
    if (Abstand <epsKreuz) and (Abstand <Minimum) then begin
      Minimum:=Abstand; Erg:=KnNr; ErgKn:=Knoten;
    end;
  end;
  if (Erg =0) then begin
    Str(x:7:2,S1); Str(y:7:2,S2);
    FehlerAusg('Punkt '+S1+'   '+S2+' keinem Knoten zuzuordnen!');
  end else GetKnotenNr:=Erg;
end;

function IsStartKn(KnNr:word; Id:rId):boolean;
var Antw:byte;
    DistS, DistE:real;
    S1,S2:String;
    Strasse:rStrasse;
var Knoten:rKnoten;
begin
  GetKnDisk(KnNr,Knoten);
  GetStrasse(Strasse,id);
  Antw:=0; Str(Strasse.Id.Name,S1); Str(KnNr,S2);
  DistS:=Abst(Knoten.X,       Knoten.Y,
              Strasse.StartX, Strasse.StartY);
  DistE:=Abst(Knoten.X,       Knoten.Y,
              Strasse.EndeX,  Strasse.EndeY);
  if (DistS <epsKreuz) then Antw:=1;
  if (DistE <epsKreuz) then begin
    if (Antw<>1) or (DistS >DistE) then begin
      if bKurzStr then Antw:=2
      else FehlerAusg('Strasse Nr '+S1+' hat gemeinsamen Start und Ziel');
    end;
```

```
  end;
  case Antw of
   0 : FehlerAusg('Strasse '+S1+' hat Knoten '+S2+' nicht als Endpunkt');
   1 : IsStartKn:=true;
   2 : IsStartKn:=false;
  end; (* case *)
end;

procedure GetEndKnotenNummer(Id:rId; var KnNr1,KnNr2:word);
var KnNr :array[1..3] of word;
    KaNr : word;
    X,Y  : real;
    i    : byte;
    AusgStr,s:string;
    Strasse:rStrasse;
var Knoten:array[1..3] of rKnoten;
begin (* Testbetreib von Changegraph *)
  KnNr1:=0; KnNr2:=0;
  GetStrasse(Strasse,Id);
  if (Strasse.Typ =sFehlstr) then exit;
  for i:=1 to 3 do begin Knoten[i].X:=unendlich; Knoten[i].Y:=unendlich; end;
  for i:=1 to 2 do begin KnNr[i]:=0;
    if i=1 then begin X:=Strasse.StartX; Y:=Strasse.StartY; end
           else begin X:=Strasse.EndeX;  Y:=Strasse.EndeY;  end;
    for KnNr[3]:=1 to getAnzKnoten do begin
      GetKnDisk(KnNr[3],Knoten[3]);
      if Abst(Knoten[3].X,Knoten[3].Y,X,Y) <epsKreuz then begin
        if (KnNr[i] <>0) then begin
          if not bKurzStr then begin
            AusgStr:='Kante ';   Str(Id.Name,s);    AusgStr:=AusgStr+s;
            AusgStr:=AusgStr+' ist Knoten '; Str(KnNr[3],s);
            AusgStr:=AusgStr+s;
            AusgStr:=AusgStr+' und Knoten '; Str(KnNr[i],s);
            AusgStr:=AusgStr+s;
            AusgStr:=AusgStr+' zugeordnet!';
            FehlerAusg(AusgStr);
          end else begin
            if i=1 then begin
              if Abst(Knoten[3].X,Knoten[3].Y,X,Y) <
                 Abst(Knoten[1].X,Knoten[1].Y,X,Y) then begin
                 KnNr[1]:=KnNr[3]; Knoten[1]:=Knoten[3]; end
            end else begin
              if Abst(Knoten[1].X,Knoten[1].Y,X,Y) <
                 Abst(Knoten[2].X,Knoten[2].Y,X,Y) then begin
```

```
                KnNr[1]:=KnNr[2]; Knoten[1]:=Knoten[2];
              KnNr[2]:=KnNr[3]; Knoten[2]:=Knoten[3]; end;
            end;
          end;
        end else begin
          KnNr[i]:=KnNr[3];
          Knoten[i]:=Knoten[3];
        end;
      end;
    end;
  end;
  if IsStartKn(KnNr[1],Strasse.Id) then begin KnNr1:=KnNr[1];
                                               KnNr2:=KnNr[2]; end
                                   else begin KnNr2:=KnNr[1];
                                               KnNr1:=KnNr[2]; end;
end;

function GetStartKnotenNummer(Id:rId):word;
var KnNr1,KnNr2,KaNr:word;
    Strasse:rStrasse;
    Knoten1,Knoten2:rKnoten;
begin
  GetStrasse(Strasse,Id);
  GetEndKnotenNummer(Strasse.Id,KnNr1,KnNr2);
  GetKnDisk(KnNr1,Knoten1); GetKnDisk(KnNr2,Knoten2);
  if Sqr(Knoten1.X-Strasse.StartX)+
     Sqr(Knoten1.Y-Strasse.StartY) <Sqr(epsKreuz) then
    GetStartKnotenNummer:=KnNr1 else
  if Sqr(Knoten2.X-Strasse.StartX)+
     Sqr(Knoten2.Y-Strasse.StartY) <Sqr(epsKreuz) then
    GetStartKnotenNummer:=KnNr2 else FehlerAusg('PF in GetStartKnotenNummer');
end;

procedure MakeFehlerEllipsen;
var KlNr,KnNr,KnNrMin:word;
    Minimum,X,Y,r1      :real;
    i        :byte;
    KnZuOrd:array[1..250,1..2] of Word; (* ehemals MaxKlassen *)
    s:string;
    Strasse:rStrasse;
    Knoten:rKnoten;
    Id:rId;
begin
  SetStatus(' Die Fehlerellipsen werden errechnet ... ');
```

```
  for KlNr:=1 to getAnzStr do for i:=1 to 2 do KnZuOrd[KlNr,i]:=0;
  for KnNr:=1 to GetAnzKnoten do begin
    for KlNr:=1 to GetAnzStr do begin Id:=NIlId; Id.Nummer:=KlNr;
      GetStrasse(Strasse,Id);
      if ((Strasse.Typ<>sFehlStr) (* and (Strasse.pStart<>NIL)*)) then
      begin
        for i:=1 to 2 do begin Minimum:=unendlich;
          if i=1 then begin X:=Strasse.StartX; Y:=Strasse.StartY; end
                 else begin X:=Strasse.EndeX;  Y:=Strasse.EndeY;  end;
          for KnNr:=1 to GetAnzKnoten do begin GetKnDisk(KnNr,Knoten);
            if Abst(Knoten.X,Knoten.Y,X,Y)<epsKreuz then begin
              if (KnZuOrd[KlNr,i]<>0) then begin Str(KlNr,s);
                if not bKurzStr then (* Programm-Fehler *)
                  Fehlerausg('Kante Nr '+s+' ist zwei Knoten zugeordnet')
                else begin
                end;
              end;  (* Summe der Fehlerquadrate *)
              KnZuOrd[KlNr,i]:=KnNr;
              Knoten.EllX:=Knoten.EllX+Sqr(X-Knoten.X);
              Knoten.EllY:=Knoten.EllY+Sqr(Y-Knoten.Y);
              Knoten.EllXY:=Knoten.EllXY+(X-Knoten.X)*(Y-Knoten.Y);
              SetKnDisk(KnNr,Knoten);
            end;
          end;
        end;
      end;
    end;
  end;
  for KlNr:=1 to GetAnzStr do begin Id:=NIlId; Id.Nummer:=KlNr;
    GetStrasse(Strasse,Id);
    if ((Strasse.Typ<>sFehlStr)(* and (Strasse.pStart<>NIL)*)) then begin
      str(KlNr,S);
      if KnZuOrd[KlNr,1] =0 then
        FehlerAusg('Strasse Nr '+s+' hat keinen Startknoten');
      if KnZuOrd[KlNr,2] =0 then
        FehlerAusg('Strasse Nr '+s+' hat keinen Endknoten');
      if (KnZuOrd[KlNr,1] =KnZuOrd[KlNr,2]) and not bKurzStr then
        FehlerAusg('Strasse Nr '+s+' hat den gleichen Start und Endknoten');
    end;
  end;
  RemStatus;
end;

procedure AddKn(KnNr:word; X,Y:real);
```

```
var Knoten:rKnoten;
begin
  if (KnNr =NoName) then begin
    Knoten.X:= X; Knoten.Y:= Y; Knoten.Anz:= 1;      Knoten.pEnde:=Nil;
    Knoten.EllX:=0; Knoten.EllY:=0; Knoten.EllXY:=0; Knoten.pStart:=NIl;
  end else begin GetKnDisk(KnNr,KNoten);
    Knoten.X:= (Knoten.Anz * Knoten.X + X)/(Knoten.Anz + 1);
    Knoten.Y:= (Knoten.Anz * Knoten.Y + Y)/(Knoten.Anz + 1);
    inc(Knoten.Anz);
  end;
  SetKnDisk(KnNr,Knoten);
end;

procedure MakeKnoten;
const NewKn=0;
var KlNr,KnNr,KnNrMin,KnNrSec,KnNrAlt:word;
    Abstand,Minimum,MinSec,X,Y      :real;
    i:byte; Strasse:rStrasse; Knoten:rKnoten;
begin
  SetStatus(' Es werden die Straenknoten errechnet ... ');
  InitKnoten;
  for KlNr:=1 to getAnzStr do begin
    GetStrDisk(KlNr,Strasse);
    if ((Strasse.Typ<>sFehlStr) or not IsStatDisk(KlNr,statVern)) then
    begin
      KnNrAlt:=0;
      for i:=1 to 2 do begin Minimum:=unendlich; MinSec:=unendlich;
        if i=1 then begin X:=Strasse.StartX; Y:=Strasse.StartY; end
               else begin X:=Strasse.EndeX;  Y:=Strasse.EndeY;  end;
        for KnNr:=1 to getAnzKnoten do begin GetKnDisk(KnNr,Knoten);
(* Suche auch den Knoten mit dem zweitgeringsten Abstand. Nur wenn dieser
   nicht im Kreuzungsbereich ist, wird eine neue Kreuzung erschaffen. *)
          Abstand:=Abst(Knoten.X,Knoten.Y,X,Y);
          if Abstand <MinSec then begin
            if Abstand <Minimum then begin
              MinSec:=Minimum;  Minimum:=Abstand;
              KnNrSec:=KnNrMin; KnNrMin:=KnNr;
            end else begin
              MinSec:=Abstand; KnNrSec:=KnNr;
            end;
          end;
        end; (* for *)
        if (Minimum <epsKreuz) then begin GetKNDisk(KnNrMin,Knoten);
          if (i =1) and (Abst(Knoten.X,Knoten.Y,
```

```
              Strasse.EndeX, Strasse.EndeY) <Minimum) then begin
            AddKn(KnNrMin, Strasse.EndeX, Strasse.EndeY);
            if (MinSec <epsKreuz) then AddKn(KnNrSec, X, Y)
                                  else AddKn(NewKn, X, Y);
            i:=2;
          end else if (i =2) and (Abst(Knoten.X,Knoten.Y,
              Strasse.StartX, Strasse.StartY) <Minimum) then begin
            if (MinSec <epsKreuz) then AddKn(KnNrSec, X, Y)
                                  else AddKn(NewKn, X, Y);
          end else
            AddKn(KnNrMin,X,Y);
        end else
          AddKn(NewKn,X,Y);
      end; (* for i *)
    end;
  end;
  RemStatus;
  MakeFehlerEllipsen;
end;

procedure KanteSpeichern(KnNr1,KnNr2:word; KantenNa:integer; dist:real);
var hp  : pKante; Knoten:rKnoten;
begin GetKnDisk(KnNr1,Knoten);
  if IsVerboten(KantenNa,0,'E') then exit;
  New(hp);
  if Knoten.pEnde=Nil then begin (* Erste Kante *)
    Knoten.pStart:=hp;
    Knoten.pEnde :=hp;
  end else begin
    Knoten.pEndeˆ.P:=hp;
    hp:=Knoten.pEnde;
    Knoten.pEnde:=hpˆ.P;
  end;
  Knoten.pEndeˆ.KnNr:=KnNr2;
  Knoten.pEndeˆ.dist:=dist;
  Knoten.pEndeˆ.KaNr:=KantenNa;
  Knoten.pEndeˆ.P:=Nil;
  SetKNDisk(KnNr1,Knoten);
end;

procedure GraphSpeichern;

  procedure SaveGraph(var DateiNr:text);
  var KnNr:word;
```

```
      S,Erg:String;
      Pt,hp:pKante;
      Knoten:rKNoten;
  begin
    for KnNr:=1 to getAnzKnoten do begin
      if KnNr=13 then
        xx:=xx;
      Str(KnNr,S); GetKnDisk(KnNr,KNoten);
      writeLn(DateiNr,sNeuerKnotenNr+s);
      Pt:=Knoten.pStart;
      while Pt <>Nil do begin
        Str(pt^.KnNr,S);
        Erg:=StrAuff(S,RealLen);
        Str(pt^.KaNr,S);
        Erg:=Erg+StrAuff(S,RealLen);
        S:=NormStr(pt^.Dist);
        Erg:=Erg+StrAuff(S,RealLen);
        writeln(DateiNr,Erg);
        hp:=Pt^.P; Pt:=hp;
      end;
    end;
  end;

var DateiNr:text;
    SchrFil:String;
begin
  SchrFil:=Pfad+fnGraph+getFileCodeNr+spDat;
  if bGraphikAusg then SetStatus('Abspeichern von '+SchrFil)
                  else writeln  ('Abspeichern von '+SchrFil);
  assign(DateiNr,SchrFil); rewrite(DateiNr);
    SaveGraph(DateiNr);
    SaveVerboteAlt(DateiNr);
  Close(DateiNr);
  if bGraphikAusg then RemStatus;
end;

procedure GraphBauen;
var KlNr:integer;
    KnNr1,KnNr2:word;
    Dist:real;
    Pt,hp:pKlElem;
    s,AusgStr :string;
    Strasse:rStrasse;
    Knoten1,KNoten2:rKNoten;
```

```
    Id:rId;
begin
  AusgStr:='  Kante Nr: ';
  for KlNr:=1 to GetAnzStr do begin Id:=NilId; Id.Nummer:=KlNr;
    GetStrasse(Strasse,Id);
    if ((Strasse.Typ<>sFehlStr) and not (IsStat(Id,StatVern))) then begin
      str(KlNr,s);
      if bGraphikAusg then SetTest(AusgStr+s)
                      else begin gotoxy(1,whereY); write(AusgStr+s+'  '); end;
      Pt:=Strasse.pStart;
      KnNr1:=GetKnotenNr(Strasse.StartX,Strasse.StartY,Knoten1);
      KnNr2:=GetKnotenNr(Strasse.EndeX, Strasse.EndeY, Knoten2);
      if (KnNr1 <>KnNr2) or true then begin
        Dist:=Sqrt(Sqr(Knoten1.X-Strasse.StartX)+
                   Sqr(Knoten1.Y-Strasse.StartY))+
              Sqrt(Sqr(Knoten2.X-Strasse.EndeX)+
                   Sqr(Knoten2.Y-Strasse.EndeY));
        while Pt <>Nil do begin
          Dist:=Dist+sqrt(Sqr(Pt^.diffX)+Sqr(Pt^.diffY));
          hp:=Pt^.P; Pt:=hp;
        end;
        if IsStartKn(KnNr1,Strasse.Id) then begin
            KanteSpeichern(KnNr1,KnNr2,-Strasse.Id.Name,Dist);
            KanteSpeichern(KnNr2,KnNr1, Strasse.Id.Name,Dist);
        end else begin
          KanteSpeichern(KnNr1,KnNr2, Strasse.Id.Name,Dist);
          KanteSpeichern(KnNr2,KnNr1,-Strasse.Id.Name,Dist);
        end;
      end;
    end;
  end;
  if bGraphikAusg then RemTest else writeln;
  GraphSpeichern;
end;

begin (*
  An dieser Stelle moechte ich meinen Lehrern danken, die mich bis zur
  Promotion begleitet haben und welchen ich viel verdanke. Speziell sind dies:
  Claus Hewig (JKG), Dr. Andreas Pechtl (Vorstudium),
  Dr. Axel Pelster (Hauptstudium), Prof. Herman Schaal (Diplomvater)
  und Juergen Michelfelder (Programmierung). *)
end.
```

Lebenslauf

von Wolfgang Schmid (geb. Knapp)

14.05.1967	Geboren in Sindelfingen
1973 - 1977	Grundschule Malmsheim
1977 - 1986	Gymnasium Weil der Stadt
1986 - 1993	Studium der Mathematik (HF) und Informatik (NF) an der Universität Stuttgart Meine Studienschwerpunkte waren: Topologie und Geometrie Betreuer meiner Diplomarbeit: Dr. Hermann Schaal Thema meiner Diplomarbeit: Geometrie der Landkartenentwürfe mit computergraphischer Darstellung
1994	wissenschaftliche Mitarbeit beim Institut für Photogrammetrie im Bereich Verkehrsmanagement
1995-1997	Stipendiat der Stiftung Besinnung und Ordnung