Entwicklung eines Graphenmodells als Grundlage für eine Routingapplikation für den motorisierten Individualverkehr basierend auf der GIP

David GEROE, Gerhard NAVRATIL und Hans FIBY

Dieser Beitrag wurde nach Begutachtung durch das Programmkomitee als „reviewed paper" angenommen.

Zusammenfassung

Dieser Artikel befasst sich mit der Frage, wie ein Graphenmodell aussehen muss, wenn beim Routing vorgegebene Qualitätskriterien einzuhalten sind. Abbiegevorschriften müssen beachtet werden, zudem sollen Einbahnen berücksichtigt und U-Turns auf freier Strecke vermieden werden.

Diese Untersuchung erfolgte im Rahmen der Entwicklung eines Routers für den motorisierten Individualverkehr (MIV), der direkt auf den Exportdaten der Graphenintegrationsplattform (GIP) läuft. Nach der Problembeschreibung werden zwei bekannte Lösungen auf das vorliegende Problem angewandt, die aber die Erfüllung der Qualitätskriterien nicht ermöglichen. Anschließend wird ein adaptiertes Graphenmodell vorgestellt, welches den Anforderungen entspricht.

1 Einleitung

1.1 Rahmenbedingungen

ITS Vienna Region, ein Projekt im Verkehrsverbund Ostregion (VOR), betreibt seit dem 18.06.2009 die intermodale Verkehrsauskunft AnachB.at, die einerseits einen Überblick über die aktuelle Verkehrslage im motorisierten Individualverkehr (MIV) und andererseits Routing für alle Modi (MIV, Rad, Fuß, ÖV) sowie deren Kombination (Park&Ride, Bike&Ride, Fahrradmitnahme) anbietet (ENGLEDER 2010). Als Datenbasis dient u. a. die Graphenintegrationsplattform (GIP). Diese ist ein laufend aktualisierter Datenpool, in dem die gesamte Verkehrsinfrastruktur der Ostregion enthalten ist.

Die Routingapplikationen von AnachB.at laufen nicht direkt auf der GIP, sondern bedingen einen periodischen Import der GIP-Exportdaten in das GIS-Verwaltungsprogramm DIVA Geo. Sowohl DIVA Geo als auch die Routingapplikationen sind Produkte der Firma MDV. Weitere Routingapplikationen laufen im Floating Car Data System (von Firma AIT entwickelt) und im Verkehrsmodell, welches für die Verkehrslageberechnung notwendig ist und von der Firma PTV entwickelt wurde. Die Datenübernahme aus der GIP in das Verkehrsmodell kann automatisiert erfolgen (EPP 2010).

Strobl, J., Blaschke, T. & Griesebner, G. (Hrsg.) (2011): Angewandte Geoinformatik 2011.

Um Qualitätsmanagement und Tests einfacher und ohne die Einbeziehung anderer Firmen abwickeln zu können, wurde ein eigenes Routingsystem für ITS Vienna Region entwickelt, das keinen Import in ein Fremdsystem benötigt, sondern direkt auf den GIP-Exportdaten basiert (GEROE 2010). Dieses System ermöglicht u. a. das Variieren von Parametern und das Testen von neuen Einflussfaktoren auf das Routing. Es berechnet derzeit ausschließlich Routen für den motorisierten Individualverkehr, wobei nicht adressscharf, sondern zwischen zwei Links geroutet wird.

1.2 Routingqualität

Eine der wesentlichen Anforderungen an das ITS-Routingsystem war, dass die Qualität des Routings mit der des MDV-Routers vergleichbar sein sollte. Diese „Routingqualität“ wurde folgendermaßen definiert:

1. Die empfohlenen Routen müssen der Straßenverkehrsordnung und den Verkehrsvorschriften genügen.
2. Die empfohlenen Routen sollten typisches Fahrverhalten und Ziele des Verkehrsmanagements berücksichtigen.

Aus der ersten Bedingung folgt, dass Abbiegeerlaubnisse, Abbiegeverbote und Einbahnregelungen korrekt berücksichtigt werden müssen. U-Turns abseits von Kreuzungen sind laut §14, STVO, nur unter bestimmten Einschränkungen erlaubt und sollten daher in der Routingapplikation grundsätzlich nicht empfohlen werden (STVO 1960).

Aus der zweiten Bedingung folgt, dass hochrangige Straßen im Routing bevorzugt behandelt werden sollten. Dies kann durch die Anwendung eines geeigneten Kürzesten-Wege-Algorithmus, zum Beispiel eines hierarchischen Routingalgorithmus, realisiert werden. Für die Erfüllung der ersten Bedingung ist das verwendete Graphenmodell von entscheidender Bedeutung. Auf diese Frage fokussiert dieser Artikel. Er beschreibt, welches Graphenmodell für das ITS-Routingsystem verwendet wurde und wie dieses das Erreichen einer hohen Routingqualität unterstützt.

2 IDF Export

Die GIP ist in einer Oracle Datenbank abgebildet, aus der die routingrelevanten Daten durch eine Prozedur namens „IDF Export“ aus verschiedenen GIP-Tabellen aggregiert und in eine einzige Textdatei exportiert werden können, welche alle erforderlichen Tabellen im .csv-Format enthält. IDF bedeutet „Intrest Data Format“ und bezeichnet ein Datenformat, das im Rahmen des bayrischen Forschungsprojektes „INTREST“ zum Austausch von intermodalen Verkehrsinfrastrukturdaten von PTV und MDV entwickelt wurde (PRISMA 2010). Es gibt mehrere Aufrufvarianten des IDF Exports; Für das ITS-Routingsystem wird der „MIV“ Aufruf verwendet, da es nur Routen für den MIV berücksichtigt und deshalb z. B. Daten über Radwege oder Eisenbahnstrecken nicht benötigt. Abbildung 1 gibt einen Überblick über die Formate und deren Anwendung.

Das IDF-File enthält drei routingrelevante Tabellen: Knoten (punktförmige Elemente, z. B. Straßenkreuzungen), Links (linienförmige Elemente mit Anfangs-, und Endknoten) und Turns (Abbiegeerlaubnisse mit Von-, und Nachlink). Sowohl Knoten als auch Links be-

inhalten geometrische Informationen. Knoten haben eine bestimmte Position, Links können zwischen Anfangs-, und Endknoten weitere Zwischenpunkte enthalten. Diese sind in einer eigenen Tabelle gespeichert.

Abbiegeerlaubnisse sind in der GIP positiv formuliert. Das Abbiegen von Link A nach Link B ist verboten, außer in der Turn-Tabelle befindet sich ein Eintrag, der diesen Abbiegevorgang dezidiert erlaubt (PRISMA 2010).

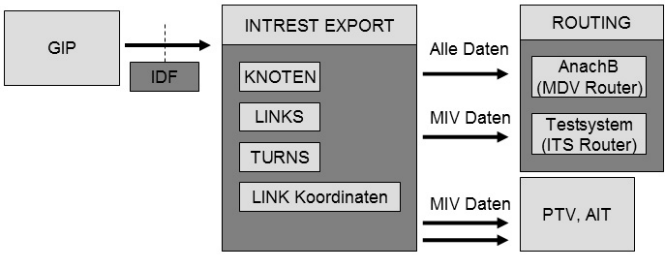

Abb. 1: Datenfluss GIP-IDF-Export

Der IDF-Graph ist ein ungerichteter Graph. Ein Graph ist als ein Paar zweier Mengen (V, E) definiert, wobei V eine endliche Anzahl an Knotenobjekten und E eine endliche Anzahl an Knotenpaaren ist. Ist die Anordnung der Knoten in den Knotenpaaren relevant, so spricht man von einem gerichteten Graphen, andernfalls von einem ungerichteten Graphen. (CORMEN et al. 2001)

Die Knotenpaare in E entsprechen den Links im IDF-File. Obwohl die Links im IDF-Graphen einen Anfangs-, und Endknoten haben, ist der Graph ungerichtet, da die Informationen über die Befahrbarkeit ausschließlich in weiteren Linkattributen liegen.

Da das IDF-File die Abbiegeerlaubnisse in einer eigenen Tabelle enthält, ist es nicht möglich, einen Kürzeste-Wege-Algorithmus auf dem Graphen, der durch die Knoten und Links des IDF-Files beschrieben wird, direkt anzuwenden. Nach dem Verbinden der Knoten, Links und Turns sind also weitere Schritte notwendig, die in weiterer Folge erklärt werden.

Im Folgenden werden zuerst zwei bekannte Lösungen vorgestellt, die den Anforderungen an die erste Bedingung der Routingqualität nicht genügen (Expandieren des Netzwerkes und Erzeugen eines Line-Graphen). Danach wird die Lösung erläutert, die für das ITS-Routingsystem verwendet wurde und das Erreichen der notwendigen Routingqualität ermöglicht (Line-Graph mit Graphknoten).

3 Netzwerk-Expansion und Erzeugen eines Line-Graphen

3.1 Lösung durch Expandieren des Netzwerkes

Eine Möglichkeit, die von ANEZ et al. (1996) und WINTER (2002) beschrieben wurde, ist das Expandieren des Netzwerkes. Hierbei wird eine Kreuzung in mehrere Knoten aufgesplittet und für jede Abbiegeerlaubnis eine eigene Kante eingeführt.

Der große Nachteil dieser Variante ist, dass sich die Anzahl der Knoten und Kanten stark erhöht. Daher wurde diese Lösung nicht verwendet.

3.2 Lösung durch Erzeugen eines Line-Graphen

Das Prinzip des Line-Graphen wurde erstmals von CALDWELL (1961) beschrieben. Seine Anwendung für die Modellierung von Abbiegeerlaubnissen wurde von ANEZ et al. (1996) untersucht, die der Struktur den Namen „Dual Graph“ gaben. WINTER (2002) konzentrierte sich auf das Abbilden von Abbiegeerlaubnissen in gewichteten und gerichteten Graphen durch einen Line-Graph.

WINTER (2002) beschreibt, dass der Line-Graph G’ des Graphen G drei wesentliche Eigenschaften hat:

- Für jede Kante e_i in G existiert ein Knoten $n(e_i)$ in G’. Die Anzahl der Knoten in G’ entspricht der Anzahl der Kanten in G.
- Für jedes Paar aufeinander folgender Kanten $\{e_1, e_2\}$ in G gibt es eine Kante in G’ zwischen den entsprechenden Knoten $n(e_1)$ und $n(e_2)$.
- Es gibt eine Kostenfunktion, die den Kanten und Knoten von G’ Kosten zuordnet.

Für den IDF-Graphen bedeutet das, dass für jeden Link ein Knoten und für jede Abbiegeerlaubnis eine Kante erzeugt werden muss. Da Knoten und Abbiegeerlaubnisse in der GIP keine Kosten zugeordnet haben, verwendet die Kostenfunktion für die Kanten des Line-Graphen die Von-Links der entsprechenden Abbiegeerlaubnisse für die Berechnung der Kosten.

Abbildung 2 zeigt den Originalgraph G und seinen transformierten Line-Graph G’. Der Originalgraph besteht aus 5 Knoten und 5 Kanten, die mit 8 Abbiegeerlaubnissen verbunden sind. Der Line-Graph beinhaltet daher 5 Knoten und 8 Kanten. Die routingrelevanten Informationen aus G bleiben in G’ erhalten.

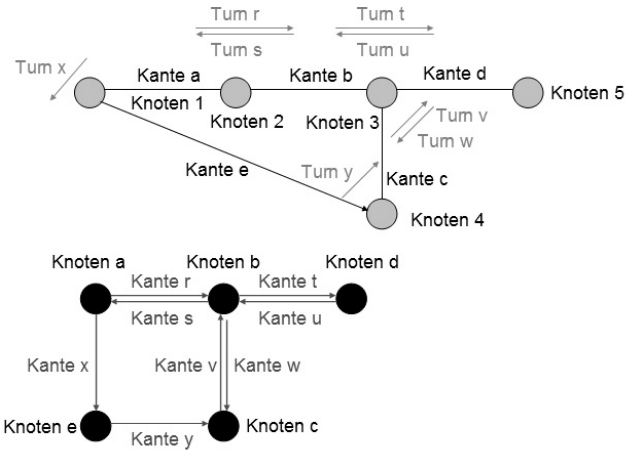

Abb. 2: Graph G (oben) und Line-Graph G’ (unten)

Aus dem IDF-Graphen kann in folgenden drei Schritten ein Line-Graph erzeugt werden:

- Für jede Abbiegeerlaubnis wird eine Kante erzeugt, die auf sie referenziert.
- Für jeden Link wird ein Knoten erzeugt, der auf ihn referenziert.
- Der Graph wird verbunden (Zuordnung von Nachfolgekanten zu den Knoten).

Der resultierende Graph kann ohne weitere Modifizierungen als Input für einen Kürzeste-Wege-Algorithmus verwendet werden, da er nur mehr aus Knoten und Kanten besteht.

3.3 Probleme bei Verwendung eines Line-Graphen

Jedoch ergeben sich dabei zwei Fehlerquellen:

- U-Turns werden nicht korrekt berücksichtigt
- Bestimmte Routen (z. B. bei Linksabbiegeverboten) werden nicht gefunden

3.3.1 U-Turn Problem

Abbildung 3 verdeutlicht das U-Turn Problem.

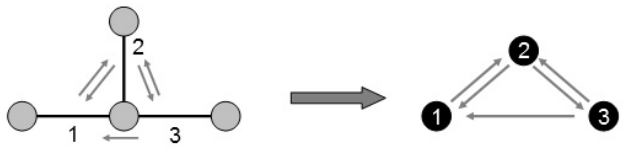

Abb. 3: U-Turn Problem

Im Originalgraph ist der Übergang von Link 1 auf Link 3 nicht erlaubt. Im Line-Graphen ist dies aber über Link 2 möglich. Das entspricht einem U-Turn auf Link 2 und sollte in einer Routingapplikation für den MIV nicht empfohlen werden.

3.3.2 Linksabbiegeverbote

Ein weiteres Problem, das generell beim Routing in Graphen mit Abbiegeverboten auftaucht, wurde von WINTER (2002) beschrieben. Ein Kürzeste-Wege-Algorithmus kann einen Knoten nicht zweimal besuchen. Daher können Linksabbiegeverbote in ungerichteten Graphen grundsätzlich nicht von Kürzeste-Wege-Algorithmen behandelt werden. Die Verwendung eines Line-Graphen löst dieses Problem zwar für gerichtete Graphen (WINTER 2002), nicht aber für ungerichtete Graphen. Abbildung 4 zeigt eine Beispielsituation.

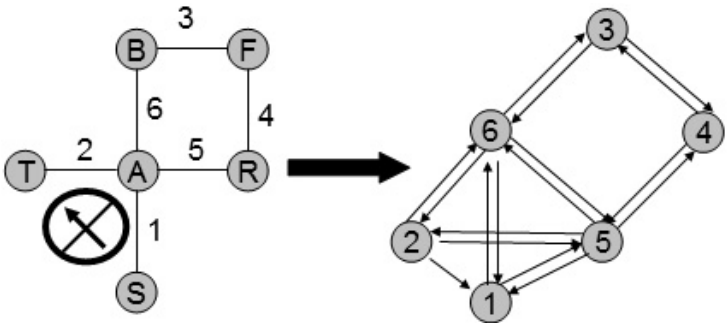

Abb. 4: Linksabbiegeverbot im Original (links) und im Line-Graphen (rechts)

Das Linksabbiegen von Link 1 nach Link 2 ist im Originalgraph verboten. Die korrekte Route ist Link 1 – 6 – 3 – 4 – 5 – 2. Ohne Line-Graph müsste Knoten A zweimal besucht werden, da S – A kürzer ist als S – A – B – F – R – A. Im Line-Graphen ist das Routing zwar möglich, jedoch wird 1 – 5 – 2 oder 1 – 6 – 2 empfohlen, was wiederum ein U-Turn ist.

3.3.3 Erste Lösung

Eine erste Lösung ist, U-Turns während des Kürzeste-Wege-Algorithmus zu detektieren und zu verwerfen. Dies funktioniert im Allgemeinen, scheitert aber bei dem in Abbildung 4 gezeigten Problem; Die Routen 1 – 6 – 2 und 1 – 6 – 5 werden zwar verworfen, dennoch wird die korrekte Route aber nicht gefunden. Die Routen 1 – 5 und 1 – 6 sind kürzer als die Routen 1 – 6 – 3 – 4 – 5 bzw. 1 – 5 – 4 – 3 – 6, daher erkennt der Algorithmus diese Routen nicht. Somit ist diese Lösung nicht brauchbar, und statt des Kürzeste-Wege-Algorithmus muss doch das Graphenmodell weiter modifiziert werden.

4 Lösung durch einen Line-Graphen mit Graphknoten

Ein Line-Graph löst das Problem aus Abbildung 4 für einen gerichteten Graphen, da es dort bei bidirektionalen Relationen für jede Richtung eine eigene Kante gibt, sodass im Line-Graphen zwei Knoten entstehen (WINTER 2002). In Abbildung 4 würde z. B. Link 5 in zwei Line-Graphen Knoten resultieren; Die Verbindung 1 – 5 würde dann nicht am selben Knoten wie die Verbindung 1 – 6 – 3 – 4 – 5 enden. Jedoch gibt es im IDF-Graphen nur einen Link, in dessen Attributen festgelegt ist, in welche Richtungen er befahrbar ist, sodass ein simpler Line-Graph das Problem nicht löst.

Jedoch folgt daraus, dass eine weitere Graphentransformation, in der die Richtungsinformation in das Graphenmodell einfließt, das Problem schließlich löst. Deshalb wird ein neues Objekt eingeführt: Der *Graphknoten*. Ein Graphknoten besteht aus dem Verweis auf einen Knoten und aus einem booleschen Wert, der die Richtung angibt, in die der zugrunde liegende Link befahren wird. Zusätzlich bekommt jeder Knoten ein Attribut „Referenzierende Graphknoten“, eine Liste, die auf seine Graphknoten verweist. Für jeden Knoten im Line-Graphen werden abhängig von der Befahrbarkeit des zugehörigen Links 0, 1 oder 2 Graphknoten erzeugt, die auf den Knoten verweisen. Der Kürzeste-Wege-Algorithmus verwendet

dann zum Routing nicht die Knoten, sondern die Graphknoten. Daher müssen auch alle für den Kürzeste-Wege-Algorithmus benötigten Attribute (z. B. Nachfolger, Gesamtkosten, …) in den Graphknoten gespeichert werden und nicht in den Knoten.

Abbildung 5 zeigt ein Beispiel. Alle Links im Originalgraphen außer Link 4 sind beidseitig befahrbar, Link 4 ist eine Einbahn, die nur gegen Richtung befahren werden kann. Der U-Turn auf Link 2 ist nicht möglich, da es zwischen den Graphknoten (2, true) und (2, false) keine Verbindung gibt. Nur wenn eine Abbiegeerlaubnis von Link 2 auf Link 2 dezidiert definiert würde (etwa falls es am Ende von Link 2 eine übersichtliche Kreuzung gibt, an der ein U-Turn erlaubt ist), gäbe es diese Verbindung. Somit werden Einbahnen, Abbiegeverbote und U-Turns korrekt berücksichtigt.

Die Transformation des GIP-Exports in einen routingfähigen Graphen, der die Qualitätskriterien erfüllt, bedingt also 3 Schritte: Zunächst wird der IDF-Graph erstellt. Dieser wird dann in einen Line-Graphen konvertiert. Abschließend werden Graphknoten hinzugefügt.

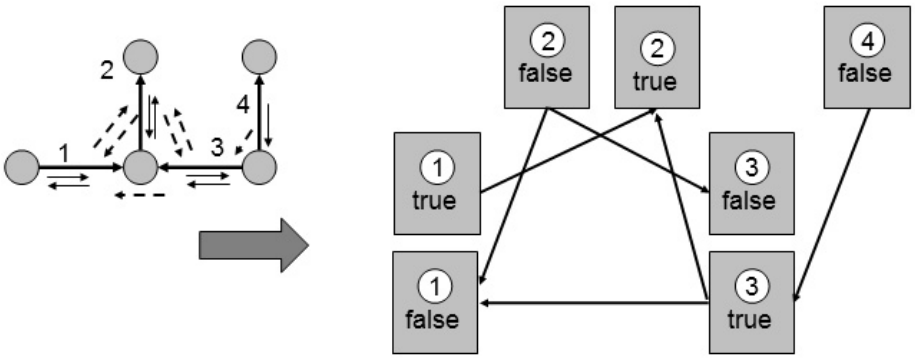

Abb. 5: Graphknoten

5 Adaptierung des Graphen während des Routingalgorithmus

Bei der Verwendung eines Kürzeste-Wege-Algorithmus (z. B. Dijkstra-, oder A* Algorithmus) führt dieses Graphenmodell zu einem Problem. Nach dessen Vorstellung werden zwei mögliche Lösungen vorgestellt.

5.1 Problembeschreibung

Wird die Kürzeste-Wege-Suche von einem bidirektional befahrbaren Link gestartet, ist die Bestimmung des Startgraphknotens problematisch. Der entsprechende Knoten ist zwar gegeben, allerdings ist die Richtung unbekannt, in der die Suche gestartet werden soll. Es gibt dafür zwei Lösungen:

- Berechnung von allen möglichen Routen und retournieren der kürzesten
- Verwendung eines virtuellen Startknotens

5.2 Berechnung von multiplen Routen

Diese Lösungsvariante wirkt sich sehr nachteilig auf die Performance aus, was durch Zeitmessungen verifiziert wurde (GEROE 2010). Bei unidirektionalen Algorithmen müssen zwei Routen berechnet werden, bei bidirektionalen sogar vier. Die Definition der Abbruchbedingungen ist sehr komplex. Daher wurde diese Variante verworfen.

5.3 Verwendung eines virtuellen Startknotens

WINTER, (2002) beschreibt ein ähnliches Problem: In einem Line-Graphen ist der kürzeste Weg von Knoten X zu Knoten Y gesucht. Die Wahl des Startknotens im Line-Graphen ist also problematisch, da der zugehörige Link jeder der ausgehenden Kanten von Knoten X sein kann, was WINTER als „Source Gateway" bzw. „Target Gateway" bezeichnet. Das Problem wird von WINTER mit einem virtuellen Startknoten (Virtual Source Node) gelöst. Hierbei handelt es sich um einen künstlichen Knoten, der mit allen Kanten des Source Gateway via Kanten mit Kosten 0 verbunden ist (WINTER 2002).

Dies entspricht dem hier vorliegenden Problem, sodass auch dieses durch einen virtuellen Startknoten gelöst werden kann. Der virtuelle Startknoten ist ein Graphknoten, der vor der Initialisierung des Kürzeste-Wege-Algorithmus über Kanten mit Kosten 0 mit beiden Kandidaten für den Startgraphknoten verbunden wird. Dieser virtuelle Startknoten wird als Startgraphknoten für den Kürzeste-Wege-Algorithmus verwendet.

Im Gegensatz zu dem Problem von WINTER ist bei unidirektionalen Algorithmen kein virtueller Zielknoten notwendig, da es egal ist, von welcher Seite das Ziel erreicht wird. Wird ein bidirektionaler Algorithmus verwendet, muss allerdings ein virtueller Zielknoten definiert werden, da die Suche dann von Start und Ziel gleichzeitig startet.

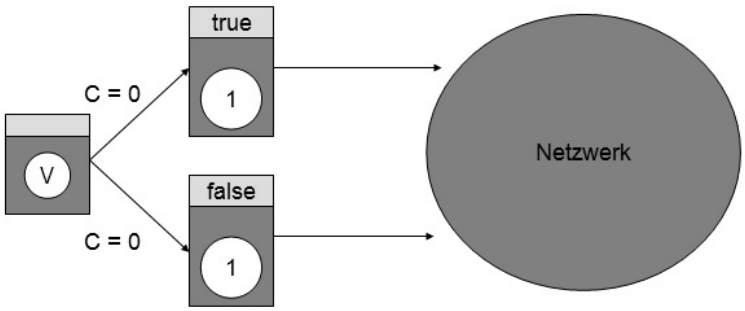

Abb. 6: Virtueller Startknoten

6 Testergebnisse

Basierend auf dem vorgestellten Graphenmodel wurden mehrere Kürzeste-Wege-Algorithmen umgesetzt. Die Implementierung erfolgte in C# (Visual Studio 2005), die

Tests wurden auf einem Laptop mit Windows Vista, 2 Prozessoren (je 2 GHz) und 3 GB RAM ausgeführt. Als bevorzugter Algorithmus wurde ein bidirektionaler A* Algorithmus mit Fibonacci Heaps ausgewählt (GEROE 2010).

Um die Anforderungen an die Routingqualität zu erfüllen, wurde der Algorithmus zusätzlich um eine hierarchische Komponente erweitert. Dabei wurde zwischen strikten Hierarchieregeln einerseits und einer parametergesteuerten hierarchischen Beeinflussung der Kostenfunktion abhängig von der „Functional Road Class" (FRC) andererseits unterschieden. Eine beispielhafte strikte Hierarchieregel lautet: „Wird eine hochrangige Straße (FRC 2,3,4) erreicht, dann darf nur mehr in eine Straße der FRC 0-4 abgebogen werden; Wird eine hochprioritäre Straße (FRC 0,1) erreicht, dann darf nur mehr in eine Straße der FRC 0-1 abgebogen werden" (GEROE 2010, S. 56). Diese Einschränkung („Erweiterte AB Hierarchie") führt aber nur unter drei Bedingungen zu korrekten Routingergebnissen:

- Während der Suche müssen die Vorwärts-, und die Rückwärtssuche immer auf derselben Hierarchieebene suchen. (Anforderung an den Algorithmus)
- Das FRC 0-4 Netz muss einen durchgängigen verbundenen Subgraphen des Netzgraphen bilden. (Anforderung an das GIP Netz)
- Es dürfen keine Echtzeitverkehrsmeldungen, die einen Link sperren können, verwendet werden, da das FRC 0-4 Netz sonst nicht mehr verbunden ist. (Einschränkung der Funktionalität)

Eine Verbesserung der Routingqualität lässt sich aber genauso gut mit parametergesteuerter Hierarchie erreichen, jedoch sind deren Auswirkungen auf die Performance geringer. Da diese drei wesentlichen Einschränkungen aber bei parametergesteuerter Hierarchie entfallen können, wird diese bevorzugt verwendet (GEROE 2010).

Anhand von 10 definierten Testrouten wurde die durchschnittliche Routingzeit über alle Testrouten ermittelt, wobei für jede Testroute die durchschnittliche Berechnungszeit aus 500 Anfragen gebildet wurde (GEROE 2010).

Mit der strikten Hierarchieregel konnte eine durchschnittliche Routingzeit von 381 ms erreicht werden, ohne diese eine Zeit von 685 ms. Kürzere Routen z. B. innerhalb Wiens werden in beiden Fällen in wenigen Millisekunden berechnet. Die Performance ist für ein Testsystem jedenfalls ausreichend, bei Verwendung von teurerer Hardware würden sich zudem sicherlich kürzere Routingzeiten ergeben (GEROE 2010).

7 Fazit

Das für MIV Routing basierend auf GIP-Exportdaten entwickelte Routingprogramm muss sicherstellen, dass Abbiegeverbote, Einbahnen und U-Turns korrekt im Routing berücksichtigt werden. Durch die Verwendung eines geeigneten Graphenmodells kann dies erreicht werden, ohne den Kürzeste-Wege-Algorithmus zu verändern.

Dazu wird der IDF-Graph zuerst in einen Line-Graphen transformiert, der aus jedem Link einen Knoten und aus jeder Abbiegeerlaubnis einen Link macht. Dieser Graph wird danach noch einmal einer Transformation unterzogen, bei der Graphknoten erstellt werden, die einerseits aus der Referenz auf einen Knoten und andererseits aus der Information, in welche Richtung der Basislink des Knotens befahren wird, gebildet werden. Damit können

bekannte Probleme, die beim Routing im Line-Graphen eines ungerichteten Graphen entstehen, vermieden werden. Bei der Initialisierung des Kürzeste-Wege-Algorithmus muss ein virtueller Startknoten erzeugt werden, da die Bestimmung des Startgraphknotens aufgrund der fehlenden Startrichtungsinformation nicht möglich ist.

Aufbauend auf diesem Graphen können bekannte Kürzeste-Wege-Algorithmen ohne Modifizierung verwendet werden. Tests zeigen, dass bei Verwendung von geeigneten Algorithmen (z. B. bidirektionaler A* mit Fibonacci Heaps und hierarchischer Komponenten) eine ausreichende Performance (kurze Routen in Wien wenige ms, lange Routen durch die Ostregion 1-2 Sekunden) erreicht werden kann. Außerdem lassen sich die genannten Anforderungen an die Routingqualität damit erfüllen.

Literatur

ANEZ, J., DE LA BARRA, T. & PEREZ, B. (1996): Dual Graph Representation of Transport Networks. Transportation Research, Part B: Methodological, 30 (3), S. 209-216.

CALDWELL, T. (1961): On finding minimum routes in a network with turn penalties. Communications of the ACM, 4 (2), S. 107-108.

CORMEN, T., LEISERSON, C., RIVEST, R. & STEIN, C. (2001): Algorithmen – Eine Einführung. Übersetzung der englischsprachigen Ausgabe „Introduction to Algorithms: Second Edition".

ENGLEDER, B. (2010): ITS Vienna Region, das innovative Verkehrsmanagementprojekt im Osten Österreichs: Erfolge – Herausforderungen – Perspektiven. In: STROBL, J. et al. (Hrsg.): Angewandte Geoinformatik 2010. Wichmann, Berlin/Offenbach, S. 336-341.

EPP, T. (2010): Automatische Datenübernahme aus der Graphenintegrationsplattform (GIP) in das Online-Verkehrsmodell. In: STROBL, J. et al. (Hrsg.): Angewandte Geoinformatik 2010. Wichmann, Berlin/Offenbach, S. 342-348.

GEROE, D. (2010): Development of a routing program for finding the optimal route in motorized individual traffic based on the graph integration platform. Masterarbeit an der FH Technikum Wien, Masterstudiengang Intelligent Transport Systems.

PRISMA SOLUTIONS (2010): INTREST Data Format IDF, Schnittstellendokumentation.

STVO (1960): Bundesgesetz vom 6. Juli 1960, mit dem Vorschriften über die Straßenpolizei erlassen werden (Straßenverkehrsordnung 1960 – StVO. 1960). StF: BGBl. Nr. 159/1960 (NR: GP IX RV 22 AB 240 S. 36. BR: S. 163).

WINTER, S. (2002): Modeling Costs of Turns in Route Planning. GeoInformatica, 6 (4), S. 345-361.