Nutzung von Echtzeit-Verkehrsdaten für die Flottensteuerung

Günter KIECHLE und Senthanal Sirpi MANOHAR

1 Einleitung und verwandte Arbeiten

Durch die technischen Entwicklungen in den Bereichen Geografische Informationssysteme, Verkehrssensorik und Informationstechnologie wurden in den letzten Jahren Reise-Informationsdienste geschaffen, die eine Berechnung von Reisezeiten im Individualverkehr in Echtzeit und unter Berücksichtigung der aktuellen Verkehrslage ermöglichen. Als Grundlage für die Bestimmung der aktuellen Verkehrslage werden beispielsweise Floating-Car-Daten (FCD) verwendet, die von mehreren, in das aktuelle Verkehrsgeschehen involvierten Fahrzeugen bereitgestellt werden. Diese Fahrzeuge sind mit einem Ortungssystem (z.B. GPS-Empfänger) und einer Funkdatenübertragungsanlage (z.B. GSM) ausgestattet und senden in regelmäßigen Abständen ihre aktuelle Position an eine zentrale Stelle. Aus den Bewegungsprofilen der einzelnen Fahrzeuge wird dann die Gesamtverkehrssituation abgeleitet. Auf diese Weise können aktuelle Reisezeiten für einzelne Straßenabschnitte ermittelt werden (MATTFELD et al. 2008).

FCD-Systeme werden häufig mit Hilfe von Taxi-Flotten betrieben und dazu verwendet, Verkehrslagebilder zu generieren oder Routing-Dienste anzubieten. Darüber hinaus wurden bisher noch wenige weiterführende Anwendungsmöglichkeiten entwickelt.

Viele Unternehmen sind mit der Aufgabe konfrontiert, zur Versorgung ihrer Kunden mit Dienstleistungen eine Flotte von Fahrzeugen zu disponieren. Neben den klassischen Logistikdienstleistern wie Kurier-, Express- und Paketdienste betrifft dies auch Unternehmen, deren Serviceleistung am Standort des Kunden erbracht wird. Dazu gehören beispielsweise Reparatur- oder Servicetechnikerdienste, Personentransportunternehmen, Einsatzorganisationen oder Zustelldienste für Speisen.

Viele dieser Aufgabenstellungen aus Flottensteuerung und Tourenplanung können mit Hilfe von Optimierungsalgorithmen gelöst werden, die als relevante Inputdaten unter anderem Reisezeitinformationen benötigen. Im allgemeinen werden dazu Distanzmatrizen mit Reisezeitwerten zwischen allen beteiligten Standorten verwendet (KIECHLE 2007).

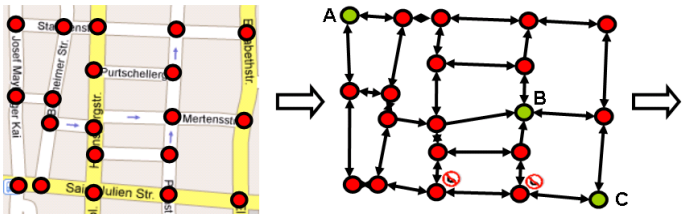
2 Problemszenario

Für die Berechnung von Distanzen in Straßennetzen werden Kürzeste-Wege-Verfahren eingesetzt. Die entsprechende Aufgabenstellung lässt sich sich als graphentheoretisches Problem formulieren. Dazu das Straßennetzwerk als Graph G = (A, N) definiert. Wobei A die Menge der Kanten (Wege) und N die Menge der Knoten (Kreuzungen) ist. Die Kanten des Netzwerkes haben ein nicht negatives Kantengewicht t(i, j) welches der Reisezeit zwischen dem Knoten i und dem Knoten j entspricht.

Derzeit verfügbare und in der Praxis eingesetzte Implementierungen von Kürzeste-Wege-Verfahren verwenden zumeist statische Durchfahrtszeitinformationen einzelner Wegesegmente als Datengrundlage. Bei dynamischen Kürzeste-Wege-Problemen werden jedoch zeitabhängige Wegeabschnittsbewertungen verwendet. Somit sind Reisezeiten zwischen zwei Punkten nicht immer gleich (statisch), sondern ändern sich zeitabhängig je nach Verkehrslage. Beispielsweise sind während der Hauptverkehrszeit, zu der viele Fahrzeuge unterwegs sind, die durchschnittlichen Fahrgeschwindigkeiten deutlich geringer als zu anderen Zeiten. Dadurch können sich auch die schnellsten Routen zwischen zwei Punkten über die Zeit verändern.

Als Lösungsalgorithmus zum Finden des schnellsten Weges wurde für die vorliegende Arbeit der 1959 von Edsger Dijkstra entwickelte Algorithmus (Dijkstra Algorithmus) gewählt, der als einer der effizientesten sequentiellen Algorithmen für gerichtete Graphen mit nicht negativen Kantengewichten gilt (ZAHN 1997). Er wurde um die Berücksichtigung von dynamischen Fahrzeiten und um die Berücksichtigung von Abbiegegeboten- und verboten erweitert und in C++ implementiert (MANHOHAR 2009).

Um Optimierungsverfahren in der Flottendisposition und Tourenplanung für dynamische Aufgabenstellungen mit entsprechenden Datengrundlagen zu versorgen, müssen Distanzmatrizen für mehrere Zeitabschnitte eines Planungszeitraumes berechnet werden. In der vorliegenden Analyse wurde ein Zeitraster von 15 Minuten gewählt, da entsprechende Informationen zu Durchfahrtszeiten aus verfügbaren FCD-Daten in diesem Intervall zur Verfügung standen. Ergebnis des Kürzeste-Wege-Verfahrens ist pro Intervall je eine Matrix der Distanzen zwischen allen relevanten Standorten der Planungsaufgabe.



Orte	A	B	C
A		72	55
B	72		38
C	55	34	

Abb. 1: Vom Straßennetz bis zur Distanzmatrix

Bei der Berechnung von Distanzmatrizen für Aufgabenstellungen in der Tourenplanung wird wie folgt vorgegangen:

1. Ausgangspunkt der Berechnungen ist ein digitales Straßennetz des Zielgebietes, das in einen Graph (Knoten-Kanten-Modell) überführt wird. In diesem Schritt sind insbesondere Einschränkungen wie Einbahnregelungen und Abbiegevorschriften zu berücksichtigen.
2. Im zweiten Schritt werden die für die Tourenplanung relevanten Standorte in einem digitalen Straßennetz verortet. Im Beispiel in Abbildung 1 sind dies die Knoten A, B und C.
3. Im dritten Schritt werden die kürzesten Wege von jedem Standort zu jedem anderen Standort berechnet und in einer Matrix abgelegt. Die Matrix ist nicht zwingend symmetrisch – aufgrund von Einbahnen oder Abbiegevorschriften können Reisezeiten zwischen den Fahrtrichtungen unterschiedlich sein.

3 Kürzeste Wege bei dynamischen Reisezeiten

Verfahren für die effiziente Berechnung von kürzesten Wegen in Graphen sind seit vielen Jahren ein bevorzugtes Thema für wissenschaftliche Untersuchungen (vgl. ZAHN 1997). Kürzeste-Wege-Verfahren, die veränderliche Kantengewichte berücksichtigen und gleichzeitig für die Berechnung von Distanzmatrizen eingesetzt werden, wurden nach bestem Wissen der Autoren in der Literatur bisher nicht beschrieben.

Bei der Verwendung von FCD-Daten als Input für Graphen mit veränderlichen Kantengewichten wird der Planungszeitraum in Intervalle unterteilt, die der Auflösung der FCD-Daten entsprechen. Im hier vorliegenden Fall haben die Intervalle eine Länge von 15 Minuten. Abbildung 2 zeigt einen Graph, in dem die Kante von u nach v eine zeitabhängige Länge (Gewicht, Durchfahrtszeit) aufweist. Die Veränderung des Kantengewichtes in Abhängigkeit von t ist nichtlinear in Zeitintervallen aufgetragen.

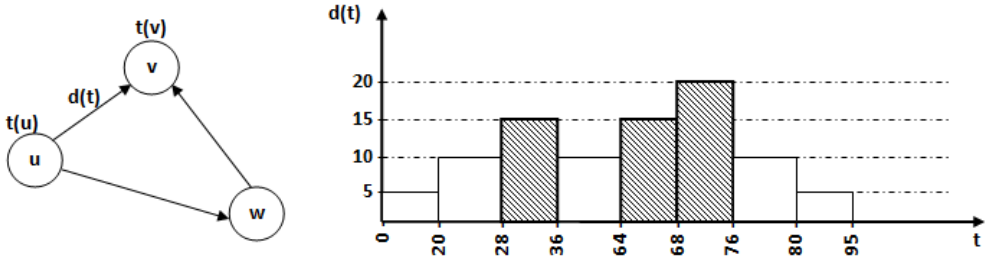


Abb. 2: Graph mit beispielhaftem Verlauf eines dynamischen Kantengewichtes

Zur Berechnung von Distanzmatrizen eignet sich das Verfahren von Dijkstra, das in einem Durchlauf die kürzesten Wege von einem Ausgangsknoten im Graph zu allen anderen bzw. zu einer Teilmenge von anderen Knoten berechnen kann. Um mit dynamischen Kantengewichten umgehen zu können, muss das Verfahren unwesentlich erweitert werden, sodaß es ausgehend vom Ausgangsknoten den Zeitverlauf beim Traversieren des Graphs verarbeiten kann. Der Zeitbedarf für das Traversieren einer Kante wird dann abhängig vom Zeitpunkt bestimmt, zu dem deren Ausgangsknoten vom Verfahren erreicht wurde. Eine Beschreibung des Verfahrens von Dijstra für dynamische Kantengewichte ist nachfolgend als Pseude-Code angeführt, wobei die Veränderungen zur statischen Variante kursiv hervorgehoben sind.[1]

```
// Initialization
for each vertex v in the graph
        arrivalTime[v] := infinity // in seconds
        prev[v] := undefined // prev node
arrivalTime [source] := departure time in seconds at source node
Q := the set of all nodes in the graph

// Computation of shortest paths
```

[1] vgl. http://de.wikipedia.org/wiki/Dijkstra-Algorithmus

```
while Q is not empty
        // find minimum distance (findMin operation)
        u := node in Q with smallest travel time
        // deleting minimum node from the list (deleteMin operation)
        remove u from Q

       // calculate time interval(each interval lasts 15 minutes)
        intervalShift = arrivalTime[u] modulo (15 * 60)
        t = (arrivalTime[u] / (15 * 60)) + intervalShift

        // Relaxation of edges
        for each neighbor v of u
                // decrease the value (decreaseKey operation)
                update := arrivalTime[u] + weight(u, v, t)
                // weight() gives travel time going from u to v in interval t
                if update < arrivalTime[v]
                        arrivalTime[v] := update
                        prev[v] := u
```

4 Berücksichtigung von Abbiegevorschriften

Der klassische Algorithmus von Dijkstra ist nicht in der Lage, Abbiegevorschriften zu berücksichtigen, was sich durch das Beispiel in Abbildung 3 veranschaulichen lässt.

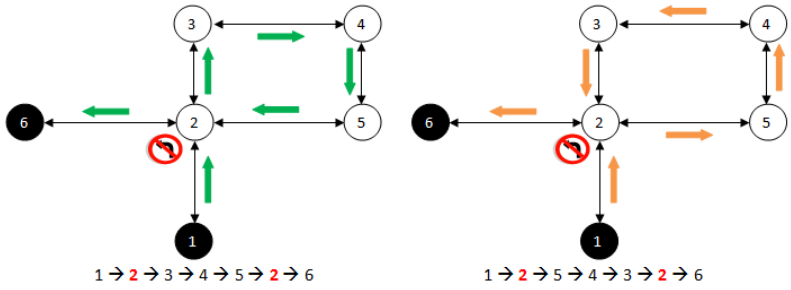


Abb. 3: Kantenabfolge bei einem Abbiegeverbot

Hier wird der kürzeste Weg vom Knoten 1 zum Knoten 6 gesucht. Der Weg 1 ➔ 2 ➔ 6 ist aufgrund des Schildes „Linksabbiegen verboten“ bei Knoten 2 nicht erlaubt, es muss also nach einer weiteren Möglichkeit gesucht werden. Es sind zwei mögliche Lösungswege, nämlich (1➔***2***➔3➔4➔5➔***2***➔6 und 1➔***2***➔5➔4➔3➔***2***➔6) dargestellt. Wie man erkennen kann, wird der Knoten 2 in beiden Lösungen zweimal besucht, was aber im klassischen Algorithmus von Dijkstra nicht möglich ist.

Das Problem kann mit dem Verfahren des Knotensplittings (Split node mechanism) gelöst werden, das nachfolgend erläutert wird. Dieser Lösungsweg hat den Vorteil, dass durch seine Anwendung der Ablauf des Dijkstra Algorithmus nicht verändert werden muss.

Bei der Methode des Knotensplittings wird die bestehende Darstellung des realen Straßennetzes derart ergänzt, dass bei jenen Knoten (Kreuzungen), bei denen eine Abbiegevorschrift vorliegt, ein weiterer, virtueller Knoten in das Netzwerk eingefügt wird. Sollten bei einem Knoten mehrere Abbiegevorschriften vorliegen, werden so viele zusätzliche Knoten eingefügt, wie Vorschriften gegeben sind (siehe Abb. 4).

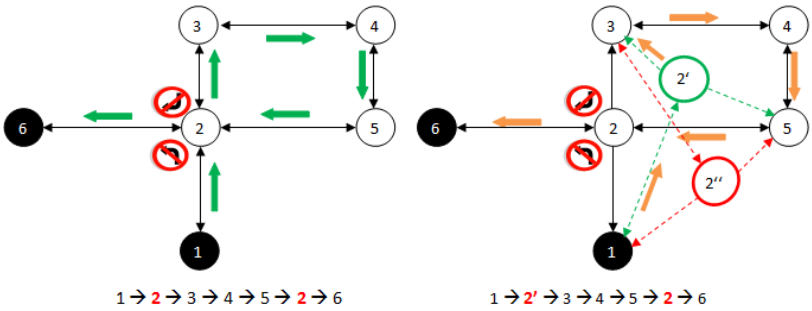


Abb. 4: Knotensplitting bei Abbiegevorschriften

Folgende Änderungen werden für das Linksabbiegeverbot bei Knoten 2 von Knoten 1 aus kommend vorgenommen: Es wird ein zweiter Knoten 2' eingeführt. Der neue (virtuelle) Knoten ist mit dem Knoten 1 über eine in beiden Richtungen zu befahrende Straße verbunden. Die Knoten 3 und 5 sind nur vom Knoten 2' aus erreichbar, nicht jedoch umgekehrt. Die Knoten 2 und 2' dürfen nicht miteinander verbunden sein. Zusätzlich ist der Knoten 2 vom Knoten 1 aus nicht mehr erreichbar (Einbahn von Knoten 2 zu Knoten 1).

Der vom Verfahren nach Dijkstra gefundene Weg vom Knoten 1 zum Knoten 6 lautet nun: 1➔*2'*➔3➔4➔5➔*2*➔6 oder 1➔*2'*➔5➔4➔3➔*2*➔6 (nicht in der Abbildung 4: Knotensplitting bei Abbiegevorschriften.

dargestellt). Man kann erkennen, dass durch den zusätzlichen Knoten 2' kein Knoten zweimal besucht wird. Das Einfügen der neuen Knoten muss nur einmal beim ersten Einlesen des Straßennetzes erfolgen. Der Speicherbedarf dieser Darstellungsform erhöht sich durch die zusätzlichen Knoten – ebenso erhöht sich die Laufzeit des Berechnungsverfahrens, da sich die Gesamtanzahl der Knoten im Netzwerk erhöht.

5 Performanceanalyse

Auf Basis der beschriebenen Implementierung wurden verschiedene Szenarien in Hinblick auf Anzahl der Intervalle und Anzahl der Standorte evaluiert und verglichen. Als Datengrundlage wurde ein digitales Straßennetz der Stadt Wien mit ca. 70.000 Kanten herangezogen und mit dynamischen Durchfahrtszeiten generiert aus FCD Daten erweitert. Um die Laufzeiten des Algorithmus für die Berechnung einer typischen Distanzmatrix zu erhalten, wurden zufällige Standorte ausgewählt. Die Anzahl der Standorte und die Anzahl der Intervalle, für die je eine Matrix errechnet wurde, wurden schrittweise erhöht.

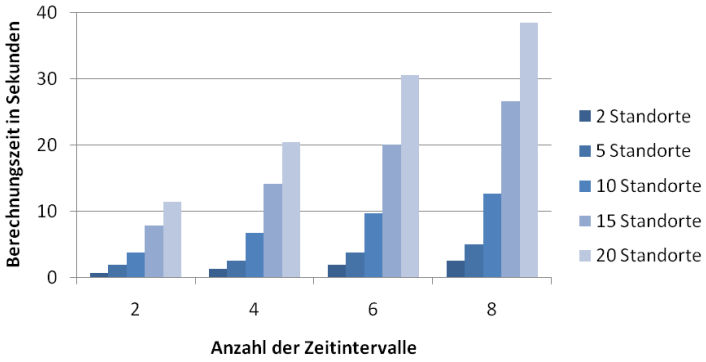


Abb. 5: Berechnungszeiten für Distanzmatrizen von 2-20 Standorten und 2-8 Intervallen

Aus den Ergebnissen ist ersichtlich, dass die Berechnungszeiten mit Anzahl der Zeitintervalle linear ansteigen, ebenso ist bei Erhöhung der Standorte mit einem annähernd linear ansteigenden Zeitbedarf zu rechnen. Um die Berechnungszeiten zu verkürzen, ist eine parallele Berechnung für die voneinander unabhängigen Zeitintervalle möglich.

Literatur

KIECHLE, G. (2007): Using GIS for Optimisation in Transportation Planning. In: ERCIM-News, 68, Jan. 2007, S. 39-40.

MANOHAR, S. (2009): Performance Evaluation of Dijkstra's Fastest Path Algorithm on Large Networks with Dynamic Travel Time. Master Thesis, Carinthia University of Applied Sciences.

MATTFELD, D., EHMKE, J. & MEISEL, S. (2008): Providing travel times for reducing uncertainty in city logistics planning. Präsentation auf der International Conference on Operations Research 2008, Augsburg.

ZAHN, F. (1997): Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures. In: Journal of Geographic Information and Decision Analysis, 1, S. 69-82.