Institut für Informatik
der Technischen Universität München

Lehrstuhl für Informatik mit Schwerpunkt
Wissenschaftliches Rechnen

Multimodale parallele Simulation des Verkehrsflusses in großen Netzen

Michael Moltenbrey

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des Akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. Dr.h.c. Manfred Broy
Prüfer der Dissertation:	1. Univ.-Prof. Dr. Hans-Joachim Bungartz
	2. Univ.-Prof. Dr. Hans Michael Gerndt
	3. Univ.-Prof. Dr. Markus Friedrich, Universität Stuttgart

Die Dissertation wurde am 4. Februar 2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 17. Juni 2009 angenommen.

Danksagung

Die vorliegende Arbeit ist während meiner Tätigkeit am Institut für Parallele und Verteilte Systeme (IPVS) der Universität Stuttgart und dem Lehrstuhl für Informatik mit Schwerpunkt Wissenschaftliches Rechnen der Technischen Universität München in den Jahren 2004 bis 2009 entstanden.

Mein besonderer und aufrichtiger Dank gilt Herrn Prof. Dr. Hans-Joachim Bungartz für die Überlassung des herausfordernden und spannenden Forschungsthemas, für die Betreuung, die fachlichen Anregungen und die überaus freundschaftliche Atmosphäre, die am Lehrstuhl sowohl in Stuttgart als auch nach dem Umzug nach München herrschte. Herrn Prof. Dr. Michael Gerndt und Prof. Dr.-Ing. Markus Friedrich danke ich für das Interesse an meiner Arbeit und für die Übernahme der Zweitgutachten.

Ich fühle mich auch meinen Kollegen Dr. Michael Bader, Martin Buchholz, Dr. Miriam Mehl, Philipp Neumann Dirk Pflüger sowie Stefanie Schraufstetter für ihre zahlreichen kritischen und konstruktiven Anmerkungen beim Lesen der Arbeit zu Dank verpflichtet. Ich bedanke mich auch bei meinem Projektpartnern Gerd Schleupen und Dr. Klaus Nökel sowie der PTV AG für deren fachliche Unterstützung und für die Bereitstellung von Verkehrsdaten.

Ganz besonderer Dank gilt meinem Kollegen Dr. Stefan Zimmer für seine zahlreichen und hilfreichen Hinweise, sowie seine motivierende Unterstützung u.a. bei dem mittäglichen Espresso-Runden. Ebenso danke ich Dr. Ralf-Peter Mundani für seine Anmerkungen und die Einführung in die Welt der italienischen Kaffeekunst.

Weiterhin gilt mein Dank meinen zahlreichen Studenten und Diplomanden.

Michael Moltenbrey

München, im Juni 2009

Zusammenfassung

Die Verkehrssimulation ist ein klassisches und zunehmend wichtiges Anwendungsgebiet für die Simulationspipeline. Immer neue Modelle entstehen und werden nach der durchgeführten Simulation evaluiert und angepasst. Auf Grund der wachsenden Komplexität von Modellen und Simulationsszenarien (große Verkehrsnetze, detailliertes Verhalten der Verkehrsteilnehmer, multimodale Simulationen und Multilevelansätze) sind effiziente Verfahren (z.B. parallel oder hierarchisch) auf allen Stufen der Simulationspipeline unabdingbar.

In dieser Arbeit wird ein Framework zur Simulation des multimodalen Straßenverkehrs (Individualverkehr (IV) und öffentlicher Verkehr (ÖV)) vorgestellt, welches basierend auf unterschiedlichen Verkehrsmodellen (ein Zellularautomatennmodell im mikroskopischen Bereich und Umlegungsverfahren im makroskopischen) und ausgehend von gleichen Datenbeständen (Nachfragedaten in Form von Quelle-Ziel-Matrizen) eine Bearbeitung auf Parallelrechnern erlaubt.

Dazu werden Algorithmen der Verkehrssimulation für den IV und den ÖV analysiert und geeignete Parallelisierungsstrategien entwickelt. Unter anderem werden Partitionierungen der Verkehrsnetze (Multilevel-Partitionierungen, raumfüllende Kurven, ...) sowie effiziente Routenfindungsstrategien (hierarchische, parallele Kurzwegsuchen, ...), betrachtet und eingesetzt.

Mit Hilfe des entwickelten Frameworks ist es möglich, Simulationen sehr großer Verkehrsszenarien auf einer einheitlichen Datenbasis (d.h. mikroskopisch und makroskopisch) durchzuführen.

Inhaltsverzeichnis

1 Einleitung

Verkehr in jedweder Ausprägung spielt in unserem täglichen Leben eine immer größere Rolle. So werden weltweit jeden Tag Güter transportiert und Menschen pendeln zwischen Wohnung und Arbeitsplätzen oder reisen. Große Teile der Globalwirtschaft hängen heute von den zahlreichen Transportmöglichkeiten ab. Bereits Mitte des 19. Jahrhunderts wurde der Ausbau des europäischen Schienennetzes forciert. 2006 wurden allein in Deutschland 1,854 Milliarden Personen von der Deutschen Bahn transportiert und die Bahn legte 74,788 Millionen Personenkilometer zurück [2]. Seit der Mitte des 20. Jahrhunderts spielt allerdings der Straßenverkehr die dominierende Rolle.

Alleine in den letzten 15 Jahren ist das Verkehrsaufkommen in Deutschland um etwa 15% gestiegen [1]. Abb. 1.1 zeigt die Entwicklung des Fahrzeugbestandes in den Jahren 1970-2006.

Mit dieser Entwicklung einher ging ein kontinuierlicher Ausbau des Straßennetzes (siehe Abb. 1.2). Beiden Entwicklungen sind allerdings durch mehrere Faktoren Grenzen gesetzt. Mit steigendem Verkehrsaufkommen häufen sich Probleme wie Staus, Unfälle und Lärmentwicklung. Auch die Umweltbelastungen durch den Schadstoffausstoß der Fahrzeuge wird immer wichtiger. Die verkehrsbedingten Emissionen von Schadstoffen sind in Deutschland zwar leicht rückläufig, der Anteil des Straßenverkehrs an den CO_2-Emissionen beträgt aber nach wie vor etwa 20% [1]. Die Kosten der Umweltbelastungen werden von der EU auf etwa 1,1% des europ. Bruttoinlandsprodukts (BIP) veranschlagt [3].

Neben Umweltproblemen entstehen aber auch immer größere ökonomische Probleme. So bewegt sich statistisch gesehen der Verkehr in städtischen Gebieten in Deutschland im Schnitt 6,6 Stunden pro Tag unter Staubedingungen [1, 163]. So ärgerlich ein Stau für den individuellen Verkehrsteilnehmer sein mag, so sehr entsteht dadurch auch ein immenser volkswirtschaftlicher Schaden. Alleine innerhalb der EU beträgt dieser Schaden – aufgrund Lagerhaltung auf der Straße und Just-in-Time-Lieferungen – knapp 1% des europäischen BIP [3].

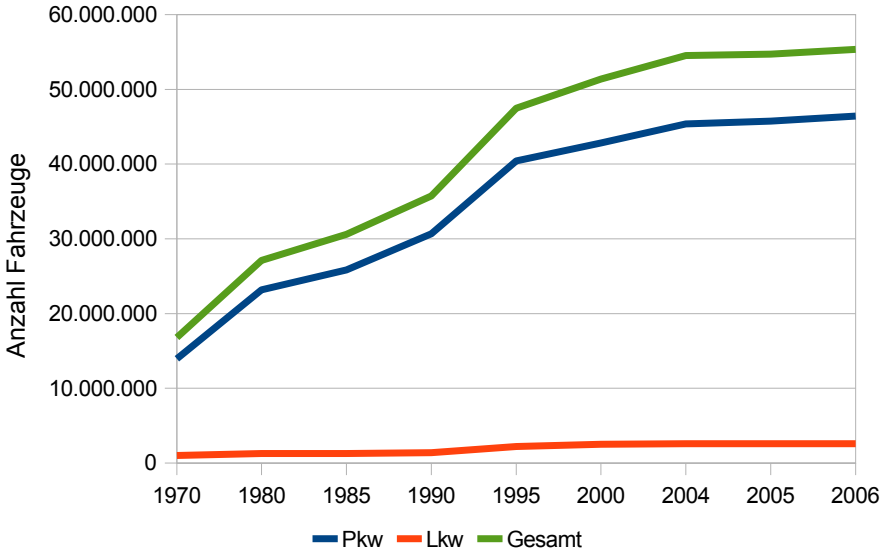


Abbildung 1.1: Entwicklung des Fahrzeugbestands in Deutschland in den Jahren 1970 bis 2006 (nach [1]).

Ein Ende dieser Entwicklung ist trotz steigender Rohstoff- und damit einhergehender wachsender Transportkosten nicht abzusehen. Hinzu kommt, dass Verkehrsnetze mit ihrer Verzahnung unterschiedlicher Verkehrssysteme (Schienen-, Straßen-, Wasser- und Luftverkehr) schon heute sehr komplex sind und selbst bei einfachen Maßnahmen, wie dem Bau einer neuen Straße, die Folgen schwer abschätzbar oder gar vollkommen entgegen der Intuition sein können. Dies ist beispielsweise anhand des Paradoxon von Braess (siehe Abschnitt 2.1.1) gut zu erkennen, wo der Bau einer neuen Straße entgegen den Erwartungen nicht zu einer Verkehrsentlastung, sondern vielmehr zu einer Verschlechterung der Verkehrssituation führt.

Experimentelle Baumaßnahmen sind zu kostspielig. Daher kommt hier die Simulation des Verkehrsflusses ins Spiel. Sie ist eine kostengünstige Möglichkeit, verkehrsplanerische Maßnahmen schon am Anfang zu unterstützen. Sie erlaubt unter anderem die Evaluation verkehrsplanerischer Maßnahmen, sowie die Identifikation von Verkehrsengpässen und Stauprognosen.

Erste einfache Verkehrssimulationen wurden schon in den 1950er Jahren entwickelt. Doch fehlte es damals an der notwendigen Rechenkapazität um sie sinnvoll zu nutzen. Moderne Rechner stellen heutzutage die notwendige Rechenkapazität preiswert zur Verfügung und neue Verkehrssimulationsmodelle wurden entwickelt, die einen sinnvollen Produktiveinsatz der Verkehrssimulation erlaubten. So ist es möglich, den Autobahnverkehr einer Region mittlerer Größe zu simulieren, z.B. in Nordrhein-Westfalen [95, 113].

Allerdings benötigt man für noch realistischere Ergebnisse oder die Berechnung des Ver-

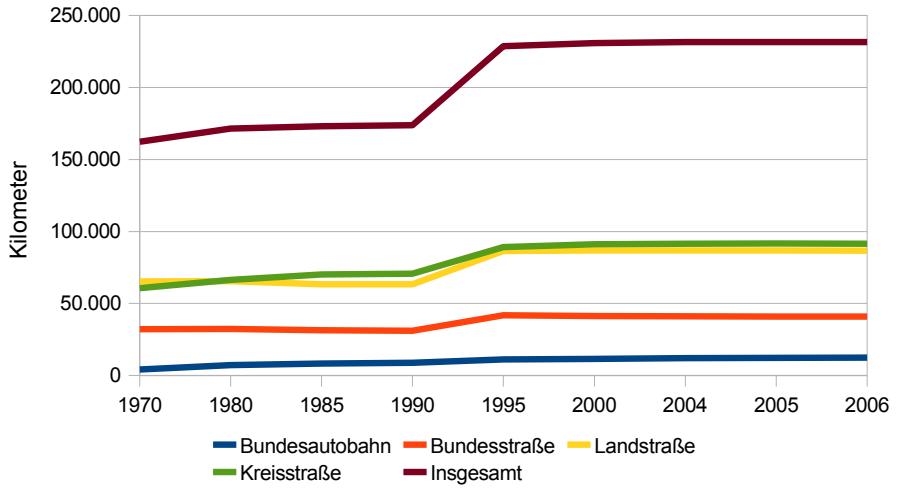


Abbildung 1.2: Ausbau des bundesdeutschen Straßennetzes (ab 1990 inklusive des Straßennetzes des ehemaligen DDR-Gebiets) (nach [1]).

kehrsflusses in größeren und detaillierten Verkehrsnetzen (beispielsweise Straßennetz der Bundesrepublik Deutschland oder der USA) immer noch sehr viel Rechenzeit. Dies erschwert das Arbeiten an entsprechenden Netzen. Zudem stößt man beim Vorhalten von Verkehrsnetzen schnell an Beschränkungen durch den Arbeitsspeicher der Rechner.

Zur Lösung der beschriebenen Probleme bietet sich der Einsatz von parallelen Rechensystemen an. Dafür sind sowohl detaillierte Kenntnisse der Verfahren als auch der zu verwendenden Rechnerarchitekturen, der zu Grunde liegenden Datenstrukturen und Programmierparadigmen essentiell. Nicht zuletzt aus den Erfahrungen des Projekts „Multimodale Simulation des Verkehrsablaufs in großen Netzen“ der Landesstiftung Baden-Württemberg, in dem Teile der kommerziellen Verkehrsplanungssoftware VISUM der PTV AG parallelisiert wurden, wurde deutlich, dass eine nachträgliche Parallelisierung problematisch ist. Daher wurden für eine umfassende Untersuchung der Parallelisierbarkeit zunächst - im Rahmen des Landesstiftungsprojektes - die rechenzeitintensiven Teile neu implementiert und anschließend ein vollständig eigenständiges System entwickelt.

1.1 Ziele dieser Arbeit

Das Kernziel dieser Arbeit ist es, ein Framework zur Verkehrssimulation zu entwickeln, welches die Verwendung unterschiedlicher Verkehrsmodelle (mikroskopisch und makroskopisch) unter Ausnutzung der gleichen Datenbasis erlaubt. Zu den Hauptaufgaben dieser Arbeit gehört es, verschiedene Parallelisierungsstrategien für die verschiedenen Simulationsmodelle zu entwerfen und innerhalb des Frameworks eine einfache Nutzung dieser

Technologien zur Verfügung zu stellen. Die durch die Anwendung der entwickelten Verfahren entstandenen Ergebnisse sollen durch das Framework für eine Weiterverarbeitung vorbereitet werden. Zu den angestrebten Zielen dieser Arbeit gehören:

- Aus der Vielzahl existierender Verkehrsmodelle gilt es eine geeignete Auswahl zu treffen, die beide großen Typen der Verkehrssimulation – mikroskopisch und makroskopische – abdecken. Gegebenenfalls gilt es bestehende Modelle zu erweitern, um komplexere und damit rechenintensivere Szenarien simulieren zu können.
- Es muss festgelegt werden, welche Datenbasis diese Modelle benötigen und wie sich diese gemeinsam durch die Modelle nutzen lässt.
- Für die ausgewählten Modelle gilt es, effiziente Parallelisierungsstrategien zu entwickeln, die sowohl eine Laufzeit- als auch eine Speicheroptimierung ermöglichen. Hierzu zählen sowohl die Betrachtung von Möglichkeiten der verteilten Nachfragenutzung, der Aufteilung der Verkehrsinfrastruktur als auch die Untersuchung von Lastverteilungsverfahren.
- Schließlich sollen die gewonnenen Ergebnisse zur Weiterverarbeitung durch das Framework vorbereitet werden.

1.2 Beiträge dieser Arbeit

Orientiert an der obigen Zielsetzung, konnten im Rahmen dieser Arbeit folgende zentrale Beiträge geleistet werden.

- Es entstand ein Framework zur Verkehrssimulation, welches flexibel erweiterbar ist und sowohl mikroskopische als auch makroskopische Simulationen ausgehend von der gleichen Datenbasis erlaubt.
- Parallelisierungsstrategien für beide Simulationsarten wurden analysiert, entworfen und implementiert. Dabei wurde unter anderem für die mikroskopische Verkehrssimulation ein effizientes Verfahren zur parallelen Berechnung in partitionierten Verkehrsnetzen unter Verwendung einer dynamischen Lastverteilung (unter Berücksichtigung der jeweils aktuellen Auslastung des Verkehrsnetzes) mittels raumfüllender Kurven entwickelt.
- Sowohl für die makroskopische Simulation des motorisierten Individualverkehrs als auch des Öffentlichen Verkehrs wurden Parallelisierungsstrategien für nachrichtengekoppelte Multiprozessorsysteme entwickelt. Diese Art der Parallelisierung ist, insbesondere für den Öffentlichen Verkehr, neu und erlaubt effiziente Berechnungen großer Simulationsszenarien.
- Für den makroskopischen Individualverkehr wurde eine stochastische Umlegung auf zwei verschiedene Arten parallelisiert, die sowohl eine Aufteilung der Nachfrage als auch eine Partitionierung des Verkehrsnetze unterstützen. Auf diese Weise konnte ein

wesentlicher Fortschritt bei der Simulation von sehr großen und detaillierten Szenarien erzielt werden, die bisher in angemessener Zeit oder wegen Speicherproblemen so nicht berechenbar waren. So ließ sich die makroskopische Simulation in beiden Ausprägungen deutlich beschleunigen.

- Es wurden hybride Simulationen entworfen, insbesondere eine Kopplung zwischen einer mikroskopischen Simulation und einer stochastischen Umlegung. Neu an diesem Ansatz ist, dass das Verkehrsnetz beliebig in makroskopische und mikroskopische Teile zerlegt und anschließend gekoppelt werden kann. Der Datenaustausch erfolgt dabei in beide Richtungen, das heißt sowohl von der makroskopischen zur mikroskopischen als auch umgekehrt.

Im Rahmen dieser Arbeit konnte ein zentraler Beitrag bei der Zusammenführung von aktuellen Modellierungstechniken im Bereich der Verkehrssimulation und den für entsprechend große Szenarien erforderlichen Parallelrechentechnologien erzielt werden.

1.3 Gliederung der Arbeit

Kapitel 2 gibt eine kurze Einführung in die grundlegenden Begriffe und Methoden der Verkehrssimulation, die zum Verständnis dieser Arbeit notwendig sind. So werden die Struktur von Verkehrsnetzen, Verkehrsnachfrage und ihr Zusammenspiel mit mikroskopischen und makroskopischen Verkehrsmodellen betrachtet. Das Kapitel schließt mit einem Überblick über bestehende Systeme zur Verkehrssimulation.

Kapitel 3 führt einige wichtige Begriffe zum Thema Parallelisierung ein, die für das weitere Verständnis dieser Arbeit notwendig sind.

Das im Rahmen dieser Arbeit entstandene Framework zur Verkehrssimulation wird in seiner Grundstruktur in **Kapitel 4** eingeführt und motiviert.

Kapitel 5 beschäftigt sich mit der mikroskopischen Simulation, die durch das Framework verwendet wird. Darin wird eine Erweiterung eines grundlegenden mikroskopischen Verkehrsflussmodells um die Behandlung heterogenen und Öffentlichen Verkehrs vorgestellt und Parallelisierungsstrategien dafür aufgezeigt.

Die Betrachtung des makroskopischen Öffentlichen Verkehrs und die Entwicklung einer Parallelisierungsstrategie ist Gegenstand von **Kapitel 6**.

In **Kapitel 7** wird ein bestehendes makroskopisches Verkehrsmodell zur Umlegung des motorisierten Individualverkehrs vorgestellt. Hierzu werden zwei Parallelisierungsstrategien aufgezeigt, die im Rahmen dieser Arbeit entstanden. Eine der Strategien begnügt sich mit der Aufteilung der Verkehrsnachfrage, die andere nutzt Partitionierungen des Verkehrsnetzes, um zusätzlich den Speicherbedarf auf den einzelnen Rechenknoten eines parallelen Rechensystem zu reduzieren.

Kapitel 8 demonstriert die Entwicklung zweier hybrider Simulationsmodelle. Im ersten erfolgt eine Kopplung der in den Kapitel 4 und 6 vorgestellten Verkehrsmodelle. Das zweite erweitert das Framework um die Möglichkeiten einer kombinierten Fußgänger- und Mikrosimulation.

Die Ergebnisse der in den Kapiteln 4 bis 7 vorgestellten Verfahren werden in **Kapitel 9** präsentiert.

Die Arbeit schließt in **Kapitel 10** mit einer Zusammenfassung, der Bewertung der Ergebnisse und gibt einen Ausblick auf noch offene Fragestellungen und Erweiterungsmöglichkeiten des Frameworks.

2 Grundlagen der Verkehrssimulation

Die Möglichkeiten den Verkehr zu simulieren sind seit den ersten Gehversuchen in den 1950er und 1960er Jahren vielfältig geworden. Versuchte man sich in den ersten Jahren an der Übertragung von Verfahren aus anderen Bereichen, wie der Strömungsmechanik, so entwickelte sich der Bereich der Verkehrssimulation bald zu einem eigenen, treibenden Motor für Innovationen.

Es existieren grundsätzlich zwei Arten den Verkehr zu betrachten und zu simulieren. Man unterscheidet die *mikroskopische* von der *makroskopischen Simulation*. Während im ersteren Fall einzelne Verkehrsteilnehmer und deren Interaktion detailliert betrachtet werden, so bestimmen Durchschnittswerte und Flüsse die letztere Simulationsart. Eine dritte, weniger bedeutende Variante, die *mesoskopisch* Simulation ist ein Mittelweg zwischen mikroskopisch und makroskopisch. Geprägt sind diese mesoskopischen Verfahren vor allem durch gaskinetische Modelle. Die mesoskopischen Verfahren werden im weiteren Verlauf nicht betrachtet.

So verschieden diese Simulationsarten auf den ersten Blick auch wirken mögen, so viel haben sie im Kern doch gemeinsam. Beide Varianten untersuchen und beschreiben im Wesentlichen die gleichen Parameter und Charakteristika. Des Weiteren benötigen sie beide als Grundlage für die Simulation ein Verkehrsnetz.

Da es sich nicht um eines der klassischen Gebiete der Informatik handelt, bedarf es einiger einleitender Begriffserklärungen und Einführungen in die verschiedenen Modellaspekte der Verkehrssimulation. Diese sollen im Folgenden in diesem Kapitel diskutiert werden.

Das Kapitel beginnt mit einer allgemeinen Einführung in die Begriffswelt der Verkehrssimulation. Hierbei werden die notwendigen Begriffe und Größen eingeführt und motiviert. Neben der Struktur von Verkehrsnetzen und der Beschreibung von Verkehrsnachfrage, werden verschiedene Arten der Routensuche in Verkehrsnetzen vorgestellt. Abschließend folgt eine Einführung in die großen „Teilwelten“ der Verkehrssimulation: mikroskopische und makroskopische Simulationen.

2.1 Das Verkehrsmodell

Jede Art der Verkehrssimulation – egal welcher Ausprägung, ob mikro-, meso- oder makroskopisch – setzt auf einem gemeinsamen Reservoir aus Grunddaten und -parametern auf, ebenso wie auf einigen grundlegenden Charakteristika. Diese werden in diesem Abschnitt eingeführt.

2.1.1 Das Paradoxon von Braess

Ein grundlegender Denkfehler besteht darin, dass man annimmt, dass es grundsätzlich zu einer Entlastung des Verkehrsaufkommens in einem Straßennetz kommt, wenn man eine zusätzliche Straße zur Entlastung baut (beispielsweise Umgehungsstraßen). Diese Grundannahme erweist sich nicht in allen Fällen als korrekt. Im Jahr 1968 konnte von Dietrich Braess gezeigt werden, dass eine zusätzliche Handlungsalternative (beispielsweise der Bau einer neuen Straße) zu einer Verschlechterung der Anfangssituation führen kann [22]. Dies ist seither als das Braess-Paradoxon bekannt und kann an einem Beispiel veranschaulicht werden.

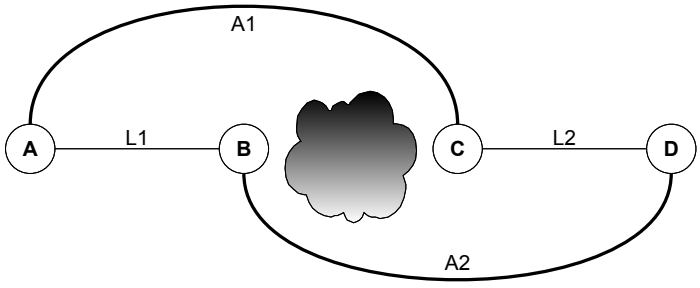


Abbildung 2.1: Ausgangssituation für ein Beispiel zur Beschreibung des Paradoxon von Braess.

Wie Abbildung 2.1 zu entnehmen ist, besteht das Beispielszenario aus vier Städten *A, B, C* und *D*. Die Städte *A* und *C* sind über eine gut ausgebaute Autobahn *A*1 miteinander verbunden. Dasselbe gilt für die Städte *B* und *D*, die über *A*2 miteinander verknüpft sind. Zusätzlich zu den beiden Autobahnen existieren noch zwei weniger gut ausgebaute Landstraßen *L*1 und *L*2. Diese verbinden die Städte *A* und *B* bzw. *C* und *D*. Die beiden Autobahnen müssen allerdings ein Hindernis in Form eines Sees (grau schattiert) umfahren und sind daher länger als die Landstraßen.

Sei x die Verkehrsdichte in Tausend Autos pro Stunde. Dann entspreche die Fahrzeit eines

einzelnen Verkehrsteilnehmers auf den Autobahnen in Minuten:

$$t_{AC}(x) = t_{BD}(x) = 50 + x$$

Da die Landstraßen zwar kürzer als die Autobahnen, aber schlechter ausgebaut sind, hängt die Fahrtdauer stärker vom Verkehrsaufkommen ab. Daher sei die Fahrtdauer auf den Landstraßen gegeben durch:

$$t_{AB}(x) = t_{CD}(x) = 0 + 10 \cdot x$$

In diesem Beispielszenario wollen alle Verkehrsteilnehmer von der Stadt A zur Stadt D fahren und zwar stündlich 6.000 Autos. Jeder Teilnehmer wählt den für sich schnellsten Weg. Dadurch stellt sich ein so genanntes Nash-Gleichgewicht ein, bei dem die Hälfte der Autos über die Stadt B fährt und die andere Hälfte über die Stadt C. Das bedeutet, dass stündlich auf jeder Strecke 3.000 Autos unterwegs sind und die einheitliche Fahrtdauer $(50+3)+10 \cdot 3 = 83$ Minuten beträgt. In einem Verkehrsnetz wird immer versucht einen Gleichgewichtszustand zu erreichen, was in den so genannten Wardrop'schen Prinzipien festgehalten ist. Dies ist in Abb. 2.2 veranschaulicht.

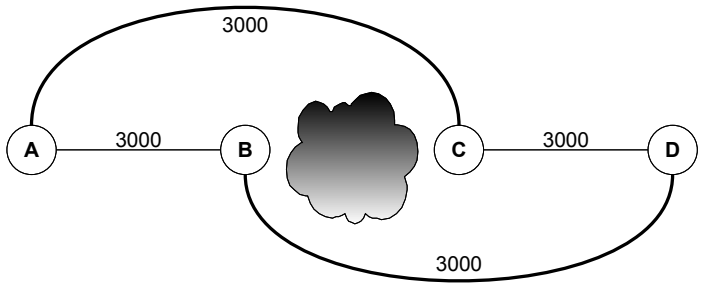


Abbildung 2.2: Belastungen im Beispielszenario aus Abb. 2.1.

Nun wird eine Brücke $B1$ über den See zur Entlastung des Verkehrs gebaut. Diese verbindet die Städte B und C. Es handelt sich dabei um eine Einbahnstraße, die nur eine Befahrung in der Richtung $B \rightarrow C$ zulässt (siehe Abb. 2.3 oben). $B1$ sei eine kurze und gut ausgebaute Straße. Dementsprechend betrage die Fahrtdauer in Minuten:

$$t_{BC}(x) = 10 + x$$

Auch hier stellt sich ein Gleichgewicht ein wie es in Abb. 2.3 unten veranschaulicht ist. Die Fahrtdauer für alle Verkehrsteilnehmer beträgt $(50+2)+10 \cdot 4 = 10 \cdot 4 + (10+2) + 10 \cdot 4 =$ 92 Minuten. Demzufolge führt der Bau einer neuen Straße, der eigentlich eine Verkehrsentlastung bringen sollte, zu einer Verschlechterung der Verkehrssituation und einer Verlängerung Fahrzeit um neun Minuten.

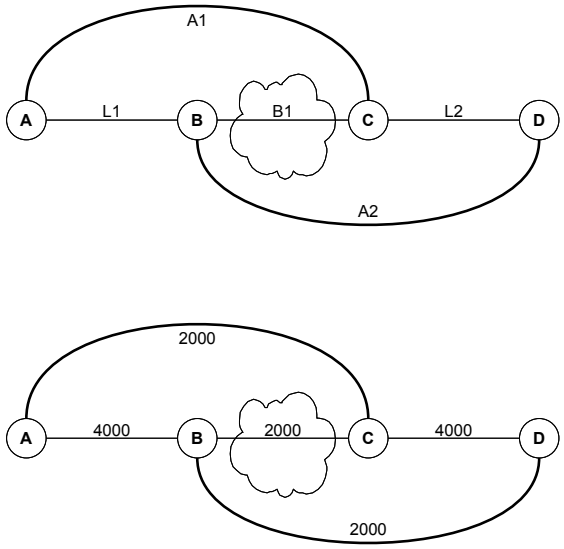


Abbildung 2.3: Erweiterung der Ausgangssituation durch Bau einer Brücke über den See.

Dieser Sachverhalt lässt sich auch gut anschaulich verstehen. Jeder Verkehrsteilnehmer muss zwangsläufig auf seiner Fahrt von *A* nach *D* einen Teil seiner Reise auf einer Landstraße absolvieren. Die Fahrtdauer darauf ist allerdings stark von der Verkehrsbelastung abhängig. Dementsprechend wird die Fahrt auf den Landstraßen zum eigentlichen Engpass, denn durch die zusätzliche Brücke $B1$ werden nun einige Fahrer die Landstraße in voller Länge (*L1–B1–L2*) verwenden. Dadurch kommt es auf den bisherigen Landstraßen $L1$ und $L2$ zu einer erhöhten Verkehrsbelastung, was zu einer Verschlechterung der Gesamtsituation führt.

Dieses Beispiel zeigt, dass nicht alle intuitiven Ideen in der Realität so einfach realisierbar bzw. sinnvoll sind. Dieses soeben vorgestellte Szenario lässt sich noch gut von Hand lösen, begibt man sich aber zu realen Szenarien, so ist die Ausgangssituation ungleich komplexer und differenzierter. Ein manuelles Lösen ist nicht mehr möglich. Die Nutzung von Verkehrssimulationen ist hier notwendig und sinnvoll. Mit deren Hilfe lassen sich schon vor einem teuren Neubau die geplanten Maßnahmen kostengünstig evaluieren.

2.1.2 Begriffe

Für das Verständnis der restlichen Arbeit ist es wichtig, einige grundlegende Begriffe einzuführen und zu motivieren. Sich bewegender Verkehr wird in der Regel unter dem Begriff *Verkehrsstrom* geführt. An Kreuzungen, also *Verkehrsknotenpunkten*, existieren für jede mögliche *Fahrbeziehung*, d.h. Möglichkeit zu fahren, Verkehrsströme. Wie Abb. 2.4 zu entnehmen ist, gibt es in einer normalen Kreuzung mit vier ein- und ausgehenden Straßen insgesamt 12 Ströme. Zusätzlich beschreiben auch die einzelnen Einträge in einer Quelle-Ziel-Matrix (siehe Abschnitt 2.1.5) jeweils einen Verkehrsstrom von einer Quelle zu einem Ziel.

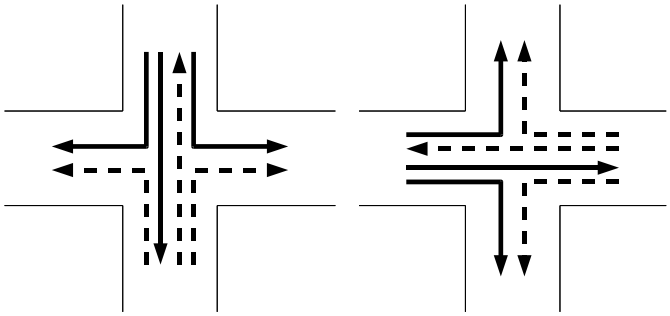


Abbildung 2.4: Verkehrsströme an einer einfachen Vierfachkreuzung.

Fließender Verkehr wird noch durch weitere charakteristische Größen beschrieben. Betrachtet man einen Straßenabschnitt bestimmter Länge, so kann es interessant sein zu wissen, wie viele Fahrzeuge sich auf diesem befinden. Diese Größe wird als *Verkehrsdichte* ρ mit der Einheit Fzg/km bezeichnet. Demgegenüber steht der *Verkehrsfluss* F in Fzg/h, der angibt wie viele Fahrzeuge pro Zeit an einem Kontrollpunkt vorbeifahren. Des Weiteren ist auch die *durchschnittliche Geschwindigkeit* $\overline{v}$ von Interesse.

Das *Fundamentaldiagramm* ist eine sehr wichtige Größe zur Charakterisierung des Verkehrs. Es stellt eine Beziehung zwischen dem Fluss F und der Dichte ρ her: $F = F(\rho)$. Abb. 2.5 zeigt ein durch Messungen auf einer zweistreifigen Autobahn in Deutschland gewonnenes Fundamentaldiagramm.

Eine schematische Darstellung eines allgemeinen Fundamentaldiagramms ist Abb. 2.6 zu entnehmen. Man kann dabei zwei wichtige Bereiche erkennen: den Bereich des *freien Flusses* und den Staubereich. Im freien Fluss ist die Dichte der Fahrzeuge so gering, dass eine Interaktion der Fahrzeuge miteinander vernachlässigbar ist. Die Fahrzeuge können sich hier mit ihrer Wunschgeschwindigkeit bewegen. Diese Wunschgeschwindigkeit entspricht häufig der Maximalgeschwindigkeit auf dem betrachteten Straßenabschnitt. Mit steigender Dichte nimmt die Bedeutung der Interaktionen zu. Am Punkt (ρ_c, f_{max}) gibt es genau einen Punkt des maximalen Flusses.

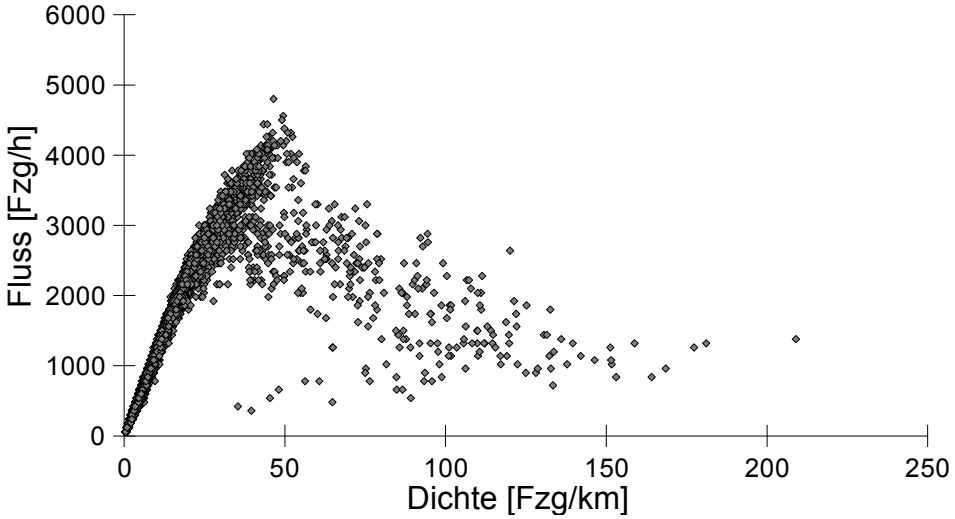


Abbildung 2.5: Gemessenes Fundamentaldiagramm einer zweistreifigen Autobahn in Deutschland (Quelle: PTV AG).

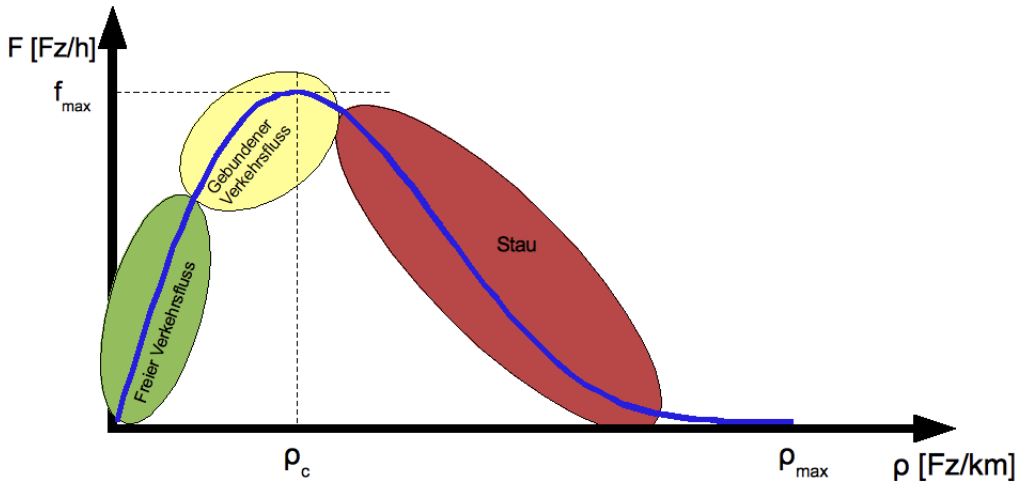


Abbildung 2.6: Schematisches Fundamentaldiagramm des Straßenverkehrs.

Übersteigt die Dichte den kritischen Wert ρ_c, übernehmen die Wechselwirkungen zwischen den Fahrzeugen die Oberhand und der Fluss sinkt. Im schlimmsten Fall entsteht dabei ein Stau. Dies wird auch an einer alternativen Darstellung des Fundamentaldiagramms (siehe Abb. 2.7), mit $v = v(\rho)$, deutlich, bei der der Geschwindigkeit die Dichte gegenübergestellt wird. Man sieht, dass mit steigender Verkehrsdichte die Geschwindigkeit abnimmt, bis sie schließlich zum Erliegen kommt.

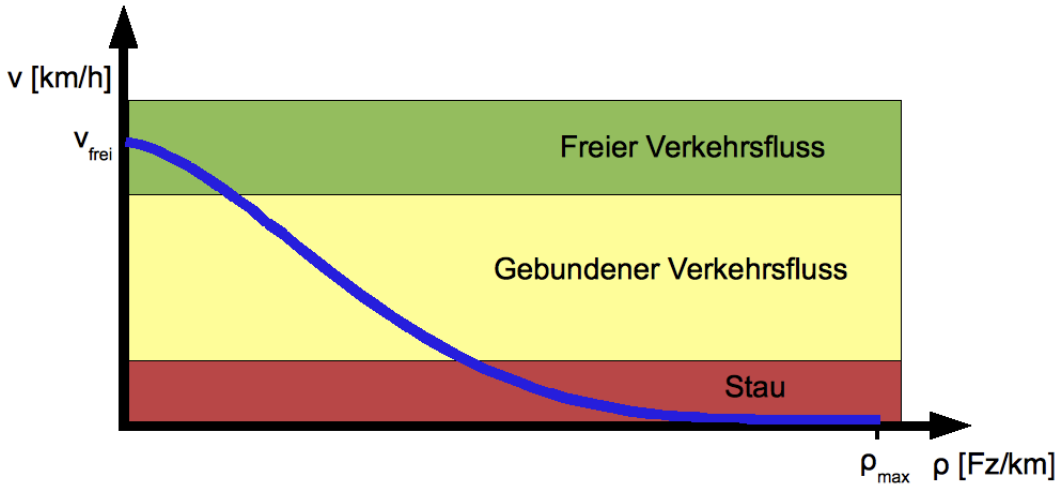


Abbildung 2.7: Schematisches Diagramm der Beziehung zwischen Geschwindigkeit und Dichte.

Eine dritte Darstellungsmöglichkeit ist die Darstellung der Beziehung von Geschwindigkeit und Fluss: $v = v(F)$ (siehe Abb. 2.8).

2.1.3 Gleichgewichtszustände

In der Verkehrssimulation wird häufig angestrebt, dass sich in einem simulierten Verkehrsnetz ein Gleichgewichtszustand ergibt. Man bezieht sich dabei auf die Grundannahme, dass ein System im Gleichgewicht eine optimale Darstellung des Verkehrs ergibt. Man muss zwischen zwei verschiedenen Gleichgewichten unterscheiden. Beim *systemoptimalen Gleichgewicht* entscheidet eine globale Instanz über die Wahl der Routen, um so eine optimale Auslastung des Verkehrsnetzes zu erzielen.

Demgegenüber steht das *Benutzergleichgewicht*. Hier entscheidet jeder Nutzer (Verkehrsteilnehmer) selbst, welche Routen er für die Fahrt im Verkehrsnetz nutzen möchte. Im Jahr 1952 formulierte John G. Wardrop zwei fundamentale Grundsätze zur Beschreibung von Gleichgewichtszuständen bei Verkehrsflüssen [166]:

1. Die Kosten aller Routen zu einem gegebenen Paar von Start- und Zielknoten sind gleich und die Kosten keiner unbenutzten Routen sind geringer.

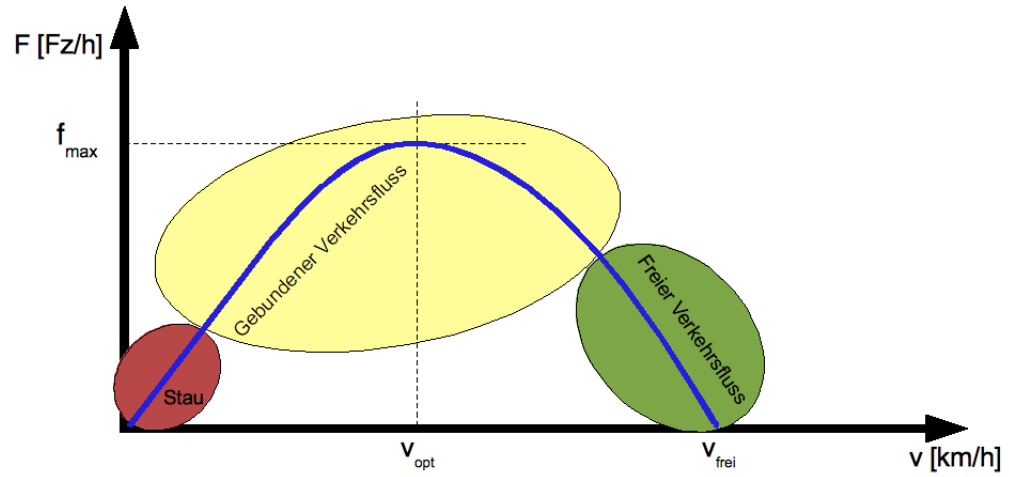


Abbildung 2.8: Schematisches Diagramm der Beziehung zwischen Geschwindigkeit und Fluss.

2. Im Gleichgewichtszustand ist die durchschnittliche Reisezeit minimal.

Diese beiden Wardrop'schen Prinzipen stellen auch heute noch eine fundamentale Grundlage im Bereich der Verkehrssimulation dar.

2.1.4 Verkehrsnetze

Verkehrsnetze sind ein wichtiger Bestandteil unserer modernen Umwelt und als Eingabeparameter für eine Verkehrssimulation unabdingbar. Verkehrsnetze beschreiben die Verkehrsinfrastruktur eines bestimmten Gebietes. Sie können dabei sowohl in ihrer räumlichen Ausbreitung als auch in ihrem Detailgrad stark variieren.

Allgemein lässt sich festhalten, dass Verkehrsnetze die Transportwege in unserer modernen Verkehrsinfrastruktur beschreiben. Diese Transportwege können eine Vielzahl von Ausprägungen repräsentieren. So existieren see- und landgebundene Wege sowie Luftwege. Im Folgenden erfolgt eine Konzentration auf die landgebundenen Wege (Straße und Schiene).

Ein Verkehrsnetz lässt sich dabei durch verschiedene Strukturen wie *Bezirke*, *Anbindungen*, *Abbiegebeziehungen*, *Strecken* und *Knoten* beschreiben. Diese *Netzelemente* sind Gegenstand der weiteren Betrachtung.

Ein Verkehrsnetz wird als ein zusammenhängender, gerichteter und gewichteter Graph $G = (V,E)$ modelliert, der aus einer Menge von Knoten $|V|$ und einer Menge von Kanten $|E|$ besteht. Die *Knoten* stellen dabei wichtige Punkte im Verkehrsnetz dar. Dies können Kreuzungen und Verzweigungen sein; aber auch Stellen an denen sich die Strukturen der Straßen ändern, beispielsweise wenn sich die Anzahl der Fahrstreifen ändert oder sich nach einem

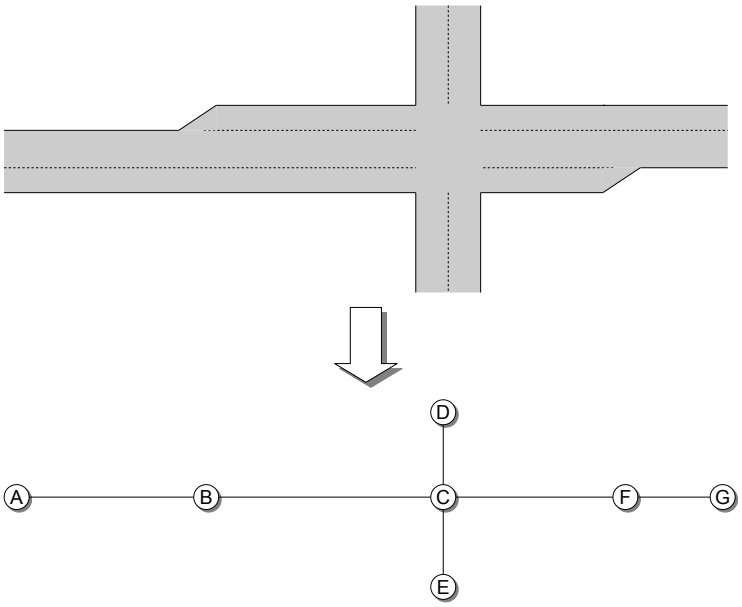


Abbildung 2.9: Umsetzung eines Straßenabschnitts in einen Graphen. An den Stellen der Fahrstreifenerweiterung beziehungsweise Verminderung werden zusätzliche Knoten eingefügt.

Verkehrszeichen die zulässige Höchstgeschwindigkeit der Straße erhöht beziehungsweise vermindert. Kanten sind die Verbindungen zwischen den Knoten und beschreiben so die Struktur des Verkehrsnetzes. Man spricht anstelle von Kanten auch von *Strecken*. Da es sich bei einer Strecke um eine gerichtete Kante im Graphen handelt, sind Hin- und Rückrichtung eigenständige Objekte im Graphen.

Zusätzlich gehören Strecken einem oder mehreren *Verkehrssystemen* an. Diese Verkehrssysteme geben an, welche Fahrzeugtypen sich auf den Strecken bewegen dürfen. Dadurch können einzelne Strecken als Schienen oder normale Straßen ausgewiesen beziehungsweise einzelne Straßen nur für den öffentlichen Verkehr zugelassen werden.

Eine einfache Umsetzung eines Straßenabschnitts in einen Graphen ist in Abb. 2.9 dargestellt. Aus Gründen der Übersichtlichkeit ist der Graph als ungerichtet visualisiert. Wie zu erkennen ist, werden an den Stellen, an denen sich die Straße verbreitet bzw. verschmälert, zusätzliche Knoten (`B` und `F`) eingefügt.

Sowohl die Knoten als auch die Kanten können eine beliebige Anzahl von Attributen besitzen, die für die Simulation eine Rolle spielen können. Diese Attribute können zur Berechnung der Gewichte herangezogen werden. Im Allgemeinen besitzen in einem Verkehrsnetz sowohl die Knoten als auch die Kanten Gewichte. Man spricht anstelle von Gewichten häufig auch von *Widerständen*. Im Folgenden werden diese Begriffe austauschbar verwendet. Von Gewichten wird zukünftig gesprochen, wenn die zu Grunde liegende Graphenstruktur

gemeint ist und von Widerständen bei den verkehrsspezifischen Aspekten.

Es existieren zahlreiche Möglichkeiten die Gewichte g der Elemente eines Verkehrsnetzes zu beschreiben. Die einfachste Variante ist die Betrachtung der Längen l der einzelnen Strecken. Allerdings ist dies eine sehr ungenaue Methode, die nicht die realen Gegebenheiten auf einer Strecke zu einem bestimmten Zeitpunkt widerspiegelt. Realistischer ist, das Gewicht g_a einer Kante a als eine Linearkombination verschiedenster Attribute $A = \{a_0, \ldots, a_n\}$ zu betrachten, d.h.

$$g_a = \alpha_0 \cdot a_0 + \alpha_1 \cdot a_1 + \ldots + \alpha_n \cdot a_n. \tag{2.1}$$

Die Attribute können dabei die Reisezeit, Mautkosten, Straßenbeschaffenheit u.v.m. sein. Durch die Faktoren $\alpha_0, \ldots \alpha_n$ können diese Attribute gewichtet werden. In der Regel nimmt dabei die Reisezeit auf einer Strecke den höchsten Einfluss ein.

Die Reisezeit t_a auf einer Strecke a kann im Falle des freien Flusses, wenn keine Behinderung durch andere Fahrzeuge existiert, mittels

$$t_a = \frac{l}{v_{max}} \tag{2.2}$$

berechnet werden, was allerdings außer in einem fast leeren Verkehrsnetz nie der Fall sein wird. Eine realitätsnähere Möglichkeit, die Reisezeit zu bestimmen, besteht unter der Annahme, dass sich die Geschwindigkeit auf einer Strecke mit ihrer Belastung q ändert. So ist die Geschwindigkeit eines Fahrzeuges auf einer freien Strecke in der Regel höher als bei dichtem Verkehr. Je höher die Belastung q einer Strecke ist, desto höher wird auch ihr Gewicht bzw. ihr Widerstand. Die Reisezeit t_a lässt sich dann wie folgt berechnen [50].

$$t_a(q) = \frac{l}{v_{max}} \cdot \left(1 + \alpha \left(\frac{q}{\gamma \cdot q_{max}}\right)^{\beta}\right) \tag{2.3}$$

In Gleichung (2.3) entspricht v_{max} der Freiflussgeschwindigkeit, also der maximal möglichen Geschwindigkeit auf dieser Strecke, und q_{max} der maximalen Kapazität dieser Strecke. Die maximale Kapazität gibt an wie viele Fahrzeuge sich auf der Strecke befinden können. Die Faktoren α, β und γ dienen der Skalierung der einzelnen Parameter und können aus empirischen Daten abgeleitet werden.

Zusätzlich zu diesen Grundelementen eines Verkehrsnetzes gibt es noch weitere Komponenten. In realen Verkehrsnetzen existieren im Allgemeinen Regeln dafür, wohin an Kreuzungen abgebogen werden darf. Diesem Sachverhalt tragen *Abbiegebeziehungen* Rechnung, die für Verkehrsgraphen modelliert sind. Diese Abbiegebeziehungen werden den Knoten des Graphen zugeordnet und geben an von welcher einmündenden Strecke an einer Kreuzung in welche ausgehende Strecke abgebogen werden darf. In ihrer einfachsten Form besteht eine Abbiegebeziehung aus einem Tripel von Knoten: Quellknoten, Zielknoten und demjenigen Knoten, der die Kreuzung repräsentiert.

Ein einfaches Beispiel für Abbiegebeziehungen ist in Abb. 2.10 dargestellt. Hier sind drei Abbiegebeziehungen an einem Kreuzungsknoten C visualisiert. Die Abbiegebeziehungen spielen bei der Routensuche (siehe Abschnitt 2.1.6) eine essentielle Rolle.

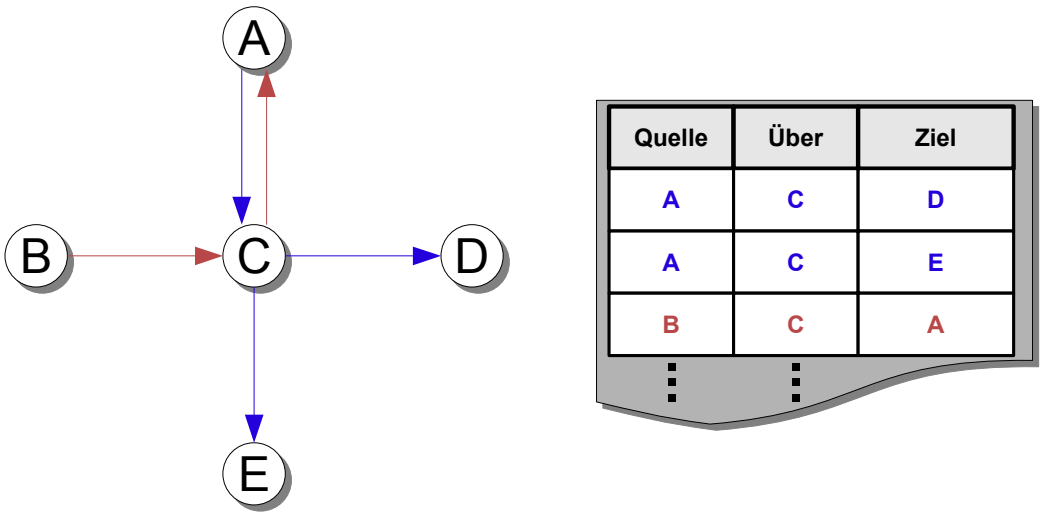


Abbildung 2.10: Abbiegebeziehungen in einem einfachen Graphen.

	Knoten	Strecken	Bezirke	Anbindungen	Abbiege-beziehungen
Karlsruhe	8.355	23.347	725	5.601	72.453
Region Stuttgart	149.652	369.009	784	14.787	1.030.269
Deutschland (IV)	109.842	758.139	6.928	55.347	643.382
Istanbul	24.486	126.980	450	4.155	174.035
New York	264.346	733.846	812	26.923	–

Tabelle 2.1: Kennzahlen einiger ausgewählter Verkehrsnetze.

Die letzten beiden Netzelemente, die *Bezirke* und *Anbindungen*, sind eng verzahnt. Bezirke beschreiben dabei Ausgangs- und Endpunkte von Verkehrsbewegungen. Sie repräsentieren beispielsweise Wohngebiete oder Arbeitsplätze. Die Verkehrsnachfrage wird mit ihrer Hilfe modelliert. Die Bezirke sind mittels Anbindungen mit den Knoten des Verkehrsnetzes verknüpft. Dies ist in Abb. 2.11 durch ein einfaches Verkehrsnetz dargestellt. Darin enthalten sind die Grundelemente wie Knoten und Strecken. Zusätzlich dazu ist dieses Grundnetz um Bezirke und Anbindungen angereichert. Wie am Beispiel von Bezirk `B1` zu erkennen ist, können Bezirke über verschiedene Anbindungen an mehrere Knoten des Verkehrsnetzes angebunden sein. Auf diese Weise wird die Verkehrsnachfrage aus einem Bezirk (beispielsweise eines Wohngebiets) an verschiedenen Stellen in das Verkehrsnetz eingespeist. Die Aufteilung der Nachfrage kann dabei über die Anbindungen gesteuert werden. Im Beispiel fließen 60% der Nachfrage von `B1` über Knoten `B` in das Verkehrsnetz ein und lediglich 40% über Knoten `A`.

Tabelle 2.1 zeigt die Kennzahlen, der oben beschriebenen Netzelemente, für einige ausgewählte Verkehrsnetze. Ein Ausschnitt des Verkehrsnetzes der Stadt New York ist in Abb.

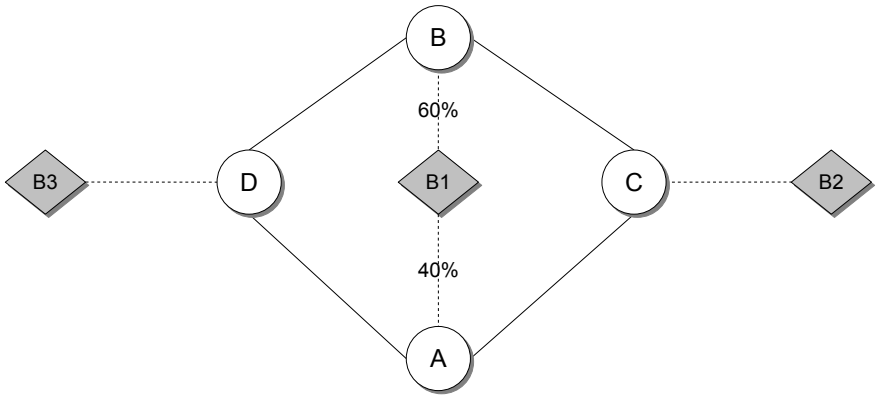


Abbildung 2.11: Erweiterte Struktur eines Verkehrsnetzes mit vier Knoten (A...D), vier Strecken und drei Bezirken (B1, ..., B3). Die Bezirke sind jeweils über Anbindungen mit Knoten verknüpft (gestrichelte Linien) (nach [135]).

2.12 abgebildet, sowie in Abb. 2.13 dasjenige der Stadt Karlsruhe.

2.1.5 Verkehrsnachfrage

Nachdem die statischen Elemente der Verkehrssimulation in den vorherigen Abschnitten beschrieben wurden, müssen diese um die dynamischen erweitert werden. Zu den dynamischen Elementen in einer Verkehrssimulation gehören in erster Linie die Verkehrsteilnehmer (beispielsweise in Gestalt von Fahrzeugen).

Ein erster naiver Ansatz, der auch in den ersten Verkehrssimulatoren Einsatz fand, besteht darin, die Fahrzeuge zufällig zu generieren und ihr Abbiegeverhalten an Kreuzungen ebenfalls dem Zufall zu überlassen. Es ist offensichtlich, dass das daraus resultierende Verhalten der Wirklichkeit nicht sehr nahe kommt. Reale Verkehrsteilnehmer bewegen sich in der Regel nicht ziellos im Verkehrsnetz, vielmehr streben sie nach Verlassen ihres Ausgangspunktes eines oder mehrere Ziele an. Man spricht auch davon, dass sie *Aktivitätspläne* haben. Fasst man diese Pläne zusammen, so ergibt sich die *Verkehrsnachfrage*. Diese Nachfrage kann allgemein für den gesamten Verkehr im System gelten oder auf einzelne *Verkehrsmodi* beschränkt sein. Unter einem Verkehrsmodus versteht man eine Verknüpfung zwischen einem Verkehrssystem (Öffentlicher Verkehr, Individualverkehr,...) und einer entsprechenden Nachfrage.

Man kann zwischen einer erhobenen und einer berechneten (künstlichen) Nachfrage, sowie zwischen einer aktuellen und einer künftigen Nachfrage unterscheiden. Erhobene Nachfragedaten werden üblicherweise durch Befragungen und sonstige demographische Erhebungen ermittelt [6]. Sie beschreiben dabei die Anzahl der Fahrzeugfahrten und deren Verteilung innerhalb des untersuchten Zeitraums (typischerweise 24h) bei einem vorhandenen

Abbildung 2.12: Ausschnitt aus dem Verkehrsnetz der Stadt New York.

Abbildung 2.13: Verkehrsnetz der Stadt Karlsruhe.

Verkehrsangebot. Allerdings stellt dies nur eine Momentaufnahme der Verkehrssituation dar. Es ist schwer die gleichen Daten nochmals zu gewinnen, wenn man die Untersuchungen zu einem späteren Zeitpunkt wiederholt. Für die Befragungen wird eine repräsentative Stichprobe der Verkehrsteilnehmer gewählt und daraus die Verkehrsnachfrage ermittelt. Bei der berechneten Verkehrsnachfrage zieht man Annahmen zur Fahrtenanzahl und deren Verteilung heran. Zur Berechnung werden unter anderem so genannte *Verkehrsnachfragemodelle* eingesetzt. Daly stellt in [40] verfeinerte Methoden der Generierung vor. Ein weiterer Überblick hierzu ist in [107] gegeben.

Generell lassen sich zwei unterschiedliche Betrachtungen anstellen. Entweder betrachtet man die Gesamtheit der Verkehrsströme von einer Quelle zu einem Ziel, das heißt man fasst die Nachfrage für diese jeweilige Relation zusammen, oder man behandelt individuelle Pläne, die in ihrer feinsten Auflösung dazu führen können, dass jedem einzelnen Verkehrsteilnehmer im System ein vollkommen individueller Aktivitätsplan zugewiesen ist.

Im ersten Fall spricht man von Quelle-Ziel-Matrizen, im zweiten ist häufig die Rede von Mulitagentensystemen bei denen jedem Agenten ein individueller Plan zugewiesen ist [11, 12].

Quelle-Ziel-Matrizen

Quelle-Ziel-Matrizen können entweder aus statistischen Erhebungen oder durch Berechnungen gewonnen werden. Die Grundstruktur einer Quelle-Ziele-Matrix kann Abb. 2.14 entnommen werden.

Q \ Z	10	20	30	40
10		27		
20				
30	12			
40			49	

Abbildung 2.14: Beispiel einer Quelle-Ziel-Matrix mit drei Elementen der Verkehrsnachfrage.

Jede Quelle-Ziel-Matrix (QZ-Matrix) ist für einen bestimmtem Simulationszeitraum gültig (beispielsweise 24h). In ihr sind jedem Paar aus Quell- und Zielbezirk – im Folgenden *QZ-Paar* – die Anzahl der Fahrzeugfahrten n innerhalb des Simulationszeitraums gegeben. Dies bedeutet, dass in dieser Zeit n Fahrzeuge von Q nach Z fahren. Im Beispiel von Abb. 2.14 bewegen sich folglich u.a. 27 Fahrzeuge von Bezirk 10 nach 20 und 12 Fahrzeuge von 30 nach 10.

Damit lässt sich das Verkehrsaufkommen recht gut beschreiben. Allerdings ist es mit diesen QZ-Matrizen nicht möglich *Aktivitätsketten* zu modellieren. Man spricht von einer Aktivitätskette, wenn ein Verkehrsteilnehmer innerhalb des untersuchten Zeitraums mehr als ein Ziel ansteuern möchte. Damit wäre es möglich ganze Tagespläne von Verkehrsteilnehmern zu modellieren. Es gibt Ansätze, diese Informationen in den QZ-Matrizen durch Erweiterungen zu beschreiben. Die Software VISEM [52, 53] unterstützt dies. Eine weitere Möglichkeit ist in Abb. 2.15 skizziert.

In diesem Fall wird versucht eine Aktivitätskette in ihre Einzelbestandteile aufzubrechen und in der Matrix abzubilden. Abb. 2.15 beschreibt eine Bewegung von $10 \rightarrow 20 \rightarrow 40 \rightarrow 30$. Die gesamte Kette wird dabei in Einzelfahrten zerlegt, die dann dem entsprechenden QZ-Paar zugeschlagen werden. Mit diesem Verfahren verlieren die Fahrten allerdings ihre Zusammengehörigkeit und können keinen individuellen Verkehrsteilnehmer mehr zugeordnet werden. Daher eignet sich diese Beschreibung vor allem für die Verwendung in makroskopischen Verkehrssimulationen (siehe Abschnitt 2.3 und Kapitel 7).

Q \ Z	10	20	30	40
10		1		
20				1
30				
40			1	

Abbildung 2.15: Beispiel einer einfachen Aktivitätskette in einer Quelle-Ziel-Matrix

Einen etwas anderen Ansatz verfolgen Balmer und Rieser [13], indem sie versuchen aus den QZ-Matrizen detaillierte Pläne abzuleiten, die sie als Eingabe für einen mikroskopischen Verkehrssimulator verwenden. Zur Generierung dieser Daten werden demographische Informationen (Bevölkerungszahlen, ...) sowie weitere statistische Verfahren herangezogen.

Aktivitätsgenerierungen

Die über die jeweiligen QZ-Paare definierten Fahrzeugfahrten fließen von Quellbezirk Q über eine entsprechende Anbindung durch einen Knoten des Verkehrsnetzes in dieses ein. Beschreibt eine Matrix nun einen Simulationszeitraum von beispielsweise 24h und 100 Fahrzeugfahrten für ein QZ-Paar, so kommt die Frage, auf zu welchen Zeitpunkten die Fahrzeuge in die Simulation eintreten. Eine Gleichverteilung der Fahrzeuge über den gesamten Simulationszeitraum ist einfach, aber nicht zielführend. Im Verkehrsaufkommen existieren schließlich tageszeitliche Schwankungen.

Diese Schwankungen werden durch die Einführung einer *Ganglinie* (wie in Abb. 2.16 dargestellt) modelliert. In dieser wird der prozentuale Anteil an der Gesamtnachfrage beispielsweise für einen Tagesverlauf angegeben [147]. In der Beispielganglinie ist ein Tag in 24 Teilintervalle unterteilt. Zu erkennen ist die tageszeitliche Schwankung des Verkehrsaufkommens mit Peaks zu Rush-Hour-Zeiten. Greift man das vorherige Beispiel mit 100 Fahrzeugfahrten innerhalb von 24h auf, so ergeben sich für den Zeitraum von 7-8 Uhr 8 Fahrzeugfahrten und für 16-17 Uhr 15 Fahrten. Diese Fahrten können dann im weiteren Verlauf – je nach Art der Simulation – mikroskopisch oder makroskopisch behandelt werden und sind schließlich Ausgangspunkt für Routensuchen.

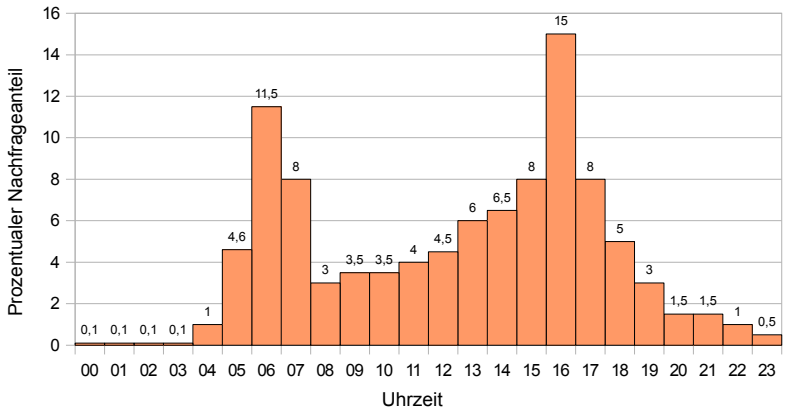


Abbildung 2.16: Beispielverlauf einer Ganglinie (nach [147])

Individuelle Aktivitätspläne für Multiagentensysteme

Neben den vorgenannten Quelle-Ziel-Matrizen gibt es weitere Möglichkeiten Aktivitäten zu beschreiben. Diese finden vor allem Anwendung in mikroskopischen Simulationsmodellen. Dabei kann es wünschenswert sein, für jeden Verkehrsteilnehmer individuelle Pläne zu besitzen, die auch Aktivitätsketten repräsentieren können. Oft erfolgt die Modellierung über *Multiagentensysteme*. Dabei wird jedem Agenten ein eigener Plan zugewiesen. Balmer et al zeigen dies in [11, 12]. Durch die Individualisierung ist es möglich, detaillierte Pläne und Zusammenhänge zu betrachten. Allerdings sind diese Verfahren auch recht speicher- und rechenintensiv. Im weiteren Verlauf dieser Arbeit spielen sie keine Rolle und werden daher nicht weiter betrachtet.

2.1.6 Routensuche in Verkehrsnetzen

In beiden Varianten der Verkehrssimulation, der makro- und der mikroskopischen, ist es notwendig, *Routen* zu berechnen. Nachdem die Verkehrsnachfrage der Bezirke eines Verkehrsnetzes über die Knoten eingespeist wurde ergibt sich das Problem, dass Fahrzeuge ausgehend von ihrem Einspeisknoten (= Quellknoten) eines oder mehrere Ziele (= Zielknoten) haben. Eine Modellierung dieser Zusammenhänge ist im einfachsten Fall durch die Quelle-Ziel-Matrizen gegeben. Die Fahrzeuge bewegen sich in der Realität also zielgerichtet. Diesem Sachverhalt muss in der Simulation Rechnung getragen werden, was mittels einer Routensuche geschieht. Dieser Abschnitt gibt einen groben Überblick über eine Auswahl von verbreiteten Routensuchverfahren.

Allgemeine Routensuche

Für die Routensuche in Verkehrsnetzen können die klassischen Algorithmen zur Wegsuche in Graphen verwendet werden. Der dem Verkehrsnetz zu Grunde liegende Graph $G = (V, E)$ besitzt eine Gewichtsfunktion $g : E \rightarrow \mathbb{R}_+$, welche einer Kante bzw. einer Strecke ihr Gewicht bzw. ihren Widerstand zuordnet. Das Gewicht einer Kante kann sich dabei auf unterschiedliche Art ergeben (siehe Abschnitt 2.1.4).

Eine Route in dem Graphen sei durch eine Folge von Knoten $(v_0, \ldots, v_n)$ beschrieben, wobei die entsprechenden Kanten im Graphen existieren müssen ($\forall i \in [0, n[: (v_i, v_{i+1}) \in E$). Ignoriert man in einem ersten Schritt die Abbiegebeziehungen und die Gewichte der anderen Netzelemente, so kann direkt einer der klassischen Algorithmen zur Kurzwegsuche verwendet werden.

Grob kann man diese Algorithmen in *label-setting* und *label-correcting* Verfahren unterteilen. Einen Überblick dazu geben Zhan und Noon [173], sowie Cormen et al [37]. Ein bekannter Vertreter der label-setting Verfahren ist der Algorithmus zur Kurzwegsuche von Dijkstra [44]. In diesem Algorithmus werden drei Listen vorgehalten: Eine enthält die bereits besuchten Knoten des Graphen, die zweite diejenigen Knoten, die von den bereits besuchten Knoten direkt erreichbar sind und die letzte Liste enthält alle noch unbekannten Knoten des Graphen. In Dijkstras Algorithmus wird immer derjenige Knoten mit der kürzesten Distanz d zum Startknoten aus der Liste der erreichbaren Knoten entfernt und zur Liste der besuchten Knoten hinzugefügt. Jeder Knoten wird dabei maximal einmal besucht, das heißt der kürzeste Weg zu einem Knoten ändert sich nicht mehr, wenn er einmal bekannt ist. Die Abstandsmarken (Entfernungen) werden also nur gesetzt, woher der Begriff label-setting herrührt.

Demgegenüber stehen die label-correcting Verfahren, deren prominentester Vertreter der Algorithmus von Bellman-Ford ist [18, 58]. Hier können Knoten mehrfach in der Liste auftauchen, wodurch die Abstandsmarken teilweise mehrfach korrigiert werden müssen.

Listing 2.1 zeigt einen allgemeinen Labeling Algorithmus zur Bestimmung eines kürzesten Weges von Startknoten $s \in V$ zu Zielknoten $t \in V$. Hierbei wird iterativ ein Suchbaum mit Wurzel s aufgebaut. Die momentane Distanz zwischen s und des während der Berechnung gerade betrachteten Knoten k wird im Feld `d[k]` gespeichert. In `pre[k]` wird zusätzlich der Vorgängerknoten von k im Suchbaum abgelegt.

Bisher wurden die Abbiegebeziehungen und die Gewichte der anderen Netzelemente neben den Strecken in der Betrachtung vernachlässigt. Eine Einbeziehung dieser Komponenten ist bei der Berechnung einer Route allerdings essentiell. Durch die Abbiegebeziehungen wird unter Umständen die Anzahl der wählbaren Strecken eingeschränkt. Zusätzlich verursachen alle überfahrenen Netzelemente Kosten (beispielsweise erhöht sich die Fahrzeit beim Überqueren einer Kreuzung), was bei der Routensuche berücksichtigt werden muss.

Sei T die Menge aller Abbiegebeziehungen. Dabei sei jedes Element $t \in T$ durch ein Tripel von Knoten gegeben: $t = (i, j, k)$ mit $i, j, k \in V$. Dieses Tripel besagt, dass ein Fahrzeug von

```
d[s] = 0
pre[s] = ⊥
for all v ∈ V, i ≠ s
   d[v] = ∞
end for
while list ≠ ∅
   remove i from list
   for all e = (i, j), e ∈ E
      if d[j] > d[i] + g(e)
        d[j] = d[i] + g(e)
        pre[j] = i
        if j ∉ list
          list = list ∪ j
        end if
      end if
   end for
end while
```

Listing 2.1: Allgemeiner Labeling Algorithmus zur Berechnung des kürzesten Wegs.

Knoten i kommend an Knoten j in Richtung Knoten k abbiegen kann. Zusätzlich ersetzen drei neue Gewichtsfunktionen g_e, g_v und g_t zur Bestimmung der Gewichte der jeweiligen Netzelemente die bisherige Gewichtsfunktion g. Dabei gelte: $g_e : E \to \mathbb{R}_+$, $g_v : V \to \mathbb{R}_+$ und $g_t : T \to \mathbb{R}_+$. Daraus ergibt sich der in Listing 2.2 skizzierte modifizierte Algorithmus von Dijkstra.

Im Algorithmus von Dijkstra ist im ungünstigsten Fall der Zielknoten der letzte Knoten, der besucht wird. In jedem Schritt muss der Knoten mit der kleinsten Distanz zum Startknoten aus der Liste entfernt werden (`extract_min()`). Zusätzlich wird jede Kante maximal einmal betrachtet. Damit ergibt sich für die Komplexität des Algorithmus $\mathcal{O}(|V|^2 + |E|)$. Verwendet man geeignete Datenstrukturen für die Verwaltung der Listen (beispielsweise Fibonacci-Heaps anstelle unsortierter Listen), kann man die Komplexität auf $\mathcal{O}(|V| \cdot \log(|V|) + |E|)$ reduzieren [60, 145].

In [89] und [172] werden weitere Kurzwegsuchalgorithmen auf ihre Tauglichkeit bei Verkehrsnetzen überprüft. Im Folgenden erfolgt eine Betrachtung ausgewählter optimierter Suchstrategien für die Berechnung einer Route in einem Verkehrsnetz.

Optimierte Routensuche

Aufgrund der meist sehr umfangreichen Quelle-Ziel-Matrizen, müssen in der Verkehrssimulation in der Regel sehr viele Routensuchen durchgeführt werden. Die klassischen Labeling Algorithmen sind für die Berechnung einer einzelnen Routen gut geeignet. Bei zu-

```
d[s] = 0
pre[s] = ⊥
for all v ∈ V, i ≠ s
   d[v] = ∞
end for
while list ≠ ∅
   i = extractMin( list )
   for all e = (i, j), e ∈ E
      t = (pre[i], i , j)
      if t ∈ T or pre[i] = ⊥
        if d[j] > d[i] + g_e(e) + g_t(t) + g_v(j)
          d[j] = d[i] + g_e(e) + g_t(t) + g_v(j)
          pre[j] = i
          if j ∉ list
            list = list ∪ j
          end if
        end if
      end if
   end for
end while
```

Listing 2.2: Modifizierter Algorithmus von Dijkstra zur Berechnung des kürzesten Wegs in einem Verkehrsnetz unter Berücksichtigung der Gewichte aller Netzelemente und der Abbiegebeziehungen.

nehmender Größe der Verkehrsnetze und wachsender Anzahl von QZ-Paaren ist es sinnvoll optimierten Routensuchstrategien den Vorzug zu geben.

Als eine erste Optimierung kann die *Veränderung des Abbruchkriteriums* des Dijkstra-Algorithmus herangezogen werden. Dijkstra berechnet ausgehend von dem Startknoten die kürzesten Wege zu allen anderen Knoten. Bei der Berechnung eines QZ-Paares genügt es bei Erreichen des jeweiligen Zielknotens abzubrechen und die weiteren Knoten nicht zu betrachten. Auf diese Weise wird die im vorherigen Abschnitt genannte Komplexität lediglich im worst-case erreicht.

Weitere Strategien zielen ebenfalls auf eine *Reduktion des Suchraums* ab, das heißt die Anzahl der betrachteten Knoten und Kanten zu minimieren. Eine *bidirektionale Suche* [110] kann dabei helfen. Bei diesen bidirektionalen Verfahren wird gleichzeitig zur Suche vom Startknoten s zum Zielknoten t eine Rückwartssuche in umgekehrter Reihenfolge von t nach s durchgeführt. Hierfür muss der Graph G des Verkehrsnetzes invertiert werden. In dem resultierenden invertierten Graphen G^{-1} sind alle Kanten $e = (i, j)$ zu $e^{-1} = (j, i)$ umgedreht. Dasselbe gilt auch für jede einzelne Abbiegebeziehung $t = (i, j, k)$, die sich zu $t^{-1} = (k, j, i)$ ändert. Beide Teilsuchen werden solange in beliebiger Reihenfolge durchge-

führt, bis ein Knoten v besucht wird, der bereits von der anderen Teilsuche betrachtet wurde. In diesem Fall verbindet v die beiden bisher ermittelten Teilsuchbäume. Der Knoten v liegt zwar nicht notwendigerweise auf dem global kürzesten Weg w zwischen s und t, aber beide Teilsuchbäume zusammen und maximal eine zusätzliche Kante enthalten w. Luby und Radge konnten in [110] zeigen, dass sich bei einer bidirektionalen Implementierung des Algorithmus von Dijkstra der Suchraum verkleinert und sich daraus eine Komplexität von $\mathcal{O}(\sqrt{|V|} \cdot \log(|V|))$ ergibt.

Bei der Simulation großer Verkehrsnetze kann es aber dennoch notwendig sein, dass noch effizientere Suchstrategien verwendet werden, das heißt Verfahren, in denen der kürzeste Weg in noch weniger Schritten ermittelt werden kann und dabei auch der Suchraum möglichst klein gehalten wird. Möhring et al [116] konnten zwar zeigen, dass bidirektionale Suchstrategien schneller sein können als der klassische Dijkstra. Dennoch wäre es wünschenswert dies weiter zu beschleunigen. Dies ist unter anderem mit Hilfe des $A*$-Algorithmus möglich [78]. Er ermöglicht durch die Anwendung von Heuristiken die Auslassung ganzer Äste im Suchbaum, wenn diese nicht zum Ziel führen. $A*$ ermittelt zunächst mit seiner Heuristik immer diejenigen Knoten, die voraussichtlich am schnellsten zum Zielknoten führen.

Eine weitere Optimierungsmöglichkeit besteht in dem Einführen einer *Vorberechnungsphase*, in der der Graph vorab analysiert wird, um bei der späteren Berechnung schneller ans Ziel zu kommen. Eine Variante ist das so genannte *reach-based Routing* [77]. Hierbei wird für jeden Knoten des Graphen ein „reach“-Wert berechnet. Dieser Wert lenkt die Suche in ihrem späteren Verlauf in Richtung des Zielknotens. Die Berechnung des kürzesten Wegs ist nach [77] etwa zehnmal schneller als der klassische Dijkstra. Allerdings nimmt die Vorberechnungsphase viel Zeit in Anspruch, so dass sich dieses Verfahren hauptsächlich bei sehr großen Graphen lohnt.

Schulz et al [150] verfolgen eine etwas andere Strategie zur Beschleunigung der Suche. Sie benutzen das Konzept von *Multi-Level-Graphen*, das heißt es werden aus dem Ursprungsgraphen in mehreren Schritten immer gröber aufgelöste Graphen mit jeweils weniger Knoten als im vorhergehenden Schritt erzeugt. Anschließend wird die Suche auf dem gröbsten Graphen mit den wenigsten Knoten durchgeführt. Nach Ermittlung des groben Weges wird dieser wieder schrittweise verfeinert. Die Berechnung des Weges ist dabei ebenfalls etwa zehnmal schneller als beim klassischen Dijkstra.

Eine ähnliche Idee verfolgen Sanders und Schultes [142, 143]. Sie versuchen die Straßenhierarchie in Verkehrsnetzen auszunutzen und auf diese Weise die Suche zu beschleunigen. Sie verwenden in ihrem Ansatz keine zusätzlichen Informationen, wie etwa den Straßentyp. In Kapitel 5 wird ein im Rahmen dieser Arbeit entwickelter Ansatz vorgestellt, der direkt die Straßentypen ausnutzt. Beiden Ansätzen liegt die Annahme zu Grunde, dass ein Fahrer für seine Fahrt diejenigen Strecken wählt, die das schnellste Vorankommen versprechen.

In [116] wird das *arc-flag*-Verfahren vorgestellt, in welchem zunächst das Verkehrsnetz partitioniert wird. In der Vorberechnungsphase, in der auch die Partitionierung stattfindet, erhält jede Kante einen Vektor, dessen Länge der Anzahl der Partitionen entspricht. Jedes

Feld dieser Vektoren ist ein bool'scher Wert. Dieser Wert ist wahr, wenn die Kante auf einem kürzesten Weg in die entsprechende Partition führt. Auf diese Weise müssen bei der Suche nur die Kanten betrachtet werden, die auf einem kürzesten Weg in die Partition liegen, in welcher der Zielknoten ist.

Wagner und Willhalm [164] nutzen geometrische Informationen, um die Suche zu beschleunigen.

Neben diesen sequentiellen Varianten der Suche existieren auch parallele Verfahren zur Berechnung kürzester Wege. Generell kann man bei der parallelen Berechnung kürzester Wege zwei Grundtypen unterscheiden: (1) Replikation des Verkehrsnetzes oder (2) Partitionierung des Verkehrsnetzes. Ein Überblick ist in [88] zu finden. Beispielsweise verwenden Adamson und Tick [4] dafür einen parallelen Greedy-Algorithmus. Lanthier et al nutzen geometrische Informationen für eine parallele Implementierung [105]. Eine genauere Betrachtung eines parallelen Kurzwegsuchealgorithmus ist in Kapitel 7 zu finden.

2.1.7 Ein allgemeiner Simulationsablauf

Nachdem die wichtigsten Größen und Strukturen in den vorherigen Abschnitten beschrieben wurden, ist es notwendig einen allgemeinen Simulationsablauf zu beschreiben. Unabhängig von der Simulationsart – mikro- oder makroskopisch – kann man die vereinfachte Grundstruktur eines Ablaufs beschreiben, da die Grundschritte ähnlich sind. Abb. 2.17 zeigt dies schematisch.

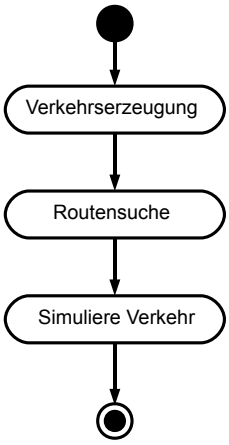


Abbildung 2.17: Schematischer Darstellung eines allgemeinen Simulationsablaufs.

Der erste Schritt ist in allen Fällen die Verkehrserzeugung. Diese hat das Heranziehen der Verkehrsnachfrage zum Ziel (siehe Abschnitt 2.1.5). Nachdem die Nachfrage in das Verkehrsnetz eingespeist wurde, werden die Routen ermittelt, die genutzt werden sollen (siehe Abschnitt 2.1.6). Abschließend erfolgt die eigentliche Simulation des Verkehrs. Die mikroskopische Simulation wird im Abschnitt 2.2 eingeführt, die makroskopische Simulation in

dem darauf folgenden Abschnitt 2.3. Im Detail kann es zu vielfältigen Abwandlungen und Rückkopplungen kommen, wenn man eine tatsächliche Simulation durchführen will.

2.2 Mikroskopische Verkehrssimulation

Bei der mikroskopischen Simulation betrachtet man das Verkehrsgeschehen aufgelöst auf die einzelnen Verkehrsteilnehmer. Modelliert werden hierfür die Interaktionen zwischen den einzelnen Teilnehmern sowie deren eigenständiges Verhalten. Durch die Wiedergabe dieses Verhaltens wird man in die Lage versetzt realistisch das Verkehrsverhalten auf den Straßen zu simulieren.

Erste Ansätze für mikroskopische Verkehrssimulationen reichen bis in die 1950er Jahre zurück. Zu den ersten einsatzfähigen und ausgereiften Modellen zählten die so genannten *Fahrzeug-Folge-Modelle* [36, 21]. Diese hatten aber neben einem höheren Rechenaufwand einige andere Nachteile, wie weiter unten gezeigt werden wird. Dies führte dann in den 1980er und 1990er Jahren zur Entwicklung anderer Modelle, die versuchten, diese Nachteile zu reduzieren. Von großer Bedeutung war hier das im Jahr 1992 von Schreckenberg und Nagel entwickelte Zellularautomatenmodell [125].

Zunächst beginnt dieser Abschnitt mit einer Betrachtung von Fahrzeug-Folge-Modellen. Anschließend werden detaillierter andere Modelle zur mikroskopischen Verkehrssimulation basierend auf Zellulären Automaten beschrieben. Diese werden für die weitere Behandlung in dieser Arbeit von Bedeutung sein.

2.2.1 Fahrzeug-Folge-Modelle

Zu den ersten entwickelten mikroskopischen Verkehrsflussmodellen gehören die *Fahrzeug-Folge-Modelle*. Diese lehnen sich an Ideen der Newton'schen Mechanik an, indem für jedes Fahrzeug eine Bewegungsgleichung angegeben wird, ähnlich wie dies bei interagierenden Partikeln im Newton'schen System geschieht. Dort reagiert ein Partikel mit einer Beschleunigung auf einen Stimulus. Überträgt man diese Idee auf den Verkehr, so reagiert ein Fahrzeug auf seine Umgebung stets mit einer Beschleunigung oder durch Abbremsen.

Im Folgenden sollen zwei Modellklassen der Fahrzeug-Folge-Modelle vorgestellt werden: *Follow-the-Leader*-Modelle und das *Optimal-Velocity*-Modell.

Follow-the-Leader Modelle

Die ersten *Follow-the-Leader Modelle* entstanden in den frühen 1950er Jahren. So entwarfen Reuschel [137] und Pipes [134] zunächst einfache Modelle, die die Geschwindigkeitsdifferenz zum Vordermann, das heißt zwischen n-tem und $n+1$-tem Fahrzeug, als *Stimulus* heranziehen. Sie legten ihren Modellen die Idee zu Grunde, dass ein Fahrer sich meistens

an der Geschwindigkeit des jeweiligen Vordermanns orientiert und versucht sich dieser anzugleichen. Daher leitet sich auch der Name „Follow-the-Leader“ zur Beschreibung dieser Modelle ab. Damit lässt sich die Bewegungsgleichung für das n-te Fahrzeug mit

$$\ddot{x}_n(t) = \frac{1}{\tau}[\dot{x}_{n+1}(t) - \dot{x}_n(t)] \tag{2.4}$$

angeben. In Gleichung (2.4) entspricht dabei τ einem Sensitivitätsfaktor, der angibt wie stark ein Fahrer auf den Stimulus reagiert.

Pipes [134] bezog in sein Modell noch folgende Überlegungen mit ein: Er ging davon aus, dass je höher die Geschwindigkeiten sind, desto höher der Abstand zum Vordermann ist und dass es zur Vermeidung von Unfällen notwendig ist, einen Sicherheitsabstand $(\Delta x)_{safe}$ zu verwenden. Dies führte zu Gleichung (2.5).

$$\Delta x_n(t) = x_{n+1}(t) - x_n(t) = (\Delta x)_{safe} + \tau \cdot \dot{x}_n(t) \tag{2.5}$$

Allerdings wiesen Chandler et al. [34] darauf hin, dass diese Annahmen für eine korrekte Beschreibung des Fahrverhaltens der Verkehrsteilnehmer zu vereinfachend sind. Sie argumentierten, dass es wichtig ist, die Reaktionszeit T eines Fahrers in die Betrachtung mit einzubeziehen. Dies führt zu folgender Beschreibung der Bewegung:

$$\ddot{x}_n(t+T) = S \cdot [\dot{x}_{n+1}(t) - \dot{x}_n(t)] \tag{2.6}$$

Hierbei gibt S einen konstanten Sensitivitätskoeffizienten wieder. Aber selbst diese Erweiterung genügt noch nicht, um hinreichend realistisch den Straßenverkehr beschreiben zu können. So existieren Probleme bei Fahrzeugen, die keinen nahen Vordermann besitzen: Sie orientieren sich mit ihrer Geschwindigkeit an einem gegebenenfalls sehr weit entfernten vorausfahrenden Fahrzeug. Dieses Verhalten entspricht aber nicht empirischen Beobachtungen. Außerdem ist in der Realität beobachtbar, dass die Sensitivität eines Fahrers bei kürzeren Abständen zum Vordermann steigt. Diesem Umstand wird in den Modellen von Gazis [68, 69] und Gipps [70] Rechnung getragen. Für einen umfangreicheren Überblick über Follow-the-Leader-Modelle sei auf [21] verwiesen.

Darüber hinaus gibt es Versuche, diese Modelle um psycho-physische Komponenten zu erweitern und so das Entscheidungsverhalten von Fahrern realitätsnäher zu gestalten. Unter diesen Gesichtspunkten entwickelte Wiedemann ein Modell [168, 169], welches unter anderem in der Simulationssoftware PELOPS Eingang fand [111, 43].

Follow-the-Leader-Modelle weisen Nachteile auf, die zum Teil bis heute noch nicht befriedigend gelöst sind. So wird in der Regel das Überholen von langsameren Fahrzeugen auf mehrstreifigen Straßen nach wie vor vernachlässigt. Ein weiteres Problem stellt die Annahme einer gleichen Beschleunigungs- und Bremsdauer dar. Eine große, ungelöstes Schwäche ist, dass sich das Verhalten von Fahrzeugen im freien Fluss nicht mit empirischen Beobachtungen deckt. Zudem erfordern viele dieser Modelle, dass zahlreiche ihrer Parameter erst gemäß empirischer Daten kalibriert werden müssen und keine in sich schlüssige Theorie dafür hergeleitet werden kann [36].

Optimal Velocity Modell

Eine andere Variante eines Fahrzeug-Folge-Modells, das eine größere Verbreitung fand, ist das von Bando et al entwickelte *Optimal Velocity Modell* [15, 14]. Anders als bei den Follow-the-Leader-Modellen betrachten sie nicht die Geschwindigkeit des vorausfahrenden Fahrzeugs, sondern bauen ihr Modell auf der Annahme auf, dass sich die gewünschte Zielgeschwindigkeit aus dem *Abstand* zum Vordermann ergibt.

$$\ddot{x}_n(t) = \frac{1}{\tau}[V_{opt}(\Delta x_n(t)) - \dot{x}_n(t)] \tag{2.7}$$

Dabei bezeichnet V_{opt} die *Optimal-Velocity-Funktion* (OV-Funktion). Diese kann je nach genauer Modellausprägung stark variieren. Eine einfache Möglichkeit sie zu definieren, ist die Einbeziehung der Treppenfunktion Θ (engl. heavyside step function) [160].

$$V_{opt}(\Delta x) = v_{max} \cdot \Theta(\Delta x - d) \tag{2.8}$$

Hier ist d eine Konstante. Der allgemeine Ablauf der Optimal Velocity Modelle gliedert sich in die vier nachfolgenden Schritte: (1) Zunächst wird der Abstand Δx zum nächsten vorausfahrenden Fahrzeug bestimmt. (2) Anschließend wird die Wunschgeschwindigkeit für jedes Fahrzeug mittels $V_{opt}(\Delta x_n)$ berechnet. (3) Diese soeben berechnete Wunschgeschwindigkeit wird nun mit der aktuellen Geschwindigkeit v_n des jeweiligen Fahrzeugs verglichen. (4) Als letztes wird das Fahrzeug gemäß der Bewegungsgleichung (2.7) entweder beschleunigt oder abgebremst.

Die Optimal Velocity Modelle erlauben eine recht realitätsnahe Nachbildung der Fundamentaldiagramme des Verkehrs (siehe Abschnitt 2.1). Dennoch erweisen sie sich als rechenintensiv und erscheinen für umfangreiche Simulationen von großen Verkehrsstrukturen als ungeeignet.

2.2.2 Das Modell von Nagel und Schreckenberg

Die im vorherigen Abschnitt beschriebenen Fahrzeug-Folge-Modelle sind zu rechenintensiv für größere Simulationen und Simulationsszenarien. Dies war lange Zeit ein Grund, warum sich die Verkehrssimulation auf den makroskopischen Bereich konzentrierte.

Mit der Verwendung zellulärer Automaten im Bereich der Verkehrssimulation [125] im Jahr 1992 tat sich allerdings zum ersten Mal eine Möglichkeit auf, eine mikroskopische Simulation schnell genug berechnen und ausführen zu können, um damit größere und realistischere Szenarien behandeln zu können.

Exkurs: Zelluläre Automaten

Zelluläre Automaten (engl. Celluar Automaton CA) wurden 1940 erstmalig von Stanislaw Ulam, der zu dieser Zeit in Los Alamos arbeitete, beschrieben. Allerdings wurden sie erst von seinem Kollegen John von Neumann zu einem universellen Berechnungsmodell erweitert.

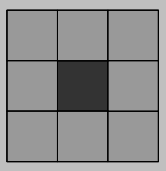
Jeder Zelluläre Automat ist dabei durch die folgenden Größen beschrieben:

- einem Zellraum Z
- einer Zustandsmenge Q
- einer endlichen Nachbarschaftsmenge N
- einer lokalen Überführungsfunktion $\delta : Q^N \rightarrow Q$

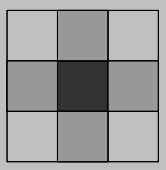
Ein CA kann dabei beliebig viele Dimensionen annehmen, wobei in den meisten Anwendungsfällen zwei Dimensionen ausreichend sind. Jeder Zelle des CA wird nun eine bestimmter Zustand zugeordnet. Der Zustandsübergang einer Zelle von einem alten Zustand Q_{alt} in einen neuen Zustand Q_{neu} wird mittels bestimmter Übergangsregeln durchgeführt. Die Zustandsübergänge erfolgen dabei stets nach der selben Überführungsfunktion. Diese Überführung erfolgt für alle Zellen gleichzeitig.

Meist spielen für die Zustandsübergänge die Zustände der Nachbarzellen eine Rolle. Man unterscheidet hier zwei Arten von Nachbarschaften. Ausgegangen wird von der dunkelgrauen Zelle in der Mitte. Die Zellen, die zur Nachbarschaft zählen, sind hellgrau eingefärbt.

1. die Nachbarschaft nach Moore



2. die Nachbarschaft nach v. Neumann



In den 1970er Jahren gerieten CA durch John Conways *Game of Life* in das Blickfeld einer breiten Öffentlichkeit. Hierbei handelt es sich um einen zweidimensionalen CA, der das Wachstum von Bakterienkulturen simulieren soll. Angenommen wird bei diesem CA eine Moore-Nachbarschaft und folgende Zustandsmenge:

$$Q = \{0,1\} = \{tot, lebendig\}$$

Das heißt eine Zelle kann entweder lebendig oder tot sein. Dies ist abhängig von der Anzahl der Nachbarn, die sie hat.

Das Basismodell

Das so genannte Nagel-Schreckenberg-Modell (NaSch-Modell) [125] macht sich nun das Konzept der zellulären Automaten zu eigen und überträgt dieses auf den Bereich der Verkehrssimulation. Dem ursprünglichen Modell liegt dabei die Idee zugrunde, dass man einen Straßenabschnitt ohne weiteres in einzelne Abschnitte (Zellen) diskretisieren kann (siehe Abb. 2.18).

Abbildung 2.18: Diskretisierung eines Straßenabschnitts gemäß des Nagel-Schreckenberg-Basismodells.

Im Basismodell beträgt die Länge einer Zelle 7,5 Meter. Dies entspricht in etwa dem Raum, den ein Fahrzeug inklusive Sicherheitsabstand auf einer Straße einnimmt. Die Zustandsmenge einer Zelle ist darin gegeben durch $Q = \{\text{belegt},\ \text{frei}\}$, das heißt in einer Zelle kann sich maximal ein Fahrzeug befinden. Mehr als ein Fahrzeug in einer Zelle entspräche einer Kollision.

Zusätzlich wird jedem Fahrzeug f_i noch eine Geschwindigkeit v_i zugewiesen. Diese diskretisierte Fahrzeuggeschwindigkeit, das heißt Anzahl der Zellen, die im nächsten Zeitschritt befahren werden, befindet sich im Intervall $[0,\ v_{max}]$, wobei es sich bei v_{max} um die Maximal- beziehungsweise Wunschgeschwindigkeit des Fahrzeugs handelt. Im einfachsten Fall kann man v_{max} für einen gesamten Straßenabschnitt (Strecke) als konstant annehmen.

Das Basismodell bearbeitet die Bewegung von Fahrzeugen schließlich in den folgenden vier Phasen. Dabei bezeichnet f_i das Fahrzeug in der i-ten Zelle und v_i die dazugehörige Geschwindigkeit.

1. **Beschleunigen:** Falls die Geschwindigkeit v_i eines Fahrzeugs f_i kleiner als v_{max} ist und der Abstand d zum nächsten Fahrzeug f_{i+j} größer als $v_i + 1$ ist, dann ergibt sich die neue Geschwindigkeit von f_i: $v_i \rightarrow v_i + 1$

2. **Bremsen:** Falls ein Fahrzeug f das nächste Fahrzeug an der Stelle $i + j$ sieht ($j \leq v$), so muss die Geschwindigkeit von f reduziert werden, um eine Kollision zu vermeiden: $v_i \rightarrow j - 1$

3. **Trödeln:** Mit einer gewissen Wahrscheinlichkeit p wird die Geschwindigkeit v_i eines Fahrzeugs reduziert: $v_i \rightarrow v_i - 1$

4. **Fortbewegung:** Jedes Fahrzeug wird gemäß seiner Geschwindigkeit v um entspre-

chend viele Zellen voran bewegt.

Die Phasen 1 und 2 entsprechen dabei dem Interesse des Fahrers, seine Wunschgeschwindigkeit zu erreichen und dann auch zu erhalten. Dies geschieht aber unter dem Vorbehalt der Kollisionsfreiheit. Diese wird durch die Einhaltung eines Sicherheitsabstands (Phase 2) gewährleistet.

Phase 3 ist essentiell um realistisches Fahrverhalten zu modellieren. Sie bringt ein stochastisches Element in die Simulation. Ohne diese Phase würde sich jeder Fahrer gleich und „optimal“ verhalten. Dieser Trödelfaktor führt Fluktuationen im Fahrverhalten in das Modell ein. Unter diesen Fluktuationen kann man Überreaktionen beim Bremsen, Einhalten eines nicht-optimalen Abstands und verzögertes Beschleunigen verstehen. All diese Komponenten sind für eine mikroskopische Verkehrssimulation imminent wichtig, bspw. um Stop-and-Go-Wellen zu simulieren.

Aber bereits diese vier einfachen Phasen oder Regeln reichen aus um realistisch Verkehr auf Straßen mit nur einer Fahrspur und ohne Überholvorgänge zu simulieren. Dennoch ist bereits hier offensichtlich, dass das Basismodell nicht annähernd ausreichend ist, um alle Facetten des modernen Straßenverkehrs abzubilden, was allerdings essentiell ist um realistische und aussagekräftige Simulationen durchführen zu können.

Weitere CA-Modelle

Mittlerweile existieren eine Vielzahl von Verkehrsflussmodellen basierend auf Zellulären Automaten. Ein Überblick ist in [36] und [112] zu finden. Exemplarisch sollen hier einige Modelle kurz erwähnt werden. Dazu gehört das Modell von Fukui und Ishibashi mit verschiedenen Erweiterungen [65, 106, 165]. Bei diesem gibt es zwei grundlegende Unterschiede gegenüber dem NaSch-Modell. Zum einen ist der Geschwindigkeitsanstieg im Vergleich zum NaSch-Modell nicht zwangsläufig graduell und zum weiteren wird der Trödelfaktor nur auf Fahrzeuge mit hohen Geschwindigkeiten angewandt.

Ein weiteres Modell betrachtet eine alternative Behandlung der Lücke (engl. *gap*) zwischen zwei Fahrzeugen. Werth [167] zieht zur Berechnung dieser Lücke nicht nur die Geschwindigkeit des vorausfahrenden Fahrzeugs heran, sondern zusätzlich noch die Geschwindigkeitsdifferenz zwischen eigener Geschwindigkeit und vorausfahrendem Fahrzeug. Dies entspricht einer häufig beobachteten Verhaltensweise von Fahrern in realem Verkehr.

Zusätzlich existieren noch Adaptionen des Optimal Velocity Modells, welches wir weiter oben bereits kennen gelernt haben, hin zur Verwendung zellulärer Automaten. Erwähnt seien hier die Arbeiten von Emmerich und Rank [47] und Helbing und Schreckenberg [84].

2.3 Makroskopische Verkehrssimulation

Eine zweite große Klasse von Verkehrssimulationsmodellen sind im makroskopischen Bereich zu suchen. Anders als bei den zuvor betrachteten mikroskopischen Modellen interessiert hier nicht die Bewegung und Interaktion einzelner Verkehrsteilnehmer. Vielmehr betrachtet man den Verkehr als kontinuierlichen Fluss und damit verbundene aggregierte physikalische Größen. Zu diesen gehören der Verkehrsfluss F, die Verkehrsdichte ρ und die mittlere Geschwindigkeit $\bar{v}$. Diese aggregierten Größen sind häufig Grundlage von verkehrsplanerischen Entscheidungen.

Man kann zwei große Klassen makroskopischer Verkehrsmodelle unterscheiden. Zum einen sind dies Modelle, die sich der Grundlagen der *Strömungsmechanik* bedienen (auch hydrodynamische Modelle genannt). Zum anderen sind es *Umlegungsverfahren*. Beide Klassen sind Gegenstand der nachfolgenden Behandlung, wobei der Schwerpunkt auf den Umlegungsverfahren liegt.

2.3.1 Makroskopische Simulation mit Hilfsmitteln der Strömungsmechanik

Die ersten Modelle der makroskopischen Simulation des Verkehrsflusses mit den Mitteln der Strömungsmechanik gehen auf die 1950er Jahre zurück. Damals entstand die Idee, man könne den Verkehr als eine Art zähfließende Flüssigkeit auffassen, die durch die Adern – die Straßen – einer Stadt fließt. Diesen Eindruck erhält man, wenn man aus größerer Höhe beispielsweise aus dem Fenster eines Flugzeugs hinab auf eine Stadt schaut.

Lighthill und Whitham entwickelten als eine der ersten dafür ein Modell, indem sie die Kontinuumstheorie der Physik auf das Verkehrsgeschehen übertrugen [109]. Nahezu zeitgleich entwickelte Richards ein sehr ähnliches Modell [138]. Man spricht vereinfachend auch vom *LWR-Modell*. In diesem Modell besteht die Forderung nach dem Erhalt der Fahrzeugzahl, das heißt auf einem Streckenabschnitt $[x,y]$, der keine Auf- und Abfahrten besitzt, dürfen keine Fahrzeuge verloren gehen. Dieser Erhaltungssatz führt zu der in Gleichung (2.9) beschriebenen Kontinuitätsgleichung.

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial F(x,t)}{\partial x} = 0 \tag{2.9}$$

Lighthill und Whitham nahmen für ihr Modell an, dass der Fluss augenblicklich auf Dichteänderungen reagiert und leiteten daraus die folgende Beziehung zwischen Fluss F und Dichte ρ ab:

$$F(x,t) = F(\rho(x,t)) \tag{2.10}$$

Aus den Gleichungen (2.9) und (2.10) erhält man die *Lighthill-Whitham-Richards*-Gleichung:

$$\frac{\partial \rho(x,t)}{\partial t} + v_g(\rho) \cdot \frac{\partial \rho(x,t)}{\partial x} = 0 \tag{2.11}$$

wobei $v_g(\rho) = \frac{dF}{d\rho}$.

Dieses einfache Modell hat allerdings einige Nachteile. Zum einen lässt sich mit seiner Hilfe das Fundamentaldiagramm nicht berechnen. Vielmehr benötigt man dies als Eingabeparameter, was in weiteren Arbeiten getan wurde (beispielsweise [36] und [81]). Zum anderen sind einige Annahmen zu vereinfachend. Der Verkehr befindet sich nicht immer im Gleichgewicht, sondern Fahrer reagieren häufig auf das Verkehrsgeschehen, indem sie Beschleunigen oder Abbremsen. Daher können Ampeln und Unfälle nicht durch dieses Modell beschrieben werden.

Kühne entwarf ein Modell basierend auf LWR, welches versuchte, dem Rechnung zu tragen [104]. Er erweiterte es um die Annahme, dass ein Fahrzeug abbremsen wird, wenn sich die Verkehrsdichte vor ihm erhöht. Ähnlich ging Payne vor, der einen Verkehrsdruck mit einbezog, der ein zusätzliches Abbremsen bei höheren Dichten zur Folge hat [133]. Hilliges entwarf ein Modell mit zeitlich verzögerter Anpassung an die Gleichgewichtsgeschwindigkeit, die er in einer neu hinzugefügten Bewegungsgleichung beschrieb [85]. Cremer [38] und Papageorgiou [130] verbesserten die numerische Umsetzung dieser Modelle.

Weitere Modellerweiterungen beziehungsweise ähnliche Modelle wurden im Laufe der Jahre in zahlreichen Arbeiten vorgestellt. Exemplarisch seien hier Kerner und Konhäuser [96], Helbing [79, 80] und Wiedemann [169] genannt.

All diese Modelle besitzen allerdings ein fundamentales Problem. Durch zum Teil erforderliche komplexe numerische Lösungsverfahren sind sie sehr rechen- und zeitintensiv. Zudem ist eine Simulation heterogenen Verkehrs mit ihnen nur schwer möglich. Dies schließt sie für unsere gewünschten Anwendungsszenarien mit großräumigen und detaillierten Verkehrsnetzen aus.

2.3.2 Umlegungsverfahren zur makroskopischen Simulation des Verkehrs

Eine andere Möglichkeit der makroskopischen Simulation des Verkehrsflusses sind *Umlegungsverfahren*. Sie werden in der Regel dazu herangezogen den Zusammenhang zwischen Angebot und Nachfrage abzubilden. Diese Verfahren bedienen sich bei der Berechnung nicht strömungsmechanischer Ansätze und können ohne die rechenintensive numerische Lösung von Differentialgleichungen auskommen. Bei diesen Verfahren wird die Verkehrsnachfrage auf die einzelnen Netzelemente verteilt. Man spricht bei diesem Vorgang auch davon, dass die Nachfrage auf die Elemente *umgelegt* wird, was zum Namensgeber für diese Gruppe der Verfahren wurde.

Abb. 2.19 zeigt den vereinfachten Ablauf eines allgemeinen Umlegungsverfahrens. Seine Grundstruktur gliedert sich in die Schritte *Routensuche*, *Routenwahl* und schließlich die *Belastung der Netzelemente*. Die gewonnenen Ergebnisse eines Umlegungsverfahren sind in aller Regel Belastungen einzelner Netzelemente (Knoten, Strecken, ...), Verkehrsströme und Kenngrößen von Ortsveränderungen. Unter Verkehrsströmen versteht man im Indivi-

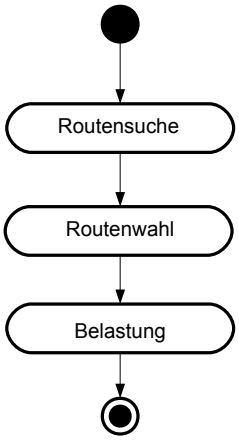


Abbildung 2.19: Ablauf eines allgemeinen Umlegungsverfahrens

dualverkehr belastete Routen beziehungsweise im Öffentlichen Verkehr belastete Verbindungen zwischen einer Quelle und einem Ziel. Unter den wichtigsten Kenngrößen einer Ortsveränderung ist die Reisezeit zu finden.

Routensuche

Bei der Routensuche finden verschiedene Wegsuchalgorithmen Einsatz. Der am häufigsten angewendete Algorithmus ist der von Dijkstra entwickelte Algorithmus zur Suche des kürzesten Weges in einem Graphen [44]. Er hat in verschiedenen Ausprägungen Einzug in die Routensuche bei Umlegungsverfahren gefunden. Eine genauere Betrachtung der Suchverfahren ist Gegenstand von Abschnitt 2.1.6. Wichtig ist hierbei die Unterscheidung der Ergebnisse, welche entweder *Verbindungen* oder *Routen* sein können. Von Routen spricht man, wenn eine Ortsveränderung durch eine Folge von Strecken und Knoten beschrieben wird. Bei Verbindungen kommt zur Beschreibung einer Route noch eine zeitliche Komponente hinzu.

Bei der Suche werden für jeden Eintrag der Quelle-Ziel-Matrix mehrere Routen bzw. Verbindungen berechnet. Ergebnis der Routensuche ist daher eine Menge von Routen bzw. Verbindungen zu jedem Quelle-Ziel-Paar.

Routenwahl

Im nächsten Schritt, der Routenwahl, muss die Verkehrsnachfrage auf die im vorhergehenden Schritt ermittelten Routen verteilt werden. Hierbei wird versucht, den individuellen Entscheidungen der Verkehrsteilnehmer in der realen Welt Rechnung zu tragen. Ein erster einfacher Ansatz, lediglich den kürzesten Weg zwischen einer Quelle und einem Ziel zu betrachten, ist nicht besonders realitätsnah. Nicht jeder Verkehrsteilnehmer wird stets

den kürzesten Weg wählen, um an sein Ziel zu gelangen. Vielmehr hängt die Routenwahl von einer Fülle individueller Präferenzen ab. Hinzu kommt, dass die Routenwahl eines Fahrers die Routenwahlentscheidungen anderer Verkehrsteilnehmer beeinflussen kann: Je mehr Fahrer eine bestimmte Route wählen, desto unattraktiver wird sie im Laufe der Zeit, da ihre Belastung wächst und sich damit potentiell die Reisezeit verlängert. Ziel der Routenwahl ist es daher möglichst ein Nutzergleichgewicht unter den Routen zu erreichen. Dieses Gleichgewicht orientiert sich an den Wardrop'schen Prinzipien.

Bei der Routenwahl im Öffentlichen Verkehr kommt neben der Umsteigehäufigkeit noch eine zeitliche Komponente (beispielsweise in Form der Abfahrtszeiten) hinzu. Aus diesem Grund spricht man hier anstelle von Routenwahl auch von einer Verbindungswahl. Für einen Verkehrsteilnehmer sind die Abfahrtszeiten und ob sich diese mit seinen Wünschen bzw. seinem Bedarf decken von entscheidender Bedeutung.

Im Allgemeinen erfolgt die Verteilung der Nachfrage auf die Routen nach Aufteilungsmodellen [135]. Die Grundstruktur dieser Modelle hat dabei folgende Gestalt.

$$P_i^a = \frac{U_i^a}{\sum_{j=1}^{n} U_j^a} \tag{2.12}$$

Durch Gleichung (2.12) wird der prozentuale Anteil P_i^a der Nachfrage für eine Route (Verbindung) i im Zeitintervall (auch *Zeitscheibe*) a bestimmt. Hierzu wird der Widerstand g_i^a von i in den *Nutzen* U_i^a umgewandelt, was durch eine vom Benutzer spezifizierte Funktion geschehen kann. Je nach Wahl dieser Funktion entstehen unterschiedliche Modelle von denen hier zwei exemplarisch vorgestellt werden sollen.

Beim Modell nach *Kirchhoff* lässt sich P_i^a durch

$$P_i^a = \frac{(g_i^a)^{-\beta}}{\sum_{j=1}^{n} (g_j^a)^{-\beta}} \tag{2.13}$$

berechnen. Hierbei dient der Parameter β zur Beschreibung der Widerstandsempfindlichkeit. Bei diesem Modell werden ausschließlich die Verhältnisse der Widerstände betrachtet. Es orientiert sich in seinem Verhalten an der bekannten Regel in elektrischen Schaltkreisen.

Das *Logit*-Modell verwendet die Folgende Gleichung [30].

$$P_i^a = \frac{e^{-\beta \cdot g_i^a}}{\sum_{j=1}^{n} e^{-\beta \cdot g_j^a}} \tag{2.14}$$

In der Praxis hat sich gezeigt [135], dass sich von diesen beiden Modellen dasjenige von Kirchhoff gut eignet. Das Logit-Modell zeigt bei kurzen und langen Routen nicht ganz realitätsnahes Verhalten.

Routenbelastung

Die Routenbelastung entspricht der Beschreibung der Fortbewegung der Fahrzeuge auf den zuvor ermittelten Routen bzw. Verbindungen. Dies geschieht durch die Anwendung so ge-

nannter *Verkehrsflussmodelle*. Es existieren eine Vielzahl von Verkehrsflussmodellen, die von einer sehr groben und einfachen Betrachtungsweise bis hin zu detaillierten mikroskopischen Modellen reichen. Eine Möglichkeit ist das Heranziehen eines *kapazitätsabhängigen Modells*. Hierbei ermittelt sich die Fahrzeit auf einer Strecke über eine *Capacity Restraint* Funktion (CR-Funktion). Diese zieht die Auslastung q heran. Die folgende Gleichung (2.15) wurde schon in Abschnitt 2.1.4 zur Ermittlung der Fahrzeit auf einer Strecke besprochen.

$$t_{akt} = t_0 \cdot \left(1 + a\left(\frac{q}{q_{max} \cdot c}\right)^b\right) \tag{2.15}$$

Sie kann als eine solche CR-Funktion herangezogen werden [50]. Es gilt dabei zu beachten, dass bei dieser vereinfachenden Funktion jede Strecke unabhängig von allen anderen betrachtet wird. Auf diese Weise ist es u.a. nicht möglich Rückstaueffekte zu simulieren. Will man dies tun, ist ein Betrachten der Nachbarstrecken notwendig. Im Modell von Papageorgiou [131] wird dem Rechnung getragen. Im weiteren Verlauf dieser Arbeit wird allerdings das in Gl. 2.15 vorgestellte Modell herangezogen werden. Für eine ausführliche Betrachtung anderer Modelle sei daher auf [131] verwiesen.

2.4 Bestehende Systeme zur Verkehrssimulation

In den letzten Jahren gab es eine stetige Weiterentwicklung von Verkehrsmodellen und Aspekten ihrer Parallelisierung. In [124] beschreiben Nagel und Schleicher eine erste parallele Implementierung des ursprünglichen NaSch-Modells [125]. Sie vernachlässigen dabei allerdings jeden Aspekt des mehrstreifigen oder heterogenen Verkehrs. Auf Grund dieser Einschränkungen erreichten sie Beschleunigungen um einen Faktor 1.000 im Vergleich zu damals vorhandenen sequentiellen Modellen.

Eine Weiterentwicklung erfuhr die Parallelisierung in der Software TRANSIMS[1] [123], welche jahrelang am Los Alamos National Laboratory, unter anderem unter Mitwirkung von Nagel, entwickelt wurde. Dabei wurde TRANSMIS unter Verwendung von Gebietszerlegungen parallelisiert. Es wurde dabei ein Szenario mit etwa 200.000 Knoten betrachtet. Sie erzielten im Durchschnitt ein Echtzeitverhältnis von über 20, das heißt ihre Simulation lief 20 mal schneller als die Realität.

TRANSIMS wurde im Laufe der Jahre um zahlreiche Komponenten erweitert und stellt mittlerweile ein umfassendes Werkzeug zur mikroskopischen Verkehrssimulation dar. Eine Erweiterung bestand in der Anwendung von Multiagententechnologien, die es erlaubte, individualisierte Aktivitätspläne zu simulieren [139, 140]. In [122] analysieren Nagel et al verschiedene Verfahren zur Simulation großräumiger Szenarien mittels TRANSIMS. Die Idee von Simao und Powell [152], Warteschlangensysteme für die mikroskopische Simulation des Straßenverkehrs einzuführen, führte zu deren Integration in TRANSIMS [155].

[1] `http://transims.tsasa.lanl.gov/`

Charypar [35] gelang es durch Verbesserungen eine deutliche Beschleunigung der Simulation zu erreichen. So war es ihm möglich eine 24h Simulation in nur 84 Sekunden durchzuführen. In [67] entwickelte Gawron ebenfalls ein Modell zur Verkehrssimulation unter Einbeziehung von Warteschlangensystemen, FASTLANE. Cetin et al [33, 32] entwarfen ebenfalls ein solches Modell, dessen Anwendungsszenario gezielt die Simulation großer Verkehrsnetze und -szenarien war.

All diese Warteschlangenmodelle schlagen andere Modelle deutlich im Hinblick auf die Rechenzeit. Allerdings erkauft man sich damit Ungenauigkeiten. Eine individuelle Betrachtung der Wechselwirkungen der einzelnen Verkehrsteilnehmer ist nicht möglich. Jede Strecke wird nämlich durch mindestens eine Warteschlange ersetzt. Die Verweildauer eines Fahrzeugs wird beim Betreten der jeweiligen Warteschlange ermittelt. Kreuzungsbehandlungen gestalten sich ebenfalls als schwierig. Eine zufriedenstellende Behandlung mehrstreifigen Verkehrs ist auch noch nicht gewährleistet. Aus diesem Grund werden diese Modelle im Rahmen diese Arbeit nicht näher beleuchtet.

Neben TRANSIMS existieren noch zahlreiche weitere mikroskopische Verkehrssimulation. Dazu zählt der kommerzielle Simulator VISSIM[2] der PTV AG [49]. Dieser sehr ausgereifte Simulator unterstützt unter anderem die Modellierung von Stadt- und Autobahnverkehr. Den Bewegungen der Fahrzeuge liegt das Fahrzeugfolgemodell von Wiedemann [169] zu Grunde. Zum Fahrstreifenwechsel können verschiedene Modelle eingesetzt werden. Es ist damit möglich heterogenen Verkehr zu simulieren. Daneben ist eine Fußgängersimulation basierend auf dem Social-Force-Modell von Helbing et al [83] möglich. Eine Parallelisierung von VISSIM ist derzeit geplant.

Das Open-Source Softwarepaket SUMO[3] setzt auf dem mikroskopischen Simulationsmodell von Krauß [102, 103] auf. Dieses Modell nutzt Warteschlangensysteme um das Verkehrsnetz zu modellieren. Die Fahrzeuge bewegen sich dabei von Warteschlange zu Warteschlange. Dabei kann eine Warteschlange eine ganze Strecke oder Teile davon darstellen. Allerdings besitzt dieses Modell dieselben Nachteile wie die anderen Warteschlangenmodellen auch.

Dupuis und Chopard gelang es in [45] einen parallelen mikroskopischen Verkehrssimulator zu entwickeln, der auf zellulären Automaten beruht und in der Lage ist den Verkehr der Stadt Genf zu simulieren. Das Stadtnetz von Genf setzt sich dabei aus ca. 3.000 Strecken und etwa 1.000 Knoten zusammen. Sie erreichen bei ihrer Simulation bei 14 Rechenknoten ein Echtzeitverhältnis von etwa 49. Eine Analyse mit mehr Rechenknoten erfolgte nicht, ebenso fand nur eine statische Lastverteilung statt.

Rickert entwarf den mikroskopischen Verkehrssimulator PAMINA[4] [139], der unter anderem auf dem klassischen Nagel-Schreckenberg-Modell [125] und seiner Erweiterung um mehrstreifigen Verkehr [126] beruht. In [139] wurden Simulationen auf dem deutschen Autobahnnetz durchgeführt, welches aus etwa 3.300 Knoten und 6.800 Strecken bestand. Das

[2] http://www.ptv.de/traffic/software-und-system-solutions/vissim/
[3] http://sumo.sourceforge.net/
[4] http://www.zaik.uni-koeln.de/~mr/PAMINA.html

erzielte Echtzeitverhältnis liegt dabei unter Verwendung von 16 Rechenknoten knapp über 1.

Neben den reinen Simulatoren existieren ebenfalls einige Erweiterungen der Zellularautomatenmodelle um die Behandlung von Stadtverkehr. Die ersten Modelle konzentrierten sich vor allem auf die Simulation des Autobahnverkehrs. Dies hatte vor allem zwei Gründe: Zum einen ist die Beschreibung des Verhaltens von Verkehrsteilnehmern einfacher zu beschreiben und zum anderen reichten die damaligen Rechenkapazitäten nicht aus, detaillierte Netzstrukturen wie sie in einer Stadt existieren zu simulieren. In [48] wird vor allem Wert auf die Behandlung von Kreuzungen gelegt. Simon und Nagel beschreiben in [154] ein vereinfachtes Modell zur Simulation von städtischem Verkehr.

Zur makroskopischen Simulation des Verkehrs mittels Umlegungsverfahren existiert in erster Linie das Softwarepaket VISUM[5] der PTV AG [135], welches vor allem zu Zwecken der Verkehrsplanung eingesetzt wird. Parallele Implementierungen von Umlegungsverfahren sind bisher, vor allem auf nachrichtengekoppelten Systemen, eher selten.

Neben den Verfahren, die im weiteren Verlauf dieser Arbeit vorgestellt werden, beschreibt Gawron in [66] und [67] weitere Möglichkeiten zur Berechnung von Umlegungen.

[5] `http://www.ptv.de/traffic/software-und-system-solutions/visum/`

3 Grundlagen paralleler Systeme

In den letzten Jahren haben parallele Rechensysteme in ihren verschiedenen Ausprägungen stetig an Bedeutung zugenommen. Die parallele Berechnung hat mittlerweile in vielen Bereichen der Informatik Einzug gehalten, nicht zuletzt auch dadurch, dass sich heute in den meisten Rechnern bereits Multicore-Prozessoren befinden. Auch für diese Arbeit spielen Parallelisierung und parallele Architekturen eine herausgehobene Rolle. Da diese Arbeit einen weiten Bogen zwischen Verkehrssimulation und Informatik spannt, ist es notwendig die Grundlagen beider Seiten zu beleuchten.

Im weiteren Verlauf der Arbeit wird davon ausgegangen, dass der Leser Grundkenntnisse der parallelen Programmierung besitzt. Daher ist es zunächst notwendig in die wichtigsten Begriffe einzuführen, die für das weitere Verständnis dieser Arbeit notwendig sind. Für detaillierte Einführungen in parallele Systeme und ihre Programmierung sei auf [59, 72, 90] und [161] verwiesen.

3.1 Gründe für eine Parallelisierung

Die Rechenleistung von Computern steigt immer weiter an. Nach dem Moore'schen Gesetz verdoppelt sich die Anzahl der Transistoren auf einem integrierten Schaltkreis etwa alle zwei Jahre. Damit einher geht auch eine Steigerung der Rechenleistung. Zur Veranschaulichung dieser Rechenleistung seien drei Beispiele genannt. Jeder dieser drei Rechner stand für eine gewisse Zeit an der Spitze der TOP500-Liste[1]. In den Jahren 2002–2004 stand der *Earth Simulator* auf Platz 1. Er besitzt 5.120 Prozessoren und hat eine Spitzenleistung von 40,96 TFlops. Er wurde 2004 von dem Rechner *BlueGene/L* abgelöst, der nach mehreren Ausbaustufen mit 212.992 Prozessoren bei 596,38 TFlops landete. Im Juni 2008 übernahm der *Roadrunner* die Führung mit einer Spitzenleistung von 1.456 TFlops und 129.600 Prozessoren.

[1] `http://www.top500.org`

Zunächst sieht man, dass bei der Rechenleistung im Bereich der Hochleistungsrechnern eine rasante Leistungsentwicklung stattfindet, die in etwa parallel zu der Entwicklung verläuft, wie man sie auch bei PC-Systemen beobachten kann. Eine erste Vermutung legt nun nahe, dass man lediglich eine gewisse Zeit warten muss, bis die Rechenleistung irgendwann ausreichend ist, um ein gewünschtes Problem zu lösen. Allerdings wächst mit steigender Rechenleistung auch der Wunsch nach neuen Anwendungen und der Betrachtung detaillierter Problemszenarien. Genügte es beispielsweise zu Beginn der Verkehrssimulation, lediglich Autobahnabschnitte mit einigen Kilometern Länge zu betrachten [125], so besteht heute der Wunsch, komplette Verkehrsinfrastrukturen zu simulieren (beispielsweise das komplette Autobahnnetz des Bundeslandes Nordrhein-Westfalen [95, 113]). Dieser Zyklus lässt sich kaum durchbrechen, der Bedarf an Rechenleistung ist keineswegs gesättigt. Vielmehr ist die Frage, was für Systeme den Ressourcenhunger der Anwendungen ökonomisch befriedigen können, nach wie vor aktuell. Das Beispiel aus der TOP500-Liste zeigt, dass dies im Hochleistungsbereich nur mit massiv parallelen Systemen möglich ist.

Es lassen sich vor allem drei Gründe für die Entwicklung paralleler Lösungsstrategien identifizieren. Erstens, kann der Wunsche bestehen, den *Durchsatz* zu erhöhen, indem N Probleme gleichzeitig berechnet werden. Beispielsweise lassen sich N Instanzen eines sequentiellen Programms heranziehen, um auf verschiedenen Teilen eines Datensatzes zu arbeiten. Dadurch lässt sich das Gesamtproblemen schneller lösen. Prominente Beispiele dieser Lösungsstrategie sind die Projekte SETI@home[2] und Folding@home[3], wobei ersteres zur Suche nach extraterrestrischen Signalen in Datensätzen eines Radioteleskops verwendet wird und bei letzterem Proteinfaltungen untersuchet werden.

Ein zweiter Grund kann der Wunsch nach einer Reduzierung der *Laufzeit* sein. Die Idee ist, dass ein paralleles Programm, welches auf N Prozessoren verteilt ist, die gemeinsam daran arbeiten ein Problem einer gewissen Größe zu berechnen, dies in kürzerer Zeit zu einem Ergebnis bringen kann. Im Idealfall kann dies in $1/N$ der ursprünglichen Zeit erfolgen. Im Gegensatz zu dem vorherigen Ansatz treten hierbei allerdings zwei Probleme auf: Zum einen muss das bestehende Programm durch eine parallele Variante ersetzt werden und zum anderen kommen zusätzlich zum eigentlichen Berechnungsaufwand noch Kommunikationskosten hinzu.

Der dritte Grund ist eine Erhöhung der *Problemgröße*. Unter Ausnutzung aller lokalen Speicher etc. der beteiligten Rechenknoten kann man größere Probleme lösen als dies vorher möglich war. Im besten Fall ist es möglich, ein Problem zu lösen, das einen N-mal so großen Datensatz zur Grundlage hat. Ab einer gewissen Größe ist es schwierig einen einzigen großen Speicher zu erwerben. Es ist einfacher einzelne kleine Speicher zu verwenden. Auch hierfür muss ein paralleles Programm entwickelt werden und Kommunikationskosten kommen hinzu.

Im Rahmen dieser Arbeit sind vor allem die Reduzierung der Laufzeit und eine Erhöhung der Problemgröße von besonderem Interesse. Heutige Simulationen des Verkehrs und Um-

[2] `http://setiathome.ssl.berkeley.edu/`
[3] `http://folding.stanford.edu/`

legungsverfahren können, abhängig von den Simulationsszenarien, mehrere Stunden Rechenzeit in Anspruch nehmen. Eine Verkürzung dieser Rechenzeit ist für den produktiven Einsatz wünschenswert, um schneller an Ergebnisse kommen zu können. Will man realistischere, großräumigere und höher aufgelöste Szenarien simulieren, die sich auf einem einzelnen Rechner nicht mehr behandeln lassen. Hinzu kommt, dass die Simulation in der Regel nur ein Schritt eines Arbeitsablaufs ist, der oft erst nach Verfügbarkeit der Simulationsergebnisse weitergeführt werden kann. Daher ist es sinnvoll Verfahren zu entwickeln die diese Erhöhung der Problemgröße ermöglichen.

3.2 Modelle für Parallelisierungen

Will man parallele Programme entwickeln, so muss zunächst festgelegt werden, welche Programmabläufe des sequentiellen Programms parallelisiert werden sollen. Abb. 3.1 zeigt einen Auswahl von wichtigen parallelen Programmstrukturen. Daneben sind noch zahlreiche weitere vorstellbar, die aber hier in der weiteren Betrachtung vernachlässigt werden.

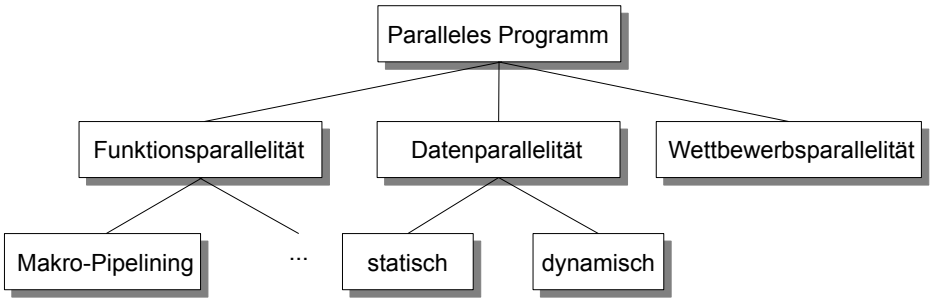


Abbildung 3.1: Übersicht über eine Auswahl paralleler Programmstrukturen

Von *Funktionsparallelität* spricht man, wenn Teilkomponenten des Programms (Funktionen, Prozeduren, Anweisungsblöcke, ...) parallel ausgeführt werden. Die verschiedenen Komponenten werden dabei auf den beteiligten Rechenknoten gleichzeitig ausgeführt. Diese Art der Parallelität ist allerdings auch mit einigen Nachteilen verbunden. Ein wichtiges Problem ist, dass für jede Teilkomponente ein separates Programm geschrieben werden muss. Da die Anzahl der Komponenten über den maximalen Parallelitätsgrad entscheidet, ist damit auch die Skalierbarkeit beschränkt. Zur Funktionsparallelität gehört auch ein Verfahren, das sich *Makro-Pipelining* nennt. Dies ähnelt Pipelining-Prozessen in modernen Prozessoren. Hierbei macht man sich zu Nutzen, dass typischerweise zwischen den Teilkomponenten Daten ausgetauscht werden müssen. Dabei übergibt eine Komponente (Erzeuger) der nächsten (Verbraucher), die von ihr bearbeiteten Daten. Auf diese Weise kann eine überlappende Pipeline für einen Datenstrom erzeugt werden.

Bei der *Datenparallelität* werden die gleichen Anweisungen oder Anweisungsblöcke eines Programms gleichzeitig auf verschiedenen Prozessoren für verschiedene Teile der Daten

ausgeführt. Ein großer Vorteil davon ist, dass lediglich ein einziges Programm für alle Prozessoren ausreicht und datenparallele Algorithmen oft ideal skalieren. Bei datenparallelen Verfahren wird zwischen *statischen* und *dynamischen* Strukturen unterschieden. Im statischen Fall entscheidet entweder der Compiler zur Übersetzungszeit oder das Programm zu Beginn seiner Ausführung, welche Teile parallel und welche sequentiell ausgeführt werden. Bei dynamischen Strukturen wird diese Entscheidung zur Laufzeit des Programms getroffen. Dadurch ist es möglich, sie zur Lastverteilung heranzuziehen. Allerdings wird diese Flexibilität durch einen höheren Organisations- und Synchronisationsaufwand für die Ausführungsreihenfolge erkauft. Diese dynamischen Strukturen lohnen sich daher nur, wenn die unterschiedlichen Teile unterschiedlich lange laufen.

In der dritten Variante, der *Wettbewerbsparallelität*, werden verschiedene Prozesse, die alle dasselbe Problem lösen, parallel ausgeführt. Die unterschiedlichen Prozesse setzen dabei verschiedene Problemlösungsstrategien ein. Sobald ein Prozess eine Lösung gefunden hat, können alle anderen Prozesse ihre Berechnung abbrechen. Diese Variante hat allerdings den Nachteil, dass man für die Ausführung viele verschiedene Programme benötigt.

Im Rahmen dieser Arbeit ist vor allem die Datenparallelität von Interesse. Das Potential der Funktionsparallelität ist auf Grund der Datenabhängigkeiten der untersuchten Verfahren nicht ausreichend für massiv parallele System. Eine Wettbewerbsparallelität scheidet auf Grund ihrer Struktur aus. Es existieren in den meisten Fällen für die Verkehrssimulation keine unterschiedlichen Problemlösungsstrategien, bei denen sich die Qualität der berechneten Ergebnisse mit geringem Aufwand vergleichen ließe, was für diese Strategien notwendig ist. Eine Datenparallelität bietet sich an, wenn man beispielsweise eine Aufteilung des Verkehrsnetzes oder der Nachfragedaten in Betracht zieht, was im Folgenden auch getan werden wird.

3.3 Parallele Architekturen

Der nächste wichtige Schritt ist, sich vor dem Entwurf paralleler Programme Gedanken über die Zielarchitektur zu machen. Flynn schlug 1972 [57] eine Kategorisierung von Rechnerarchitekturen vor, die mit Einschränkungen auch noch heute Gültigkeit besitzt. Er charakterisierte Rechner durch die Art und Weise wie sie mit einem *Befehlsstrom* (engl. instruction stream) und einem *Datenstrom* (engl. data stream) umgehen. Dabei identifizierte er vier Architekturen:

- SISD (Single instruction stream – single data stream)
  Hierbei handelt es sich um *Einprozessorrechner*, die der von-Neumann-Architektur folgen
- SIMD (Single instruction stream – multiple data stream)
  Diese Kategorie umfasst Feld- und Vektorrechner
- MISD (Multiple instruction stream – single data stream)
  Die Gruppe dieser Architektur wird in der Regel als leer betrachtet (siehe [161])

- MIMD (Multiple instruction stream – multiple data stream)
  Dies sind klassische Multiprozessorsystem wie sie heute primär zum Einsatz kommen

Die *Multiprozessorsysteme* sind heute die verbreitetste Variante bei parallelen Systemen. Man unterscheidet hierbei zwischen *speichergekoppelten* und *nachrichtengekoppelten* Systemen. Bei speichergekoppelten Multiprozessorsystemen (engl. shared memory multiprocessors) haben alle Prozessoren Zugriff auf einen gemeinsamen Speicherbereich. Eine Synchronisation und Kommunikation erfolgt über einen Datenaustausch in diesem gemeinsamen Speicher. Viele heute verfügbare Arbeitsplatzrechner sind indirekt schon speichergekoppelte Systeme, da die in ihnen verwendeten Prozessoren auf Multicore-Technologie setzen [5, 159]. Zur Programmierung für speichergekoppelte Systeme bietet sich der Einsatz von OpenMP [17, 87] an. Dieser Standard unterstützt die Parallelisierung durch Threads und Schleifen, indem er bestimmte Compiler-Direktive dafür anbietet.

Nachrichtengekoppelte Multiprozessorsysteme bestehen aus mehreren über ein Netzwerk verbundenen Rechenknoten. Man spricht auch von Rechenclustern. Die Rechenknoten besitzen keinen gemeinsamen Speicher, sondern vielmehr physikalisch getrennte Speicher. Die Kommunikation erfolgt über einen Nachrichtenaustausch zwischen den beteiligten Rechenknoten. Eine mögliche Synchronisation muss ebenfalls über Nachrichten erfolgen. Die einzelnen Komponenten eines Rechenclusters sind dabei über ein Netzwerk (beispielsweise Ethernet, InfiniBand oder Myrinet) miteinander verbunden. Zur Entwicklung paralleler Programme für nachrichtengekoppelte Systeme existieren Bibliotheken, die den Standard des *Message Passing Interface* (MPI) [115] umsetzen. Dieser stellt Kommunikationsprimitive zur Verfügung, die die Entwicklung von Programmen unterstützen. Eine weit verbreitete MPI-Bibliothek, die auch im Rahmen dieser Arbeit zum Einsatz kam, ist MPICH [73, 74]. Weitere Informationen zu MPI sind unter anderem in [129] und [136] zu finden.

Heutige Rechencluster sind häufig keine reinen nachrichtengekoppelten Multiprozessorsysteme mehr. Es handelt sich dabei in der Regel um *hybride Systeme*: Jeder Rechenknoten ist für sich ein kleines speichergekoppeltes System. Dementsprechend bietet sich für diese Fälle auch eine entsprechend angepasste Programmierung an.

Im Bereich der Verkehrssimulation ist es zunächst naheliegend, von Einprozessorsystemen auf speichergekoppelten Multiprozessorsysteme umzusteigen. Simulationen des Verkehrs werden heutzutage häufig in Ingenieurbüros durchgeführt, in denen unter Umständen entsprechende Rechner als Arbeitsplatzrechner zur Verfügung stehen. Diese Systeme ermöglichen Teile der Simulationen parallel auszuführen (beispielsweise mehrere Routensuchen auf dem gleichen Verkehrsnetz, das komplett im gemeinsamen Speicher gehalten wird) und der zusätzliche Programmieraufwand für die parallele Ausführung ist - im Vergleich zu nachrichtengekoppelten Systemen - moderat. Allerdings ist der gemeinsame Speicher der limitierende Faktor. Zusätzlich lässt sich eine Reduktion der Laufzeit auch nur bedingt erreichen: Die Anzahl an Prozessoren ist deutlich beschränkt, weil die Ankopplung vieler Prozessoren an den Speicher (die eine sehr hohe Bandbreite verlangt) technisch aufwändig und somit teuer ist. Üblicherweise findet man heutzutage speichergekoppelte Systeme mit bis zu 16 Prozessoren. Will man nun sehr große Verkehrsnetze nutzen und detaillierte Szenarien simulieren oder eine weitere deutliche Reduzierung der Laufzeit erreichen, ist man

zwangsläufig auf den Einsatz von nachrichtengekoppelten Multiprozessorsystemen angewiesen. Aus diesem Grund werden im weiteren Verlauf dieser Arbeit Rechencluster, also nachrichtengekoppelte Systeme, betrachtet.

3.4 Lastverteilung

Bei der Ausführung eines parallelen Programms ist es wichtig, eine gleichmäßige Lastverteilung unter allen beteiligten Prozessoren zu erreichen. Haben alle beteiligten Prozessoren ungefähr die gleiche Rechenlast, so ist die Rechenzeit möglichst gering, denn die Ressourcen werden am besten genutzt. Existiert ein Ungleichgewicht in der Verteilung der Arbeitslast, kommt es zwangsläufig dazu, dass einzelne Prozessoren ihre Arbeit beendet haben und nichts mehr tun (*idle*), während andere noch viel zu berechnen haben. Dabei wird die Rechenkapazität der freien Prozessoren verschenkt. Es ist daher wünschenswert, die Arbeitslast auf die Prozessoren möglichst gleichmäßig zu verteilen.

Es gibt zwei grundsätzlich unterschiedliche Strategien eine Lastverteilung zu erreichen: *Statisch* und *dynamisch* (siehe Abb. 3.2). Bei der statischen Lastverteilung wird zu Beginn des Programms, noch vor der Ausführung der parallelen Teile, eine Aufteilung der Arbeit vorgenommen und anschließend diese verteilt. Während der Ausführung des Programms ist eine nachträgliche Änderung der Arbeitsverteilung nicht mehr vorgesehen. Der Vorteil dieser Art der Lastverteilung ist, dass sich die Aufteilung schnell und effizient bestimmen lässt. Allerdings erkauft man sich diesen Vorteil mit dem Nachteil, dass man meist kein detailliertes Wissen darüber besitzt, wie lange genau die Ausführungszeit der einzelnen Programmteile bei einer beliebigen Aufgabe sind: Will man ein Beispiel aus der Verkehrssimulation zur Hand nehmen, so eignet sich dafür die Routensuche im Verkehrsnetze. Will man die Routen zwischen beliebigen QZ-Paaren berechnen, so kann diese Suche unterschiedlich lange dauern (abhängig davon, wo im Verkehrsnetz die jeweilige Quelle und das Ziel liegen). Dadurch kommt es zwangsläufig zu Ungleichverteilungen.

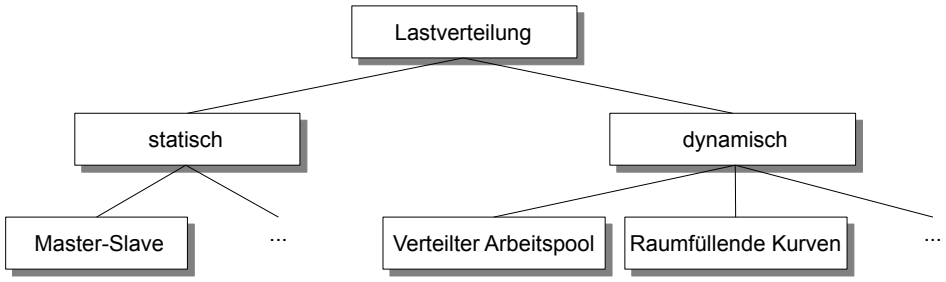


Abbildung 3.2: Übersicht über Lastverteilungsstrategien

Um diese Probleme zu lösen, bietet es sich an, dynamische Lastverteilungsstrategien zu betrachten [149]. Hierbei wird die Arbeit während der Laufzeit dynamisch verteilt. Dadurch kann in der Regel eine bessere Lastverteilung erreicht werden als bei statischen Verfahren.

Allerdings erkauft man sich dies durch einen zusätzlichen Overhead, der für die Organisation und die Kommunikation der Lastverteilung notwendig ist. Man unterscheidet bei den dynamischen Lastverteilungen zwischen *zentralisierten* und *dezentralisierten* Verfahren. Bei den zentralisierten Verfahren verteilt eine zentrale Instanz die Arbeit. Ein prominentes Beispiel dieser Verfahren ist das so genannte *Master-Slave-Prinzip*, bei dem der zentrale Prozess (Master) Arbeitsaufträge an die einzelnen Prozessoren (Slaves) verteilt. Hat ein Slave seinen Arbeitsauftrag abgearbeitet, signalisiert er dies dem Master und erhält daraufhin neue Arbeit.

Bei den zentralisierten Verfahren kann die zentrale Instanz, der Master, bei einer großen Anzahl von Arbeitsknoten, den Slaves, zu einem Flaschenhals werden. Durch Anwendung der dezentralisierten Verfahren wird versucht, dieses Problem zu lösen. Es gibt dabei einen Zwischenschritt: Es werden mehrere Master eingesetzt, von denen jeder eine Teilmenge der Slaves kontrolliert. Geht man noch einen Schritt weiter, so kann man das Prinzip des Masters vollkommen auflösen und ein vollkommen verteiltes Verfahren betrachten, bei dem die Slaves untereinander kommunizieren und die Arbeitsaufträge austauschen. Des Weiteren können, anstelle eines unstrukturierten verteilten Arbeitspools, raumfüllende Kurven zur dynamischen Lastverteilung herangezogen werden [141].

Für die Berechnungen, die in dieser Arbeit verfolgt werden, ist es ausreichend ein zentralisiertes Verfahren heranzuziehen. Die Anzahl der genutzten Slaves und die Art der Kommunikation, die eingesetzt werden wird, sind nicht groß genug, um den Master zu einem Flaschenhals werden zu lassen. Aus diesem Grund wird in erster Linie das Master-Slave-Prinzip angewendet werden.

3.5 Kenngrößen zur Evaluierung paralleler Programme

Von besonderem Interesse, neben der reinen *Laufzeit* und dem *Speicherverbrauch* eines parallelen Programms, ist die *Beschleunigung* (engl. *Speedup*) des parallelen Programms gegenüber seiner sequentiellen Implementierung. Der Speedup gibt folglich an wieviel schneller ein Programm in seiner parallelen Variante abgearbeitet werden kann.

$$S(n) = \frac{T(1)}{T(n)} \tag{3.1}$$

Gleichung (3.1) beschreibt die Bestimmung des Speedups S bei einer vorgegebenen Prozessorzahl n. Dabei stellt $T(1)$ die Ausführungszeit auf einem Einprozessorsystem (sequentiell) dar und $T(n)$ die Ausführungszeit auf einem Multiprozessorsystem mit n Prozessoren. Der Speedup bewegt sich in der Regel im Bereich $1 \leq S(n) \leq n$. Die Ausführungszeit $T(n)$ eines parallelen Programms setzt sich, vereinfacht betrachtet, dabei aus zwei Komponenten zusammen: (1) der Kommunikationszeit $T_{comm}(n)$, das heißt der Zeit, die für den Nachrichtenaustausch zwischen den beteiligten Prozessoren benötigt wird und (2) der eigentlichen Berechnungszeit $T_{comp}(n)$.

$$T(n) = T_{comp}(n) + T_{comm}(n) \tag{3.2}$$

Eine weitere wichtige Kenngröße ist die parallele *Effizienz* (engl. *efficiency*) $E(n)$, die sich aus Gleichung (3.1) durch Normierung mittels der Anzahl der Prozessoren ergibt.

$$E(n) = \frac{S(n)}{n} \tag{3.3}$$

Die Effizienz liegt typischerweise in $\frac{1}{n} \leq E(n) \leq 1$.

Bei der Ermittlung des Speedup und der Effizienz kann man nach [59] zwischen *relativen* und *absoluten* Werten unterscheiden. Bei ersteren verwendet man das parallele Programm auch für die Ermittlung der sequentiellen Werte. Dabei fließen allerdings unter Umständen die Zeiten für Kommunikation und Synchronisationsoverhead mit ein. „Gerechter“ ist es das parallele Verhalten des Algorithmus mit der besten bekannten sequentiellen Implementierung zu vergleichen, wie dies in der zweiten Variante der Fall ist.

Da, im Rahmen dieser Arbeit entstandenen Programme nur als parallele Programme existieren und in Ermangelung frei verfügbarer sequentieller Gegenstücke, beschränkt sich im weiteren Verlauf die Betrachtung auf die relativen Werte von Speedup und Effizienz.

Generell lässt sich feststellen, dass der maximale Speedup, den man durch die Parallelisierung eines Algorithmus erhalten kann, nicht nur von der Anzahl der verwendeten Prozessoren abhängt, sondern auch von weiteren limitierenden Faktoren. Jeder Algorithmus kann aufgeteilt werden in einen *sequentiellen* und einen *parallelen* Teil. Dabei beschreibt der sequentielle Teil genau diejenigen Codefragmente, die zwingend sequentiell ausgeführt werden müssen. Demgegenüber umfasst der parallele Teil diejenigen Teile des Codes, die parallelisiert werden können. Unter Zuhilfenahme dieses Aspekts formulierte Gene Amdahl 1967 eine Gesetzmäßigkeit [7], die seither mit seinem Namen verknüpft ist: *Amdahls Gesetz*.

$$S(n) \leq \frac{1}{a} \tag{3.4}$$

Gleichung (3.4) beschreibt den maximalen Speedup, der erreicht werden kann, wobei a $(0 \leq a \leq 1)$ dem sequentiellen Teil eines Algorithmus entspricht.

Bei der Arbeit mit massiv-parallelen Systemen stellten allerdings Gustafson et al [76] fest, dass die Vorhersagen nach Amdahl nicht mit den Beobachtungen auf ihren Systemen übereinstimmten. Gustafson nahm an, dass sich jedes genügend große Problem effizient parallelisieren ließe. Er entwickelte daraus eine Gesetzmäßigkeit. Bei großen Problemgrößen erweist sich das Gesetz von Gustafson häufig als realistischer. Bei wachsender Problemgröße wächst der parallelisierbare Anteil, da der Anteil der Rechenzeit gegenüber der Ausführungszeit des sequentiellen Teils (Deklarationen, ...) ansteigt.

Zur Beurteilung der Leistungsfähigkeit von Verkehrssimulatoren wird häufig noch der Begriff des *Echtzeitverhältnisses* (engl. *real time ratio*) eingeführt. Dieses beschreibt das Verhältnis zwischen Echtzeit t_{sim}, das heißt der Zeitraum, der durch die Simulation abgedeckt wird, und Simulationszeit t_{wall}, das heißt der Zeit, die der Simulator für die Berechnungen benötigt (auch wall clock time):

$$rtr = \frac{t_{sim}}{t_{wall}} \tag{3.5}$$

Je größer der Wert desto schneller ist die Simulation. Beispielsweise besagt ein Echtzeitverhältnis von 2, dass die Simulation doppelt so schnell abläuft wie die Realität, das heißt 24h Realzeit könnten in 12h simuliert werden.

4 Ein Framework zur Verkehrssimulation

Verkehrssimulationen setzen in ihren Grundversionen häufig die gleiche Datenbasis als Eingangsparameter ein. So benötigen sie neben Strukturbeschreibungen des zu simulierenden Verkehrsgebiets in Form von Verkehrsnetzen auch Nachfragedaten, welche im Rahmen dieser Arbeit als QZ-Matrizen vorliegen. Im Rahmen dieser Arbeit entstand eine flexibles und erweiterbares Framework zur Verkehrsimulation, welches diese gemeinsame Datenbasis ausnutzt. Dieses Framework unterstützt sowohl mikroskopische als auch makroskopische Verkehrssimulationen. Es ist dabei möglich, jede Simulationsart sequentiell oder parallel auszuführen. Die Ergebnisse können abschließend in vielfältiger Form gespeichert und visualisiert werden. Abb. 4.1 zeigt die Grundstruktur des Frameworks.

4.1 Eingangsdaten

Die Verkehrsnetze werden in verschiedenen Formaten unterstützt. Sie können in einem XML-, einem ASCII- oder einem speziellen Binärformat vorliegen. Das ASCII-Format orientiert sich an dem Format der Netzdateien der Software VISUM[1] der PTV AG und kann daher gut als Dateiaustauschformat genutzt werden. Die XML-Beschreibung von Verkehrsnetzen erlaubt eine gute Lesbarkeit der Strukturen durch den Benutzer. Außerdem stellt XML ein flexibles Format zum Austausch von Daten zur Verfügung. Beide Formate erkaufen diese Vorteile allerdings mit zwei schwerwiegenden Nachteilen gegenüber dem Binärformat. Letzteres erlaubt eine deutlich kompaktere, komprimiertere Speicherung der Daten. Das im Framework eingesetzte Binärformat reduziert die Dateigröße zum Teil deutlich. Bei den hier verwendeten Verkehrsnetzen verringert sich die Dateigröße im Binärformat um durchschnittlich 22% gegenüber den beiden textbasierten Formaten. Diese geringe Verkleinerung liegt darin begründet, dass dieses Binärformat bereits zur Aufnahme von Simulationsergebnissen vorbereitet ist. Außerdem können die Daten der Binärdateien schneller eingelesen und verarbeitet werden.

[1] `http://www.ptv.de/traffic/software-und-system-solutions/vissum/`

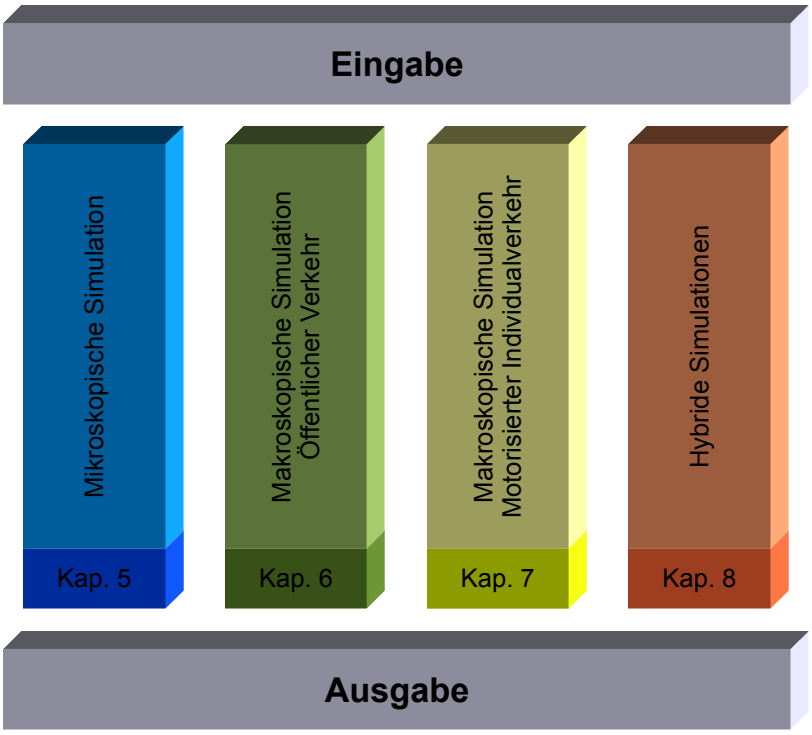


Abbildung 4.1: Überblick über das im Rahmen dieser Arbeit entstandene Framework zur Verkehrssimulation.

Die Verkehrsnachfrage in Gestalt von Quelle-Ziel-Matrizen kann über zwei unterschiedliche Formate eingelesen werden: ASCII oder XML. Das Framework stellt Datenstrukturen zur internen Repräsentation und Funktionen zum Einlesen dieser Daten zur Verfügung, die so von den verschiedenen Simulationsarten ohne Zusatzaufwand verwendet werden können.

4.2 Simulationsmodelle

Auf Grund der durch das Framework zur Verfügung gestellten Funktionalitäten ist es möglich, Simulationen durchzuführen, ohne sich Gedanken über die Ein- und Ausgabe der Daten machen zu müssen. Exemplarisch sind einige Simulationsmodelle vorhanden (siehe Abb. 4.1), die durch das Framework genutzt werden können. Dazu gehören eine *mikroskopische* und eine *makroskopische* Verkehrssimulation. Die mikroskopische Simulation basiert dabei auf dem Zellularautomatenmodell von Nagel und Schreckenberg [125] und wurde in dieser Arbeit um die Möglichkeiten der Simulation heterogenen Verkehrs erweitert. Dadurch unterstützt sie die Simulation unterschiedlicher Fahrzeugtypen (Pkw, Lkw, Motorräder, ...). Zusätzlich ist der motorisierte Öffentliche Verkehr in Form von Linienbussen in

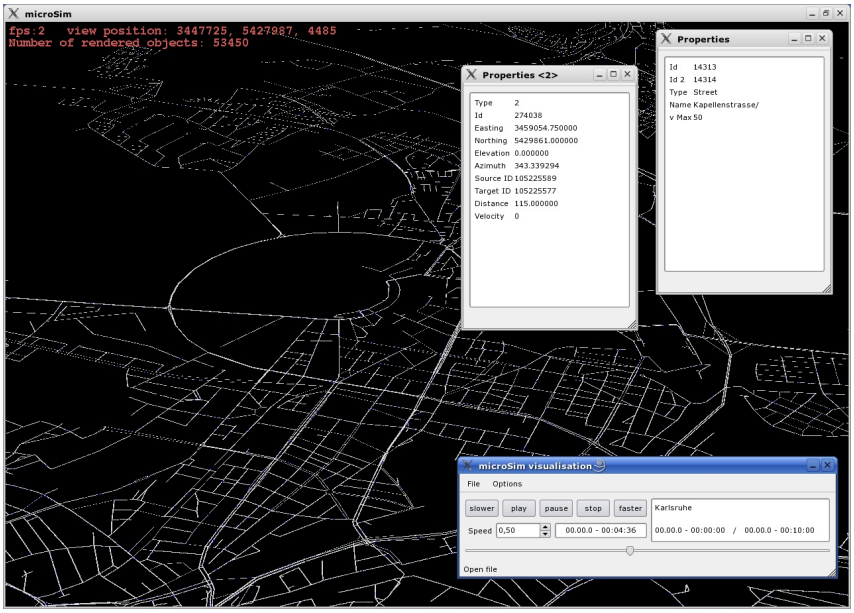


Abbildung 4.2: Bildschirmfoto des mikroskopischen Verkehrssimulators mit Grobansicht auf das Verkehrsnetz.

das System integriert. Eine parallele Variante dieser Simulation für nachrichtengekoppelte Systeme nutzt eine Partitionierung des Verkehrsnetzes mittels raumfüllender Kurven, um die Simulation beschleunigt zu berechnen.

Die makroskopische Simulation setzt sich sich aus zwei Komponenten zusammen: (1) Simulation des Öffentlichen Verkehrs mittels einer fahrplanfeinen Umlegung und (2) Simulation des motorisierten Individualverkehrs mittels einer stochastischen Umlegung. Für beide Teile werden Parallelisierungsstrategien für nachrichtengekoppelte Systeme unterstützt. Bei (1) erfolgt eine Parallelisierung durch Aufteilung der Nachfragedaten auf die beteiligten Rechenknoten. Dieser Ansatz wird auch bei (2) verfolgt, wobei dort zusätzlich eine Variante ohne repliziertes Verkehrsnetz existiert, welche vielmehr mit einem partitionierten Verkehrsnetz arbeitet.

Als Erweiterungsmöglichkeiten werden zwei hybride Simulationen aufgezeigt. Die erste ermöglicht eine Kopplung der mikroskopischen und der makroskopischen Simulation. Die zweite hybride Simulation zeigt die Verknüpfung einer einfachen Personensimulation mit der mikroskopischen Verkehrssimulation.

4.3 Ausgabe

Die Weiterverarbeitung der Simulationsergebnisse wird ebenfalls durch das Framework unterstützt. Grundsätzlich erfolgt zunächst eine Speicherung der Ergebnisse in Dateiform. So

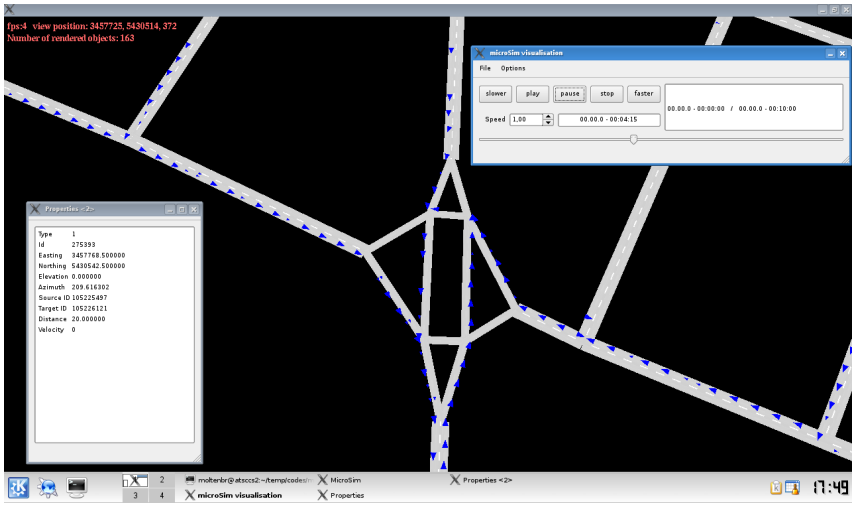


Abbildung 4.3: Visualisierung der mikroskopischen Verkehrssimulation in Detailansicht. Die Dreiecke stellen die Fahrzeuge dar.

wird bei der mikroskopischen Simulation eine *Trace*-Datei ablegt, die die Bewegung aller Fahrzeuge im Verkehrsnetz in jedem Zeitschritt protokolliert. Aus Gründen der Reduktion des Speicherbedarfs erfolgt dies in einem Binärformat. Bei der makroskopischen Simulationen werden die Umlegungsergebnisse, also die Belastungen der einzelnen Netzelemente eines Verkehrsnetzes, ebenfalls in einer Datei abgelegt.

Das Framework bringt zwei Werkzeuge zur Weiterverarbeitung dieser Daten mit. Das erste ist ein *Player*, der die Trace-Datei der Mikrosimulation einliest und diese abspielt. Es kann dabei zu beliebigen Zeitpunkten innerhalb der Simulation gesprungen und sämtliche Informationen aus dieser abgerufen werden. Zwei Beispiele sind in den Abb. 4.2 und 4.3 gegeben. Ein *Visualisierer* erlaubt das effiziente und schnelle Darstellen von großen Verkehrsnetzen und von Umlegungsergebnissen. Abb. 4.4 zeigt einen exemplarischen Einsatz des Visualisierers auf das Verkehrsnetz der Stadt New York.

4.4 Zusammenfassung

Das Framework stellt generische Ein- und Ausgaberoutinen zur Verfügung, die potentiell von jeder Art von Verkehrssimulation genutzt werden können. Das Framework ist dabei hinsichtlich der Eingabe- und Ausgabeformate und der verwendeten Simulationsmodelle flexibel erweiterbar und kann so leicht auf die jeweiligen Bedürfnisse des Nutzers angepasst werden. Auf diese Weise ist es für den Nutzer möglich, sich alleine auf die Entwicklung der internen Struktur und der Logik der Simulationsmodelle zu konzentrieren. Das Framework unterstützt den Nutzer anschließend bei der Weiterverarbeitung der Simulationsergebnisse.

Abbildung 4.4: Visualisierung eines Ausschnitts des Verkehrsnetzes von New York.

5 Mikroskopische Verkehrssimulation

Die mikroskopische Simulation des Verkehrs ist eine wichtige Domäne innerhalb der Verkehrssimulation. Daher nimmt sie auch einen besonderen Platz innerhalb des in dieser Arbeit entstandenen Frameworks zur Verkehrssimulation ein.

Im Rahmen dieser Arbeit wurde das Basismodell von Nagel und Schreckenberg um die Unterstützung heterogenen Verkehrs und des motorisierten Öffentlichen Verkehrs in Form von Linienbussen erweitert. Das Ziel ist, neben der Entwicklung eines einfachen Modells, auch die Erstellung eines hinreichend komplexen Systems zur mikroskopischen Verkehrssimulation, das sich für eine Parallelisierung eignet. Dies wird zu Beginn dieses Kapitels näher betrachtet. Die Behandlung beginnt zunächst mit einer Beschreibung des heterogenen Modells, das dann von der einstreifigen zur mehrstreifigen Verkehrsbehandlung erweitert wird. An diese Betrachtung des heterogenen Modells schließt sich die Modellerweiterung um den Öffentlichen Verkehr an, die sich das zuvor beschriebene heterogene Modell zu Nutzen macht. Abschließend werden die verschiedenen umgesetzten Strategien zur Parallelisierung des Simulationsablaufs diskutiert, die eine detaillierte Simulation des Straßenverkehrs auch auf sehr großen Verkehrsnetzen ermöglichen.

5.1 Ein erweitertes Zellularautomaten-Modell

Das Basismodell von Nagel und Schreckenberg, wie es in Kapitel 2.2 beschrieben wurde, ist in der Lage einfachen, einstreifigen Verkehr auf Autobahnen abzubilden. Für eine realistische Simulation des tatsächlichen Verkehrs ist es allerdings aus offensichtlichen Gründen nicht ausreichend. So fehlt beispielsweise im Basismodell die Betrachtung verschiedener Fahrzeugtypen und die Behandlung von Mehrstreifigkeit und Überholvorgängen.

5.1.1 Heterogener Verkehr

Realer Verkehr setzt sich aus einer Vielzahl unterschiedlichster Fahrzeugtypen und Verkehrsteilnehmer zusammen. Existiert nur ein Fahrzeugtyp beziehungsweise eine Art von Verkehrsteilnehmer wie etwa Pkw, so spricht man von *homogenem Verkehr*, in allen anderen Fällen von *heterogenem Verkehr*. Dessen Betrachtung ist für eine realistische Simulation des Straßenverkehrs unabdingbar. Wie in Kapitel 2.2 beschrieben, ist das klassische Zellularautomatenmodell von Nagel und Schreckenberg in seiner Grundform dazu nicht in der Lage.

In den letzten Jahren sind daher eine Reihe von Erweiterungen entstanden. Vor allem im asiatisch-indischen Raum entstanden dabei Arbeiten zu diesen Themen [8, 98], während dieses Thema in den westlichen Ländern lange Zeit keine sehr große Rolle spielte. Im asiatischen Raum ist die Bedeutung des heterogenen Verkehrs wichtiger als bei uns: Man betrachte nur die Anzahl an Zweirädern in Ballungszentren der Dritten Welt. Die Charakteristika dieser Modelle, wie die Nutzung der Fahrstreifen und das allgemeines Fahrverhalten, weichen deutlich von den Mustern in westlichen Ländern ab. Deshalb gab es dort eigene Entwicklungen von Modellen zur Simulation des heterogenen Verkehrs. In [42] wird beispielsweise ein Modell zur Simulation heterogenen Verkehrs vorgestellt. Die Autoren setzen dabei auf eine Reduktion der Standardzelllänge. Sie lassen dabei zwei Fahrzeuggrößen zu: Fahrzeuge, die eine Zelle belegen, und solche, die zwei Zellen belegen. Weitere Größen sind in diesem Modell nicht vorgesehen, was für eine Modellierung beliebigen heterogenen Verkehrs zu unflexibel erscheint. Mathew et al verfolgen einen ähnlichen Ansatz [114] und erweitern ihr Modell zusätzlich um die Behandlung von mehrstreifigem Verkehr.

Im Rahmen dieser Arbeit (siehe auch [117]) entstand eine flexible Erweiterung des Basismodells, die eine große Bandbreite an Einsatzmöglichkeiten unterstützt. So lassen sich damit beliebige Fahrzeugtypen simulieren und deren Interaktion feingranular einstellen. Außerdem bietet sie eine elegante Möglichkeit, neben dem motorisierten Individualverkehr (mIV), auch den motorisierten Öffentlichen Verkehr (mÖV) darzustellen. Diese Konzepte werden im folgenden Abschnitt erläutert.

Modellerweiterungen um heterogenen Verkehr

Heterogener Verkehr setzt sich aus einer Vielzahl unterschiedlichster Fahrzeugtypen zusammen. Jeder Typ für sich hat einen bestimmten eigenen Einfluss auf das Verkehrsverhalten und unterscheidet sich durch besondere Verhaltenscharateristika von den anderen Typen. Wir wollen uns mit einer Betrachtung von vier unterschiedlichen Typen begnügen: zum einen normale Pkw, die den Großteil der Verkehrsteilnehmer ausmachen; des Weiteren Lastwagen in verschiedenen Ausprägungen, deren Einfluss auf den Verkehrsfluss nicht zu vernachlässigen ist. Hinzu kommt die Betrachtung kleinerer, flexiblerer Verkehrsteilnehmer, wie beispielsweise Motorräder, die je nach geographischer Region eine große Rolle spielen und deren Verkehrsverhalten von dem anderer Verkehrsteilnehmer zum Teil deutlich differiert. Als letzter Typ sind Linienbusse zu nennen, die zwar von ihrer Größe manchen Lkw

ähneln, aber dennoch im Verhalten grundlegende Unterschiede aufweisen.

Die Grundidee hinter der hier vorgestellten Modellerweiterung ist eine zweidimensionale Erweiterung bzw. Verfeinerung des NaSch-Modells. Betrachtet man einen Streckenabschnitt mit nur *einem* Fahrstreifen, so wird dieser im Basismodell durch einen eindimensionalen zellulären Automaten diskretisiert, das heißt als Ergebnis erhält man eine Folge von Zellen, deren Länge im Basismodell jeweils 7,5 Meter beträgt. Dies hat allerdings zwei Nachteile im Hinblick auf die Modellierung heterogenen Verkehrs: (1) Zum einen entspricht – wie in Abschnitt 2.2.2 geschildert – die Länge von 7,5 Metern dem durchschnittlichen Platzbedarf eines normalen Pkw in Stausituation auf deutschen Autobahnen (Länge des Fahrzeugs plus Sicherheitsabstand). Andere Fahrzeugtypen benötigen allerdings mehr (beispielsweise Lkw und Linienbusse) oder erheblich weniger (beispielsweise Motorräder oder Fahrräder) als Pkw. Bei letzteren könnte man eine höhere Dichte erreichen, die dann der Realität eher entspricht. (2) Zum anderen benötigen kleinere Fahrzeuge nicht immer die volle Fahrstreifenbreite. Ein bekanntes Phänomen sind die häufig nebeneinander fahrenden Zweiräder jeder Ausprägung. Dieses Verhalten ist zwar gesetzlich nicht erlaubt, tritt aber in der Realität so häufig auf, dass man es für eine realistische Simulation in Betracht ziehen muss.

Das im Rahmen dieser Arbeit entstandene Modell wird nun dahingehend gegenüber dem Basismodell erweitert, dass zunächst die Länge einer Zelle verändert wird. Die Länge einer Zelle kann dabei eine beliebige Größe annehmen (beispielsweise zwischen $0,3$m und $7,5$m), wobei alle Zellen im gesamten Verkehrsnetz die selbe Länge besitzen müssen. Auf diese Weise lässt sich leicht und elegant Problem (1) beheben. Die Abstände zwischen Fahrzeugen können auf diese Weise flexibler und detaillierter eingestellt werden. Die Dichte bei erhöhtem Zweiradaufkommen kann gesteigert werden. Außerdem lassen sich mit dieser Methode eine Vielzahl unterschiedlicher Fahrzeugtypen modellieren. Je nach Typ werden eine oder mehrere Zellen benötigt. So kann beispielsweise ein Motorrad eine Zelle belegen, ein Pkw zwei und ein Bus oder Lkw dahingegen mehrere. In Abb. 5.1 ist die schematische Darstellung einer Fahrbahn mit zwei Fahrstreifen im Nagel-Schreckenberg-Modell gegeben, in dem eine Zelle immer durch maximal ein Fahrzeug belegt ist. Demgegenüber zeigt Abb. 5.2 dieselbe Fahrbahn nach der soeben beschriebenen Verfeinerung der Zellen. Hierbei belegt ein Fahrzeug mehr als eine Zelle.

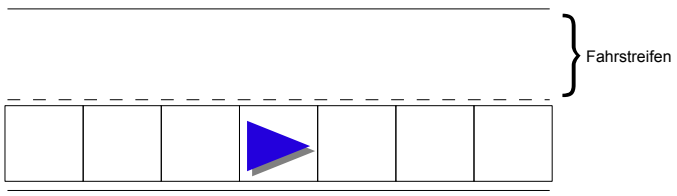


Abbildung 5.1: Darstellung einer Fahrbahn mit zwei Fahrstreifen im Basismodell von Nagel und Schreckenberg: Ein Fahrzeug belegt eine Zelle.

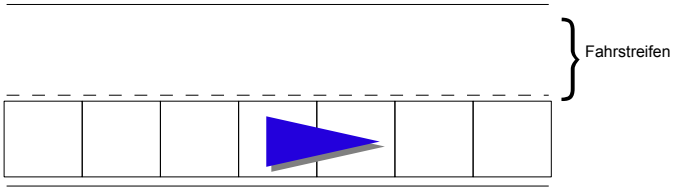


Abbildung 5.2: Darstellung einer Fahrbahn mit einem verfeinerten Zellmodell: Ein Fahrzeug kann nun mehr als eine Zelle belegen.

Will man allerdings auch Problem (2) lösen, so bedarf es zusätzlich einer Ergänzung des Basismodells um eine zweite Dimension. Hier soll das Verhalten von Zweiradfahrern modelliert werden, die stets dazu neigen – auch gegen entsprechende, verbindliche Verkehrsvorschriften – nebeneinander zu fahren. Ein gutes Beispiel hierfür sind Schüler auf ihrem morgendlichen Weg zur Schule, die durch ihr Verhalten oft unbewusst zu einem Verkehrshindernis werden.

Um in der Lage zu sein dieses Verhalten zu beschrieben, wird ein *Fahrstreifen* in eine bestimmte Anzahl von *Spuren* aufgespalten, die parallel zueinander verlaufen, wie in Abb. 5.3 dargestellt.

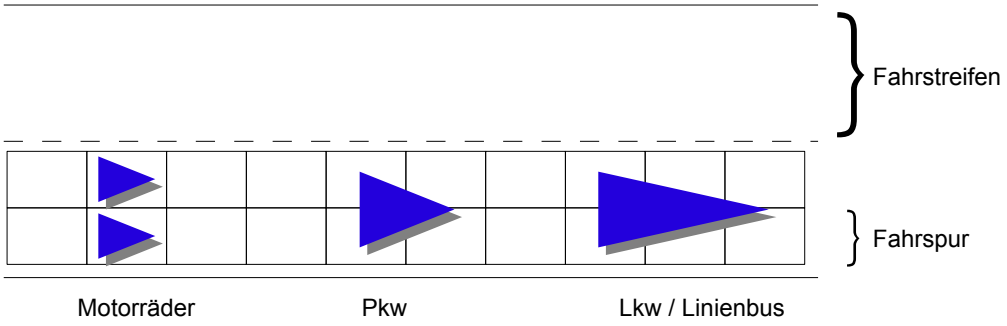


Abbildung 5.3: Darstellung der Fahrbahn aus Abb. 5.2 nach Aufspaltung eines Fahrstreifens in mehrere Spuren. Die unterschiedlichen Fahrzeugtypen sind zusätzlich mit möglichen Zellbelegungen dargestellt.

Die Anzahl der Fahrspuren innerhalb eines Fahrstreifens kann dabei beliebig variiert werden. Allerdings ist es aus äusseren Vorgaben (Fahrstreifenbreite, Fahrzeugbreite, ...) nur selten sinnvoll mehr als drei Fahrstreifen zu verwenden.

Um diese Modellerweiterung umzusetzen, ist es allerdings notwendig, Anpassungen an den Phasen des Basismodells von Nagel und Schreckenberg vorzunehmen. Die neuen Phasen gehen dabei aus den ursprünglichen hervor, indem sie um das Konzept der Mehrzelligkeit

und Mehrspurigkeit erweitert werden. Ein Fahrzeug muss immer auf allen Spuren, die es belegt, nach vorne schauen. Befindet sich auf einer der Spuren ein anderes Fahrzeug innerhalb des Sicherheitsabstands, so muss das Fahrzeug abgebremst werden. Belegt ein Fahrzeug mehr als eine Spur und in jeder der Spuren vor ihm befindet sich ein anderes Fahrzeug, so wird dasjenige Fahrzeug mit minimalem Abstand als Referenzvordermann ausgewählt. Hierfür sind Anpassungen an den ersten beiden Phasen des Basismodells notwendig. Die dritte und vierte Phase bleiben unverändert. Formal lässt sich dieses Verhalten wie folgt beschreiben.

Sei c_i eine Zelle auf Spur i und sei a ein Fahrzeug einer gewissen Breite (bedeckt also Spuren i bis j mit $j \geq i$). Seit v die Geschwindigkeit von a. Dann ergibt sich der Abstand zum nächsten vorausfahrenden Fahrzeug auf Spur i durch $d_i(a)$. Die maximal zulässige Geschwindigkeit auf dem Fahrstreifen ist wieder durch v_{max} gegeben. Dann gilt:

1. **Beschleunigen:** Wenn $(v < v_{max})$ und für alle Spuren t $(i \leq t \leq j)$ gilt: $d_t(a) > v+1$, dann ergibt sich die neue Geschwindigkeit von a: $v \rightarrow v+1$
2. **Bremsen:** Für jede Spur t, die das Fahrzeug a belegt, muss überprüft werden, ob die Bedingung $d_t(a) < v$ erfüllt ist. Ist dem so, dann ergibt sich: $v \rightarrow min(d_t(a); i \leq t \leq j)$
3. **Trödeln:** Mit einer gewissen Wahrscheinlichkeit p wird die Geschwindigkeit v des Fahrzeugs um $v \rightarrow v-1$ reduziert.
4. **Fortbewegung:** Jedes Fahrzeug wird gemäß seiner Geschwindigkeit v um entsprechend viele Zellen voran bewegt. Dabei ist jede Zelle zu beachten, die Fahrzeug a belegt.

Diese Modellerweiterung bringt allerdings zwei Nachteile mit sich. Ein Nachteil ist der erhöhte Rechenaufwand, der sich durch die Betrachtung des verfeinerten Zellmodells ergibt. Die Rechenzeit erhöht sich dabei durchschnittlich etwa um den Faktor 2. Ein weiterer Nachteil ist, dass man – anders als im Basismodell – schon jetzt mehrere Spuren betrachten muss, und will man hier ausreichend realistisch Verkehr simulieren, so bedarf es bereits hier einer Betrachtung von Überholvorgängen und Spur- beziehungsweise Streifenwechseln. Diese wurden im Basismodell noch nicht betrachtet. Im folgenden Abschnitt werden nun zunächst verschiedene, existierende Erweiterungen des NaSch-Modells diskutiert, bevor daran anschließend zu der hier erarbeiteten Modellanpassung für den heterogenen Verkehr übergegangen wird.

Erweiterung des Basismodells um Mehrstreifigkeit

Ein wichtiger Aspekt ist die Erweiterung des Basismodells um das Konzept der Mehrstreifigkeit: Viele moderne Straßen besitzen heute mehrstreifige Komponenten und nur dadurch wird eine Beschreibung und Behandlung von Überholvorgänge möglich.

Der wichtigste Aspekt bei der Integration von Mehrstreifigkeit in ein mikroskopisches Verkehrsflussmodell ist die Entwicklung einer Behandlungslogik von *Streifenwechseln*. Vie-

le existierende Modellerweiterungen für Streifenwechsel in mikroskopischen Simulationen bezogen sich lange Zeit allerdings nicht auf Modelle mit zellulären Automaten, auf denen in dieser Arbeit ein Fokus liegt. Vielmehr behandeln sie entsprechende Vorgänge in anderen Modellen, beispielsweise Follow-the-Leader-Modellen. So beschreiben Wiedemann [168] und Sparmann [156], der mit seiner Arbeit auf Wiedemann aufsetzt, entsprechende Modelle. Ebenso entwickelte Gipps basierend auf seinem Modell [70], ein Simulationsmodell für den Streifenwechsel [71]. Die wesentliche Grundidee hinter diesen Modellen ist es, die Entscheidungsfindung bei einem Streifenwechsel in einen *Wunsch* und eine *Entscheidung* aufzuteilen. Ein Wunsch einen Streifenwechsel durchzuführen entsteht dann, wenn sich vor dem „eigenen“ Fahrzeug ein anderes Fahrzeug befindet, das den weiteren Verkehrsfluss behindert. Dieser Wunsch reift schließlich zu einer Entscheidung, wenn genügend Raum (inklusive eines Sicherheitsabstands) auf dem anderen Fahrstreifen besteht. Es konnte jedoch gezeigt werden, unter anderem in [126], dass diese Modelle zum Teil deutliche Abweichungen von realem Verhalten aufweisen.

Für die weitere Betrachtung von Bedeutung sind vor allem Ansätze, die auf der Verwendung von zellulären Automaten beruhen. Erste entsprechende Vorschläge sind beispielsweise in [39] und [151] zu finden.

Cremer und Ludwig [39] entwickelten eine Modell zum Fahrstreifenwechsel. Ein Wechsel wird dann durchgeführt, wenn sich durch diesen eine Verbesserung in Reisezeit und Geschwindigkeit ergeben könnte. Der Wechsel ist allerdings nur erlaubt, wenn genügend freier Platz auf dem rechten Fahrstreifen vorhanden ist. Allerdings gibt es bei diesem Modell Probleme, alle relevanten Charakteristika eines realen Fahrstreifenwechsels abzubilden (vgl. [126]).

Nagel et al schlagen daher in [126] eine neue Modellerweiterung zum NaSch-Basismodell vor. Sie legen bei der Modellierung neben möglichst hohem Realismus einen Schwerpunkt auf die Aspekte *Sicherheit*, *Reisezeitminimierung* und *gesetzliche Vorgaben*. Sie schlagen zwei Varianten für die Modellierung des Fahrstreifenwechsels vor: Eine amerikanische und eine deutsche. Der Grund für diese beiden Varianten ist, dass die gesetzlichen Regelungen für den Fahrstreifenwechsel in Deutschland und den USA variieren. Im Folgenden erfolgt eine Konzentration auf die deutschen Regeln.

Auch in ihrem Modell ist das Ziel eine verkürzte Reisezeit zu erreichen. Um dies zu bewerkstelligen, signalisiert ein Fahrzeug den Wunsch nach einem Fahrstreifenwechsel, wenn sich ein langsameres Fahrzeug auf demselben Fahrstreifen voraus befindet. Durch dieses Fahrzeug wird die eigene Fahrt verlangsamt, was sich möglicherweise durch einen Fahrstreifenwechsel beheben lässt. Die Regeln zu einem Fahrstreifenwechsel sehen dabei, gemäß [126], folgendermaßen aus.

Nach deutschen Verkehrsregeln ist es nicht gestattet ein Fahrzeug rechts zu überholen. Dies bedeutet, dass man sich, falls sich ein langsameres Fahrzeug auf dem *linken Fahrstreifen* befindet, hinter diesem einordnen muss. Man führt daher einen Fahrstreifenwechsel vom rechten auf den linken Fahrstreifen durch, wenn sich ein langsamer fahrendes Fahrzeug auf dem linken oder dem eigenen Fahrstreifen befindet. Dieser Sachverhalt lässt sich durch

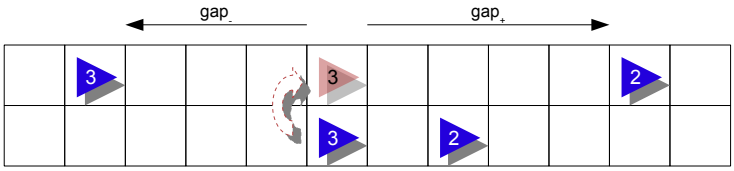


Abbildung 5.4: Durchführung eines Fahrstreifenwechsels nach Nagel et. al unter Einhaltung des Sicherheitskriteriums $[-gap_-, gap_+]$. Die Fahrzeuge sind mit ihren jeweiligen Geschwindigkeiten beschriftet.

folgende Regel fassen, wobei v_l und v_r die Geschwindigkeiten vorausfahrender Fahrzeuge auf dem linken bzw. rechten Fahrstreifen bezeichnen und v die eigene Geschwindigkeit.

$$v_r \leq v \text{ oder } v_l \leq v \tag{5.1}$$

Ein Rückwechsel vom linken auf den rechten Fahrstreifen sollte dann durchgeführt werden, wenn die Geschwindigkeiten der vorausfahrenden Fahrzeuge auf beiden Fahrstreifen genügend hoch sind. Damit ergeben sich die folgenden Regeln:

$$v_r > v \text{ und } v_l > v \tag{5.2}$$

In allen Fällen ist die Einhaltung eines Sicherheitsabstandes notwendig, was in Abb. 5.4 skizziert ist. Für die einzelnen Abstandswerte nach vorne gap_+ und nach hinten gap_- lassen sich eine Vielzahl von Werten annehmen. Es erwiesen sich allerdings die Werte $gap_- = v_{max}$ und $gap_+ = v$ in Tests als zweckmäßig. Damit ergibt sich für den Sicherheitsabstand folgender Bereich:

$$[-gap_-, gap_+] = [-v_{max}, v]. \tag{5.3}$$

Da sich die beiden Regeln (5.1) und (5.2) an den vorausfahrenden Fahrzeugen orientieren, ist es wichtig, welcher Bereich vorausschauend betrachtet werden soll. Der Vorschaubereich beschreibt dabei die Anzahl der Zellen d zwischen einem Fahrzeug und seinem vorausfahrenden Vorgänger. Wählt man beispielsweise d groß, so ergibt sich eine größere Tendenz auf den linken Fahrstreifen zu wechseln, obwohl der Abstand zum Vorgänger noch ausreichend groß wäre. Untersuchungen durch Nagel et al in [126] haben zu $d = 16$ als optimalen Wert bei einer Zelllänge von 7,5m geführt, um die realen Phänomene hinreichend genau reproduzieren zu können.

Mit diesen Regeln lässt sich gut und realistisch Verkehr auf zwei Fahrstreifen mit vielen seiner Facetten beschreiben. Allerdings können Konflikte bei mehr als zwei betrachteten Fahrstreifen auftreten (siehe Abb. 5.5). So kann es passieren, dass sich parallel fahrende Fahrzeuge, die sich auf dem ersten oder dritten Fahrstreifen befinden, auf die gleiche Zelle auf dem zweiten Fahrstreifen wechseln wollen. Eine Lösung für dieses Problem besteht beispielsweise darin, dass man die Zeitpunkte der Fahrstreifenwechsel beschränkt: in geraden Zeitschritten ist nur ein Wechsel nach rechts möglich und in ungeraden nach links [126]. Auch im Rahmen dieser Arbeit hat man sich für dieses Konzept entschieden.

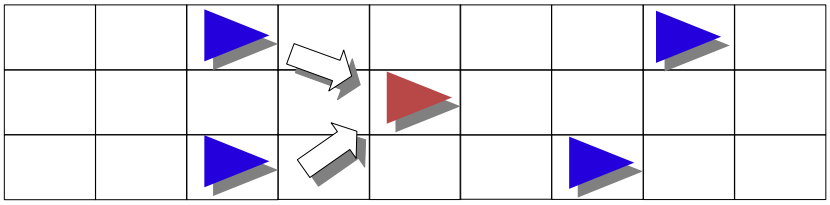


Abbildung 5.5: Konflikt bei mehr als zwei Fahrstreifen. Zwei Fahrzeuge versuchen auf dieselbe Zelle auf dem mittleren Streifen zu wechseln.

Das Konzept der Mehrstreifigkeit bei heterogenem Verkehr

Will man nun Mehrstreifigkeit bei heterogenem Verkehr betrachten, wie es in der im Rahmen dieser Arbeit entstandenen Modellerweiterung vorgestellt wurde, so bedarf es Anpassungen am eben betrachteten Modell [117]. Diese Anpassungen sind notwendig, da (1) bei verschiedenen Fahrzeugtypen unterschiedliche Verhaltensmuster auftreten und (2) da nun ein Fahrzeug mehr als eine Zelle belegen kann und dementsprechend einer komplizierteren, neuen Behandlung bedarf. Im heterogenen Fall können jetzt allerdings zwei unterschiedliche Arten von Wechseln auftreten. Zum einen ein Wechsel innerhalb eines Fahrstreifens zwischen zwei Fahrspuren und zum anderen zwischen zwei Fahrstreifen. Diese beiden Fälle gilt es genauer zu betrachten.

Die wichtigste Aufgabe ist auch hier das Auffinden der unmittelbaren Vorgänger desjenigen Fahrzeugs, welches einen Spurwechsel bzw. Fahrstreifenwechsel durchführen möchte. Im heterogenen Fall unseres Modells ist ebenso die Einhaltung eines Sicherheitskriteriums von Bedeutung, da auf diese Weise gewährleistet werden kann, dass kein anderes involviertes Fahrzeug zum Bremsen genötigt werden muss. Ein Spurwechsel entspricht einer einfachen Seitwärtsbewegung (siehe Abb. 5.6 und 5.7). Demzufolge muss in der nebenliegenden Fahrspur mindestens dieselbe Anzahl an Zellen frei sein wie in der aktuellen Ausgangsspur. Sei nun R die Menge an Zellenreihen (Spuren), die das Fahrzeug nach dem Wechsel für sich beansprucht. Ausgehend von den besetzten Zellen des Fahrzeugs gibt $\text{gap}_+(r)$ bzw. $\text{gap}_-(r)$ die Länge des Blocks der freien Zellen vor bzw. hinter dem Fahrzeug für jede Spur $r \in R$ an. Das Sicherheitskriterium $[\text{gap}_-, \text{gap}_+]$ definieren wir durch $\text{gap}_+ = \min_{r \in R} \text{gap}_+(r)$ und $\text{gap}_- = \min_{r \in R} \text{gap}_-(r)$. Für gap_- wird wie im vorherigen Abschnitt als Mindestwert v_{max} und für gap_+ die Geschwindigkeit v des Fahrzeugs angenommen.

Betrachtet man zunächst nur kleinere Fahrzeuge, also Fahrzeuge wie Motorräder, die lediglich eine Spur belegen (siehe Abb. 5.6), so lässt sich der Spurwechsel eines Motorrads f folgendermaßen beschreiben. Motorrad f wird nur dann auf eine andere Spur wechseln, wenn sich daraus eine Verbesserung in Bezug auf die Reisezeit oder eine Annäherung an die eigene Wunschgeschwindigkeit ergeben könnte. Daher überprüft f die Geschwindigkeiten der vorausfahrenden Fahrzeuge und zwar in der eigenen Spur und in der linken bzw. rechten Nachbarspur. Ist die Geschwindigkeit des in der eigenen Spur vorausfahrenden Fahrzeugs geringer (ein Abbremsen von f wäre folglich notwendig) und ist das Sicherheitskriteri-

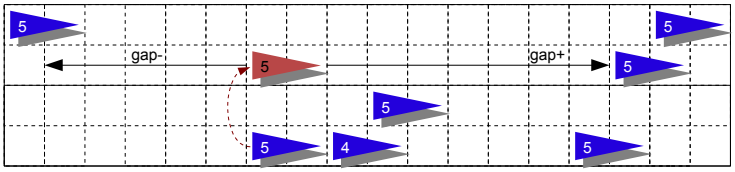


Abbildung 5.6: Spurwechsel bzw. Überholvorgänge im heterogenen Verkehrsflussmodell. Dargestellt sind Motorräder die mit ihren jeweiligen Geschwindigkeiten beschriftet sind, auf einer Fahrbahn aus zwei Fahrstreifen. Jeder Fahrstreifen ist dabei in zwei Spuren unterteilt. Ein Motorrad mit der Geschwindigkeit fünf wechselt auf eine linke Spure, da es durch ein weiteres Motorrad mit der Geschwindigkeit vier ausgebremst würde. Das rote Motorrad gibt dabei die Position nach dem Spurwechsel wieder.

um in den Nachbarspuren erfüllt, so erfolgt ein Wechsel von f auf eine linke oder rechte Spur. Ein Wechsel auf eine rechte Spur wird erlaubt, um realem Verhalten von Zweirädern Rechnung zu tragen. Durch einen Widerstandsfaktor wird sichergestellt, dass ein Wechsel auf eine linke Spur bevorzugt wird. Bei Zweirädern ist es nicht relevant zu unterscheiden, ob sich die Spur auf die gewechselt wird im eigenen oder einem anderen Fahrstreifen befindet. Abb. 5.6 skizziert den Spurwechsel eines Motorrads. Das Motorrad in der unteren Spur mit Geschwindigkeit 5 stellt fest, dass es seine Geschwindigkeit auf Grund eines mit Geschwindigkeit 4 vorausfahrenden Fahrzeugs verlangsamen müsste. Das Sicherheitskriterium in einer linken Spur ist erfüllt, so dass im nächsten Schritt ein Wechsel auf diese Spur erfolgen kann.

Eine Rückkehr auf die rechte Spur bzw. den rechten Fahrstreifen kann auf dieselbe Weise modelliert werden wie in [126] beschrieben: Wenn die Geschwindigkeit auf der alten Spur und die Geschwindigkeit des vorausfahrenden Fahrzeugs höher ist als die des Motorrads f, so erfolgt ein Wechsel zurück auf die alte Spur. Es ist notwendig diese Regelung aufgrund des Rechtsfahrgebots der deutschen Straßenverkehrsordnung zu modellieren.

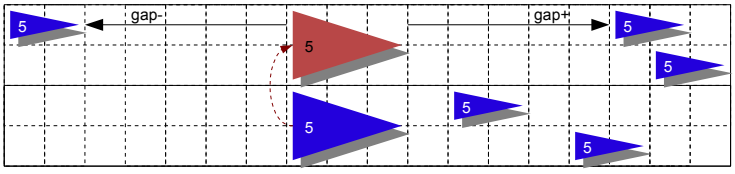


Abbildung 5.7: Spurwechsel eines Pkw. Für den Pkw mit Geschwindigkeit fünf besteht ein Anreiz zum Spurwechsel auf Grund des vorausfahrenden Zweirads mit Geschwindigkeit fünf. Andernfalls wäre ein Abbremsen notwendig. Da das Sicherheitskriterium erfüllt ist, findet ein Spurwechsel statt. Der rote Pkw gibt dabei die Position nach dem Wechsel wieder.

Analog dazu kann der Spurwechsel bzw. Fahrstreifenwechsel eines größeren Fahrzeugs beschrieben werden. Allerdings besteht hier eine zusätzliche Einschränkung: Es sind nur Wechsel auf einen anderen Fahrstreifen möglich, aber nicht auf zwei oder mehr Spuren verschiedener Fahrstreifen. Dies entspräche dem Fahren auf einer Fahrstreifenbegrenzung, was nach gesetzlichen Vorgaben nicht gestattet ist.

Damit lässt sich das Verhalten eines Wechsels von Fahrzeug a so beschreiben: Falls sich das vorausfahrende Fahrzeug langsamer bewegt als a, dann wechselt a auf die neue Spur, wenn diese nicht zu einem neuen Fahrstreifen gehört. Es kann – wie bereits erwähnt – nur eine Spurwechsel innerhalb eines Fahrstreifen oder ein Fahrstreifenwechsel zwischen zwei Fahrstreifen erfolgen. Das Verhalten bei einem Fahrstreifenwechsel entspricht dem aus [126].

Abb. 5.8 zeigt zwei Verkehrssituationen an derselben Kreuzung in der Stadt Karlsruhe zu verschiedenen Tageszeiten. In beiden Fällen wird heterogener Verkehr betrachtet; gut sind dabei die unterschiedlichen Fahrzeugtypen sichtbar. Oben ist die Verkehrssituation in entspannter Lage zu erkennen, während unten ein höheres Verkehrsaufkommen existiert.

5.1.2 Öffentlicher Verkehr

Ein weiterer wichtiger Aspekt des täglichen Verkehrsgeschehens (insbesondere in städtischen Ballungszentren) ist der *Öffentliche Verkehr* (ÖV). Hierunter versteht man sowohl den schienenbasierten Verkehr (Züge, Straßenbahnen, U-Bahnen, ...) als auch den motorisierten Öffentlichen Verkehr (Linienbusse, ...). Dieser ÖV weist einige Besonderheiten im Vergleich zum bisher betrachteten motorisierten Individualverkehr auf.

Linienbusse können in einer ersten Näherung wie normale Fahrzeuge aus den vorhergehenden Abschnitten betrachtet werden. Allerdings fahren sie und vergleichbare Verkehrsmittel stets nach Fahrplänen und bleiben bei ihren Fahrten auf festgelegten Routen. Eine Alternativroutenwahl findet im Allgemeinen nicht statt. Des Weiteren haben sie Passagiere und damit Haltestellen zu bedienen. Dies beeinflusst ihren Verkehrsfluss zum Teil deutlich, und durch das Stoppen an Haltestellen wirken sie auch auf den normalen Fluss des restlichen Verkehrs ein.

Im Folgenden wird ein Modell vorgestellt, das eine fahrplanfeine Modellierung von Linienbussen in das erweiterte Zellularautomatenmodell der mikroskopischen Simulation integriert. Fahrplanfein bedeutet in diesem Zusammenhang, dass sich die beteiligten Fahrzeuge an detaillierte Fahrpläne halten müssen. Es wird sich zeigen, dass sich Linienbusse nahtlos in das bestehende Modell des heterogenen Verkehrs integrieren lassen und sich gut deren Fahrpläne und Verhalten beschreiben lassen.

Mit der Integration des Öffentlichen Verkehrs kann man nicht nur den Realitätsgrad einer bestehenden mikroskopischen Simulation erhöhen, sondern das im Rahmen dieser Arbeit vorgestellte Modell gestattet es auch, Fahrpläne zu evaluieren und im Endeffekt quasioptimale Pläne zu erhalten.

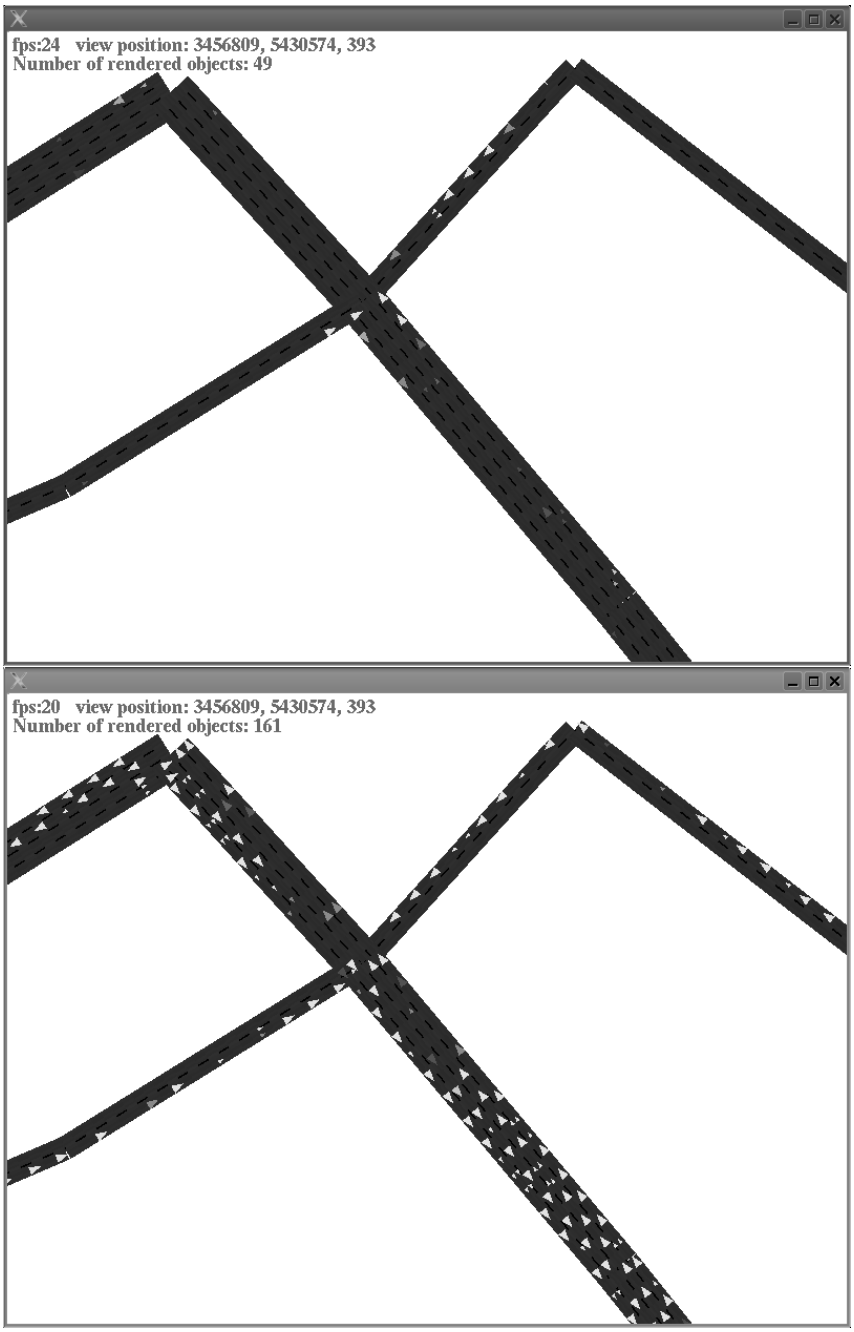


Abbildung 5.8: Mikroskopische Simulation heterogenen Verkehrs in ruhiger Verkehrslage (oben) und in Rush-Hour (unten).

Als einen ersten Schritt müssen jedoch die verschiedenen Teilkomponenten definiert werden, die für eine Integration des ÖV unablässig sind. Dazu zählt neben der Beschreibung der Linienbusse als neuer Fahrzeugtyp in unserem mikroskopischen Modell auch die Modellierung und Festlegung von Busrouten und natürlich das Verhalten an Bushaltestellen. Letzteres erweist sich als nicht ganz einfaches Problem, da hier weitere menschliche Verhaltenskomponenten ins Spiel kommen. Dazu zählt das Verhalten ein- und aussteigender Passagiere.

Modellierung des ÖV-Netzes

Linienbusse und andere öffentliche Verkehrsmittel bewegen sich im Verkehrsnetz auf bestimmten Routen. Man spricht hier auch von *Buslinien*. Diese Linien sind Bestandteil des ÖV-Netzes, welches ein Teil des Gesamtverkehrsnetzes ist. Das ÖV-Netz setzt sich wiederum aus *Bushaltestellen* zusammen, die über Strecken miteinander verbunden sind. Das ÖV-Verkehrsnetz ist eine einfache Überlagerung des allgemeinen Verkehrsnetzes. Hierzu werden einzelne Knoten im Verkehrsnetz um die Eigenschaften einer Bushaltestelle erweitert. Diese Knoten beziehungsweise Haltestellen dienen dann auch als Schnittstelle zu einer kombinierten Verkehrs- und Personensimulation, wie sie in Kapitel 8 vorgestellt wird.

Abb. 5.9 zeigt exemplarisch die Gestalt einer einfachen Buslinie. Dabei handelt es sich um einen gerichteten Pfad von einer *Starthaltestelle* über mehrere weitere *Haltestellen* bis hin zu einer *Ziel-* oder *Endhaltestelle*. Den Pfad zwischen zwei aufeinander folgenden Haltestellen (beispielsweise den Haltestellen *A* und *B*) wird als *Sub-Linie* bezeichnet.

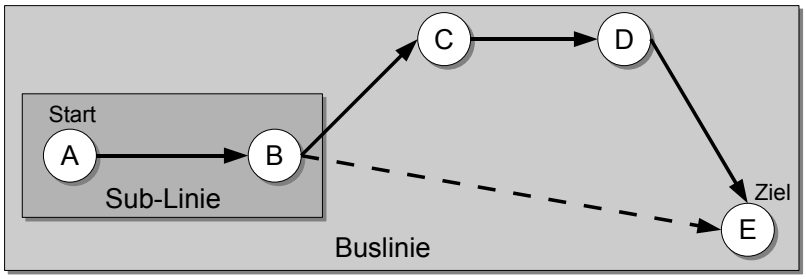
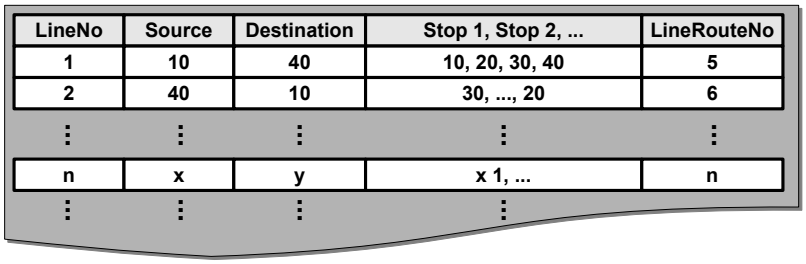


Abbildung 5.9: Struktur einer Buslinie mit Unterteilung in Sub-Linien und einer Alternativstrecke (gestrichelt). Die Buslinie besteht aus fünf Haltestellen (A – E). Exemplarisch ist eine Sub-Linie (A – B) hervorgehoben.

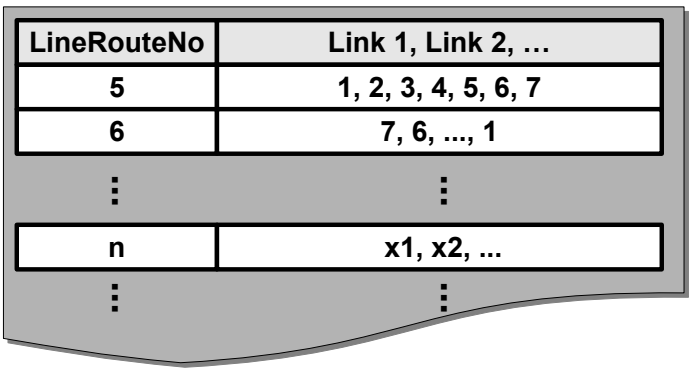
Haltestellen sind bestimmte Punkte im Verkehrsnetz, die unter anderem als Schnittstelle für eine Personensimulation dienen. Sie sind im Verkehrsnetz der Beginn beziehungsweise das Ende einer Linie oder einer Sub-Linie. Auf einer Linie befinden sich meist noch zahlreiche weitere Knoten, die allerdings nicht als Haltestellen ausgewiesen sind. Sie dienen der klassischen Beschreibung eines Straßenverlaufs und beschreiben markante Punkte im Verkehrsnetz, wie Kreuzungen etc.



LineNo	Source	Destination	Stop 1, Stop 2, ...	LineRouteNo
1	10	40	10, 20, 30, 40	5
2	40	10	30, ..., 20	6
⋮	⋮	⋮	⋮	⋮
n	x	y	x 1, ...	n
⋮	⋮	⋮	⋮	

Abbildung 5.10: Beispieltabelle einer Buslinie. Es wird dabei exemplarisch eine vollständige Buslinie (`LineNo 1`) vorgestellt, die bei Knoten 10 beginnt und bei Knoten 40 endet. Diese Linie ist auch in Abb. 5.12 veranschaulicht.

Diesem Umstand wird durch die Unterscheidung von *Buslinien* und *Linienroutenverläufen* Rechnung getragen. Es besteht dabei eine Verknüpfung zwischen diesen beiden Strukturen (Tabellen) über die so genannte *Linienroutennummer*. Diese Unterscheidung zwischen Linienbeschreibung und Linienroutenverlauf ist an [135] angelehnt.



LineRouteNo	Link 1, Link 2, …
5	1, 2, 3, 4, 5, 6, 7
6	7, 6, ..., 1
⋮	⋮
n	x1, x2, ...
⋮	⋮

Abbildung 5.11: Beispieltabelle eines Linieroutenverlaufs. Gezeigt wird exemplarisch der Verlauf einer vollständigen Buslinie (`LineRouteNo 5`). Dieser Verlauf ergibt sich über die Strecken mit den Nummern 1 bis 7.

Buslinien beschreiben dabei die Folge von Haltestellen und damit die im Fahrplan gegebene Struktur. Ein Beispiel für eine Buslinie ist in Abb. 5.10 auszugsweise dargestellt. Darin sind die Haltestellen durch ihre jeweiligen Knotennummern identifiziert. Demhingegen beschreiben Linienroutenverläufe den tatsächlichen Verlauf einer Buslinie, indem alle befahrenen Strecken (engl. Links) aufgenommen werden. Exemplarisch wird dies für die eben vorgestellte Buslinie in Abb. 5.11 illustriert.

Fasst man diese beiden Informationen zusammen, erhält man eine vollständige Beschreibung der Buslinie und ihres Verlaufs, ohne jedoch bisher einen Fahrplan einzusetzen. Abb. 5.12 zeigt das durch die vorhergehenden Tabellen beschriebene Beispielnetz. Die Buslinie

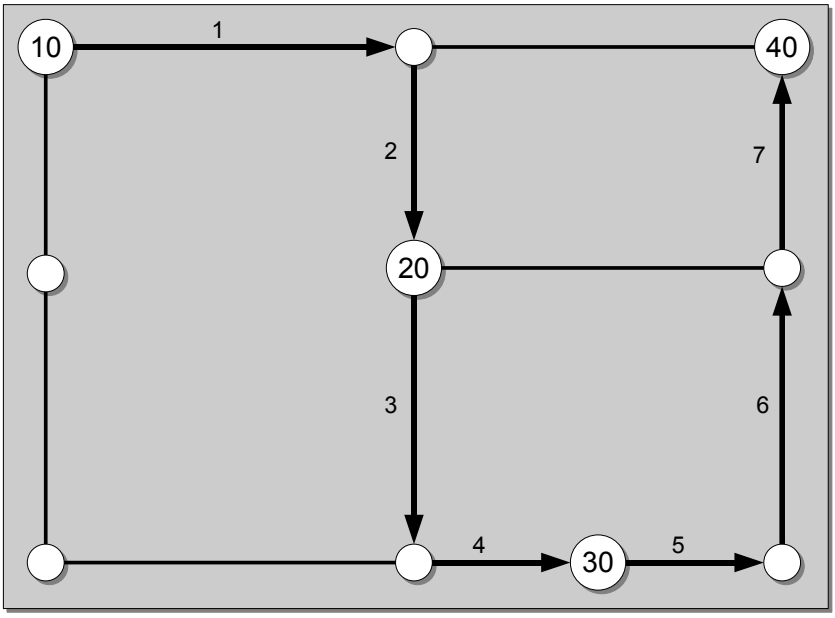


Abbildung 5.12: Einfaches Verkehrsnetz mit vier Haltestellen (10, 20, 30, 40) und einer Buslinie, die sich über die Strecken mit den Nummern 1 bis 7 zusammensetzt.

setzt sich aus vier Haltestellen (10, 20, 30, 40) zusammen. Die Buslinie selbst verläuft dabei über die Strecken 1 bis 7. Dargestellt ist durch Verwendung von Pfeilen die Hinrichtung der Buslinie (Linienroutenverlauf 5). Hin- und Rückrichtung müssen immer getrennt modelliert werden. Dies trägt dem Umstand Rechnung, dass ein und dieselbe Buslinie für die beiden Richtungen andere Strecken wählen kann und auch im allgemeinen Verkehrsnetz, welches als gerichteter Graph vorliegt, stets Hin- und Rückrichtung jeder Strecke getrennt sind.

Modellierung von Linienbussen und Aktivitäten

Das vorgestellte Modell des heterogenen Verkehrs mit seiner Beschreibung durch zweidimensionale zelluläre Automaten und die daraus gewonnene Flexibilität in der Erstellung verschiedener Fahrzeugtypen, erleichtert auch die Definition von Linienbussen.

Ein *Linienbus* kann in diesem Modell zunächst wie ein „normales“, größeres Fahrzeug (etwa wie ein Lkw) aufgefasst werden, das heißt es belegt mehrere Spuren und in jeder Spur nochmals mehrere Zellen. Eine Übersicht über mögliche Fahrzeugtypen einschließlich Linienbusse ist in Abb. 5.13 skizziert.

Neben der strukturellen Beschreibung eines Linienbusses (Anzahl und Art der belegten Zellen), bedarf es allerdings auch noch einer angepassten Verhaltensbeschreibung. Anders als bei den übrigen Fahrzeugtypen werden in unserem Modell keine Spur- und Fahrstreifen-

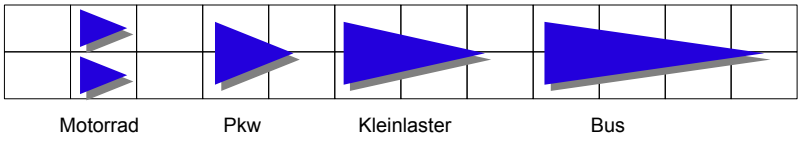


Abbildung 5.13: Übersicht verschiedener Fahrzeugtypen im Modell des heterogenen Verkehrs.

Q \ Z	A	B	C	...
A	0	40	20	...
B	40	0	30	...
C	10	20	0	...
⋮	⋮	⋮		

Abbildung 5.14: Auszug aus einer QZ-Matrix für Passagierströme im Öffentlichen Verkehr. Ein Eintrag gibt dabei an wieviele Personen sich von der Quell-Haltestelle (Q) zur Zielhaltestelle (Z) innerhalb des Simulationszeitraums bewegen. Beispielsweise fahren innerhalb des betrachteten Zeitraums 30 Personen von der Haltestelle *B* zur Haltestelle *C*.

wechsel zugelassen. Damit sind keine Überholvorgänge von Linienbussen möglich. Demzufolge wird eine Benutzung des rechten Fahrstreifens erzwungen. Des Weiteren existiert noch eine Einschränkung ihrer Maximalgeschwindigkeit. Bisher ist die Maximalgeschwindigkeit eine Eigenschaft der befahrenen Straße, was der zulässigen Höchstgeschwindigkeit auf dem entsprechenden Streckenabschnitt entspricht. Für Linienbusse wird nun eine davon abweichende, niedrigere Maximalgeschwindigkeit erzwungen, um damit verschiedenen gesetzlichen Vorschriften Geltung zu verschaffen.

Zusätzlich zu diesen Eigenschaften besitzen Linienbusse in diesem Modell eine Kapazität q. Diese beschreibt das Fassungsvermögen eines Linienbusses, also die Anzahl von Personen, die in ihm Platz finden können. Die mikroskopische Simulation unterstützt dabei zwei Arten der Behandlung von Haltestellen: (1) ohne angeschlossene Personensimulation und (2) mit Personensimulation.

Für den ersten Fall ohne Personensimulation werden die Passagiere der Linienbusse über eine Abwandlung des QZ-Matrizen-Konzepts in die Simulation eingeführt (siehe Abschnitt 2.1.5). Das allgemeine Vorgehen bei der Verwendung dieser Matrizen sieht dabei folgendermassen aus: Zunächst existieren allgemeine Matrizen, die Auskunft darüber geben, wieviele Passagiere von Haltestelle x zu Haltestelle y innerhalb des untersuchten Zeitraums fahren, was in Abb. 5.14 angedeutet ist.

Bus Stop	Time	Passengers	Lower Bound (dep. pass.)	Upper Bound (dep. pass.)	Lower Bound (arr. pass.)	Upper Bound (arr. pass.)
10	9	40	4%	9%	6%	7%
20	9	23	5%	7%	7%	7%
40	9	20	2%	6%	5%	9%
⋮	⋮	⋮	⋮	⋮		

Abbildung 5.15: Auszug aus einer Quelle-Ziel-Matrix für Passagierströme im Öffentlichen Verkehr.

Diese Matrizen werden für die Simulation verfeinert. Dabei wird für jede Haltestelle angegeben, wann innerhalb des Simulationszeitraums wie viele Passagiere warten und in den Linienbus einsteigen wollen. Außerdem wird an jeder Haltestelle ein Prozentsatz ermittelt, wie viele Passagiere an Bord eines Linienbusses aussteigen möchten. Diese Informationen werden anschließend zur Berechnung der Verweildauer des Linienbusses herangezogen. Dies ist exemplarisch in Abb. 5.15 festgehalten.

Für Fall (2) existieren Warteschlangen in jedem Haltestellenobjekt, die jeweils eine gewisse Kapazität besitzen. Sie dienen dabei als Austauschschnittstelle zwischen der Verkehrs- und der Personensimulation. Solange die Warteschlange ihre Kapazität nicht überschritten haben, können Personen aus der Personensimulation eingereiht werden. Aussteigende Passagiere werden über diese Warteschlange an die Umgebung (beispielsweise Fußweg) abgegeben.

Modellierung von Fahrplänen

Linienbusse und andere öffentliche Verkehrsmittel orientieren sich bei Ihren Fahrten an Fahrplänen. Diese gilt es nun für die Simulation zu beschreiben. Bei der Modellierung sind einige offensichtliche Punkte zu beachten: Der wichtigste ist dabei sicherlich, dass die in einem Fahrplan angegebenen Abfahrtszeiten erreichbar sein müssen.

TimetableNo	LineNo	Dep 1, Dep 2, ...	Traveltime 1, Traveltime 2, ...
1	1	09:00, 09:10, ..., 17:15	5 min, 10 min, ..., 8 min
2	2	09:15, 09:45, ..., 17:00	8 min, 10 min, ..., 5 min
⋮	⋮	⋮	⋮
n	x	d 1, d 2, ...	t 1, t 2, ...
⋮	⋮	⋮	⋮

Abbildung 5.16: Beispieltabelle eines Fahrplans.

Die Struktur eines Fahrplans ist in Abb. 5.16 skizziert. Jeder Buslinie ist dabei eine einzelne

„Zeile“ der Tabelle zugeordnet. Die Liniennummer stellt dabei die Beziehung zur aktuellen Buslinie her. Die Parameter *Departure Time* $Dep_1, \ldots, Dep_n$ geben die Abfahrtszeiten an den entsprechenden Haltestellen wider. Da Busse ggf. auch Wartezeiten an Haltestellen besitzen, ist die avisierte Fahrzeit durch die Parameter $Traveltime_1, \ldots, Traveltime_n$ gegeben. Dabei entspricht $Traveltime_1$ der Fahrzeit zwischen Haltestelle 1 und 2.

Verhalten an Bushaltestellen

Es ist wichtig das Verhalten von Linienbussen an Haltestellen zu beschreiben. Es beeinflusst die Fahrzeiten eines Linienbusses deutlich. An Haltestellen müssen Linienbusse warten bis Passagiere ein- und ausgestiegen sind. Dies kann zu variablen Wartezeiten führen, die den weiteren Verlauf der Route und die Einhaltung des Fahrplans beeinflussen. Diese *Verweilzeit* (engl. *dwell time*) gilt es zu modellieren. Es existieren dazu eine Vielzahl von Modellen – von sehr einfachen bis hin zu komplexen. Ein detaillierter Überblick ist in [119] zu finden.

Die einfachste Möglichkeit, die Verweilzeit zu beschreiben, besteht darin, sie als eine lineare Funktion der ein- und aussteigenden Passagiere zu modellieren:

$$T = \alpha + \beta \cdot N. \tag{5.4}$$

In (5.4) stellt N die Summe aller ein- und aussteigenden Passagiere an einer Haltestelle dar. Der Parameter α beschreibt die so genannte *Totzeit* eines Linienbusses: die Zeit, die ein Linienbus benötigt, an der Haltestelle zu stoppen, die Türen zu öffnen und nach dem Halt wieder anzufahren. Die durchschnittliche Einstiegs- bzw. Ausstiegszeit eines einzelnen Passagiers ist durch β gegeben. Levinson ermittelte für diese Parameter α und β die Werte $5,0$ beziehungsweise $2,75$ [108].

Diese lineare Beschreibung des Verhaltens ist aus offensichtlichen Gründen nicht ausreichend. Empirische Untersuchungen zeigen, dass bei größeren Menschenansammlungen an einer Haltestelle die Verweildauer des Linienbusses an sich steigt, aber die durchschnittliche Verweildauer eines einzelnen Passagiers sinkt. Guenthner und Sinha tragen dem in [75] durch nachfolgende nicht-lineare Beschreibung der Verweilzeit Rechnung.

$$T = 5,0 - 1,2 \cdot \ln(N). \tag{5.5}$$

Guenthner und Sinha vermuteten darüber hinaus, dass die Verweilzeit wohl nicht nur von den ein- und aussteigenden Passagieren abhängt, sondern von weiteren Faktoren (Anzahl der Türen, Tarifsystem, Tarifbezahlung, ...) beeinflusst wird. Letzteren Punkt – die Bezahlung des Fahrpreises – griffen Vandebona und Richardson [162] auf und entwickelten verschiedene Strategien für die Modellierung verschiedenen Bezahlverhaltens und deren Auswirkung auf die Verweildauer. Exemplarisch sind hier die folgenden vier Strategien genannt.

In der ersten einfachen Strategie, dem so genannten *sequentiellen Modell*, werden die Ein- und Ausstiegszeiten der Passagieren getrennt betrachtet und ihr Einfluss auf die Gesamtverweildauer des Linienbusses an der Haltestelle wird unterschiedlich gewichtet. Seien γ

die Totzeit des Linienbusses, α die durchschnittliche Ausstiegszeit eines Passagiers und β die durchschnittliche Einstiegszeit, sowie A und B die Anzahl der aus- bzw. einsteigenden Passagiere, dann ergibt sich die Verweildauer gemäß (5.6).

$$T = \gamma + \alpha \cdot A + \beta \cdot B. \tag{5.6}$$

Das *Interaktionsmodell* ist eine einfache Erweiterung des sequentiellen Modells. Der Berechnung der Verweilzeit aus (5.6) wird lediglich der Term $f(AB)$ hinzugefügt, der die Interaktion zwischen ein- und aussteigenden Passagieren beschreibt.

$$T = \gamma + \alpha \cdot A + \beta \cdot B + f(AB). \tag{5.7}$$

Das *Simultanmodell* hingegen verfolgt einen anderen Ansatz. Hier spielt die Anzahl der Türen für das Ein- und Aussteigen eine zentrale Rolle. Die Parameter λ_A und λ_B entsprechen den Totzeiten der Ausstiegs- beziehungsweise Eingangstüren. Dann ergibt sich die Verweilzeit T nach:

$$T = max \begin{pmatrix} \lambda_A + \alpha \cdot A \\ \lambda_B + \beta \cdot B \end{pmatrix}. \tag{5.8}$$

Diese Strategie ist allerdings nur dann einsetzbar, wenn in den Linienbussen bestimmte Gruppen von Türen lediglich für das Ein- bzw. andere für das Aussteigen eingesetzt werden. Das letzte Modell, dass wir betrachten, das so genannte *Multi-Rate-Boarding Modell* ist in seiner Funktionsweise ausgefeilter. Es erlaubt die Einsteigrate mit der Anzahl der einsteigenden Passagiere zu variieren. Mit seiner Hilfe konnten die Beobachtungen von Guenthner und Sinha aus [75] recht gut beschrieben werden.

$$T = \begin{cases} \gamma + \beta_1 \cdot B_i & ; 0 < B_i < x \\ \gamma + \beta_1 \cdot B_i + \beta_2 (B_i - x) & ; x < B_i \end{cases} \tag{5.9}$$

In (5.9) entspricht β_1 der Einsteigrate für B_i Passagiere, wenn die Anzahl der einsteigenden Personen kleiner als x ist. Entsprechend stellt β_2 die Einsteigrate bei mehr als x Personen dar.

Der im Rahmen dieser Arbeit entstandene Verkehrssimulator erlaubt einen flexiblen Austausch und das Hinzufügen der Berechnung der Verweilzeiten an Bushaltestellen. Alle vorgestellten Möglichkeiten sind integriert und lassen sich nach Belieben mit der Simulation verknüpfen.

Einsatz zur Fahrplanoptimierung

Das beschriebene und im Rahmen dieser Arbeit entwickelte Vorgehen bei der Simulation von Linienbussen lässt sich einerseits zum Simulieren des allgemeinen Straßenverkehrs nutzen. Andererseits eröffnet sich damit eine elegante Möglichkeit zur Fahrplanoptimierung beziehungsweise -erstellung.

Oft stellt sich die Frage, inwieweit Fahrpläne, die ursprünglich entworfen wurden, adaptiert werden müssen, um den aktuellen Verkehrsbelastungen und -situationen gerecht zu werden. Im Laufe der Zeit kann es passieren, dass das Verkehrsaufkommen auf den Straßen steigt beziehungsweise durch Baustellen oder andere Maßnahmen Engpässe im Verkehrsfluss entstehen. Dies führt dann häufig dazu, dass bestehende Fahrpläne nicht mehr eingehalten werden können.

Mit Hilfe einer mikroskopischen Verkehrssimulation lassen sich solche Problemfälle erkennen und Lösungsstrategien erarbeiten. Das im Rahmen dieser Arbeit entstandene Verfahren, welches auf der Verwendung des vorgestellten heterogenen Modells mit integriertem Öffentlichen Verkehr basiert, verfolgt dabei folgende Strategie: Ausgehend von Initialfahrplänen wird die Simulation schrittweise durchgeführt, und dabei werden in jeder Iteration die Pläne aktualisiert, die dann schließlich in den meisten Fällen konvergieren. Das grundlegende Vorgehen ist in Abb. 5.17 dargestellt.

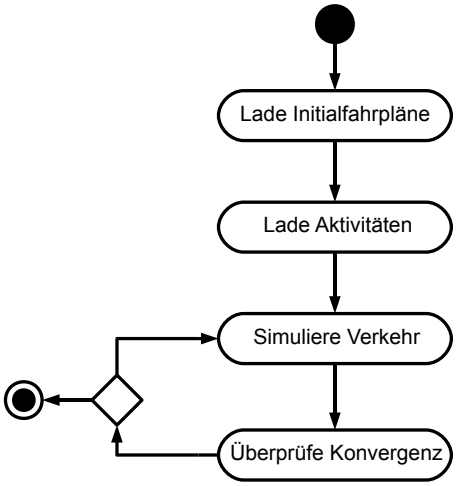


Abbildung 5.17: Ablauf der Fahrplanoptimierung: Ausgehend von Initialfahrplänen basierend auf einem unbelasteten Verkehrsnetz werden iterativ aktualisierte Fahrpläne ermitteln, die ein belastetes Netz betrachten.

Die *Initialfahrpläne* können dabei entweder von außen definiert und eingelesen werden oder automatisch bestimmt werden. Für letzteres erfolgt zunächst eine einfache Analyse einer Buslinie. Es werden die entsprechende Sub-Linien mit ihren Streckenabschnitten betrachtet. Für die Reisezeit des Linienbusses wird nun die Fahrt bei freiem Fluss betrachtet. Jede Strecke i mit Länge s_i besitzt eine Freiflussgeschwindigkeit v_i, welche der Maximalgeschwindigkeit entspricht. Wenn LR die Menge der Strecken einer Linienroute ist, so lässt sich die Gesamtreisezeit t des Linienbusses aus den Einzelreisezeiten t_i zwischen je zwei Haltestellen ermitteln zu:

$$t = \sum_{i \in LR} t_i = \sum_{i \in LR} \frac{s_i}{v_i} \tag{5.10}$$

Diese Pläne bedürfen natürlich noch deutlicher Korrekturen, da sie von einem leeren, unbelasteten Verkehrsnetz ausgehen. In einem ersten Iterationsschritt der Simulation wird die Mikrosimulation durchgeführt. Dabei werden auch die anderen Verkehrsteilnehmer (Pkw, Lkw, ...) betrachtet, die in diesem Schritt die Linienbusse in ihrer Fahrt beeinflussen. Dadurch werden unter Umständen die vormals berechneten Fahrzeiten (aus dem unbelasteten Netz) nicht eingehalten werden können. Es kommt zu Verspätungen, die sich als Unterschied zwischen der tatsächlichen und der geplanten Ankunftszeit ergeben.

$$\Delta = |t_{real} - t_{planned}| \tag{5.11}$$

Am Ende jedes Iterationsschritts wird nun überprüft, ob die Abweichungen jeder Buslinie ein gewisses Konvergenzkriterium α erfüllen. Dieses ist eine Schranke, die eine gewisse Ungenauigkeit zulässt. Gilt für die Verspätungen $\Delta < \alpha$, so ist der betrachtete Fahrplan genau genug und bedarf keiner weiteren Anpassung. Hinzu kommt ein globales Konvergenzkriterium Ψ, das alle Buslinien B im Netz betrachtet:

$$\Psi = \sum_{i \in B} \alpha_i \cdot \Delta_i \tag{5.12}$$

Dabei entspricht α_i einem Gewichtungsfaktor mit $0 < \alpha \leq 1$.

Wird das Konvergenzkriterium nicht erfüllt, so wird ein neuer Iterationsschritt der Simulation durchgeführt. Für diesen werden die bestehenden Fahrpläne mit den im aktuellen Schritt berechneten Zeiten aktualisiert. Im nächsten Schritt wird die vollständige Simulation erneut durchgeführt. Der einzige Unterschied besteht in der Verwendung der aktualisierten Fahrpläne.

5.1.3 Zusammenfassung

In diesem Abschnitt wurde ein Modell des heterogenen Verkehrs vorgestellt, welches im Rahmen dieser Arbeit entwickelt wurde. Es wurde gezeigt, wie sich beliebige Fahrzeugtypen durch dieses Modell beschreiben lassen und wie das Konzept der Mehrstreifigkeit (wie es aus der Literatur entnommen wurde) daran adaptiert werden kann. Aufbauend auf diesem heterogenen Verkehr wurde das Konzept des motorisierten Öffentlichen Verkehrs integriert. Dabei wurden alle notwendigen Bestandteile, wie Fahrpläne, Linienbusse und die Einbettung in das allgemeine Verkehrsnetz, beschrieben.

5.2 Routenplanung

Nachdem die wesentlichen statischen Elemente der mikroskopischen Verkehrssimulation eingeführt wurden, wie etwa das Verkehrsnetz, so bedarf es jetzt einer Beschreibung der beweglichen, dynamischen Elemente. Dabei stellen sich unter anderem folgende Fragen: Welche Strecken nutzen Fahrzeuge? Wie werden Fahrzeuge generiert? In der realen Welt verfolgen Fahrer bestimmte Ziele sobald sie in den Verkehr eintreten. Diese lassen sich

schließlich zu Plänen zusammenfassen. Ein Plan kann beispielsweise die vollständige Tagesplanung eines Verkehrsteilnehmers beschreiben: Morgens die Fahrt zur Arbeit, Mittagessen in einem naheliegenden Restaurant, Einkaufen und Freizeitaktivitäten nach Feierabend, sowie die Rückkehr zur Wohnung am Abend.

Die Bewegungsprofile von Fahrzeugen werden als *Aktivitätspläne* bezeichnet. Dabei kann ein Plan ein oder mehrere Punkte umfassen. Die Daten für Aktivitäten können dabei entweder aus demographischen Untersuchungen gewonnen werden oder künstlich generiert werden. Hat man die Datengrundlage, so gilt es diese in Bewegungen umzusetzen, wofür eine Routenplanung notwendig ist. Die Aktivitäten werden mit Hilfe von QZ-Matrizen, wie in Abschnitt 2.1.5 beschrieben, generiert.

Die Aktivitätspläne jedes Verkehrsteilnehmers müssen nun in tatsächliche Bewegungen umgesetzt werden. Dafür gilt es die Fahrtstrecke zwischen den einzelnen Aktivitätszielen (beispielsweise Start- und Endpunkt eines QZ-Paares) miteinander zu verknüpfen. Für diese Art der Verknüpfung bieten sich verschiedene Algorithmen zur Bestimmung des kürzesten Weges an. Wie bereits in Kapitel 2 gezeigt, kann man ein Verkehrsnetz vereinfacht als einen gerichteten Graphen $G = (V,E)$ auffassen, der eine Gewichtsfunktion $g : E \rightarrow \mathbb{R}_+$ besitzt. Diese bildet eine Strecke auf ihren Widerstand im Verkehrsnetz ab. Eine Route R von Knoten v_0 zu Knoten v_n entspricht einer Folge $(v_0, \ldots, v_n)$ von Knoten in G, so dass gilt: $(v_i, v_{i+1}) \in E$ für alle $1 \leq i < n$. Die Länge der Route R setzt sich dann aus der Summe der Einzelgewichte aller Kanten (Strecken) zusammen.

Die Gewichte können dabei auf die unterschiedlichste Weise berechnet sein. Die einfachste Möglichkeit ist es, das Gewicht einer Kante durch ihre Länge zu repräsentieren. Dies ist ein Verfahren, welches in einfachen Navigationssystemen, die sich derzeit auf dem Markt befinden, Anwendung findet. Leider trägt dieses Methode nicht dem Umstand Rechnung, dass eine Strecke umso unattraktiver wird, desto stärker sie befahren wird, das heißt umso höher ihre Auslastung ist. Dieses Problem kann dadurch gelöst werden, dass anstelle der Länge die Reisezeit t_0 auf der Strecke herangezogen wird. Diese lässt sich in einer ersten Näherung über die Beziehung von Geschwindigkeit v und Länge s berechnen:

$$t_0 = \frac{s}{v} \tag{5.13}$$

Aber auch hierbei wird davon ausgegangen, dass die Strecke immer die gleiche Belastung hat. Um dies auszugleichen muss noch die aktuelle Auslastung der Strecke hinzugefügt werden. Hierfür benötigt man zunächst die Kapazität q. Jede Strecke besitzt eine maximale Kapazität q_{max}. Sie gibt dabei die Anzahl der Fahrzeuge an, die eine Strecke aufnehmen kann. Geht man vom Basismodell aus – also von einer einstreifigen Straße mit Zellen der Länge 7,5m – so ergibt sich die maximale Kapazität einer Strecke der Länge L zu:

$$q_{max} = \left\lceil \frac{L}{7,5} \right\rceil \tag{5.14}$$

Dann kann man die aktuelle Auslastung A der Strecke über die Gleichung (5.15) berechnen.

$$A = \frac{q}{q_{max} \cdot c} \tag{5.15}$$

Für das hier vorgestellte heterogene mikroskopische Modell bedarf es allerdings einiger Anpassungen. Seien l_{min} die Länge einer Zelle, N_s die Anzahl der Fahrstreifen auf der Strecke und N_t die Anzahl der Spuren auf dem Fahrstreifen, dann ergibt sich q_{max} aus:

$$q_{max} = \frac{L}{l_{min}} \cdot N_s \cdot N_t \tag{5.16}$$

Um die aktuelle Kapazität einer Strecke zu bestimmen, muss man die Anzahl der Fahrzeuge auf dieser bestimmen.

$$q = \sum_{i \in FT} c_i \cdot n_i \tag{5.17}$$

In Gleichung (5.17) beschreibt FT Die Menge der Fahrzeugtypen, c_i die Anzahl der Zellen, die ein Fahrzeugtyp belegt und n_i die Anzahl der Fahrzeuge vom Typ i, die sich auf der Strecke befinden. Anschließend lässt sich die aktuelle Auslastung analog zum vorherigen Fall mittels (5.15) berechnen.

Die aktuelle Reisezeit t_{akt} und damit der Widerstand einer Strecke[1] ergibt sich dann aus folgender Gleichung, wobei α und β Gewichtungsfaktoren sind:

$$t_{akt} = t_0 \cdot (1 + \alpha \cdot A^{\beta}). \tag{5.18}$$

Eine *Route* in einem Verkehrsnetz kann – wie bereits beschrieben – durch eine Folge von Knoten im Graphen repräsentiert werden. Neben den reinen Gewichten der einzelnen Strecken setzt sich der Gesamtwiderstand einer Route aber noch aus weiteren Komponenten zusammen. Hinzu kommen Teilwiderstände für das Überfahren von Kreuzungen und das Einfließen von Fahrzeugen in den Straßenverkehr von den Bezirken über die Anbindungen auf eine Anfangsstrecke. Dementsprechend ergibt sich der Gesamtwiderstand t_{gesamt} zu:

$$t_{gesamt} = \sum_{i \in S} t_{akt_i} + \sum_{j \in K} t_{k_j} + t_{quelle} + t_{ziel} \tag{5.19}$$

Dabei sei S die Menge aller Strecken, die Teil der Route sind, K die Menge der Knoten auf der Route, sowie t_k die Zeit, die für das Überfahren eines Knotens notwendig sind. Die Werte t_{quelle} und t_{ziel} charakterisieren die Zeiten, die benötigt werden in den Verkehr einzufliessen beziehungsweise am Ziel das Straßennetz zu verlassen.

Nach der Charakterisierung der Widerstände im mikroskopischen Verkehrsnetz ist es notwendig, die eigentliche Routensuche zu beschreiben. Im Rahmen dieser Arbeit wurde eine hierarchische Routensuche entwickelt und umgesetzt, die auf der klassischen Kurzwegsuche von Dijkstra beruht. Jedes Straßennetz und damit Verkehrsnetz lässt sich in verschiedene Hierarchiestufen auflösen. Dies kann, wie in Abschnitt 2.1.6, durch automatisierte Verfahren ähnlich dem von Sanders und Schultes [142, 143] geschehen, die keine weiteren Informationen für die Herausbildung der Hierarchie nutzen.

Die in diesem Framework entstandene Routensuche nutzt die in einem Verkehrsnetz häufig schon gegebene Typisierung der Strecken, das heißt einer Strecke ist häufig schon ein so

[1] In mikroskopischen Flussmodellen auch $t_{akt} = t_{ausfahrt} - t_{einfahrt}$

genannter *Streckentyp* zugewiesen. Existiert dieser Typ nicht, so lässt sich eine Hierarchisierung in hinreichender Genauigkeit durch das Heranziehen weiterer Parameter durchführen. In erster Linie ist hierfür die Maximalgeschwindigkeit einer Strecke relevant. Aber auch andere Werte wie die Anzahl der Streifen, etc. können genutzt werden. Abb. 5.18 zeigt exemplarisch das Verkehrsnetz Deutschlands in verschiedene Hierarchiestufen aufgelöst.

Um eine hierarchisierte Routensuche durchzuführen, wurde der um die Behandlung von Abbiegebeziehungen modifizierte Dijkstra aus Listing 2.2 angepasst. Zur Speicherung der Knoten wird eine Prioritätswarteschlange eingesetzt. Für das Einsortieren der Knoten in die Warteschlange sind die *Schlüssel* der Knoten zuständig. Beim klassischen Dijkstra wird dafür in der Regel die Distanz zum Startknoten herangezogen. Im Fall der hierarchisierten Suche sind diese Schlüssel verändert. Sei $K[v]$ der Schlüssel von Knoten v und u der Startknoten der Suche, so besitzt $K[v]$ drei Parameter. Die *Distanz* gibt die Luftlinienentfernung von v zum Zielknoten an. Der Parameter *Nähe* ist ein bool'scher Wert, der `true` ist, wenn die Distanz von v kleiner ist als die seines Vorgängers im Pfad von u nach v. Der letzte Parameter ist die *Stufe*. Sie gibt an welche Hierarchiestufe die Kante zwischen v und `pre(v)` hat. Damit ergibt sich folgende Vergleichsmethode zum Einsortieren der Schlüssel in die Warteschlange.

```
bool kleiner ( Schluessel x, Schluessel y )
    if ( y.Naehe == x.Naehe )
       if ( y.Distanz == x.Distanz )
          return ( y.Stufe < x.Stufe )
       else
          if ( y.Distanz > x.Distanz )
             return true
          else
             return false
       end if
    else
       return x.Naehe
    end if
end kleiner
```

Listing 5.1: Modifizierte Vergleichsoperation zum Einfügen in die Prioritätswarteschlange

Unter Verwendung der eben vorgestellten Methode entfernt `extract_min()` immer denjenigen Knoten aus der Prioritätswarteschlange, der ein möglichst geringes Gewicht für seinen Pfad zum Startknoten hat und gleichzeitig auf einer möglichst hohen Hierarchiestufe liegt. Durch diese Routensuche wird der Suchraum gegenüber dem klassischen Dijkstra deutlich reduziert. Ergebnisse sind in Kapitel 9 zu finden.

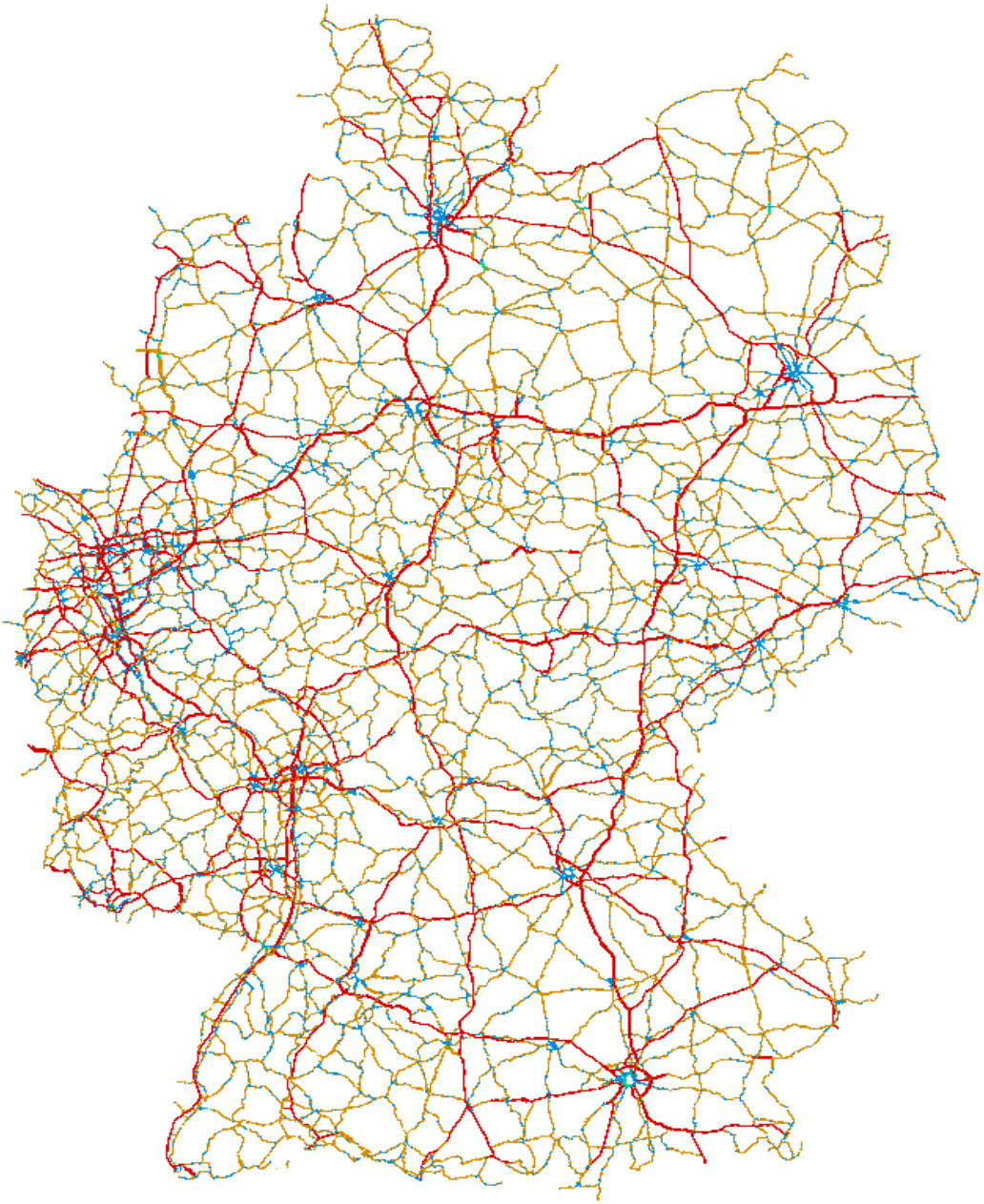


Abbildung 5.18: Das deutsche Verkehrsnetz mit verschiedenen Hierarchiestufen. Die Farben geben die jeweilige Hierarchiestufe an, wobei rot die oberste Hierarchiestufe darstellt und blau die niedrigste.

5.3 Simulationsablauf

Der Ablauf der sequentiellen mikroskopischen Simulation ist in Abb. 5.19 dargestellt. Zunächst werden alle notwendigen Daten geladen. Dazu gehören die Strukturdaten wie das Verkehrsnetz. Zusätzlich werden die Fahrpläne für die im Verkehrsnetz modellierten Buslinien und die Aktivitätspläne der Verkehrsteilnehmer geladen.

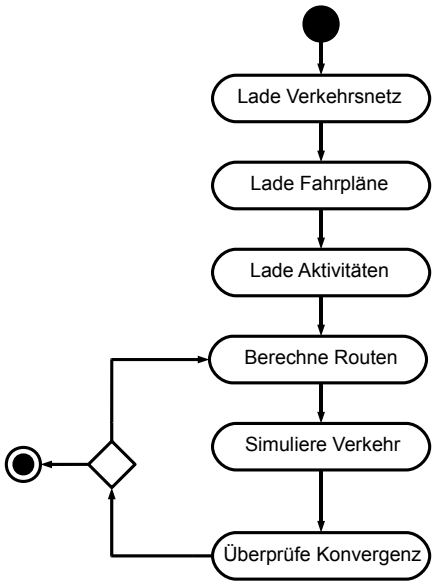


Abbildung 5.19: Allgemeiner Ablauf einer Verkehrssimulation.

Anschließend werden die Routen für die QZ-Paare der Aktivitäten berechnet. Darauf folgt die eigentliche Simulation des Verkehrs. Ist der komplette Simulationszeitraum abgearbeitet, wird überprüft, ob sich ein Gleichgewichtszustand im Verkehrsnetz eingestellt hat. Ist dies nicht der Fall, so startet eine neue Iteration der mikroskopischen Simulation, die Informationen aus der vorherigen nutzt.

5.4 Parallelisierung

In den vorherigen Abschnitten wurden die Erweiterungen des Basismodells von Nagel und Schreckenberg vorgestellt, welche im Rahmen dieser Arbeit entstanden und Teil des entwickelten Frameworks sind. Je komplexer, detaillierter und umfangreicher eine solche mikroskopische Simulation wird, desto rechenintensiver wird diese auch. Die vielfältigen und zahlreichen Interaktionen zwischen allen individuellen Verkehrsteilnehmern in solch einer Mikrosimulation treiben den Aufwand zusätzlich in die Höhe. Die Anwendung von Parallelisierungen stellt eine Möglichkeit dar, diese Probleme zu lösen.

Das vorgestellte mikroskopische Modell basierend auf zellulären Automaten bietet hierfür erst einmal gute Voraussetzungen. Zum einen ist es von Vorteil, dass ein Zeitschritt t lediglich vom direkt vorhergehenden Schritt $t-1$ abhängt. Zum anderen bestehen in erster Linie nur lokale Abhängigkeiten, das heißt die Bewegung der Fahrzeuge auf einem Straßenabschnitt erfordert nur die Betrachtung der Zellen dieses Abschnitts beziehungsweise geht die Bewegung über eine Kreuzung hinaus, so müssen lediglich die direkt darüber angebundenen Straßenabschnitte hinzugezogen werden.

In diesem Abschnitt wird eine Strategie zur Parallelisierung der mikroskopischen Verkehrssimulation vorgestellt, die auf einer Partitionierung der Verkehrsnetze und einer hierarchischen Routensuche beruht. Die Leistungergebnisse dieser Strategie sind in Kapitel 9 zu finden.

5.4.1 Partitionierung des Verkehrsnetzes

An die Partitionierung eines Verkehrsnetzes werden unterschiedliche Anforderungen gestellt. Zum einen sollte im Hinblick auf eine Lastverteilung das Gewicht jeder Partition und damit der mit ihr verbundene Arbeitsaufwand möglichst gleich sein. Zum anderen ist es erstrebenswert, dass die Partitionen so gewählt sind, dass die Anzahl der Schnittkanten zwischen den Partitionen möglichst minimal ist. Da diese Schnittkanten Orte der Kommunikation sind, sinkt dadurch der Kommunikationsaufwand.

Es bleibt in dieser Hinsicht zu klären, wie sich das Gewicht einer Partition definiert. In einer ersten Annahme kann man davon ausgehen, dass sich das Gewicht einer Partition aus der Anzahl der Netzelemente (Knoten, Strecken, Abbiegebeziehungen) ergibt. Dies trägt aber nicht dem Umstand Rechnung, dass die Rechenlast für eine Partition nicht alleine von diesen Elementen abhängt. Vielmehr ist die Anzahl der Verkehrsteilnehmer von entscheidender Bedeutung. Jedes Fahrzeug muss angefasst werden und entsprechend der beschriebenen Regeln fortbewegt werden. Im Gegensatz zu einem statischen Verkehrsnetz existiert ein Netz mit dynamischen Elementen.

Verfahren wie die rekursive Koordinatenbisektion und die Multilevel-Partitionierung mittels METIS/ParMETIS [93], wie sie bei der Parallelisierung des makroskopischen Individualverkehrs zum Einsatz kommen werden (siehe Kapitel 7), scheinen für die mikroskopische Simulation nicht ausreichend zu sein. Diese Verfahren versuchen die Menge der Schnittkanten zu reduzieren, ignorieren dabei aber die dynamischen Elemente des Verkehrsnetzes. Die Rechenlast in den einzelnen Partitionen kann daher unter Umständen deutlich variieren.

Deshalb entstand im Rahmen dieser Arbeit ein neues Verfahren zur Partitionierung von Verkehrsnetzen unter der Ausnutzung *raumfüllender Kurven* [141]. Diese Methode wurde zusätzlich im Rahmen einer Diplomarbeit weiter untersucht [170].

Zunächst gilt es die Anforderungen an eine Partitionierungen genauer zu definieren. Gewünscht ist eine Partitionierung, die die aktuelle Verkehrsauslastung in der Partition betrachtet und somit eine realitätsnahe Lastverteilung ermöglicht. Zusätzlich sollte die Parti-

tionierung so durchgeführt werden, dass die Anzahl der Schnittkanten möglichst gering ist. Im Hinblick auf das Laufzeitverhalten der Simulation und einer dynamischen Lastverteilung ist eine effiziente Berechnung beziehungsweise Neuberechnung der Partitionen wichtig.

Die Partitionierung erfolgt nun in einem dreistufigen Prozess.

1. Aufteilung des Verkehrsnetzes in *Sektoren*
2. Gewichtung der Sektoren
3. Zusammenfassen von Sektoren zu *Partitionen* mittels raumfüllender Kurven

In der ersten Phase dieses Prozesses, der *Sektoreinteilung*, wird das Verkehrsnetz in einzelne Sektoren aufgeteilt. Die Sektoren werden so gewählt, dass ihre Gewichte g möglichst gleich sind. Diese Gewichte setzen sich aus der Summe der enthaltenen Streckenlängen und Anteil der Straßen höherer Hierarchiestufen. Die Größe der dabei entstehenden Sektoren ist deutlich kleiner als die der späteren Partitionen. Die Aufteilung erfolgt dabei mit Hilfe von Quadtrees [41, 54]. Ein schematischen Beispiel einer möglichen Aufteilung in Sektoren ist in Abb. 5.20 dargestellt.

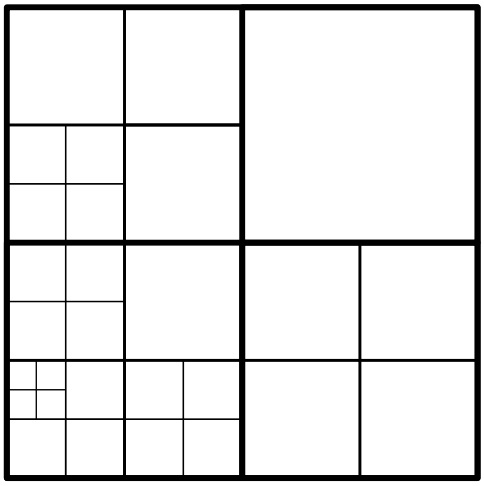


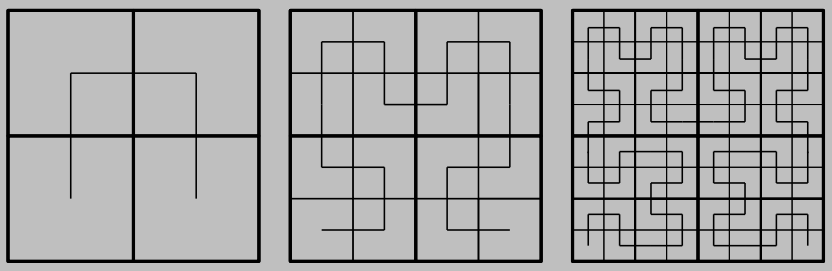
Abbildung 5.20: Aufteilung eines Verkehrsnetzes in Sektoren. Die Aufteilung basiert auf dem Prinzip eines Quadtrees bei dem die Ausgangsfläche rekursiv in vier Quadrate aufgeteilt wird.

Die erste Phase wird nur ein einziges Mal zu Beginn der Simulation durchgeführt, das heißt, die Sektoren ändern sich anschließend nicht mehr. Die beiden weiteren Phasen werden im Laufe der Simulation immer wieder durchgeführt, um eine dynamische Lastverteilung zu ermöglichen. Ist die Sektoreinteilung abgeschlossen, erfolgt in der zweiten Phase eine erneute Gewichtung der Sektoren. Neben den bereits genannten Parametern kommen hier noch weitere hinzu, wie etwa die Anzahl der Fahrzeuge, die sich in einem Sektor befinden.

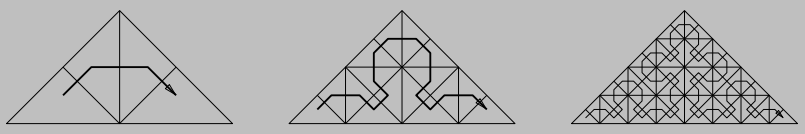
Die dritte Phase beinhaltet die eigentliche Partitionierung, in der mehrere Sektoren zu einer Partition zusammengefasst werden. Die Partitionen werden dabei so zusammengestellt, dass ihre Gewicht möglichst gleich sind.

Exkurs: Raumfüllende Kurven

Bei raumfüllenden Kurven (engl. space-filling curve, SFC) handelt es sich um eine Möglichkeit multidimensionale Daten zu linearisieren. Diese stetige Kurve traversiert alle Elemente eines n-dimensionalen Raumes. Die Elemente werden dabei immer nur ein einziges Mal betrachtet. Ein Beispiel für eine solche raumfüllende Kurve ist die *Hilbert-Kurve*. Ihre Konstruktion beruht auf einem rekursiven Verfahren. Sie durchläuft in ihrer ersten Iteration ein Quadrat, das in vier Quadranten unterteilt ist. Jeder dieser Quadraten kann dann wiederum in wieder vier Teilquadranten unterteilt werden. Diese können dann durch geeignete Rotationen bzw. Spiegelungen des *Grundmotivs* der ersten Iteration erneut durchlaufen werden. Im Folgenden sind die ersten drei Iterationen der Konstruktion der Hilbert-Kurve gezeigt.



Eine weitere raumfüllende Kurve ist die Kurve von *Sierpinski*. Sie durchläuft rechtwinklige, gleichschenklige Dreiecke anstelle von Quadraten. Die ersten drei Iterationen sind nachfolgend dargestellt.



SFC haben weitere wichtige Eigenschaften. Zum einen gehört dazu die *Erhaltung der Nachbarschaften*. Die Grundmotive werden in der nächsten Iteration immer weiter durch das Grundmotiv (rotiert oder gespiegelt) ersetzt. Dadurch werden die Gebiete der Iteration $i-1$ auch in der Iteration i in der gleichen Reihenfolge durchlaufen, d.h. es kommt immer zu lokalen Verfeinerungen. Daraus kann man ableiten, dass Elemente auf der SFC, die kurz hintereinander traversiert werden, auch nachbarschaftlich nahe sind. Ein letzter Aspekt ist die *lineare Komplexität*, da jedes Element genau einmal besucht wird.

Um die Partitionierung durchzuführen werden die Sektoren mit Hilfe raumfüllender Kurven durchlaufen (siehe Exkurs). Abb. 5.21 zeigt eine vierfache Partitionierung des Verkehrsnetzes der Stadt Karlsruhe unter Anwendung der Sierpinski-Kurve (links) und der Hilbert-Kurve (rechts). Beide Kurven erweisen sich in ihrer Qualität der Partitionierung als etwa gleich gut, wobei die Hilbert-Kurve geringfügig zusammenhängendere Partitionen liefert.

Exemplarisch ist in Abb. 5.22 die Partitionierung des Verkehrsnetzes von Deutschland dargestellt.

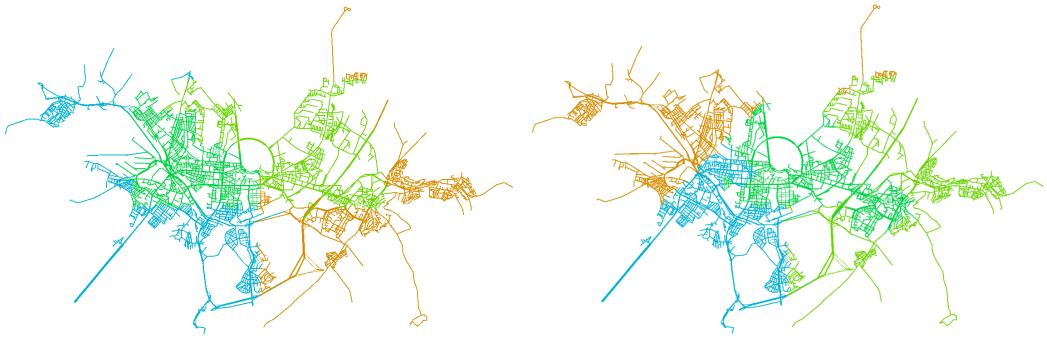


Abbildung 5.21: Partitionierung des Stadtnetzes von Karlsruhe mittels Sierpinski (links) und Hilbert (rechts) in vier Teile. Die unterschiedlichen Farben stellen die verschiedenen Partitionen dar.

Wichtig anzumerken ist noch, dass der dem Verkehrsnetz zu Grunde liegende Graph an den Knoten aufgeteilt wird, anders als beispielsweise in der Arbeit von Dupuis und Chopard [45], die eine Aufteilung an den Mitten von Kanten, das heißt Strecken durchführen. In der hier vorgestellten Partitionierung wird eine Grenzkante, das heißt eine Kante zwischen zwei Partitionen, einer der Partitionen zugewiesen. Die Kreuzungen im Verkehrsnetz, die durch die Knoten repräsentiert werden, sind folglich mögliche Schnittpunkte der Partitionen.

5.4.2 Ablauf der parallelen mikroskopischen Simulation

Nach der erfolgten Beschreibung der Partitionierung gilt es den Ablauf der parallelen mikroskopischen Simulation aufzuzeigen. Gegenüber dem sequentiellen Ablauf in Abb. 5.19 haben sich in Abb. 5.23 die grau hinterlegten Operationen geändert.

Auch im parallelen Fall beginnt der Simulator mit dem *Laden und Initialisieren* aller notwendigen Daten, wozu das Verkehrsnetz, die Fahrpläne und die Aktivitäten zählen. Nachdem diese Daten geladen wurden, erfolgt die *Partitionierung* des Verkehrsnetzes, wie sie im Abschnitt 5.4.1 beschrieben wurde. Daran schließt sich die *Routenberechnung* (siehe Abschnitt 5.2) an. Nach der Routenberechnung erfolgt eine *Kommunikation* der Daten zwischen den beteiligten Rechenknoten. Dazu gehören die Routen, die vollständig an die Rechenknoten übermittelt werden und die Partitionsinformationen, sowie die zu den Partitionen gehörenden Aktivitäten. Als erster, echter Schritt der Simulation werden nun die *Fahrzeuge* entsprechend der Aktivitäten *eingefügt*. Zunächst werden nun die *Bewegungen* an den Kreuzungen ausgeführt. Da diese Orte der Partitionsgrenzen sein können, kann es sein, dass ein Fahrzeug beim Überqueren einer Kreuzung seine bisherige Partition verlassen wird. Informationen darüber werden an die neu zuständigen Partitionen in einem neuen

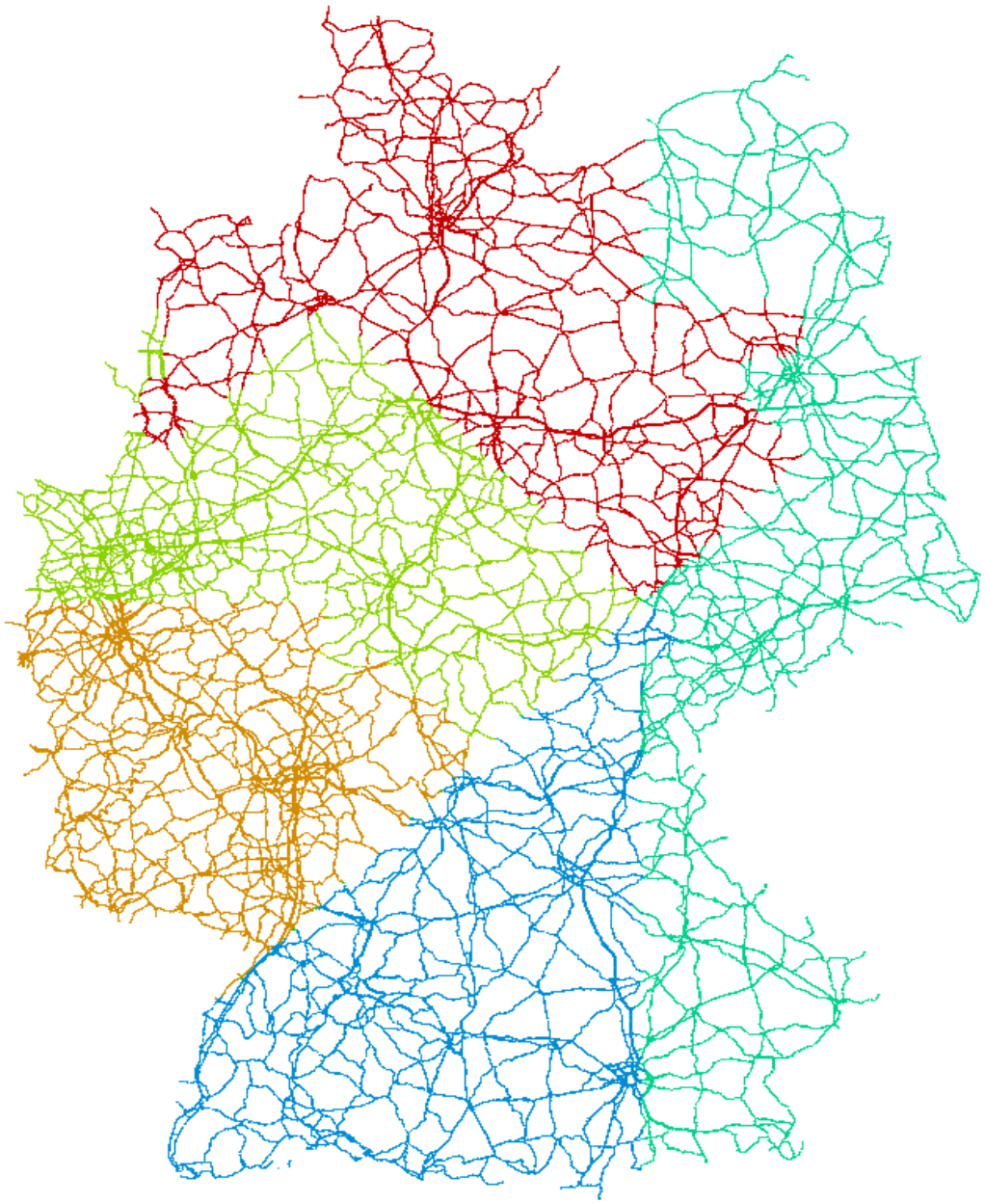


Abbildung 5.22: Partitionierung des Verkehrsnetzes Deutschlands mittels der Sierpinski-Kurve

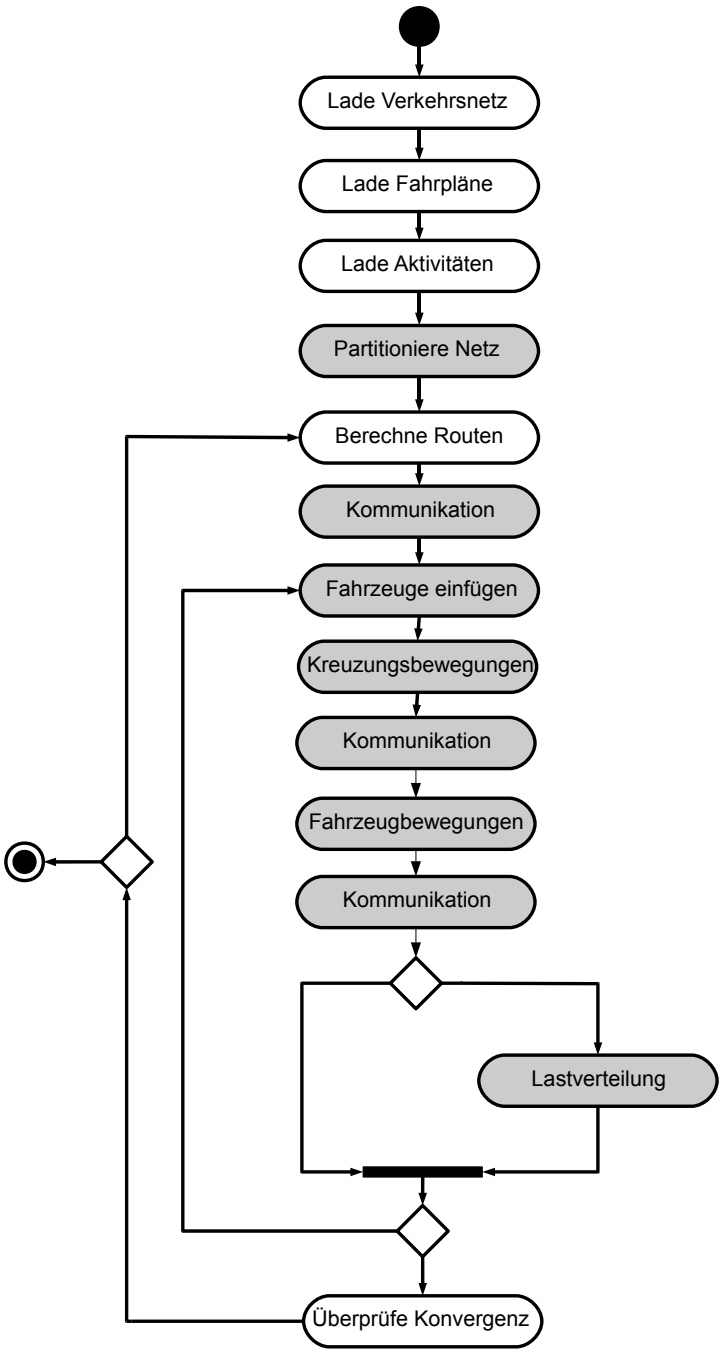


Abbildung 5.23: Ablauf der parallelen mikroskopischen Verkehrssimulation.

Kommunikationsschritt übermittelt, so dass diese die „Ankunft“ eines neuen Fahrzeugs registrieren und so eine Kollisionsfreiheit innerhalb der Simulation gewährleisten können. Alle anderen Fahrzeuge werden keine Partitionsgrenze überschreiten. Daher wird ihre Bewegung anschließend an diesen Kommunikationsschritt durchgeführt. Jetzt ist die Auslastung des Verkehrsnetzes im nächsten Simulationszeitschritt bekannt. Dementsprechend wird jetzt entschieden, ob eine erneute *Lastverteilung* durchgeführt wird. Dadurch wird überprüft, ob die Differenz der Gewichte der Partitionen sich um einen benutzerdefinierten Grenzwert unterscheiden. Ist dies der Fall, so wird eine neue Partitionierung durchgeführt, indem die Sektoren des Verkehrsnetzes, wie in Abschnitt 5.4.1 beschrieben, neu gewichtet werden und anschließend Phase drei der Partitionierung ausgeführt. Dabei werden bestehenden Partitionen in der Regel lediglich einige bisherige Sektoren entfernt oder neue hinzugefügt. Nun wird überprüft, ob bereits alle Simulationszeitschritte abgearbeitet wurden. Ist dies der Fall, so wird die *Konvergenz überprüft* (analog zum sequentiellen Fall). Ist der Simulationszeitraum noch nicht beendet, beginnt die Bearbeitung des nächsten Zeitschritts.

5.5 Zusammenfassung

In diesem Kapitel wurde ein mikroskopisches Modell vorgestellt, dass im Rahmen dieser Arbeit entstand. Dieses unterstützt die Behandlung allgemeinen heterogenen Verkehrs. Es ist dabei möglich beliebige Fahrzeugtypen hinzufügen. Es wurde gezeigt, dass sich dieses heterogene Modell leicht zur Unterstützung von motorisiertem Öffentlichem Verkehr heranziehen lässt. Dazu wurde ein Modell des motorisierten ÖV vorgestellt, welches eine fahrplanfeine Modellierung der Buslinien unterstützt und sich nahtlos in die bisherige Simulation integrieren lässt. Das Kapitel endete mit einer Beschreibung der im Rahmen dieser Arbeit entwickelten Parallelisierung der mikroskopischen Simulation. Dafür wurde ein Verfahren zur Partitionierung von Verkehrsnetzen entworfen, das raumfüllende Kurven nutzt. Die Ergebnisse sind in Kapitel 9 zu finden.

6 Makroskopische Simulation des Öffentlichen Verkehrs

Eine weitere Möglichkeit Verkehr zu simulieren besteht in einer makroskopischen Betrachtung. Hierbei kann man zwei Sichtweisen unterscheiden: Zum einen die strömungsmechanische Betrachtung des Verkehrs und in Form von Umlegungsverfahren. Die letzteren sind Gegenstand der Betrachtung in diesem und dem nächsten Kapitel.

Um den ÖV mit Hilfe von Umlegungsverfahren zu simulieren, gibt es verschiedene Möglichkeiten. Mit diesen beginnt die Betrachtung, bevor zu einem speziellen *fahrplanfeinem* Verfahren übergegangen wird. Dieses Verfahren, dass von Friedrich et al [62, 64] entwickelt wurde, ist schnell, liefert sehr gute Ergebnisse und hat Eingang in kommerzielle Softwarewerkzeuge wie die Verkehrsplanungssoftware VISUM der PTV AG gefunden. Im letzten Abschnitt dieses Kapitels wird eine Möglichkeit zur Parallelisierung dieses Verfahrens aufzeigt, die im Rahmen dieser Arbeit entstanden ist.

6.1 Umlegungsverfahren für den Öffentlichen Verkehr

Bei Umlegungsverfahren werden nicht die einzelnen Verkehrsteilnehmer und deren Interaktion betrachtet. Daher kann man sie zu den makroskopischen Simulationsmodellen zählen. Generell kann man drei große Typen von Umlegungsverfahren für den ÖV unterscheiden. Dies sind zum einen *verkehrssystemfeine* und *taktfeine* Verfahren und zum anderen *fahrplanfeine* Verfahren.

Alle Verfahren haben eine Grundablauf gemeinsam, wie er in Abb. 6.1 skizziert ist. Die beiden Hauptteile jeder Umlegung sind dabei die *Routensuche* und die anschließende *Routenbelastung*.

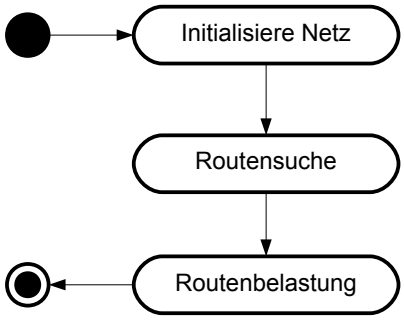


Abbildung 6.1: Grundablauf der Umlegungsverfahren für den Öffentlichen Verkehr.

Verkehrssystemfeine Verfahren

In einer ersten Näherung kann man die Existenz von Fahrplänen und Linien vernachlässigen und lediglich unterschiedliche Verkehrssysteme (U-Bahn, Linienbusse,...) unterscheiden. Hierbei spricht man von verkehrssystemfeinen Verfahren. Sie gehören zu den einfachsten Methoden, bieten aber daher auch nur einen ersten groben Überblick über die Strukturen der Verkehrsnachfrage. Es werden nur die Strecken des ÖV-Netzes betrachtet, welche in der Regel eine Teilmenge des kompletten Verkehrsnetzes sind. Den auf diese Weise gewonnenen Teilgraphen kann man für eine Wegsuche verwenden. Es wird dabei eine Route gesucht, die minimalen Widerstand (Gewicht) besitzt. Ähnlich zur Widerstandsdefinition beim mikroskopischen Verkehr (vgl. Abschnitt 5.2) kann man diesen auch beim verkehrssystemfeinen Verfahren beschreiben. Aufgrund der nur groben Betrachtungsweise dieses Verfahrens genügt es allerdings in den meisten Fällen, den Widerstand einer Strecke als die darauf benötigte Reisezeit zu definieren. Der Gesamtwiderstand einer Route, die sich aus mehreren Strecken und Netzelementen ergibt, kann sich dabei – neben dem reinen Streckenwiderstand – aus weiteren Komponenten zusammensetzen. Typischerweise werden hierfür Umsteigezuschläge für Haltestellen und beim Wechsel des Verkehrssystems verwendet. Dies trägt der Situation Rechnung, dass man normalerweise mit Wartezeiten zu rechnen hat, wenn man ein Verkehrsmittel wechselt (beispielsweise bei einem Umstieg von einem Linienbus auf Zug). Die Routensuche erfolgt für jedes relevante QZ-Paar. Hat man die widerstandsminimale Route im ÖV-Netz für das jeweilige QZ-Paar gefunden, wird die komplette Nachfrage dafür auf diese Route umgelegt.

Taktfeine Verfahren

Ein etwas höheres Maß an Realität bieten die taktfeinen Verfahren. Bei diesen Verfahren versucht man, auf eine einfache Weise das Prinzip der Fahrpläne zu imitieren. Jede Linie des ÖV-Angebots wird durch ihren Linienroutenverlauf, die Fahrzeit zwischen je zwei Haltestellen und durch die Angabe eines *Taktes* beschrieben. Dabei sind die durch die Linie verwendeten Strecken des ÖV-Netzes durch den Linienroutenverlauf gegeben. Der Takt gibt

dabei den Abstand zwischen den Abfahrtszeiten der Linie an.

Das Grundprinzip der Umlegungsverfahren, wie in Abb. 6.1 angedeutet wurde, wird dahingehend erweitert, dass sich das taktfeine Verfahren neben der Initialisierung des Verkehrsnetzes in die Schritte *Taktermittlung*, kombinierte *Routensuche und Routenwahl* und in eine *Routenbelastung* aufteilt.

Der Takt einer Linie kann dabei auf verschiedene Weisen berechnet werden. Exemplarisch sollen hier zwei Möglichkeiten aufgezeigt werden. Weitergehende Möglichkeiten und Methoden können beispielsweise [135] entnommen werden. Eine Möglichkeit besteht darin, den Takt einer Linie direkt anzugeben, das heißt als eine feste benutzerdefinierte Größe. Dies hat den Vorteil, dass man auf Fahrpläne generell verzichten kann. Die verlorene Flexibilität stellt allerdings einen nicht zu vernachlässigenden Nachteil dar.

Daher kann man als weitere Möglichkeit durch eine einfache Erweiterung, unter Verwendung von Fahrplänen, realistischere Ergebnisse erzielen. Hierzu betrachtet man für jede Linie verschiedene Zeitintervalle $I = [x, y[$. Diese müssen sich allerdings innerhalb des betrachteten Umlegungszeitraums, das heißt dem Simulationszeitraum, befinden. Damit lässt sich der Takt T mittels

$$T_{x,y} = \frac{y-x}{n} \tag{6.1}$$

ermitteln. Dabei gibt n die Anzahl der Abfahrten an, die in diesem Zeitraum stattfinden sollen. Dieses Methode ist genauer als die die vorherige mit der direkten Angabe des Taktes, da sie durch verschiedene Intervalle mehr Flexibilität bietet.

Die Routensuche läuft anders ab als bei den verkehrssystemfeinen Verfahren: Ausgangspunkt ist wieder ein QZ-Paar, also eine Quelle-Ziel-Beziehung. Ausgehend von dem gegebenen Zielbezirk werden rückwärts alle Möglichkeiten berechnet, zu diesem Zielbezirk zu gelangen. Auf diese Weise entsteht für jeden Zielbezirk ein Entscheidungsbaum, dessen Zweige mögliche Routen auf dem Weg vom Quell- zum Zielbezirk repräsentieren. Die Nachfrage, also die Anzahl der Fahrzeugfahrten, wird nun auf diese Routen verteilt.

Dieses Verfahren liefert bessere Ergebnisse als die verkehrssystemfeinen Verfahren, berücksichtigt aber weiterhin keine Fahrpläne, die für detaillierte Betrachtungen aber unabdingbar sind. Zu diesem Zweck werden so genannte fahrplanfeine Verfahren eingesetzt.

6.2 Fahrplanfeine Umlegung des Öffentlichen Verkehrs

Im folgenden wird ein Verfahren betrachtet, das 1994 in [61] entwickelt und weiter in [62, 64] beschrieben wurde. Dieses Verfahren ist Ausgangspunkt für die im Rahmen dieser Arbeit entstandene Parallelisierung.

Bei fahrplanfeinen Verfahren werden die genauen Fahrpläne jeder einzelnen Linie betrachtet. Dazu gehören die genauen Abfahrts- und Ankunftszeiten des jeweiligen Verkehrsmittels. Auch bei diesem Verfahren kommt eine QZ-Matrix zum Einsatz. Der Ablauf ist aller-

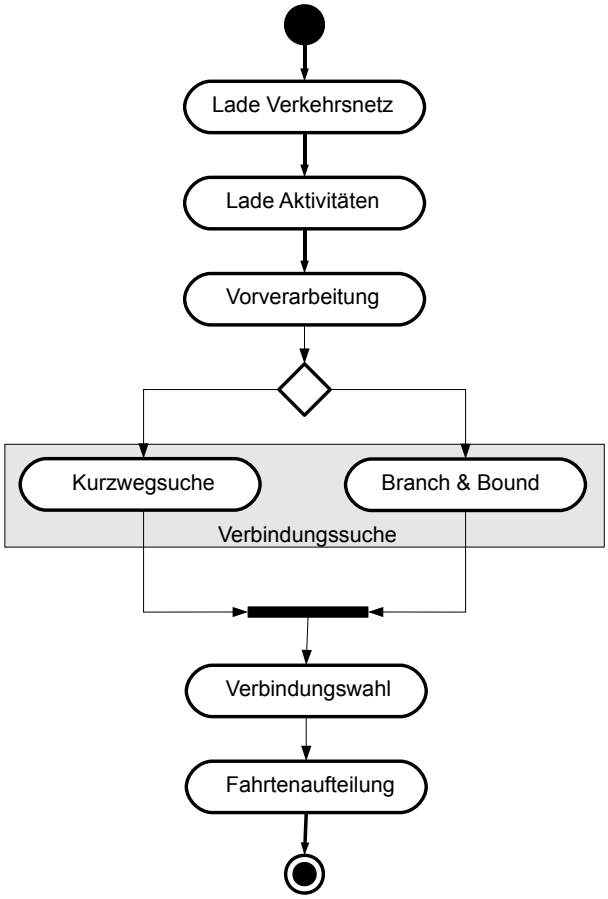


Abbildung 6.2: Ablauf der sequentiellen fahrplanfeinen Umlegung des Öffentlichen Verkehrs.

dings aufwändiger als bei den beiden vorgenannten Verfahrensklassen. Er ist in Abb. 6.2 skizziert.

Auffällig ist, dass es für die die Verbindungssuche zwei Varianten gibt: eine klassische Kurzwegsuche, der der Algorithmus von Dijkstra zugrunde liegt, und ein Branch & Bound Algorithmus. Der Schwerpunkt dieser Arbeit liegt auf dem Branch & Bound Verfahren. Dennoch soll kurz auch die Variante der Kurzwegsuche skizziert werden.

6.2.1 Kurzwegsuche

Bei der Kurzwegsuche werden die möglichen Abfahrts- und Ankunftszeiten innerhalb des Simulationszeitraums untersucht. Dazu wird für jede darin mögliche Abfahrtszeit eine Kurzwegsuche für jedes vorhandene QZ-Paar i, j durchgeführt. Der Widerstand der einzelnen

Netzelemente (Strecken, Knoten, ...) ergibt sich analog zu den vorherigen Verfahren. Zusätzlich wird aber noch die Umsteigehäufigkeit betrachtet. So wird der Widerstand bei jedem Umstieg um einen gewissen Wert erhöht. Damit lässt sich die Suche steuern: Bei einem hohen Wert werden bevorzugt die Routen gefunden, die ein selteneres Umsteigen erfordern; bei einem niedrigem Wert werden die zeitkürzesten Routen bevorzugt.

Basierend auf den Fahrplänen werden zunächst die möglichen Abfahrtszeitpunkte im Bezirk i, der Quelle, bestimmt. Anschließend wird für jeden dieser Zeitpunkte eine Kurzwegsuche durchgeführt. Diese ermittelt die für diesen Zeitpunkt beste Route von Bezirk i zu Bezirk j, dem Ziel. Der Gesamtwiderstand $W_{i,j}$ der Route ergibt sich dabei aus der Summe von Zugangszeit, Abgangszeit, der Fahrzeit in den einzelnen Verkehrsmitteln, den Umsteigegehzeiten für den Fußweg zwischen zwei Umsteigehaltestellen, der eigentlichen Umsteigewartezeit und dem Strafwert für die Umsteigehäufigkeit.

Die häufige Kurzwegsuche für alle möglichen Abfahrtszeitpunkte für jedes QZ-Paar stellt allerdings hohe Kosten dar, und wie in [61] gezeigt, gibt es auch Einschränkungen in der Qualität, was zur Entwicklung des Branch & Bound Verfahrens führte.

6.2.2 Branch & Bound

Die Grundidee hinter diesem Verfahren ist, dass zu jedem Quellbezirk ein Suchbaum mit allen Teilrouten generiert wird, die hinreichend gut sind. Auf diesen Routen werden anschließend die Fahrten verteilt. Das Verfahren unterscheidet dabei zwischen *Routen* und *Verbindungen*. Letztere sind dabei um Fahrplaninformationen angereicherte Routen. Das Verfahren soll im folgenden kurz skizziert werden.

Nach dem Einlesen des Verkehrsnetzes und der Aktivitäten in Form einer Quelle-Ziel-Matrix beginnt ein *Vorverarbeitungsschritt*, welcher die Generierung von *Verbindungssegmenten* zum Ziel hat. Diese beschreiben jeweils den Teil einer Reise und werden für die Gewinnung der endgültigen Verbindungen verwendet.

Um die Verbindungssegmente zu gewinnen, werden zunächst alle *Routensegmente* einer Linie ermittelt. Diese werden, ausgehend von dem Verkehrsnetz und den Linien, noch ohne Verwendung von Fahrplänen gewonnen. Ein Beispiel für die Erstellung von Routensegmenten ist in Abb. 6.3 dargestellt.

Ein Routensegment einer Linie ist ein Abschnitt der Linie zwischen einer Einstiegs- und einer Ausstiegshaltestelle. Es ergeben sich dabei sämtliche Routensegmente einer Linie, wenn man jede mögliche Kombination von Ein- und Ausstiegshaltestellen einer Linie betrachtet. Zieht man als Beispiel Abb. 6.3 heran, so lassen sich für die Einstiegshaltestelle die Routensegmente ($A \rightarrow B$, $A \rightarrow B \rightarrow C$, $A \rightarrow B \rightarrow C \rightarrow E$) definieren. Als nächstes betrachtet man die auf A folgende Haltestelle B und generiert von dieser ausgehende die möglichen Routensegmente ($B \rightarrow C$ und $B \rightarrow C \rightarrow E$). Dies wird solange fortgesetzt bis keine weiteren Haltestellen auf der Linie mehr zur Verfügung stehen.

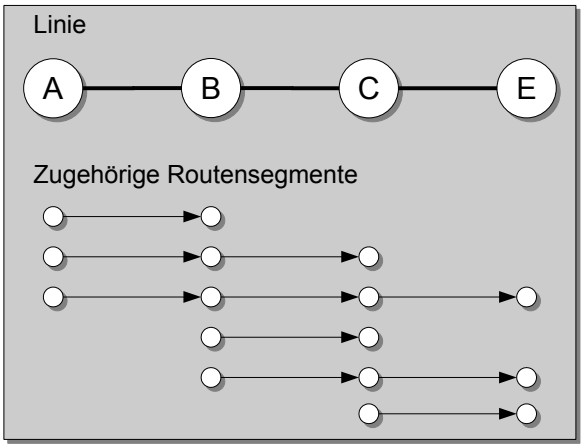


Abbildung 6.3: Gestalt der Routensegmente für die Linie $A \rightarrow B$. Die Routensegmente entstehen, indem man jede möglichen Kombination von Ein- und Ausstiegshaltestellen einer Linie betrachtet (nach [62]).

Die Verbindungssegmente werden nun gewonnen, indem man die Routensegmente um Abfahrts- und Ankunftszeitpunkte anreichert. Mit dieser Anreicherung ist die Vorverarbeitung abgeschlossen.

Als nächster Schritt erfolgt die *Routen- bzw. Verbindungssuche*. Dabei wird ausgehend vom Quellknoten mit Hilfe eines Branch & Bound Suchverfahrens[1] ein Verbindungsbaum (siehe Abb. 6.4) aufgebaut. Der Algorithmus geht dabei immer Baumebene für Baumebene durch. Hat man ein Verbindungssegment der aktuell betrachteten Baumebene, so werden alle potentiellen Nachfolger betrachtet und bewertet. Der Widerstand w_c einer Verbindung c ergibt sich schließlich folgendermaßen:

$$w_c = \alpha \cdot t_c + \beta \cdot nt_c + \gamma \cdot f_c \tag{6.2}$$

Dabei sind t_c die Reisezeit auf der Verbindung, nt_c die Umsteigehäufigkeit, und f_c modelliert den Fahrpreis. Die Faktoren α, β und γ dienen der Gewichtung der einzelnen Komponenten.

Ein neues Verbindungssegment wird anschließend zur bereits bestehenden Verbindung hinzugefügt, sofern es einige Kriterien erfüllt: zeitliche Eignung, Relevanz und Toleranzbeschränkungen. Unter zeitlicher Eignung versteht man, dass die Abfahrtszeit des betrachteten Segments nicht vor der Ankunft der bisherigen Verbindung zuzüglich einer minimalen Umsteigegehzeit liegt. Die Relevanz gibt an, dass es bezüglich der zeitlichen Eignung keine bereits gefundene bessere Verbindung gibt. Zusätzlich darf bei keiner Verbindung die Um-

[1] Branch & Bound Verfahren kommen häufig bei Optimierungsproblemen zum Einsatz. Hier ist es in der Regel notwendig alle Möglichkeiten durchzuprobieren, die zur Lösung des Problems führen könnten, was sich in einem häufig inakzeptabel hohen Rechenaufwand niederschlägt. Daher wird bei Branch & Bound Verfahren der Suchraum in Teilmengen aufgeteilt (Branch) und versucht suboptimale Lösungen möglichst frühzeitig zu auszuschließen (Bound).

steigehäufigkeit bzw. der Fahrpreis geringer sein. Die Grundstruktur des Verbindungsbaums ist in Abb. 6.4 dargestellt.

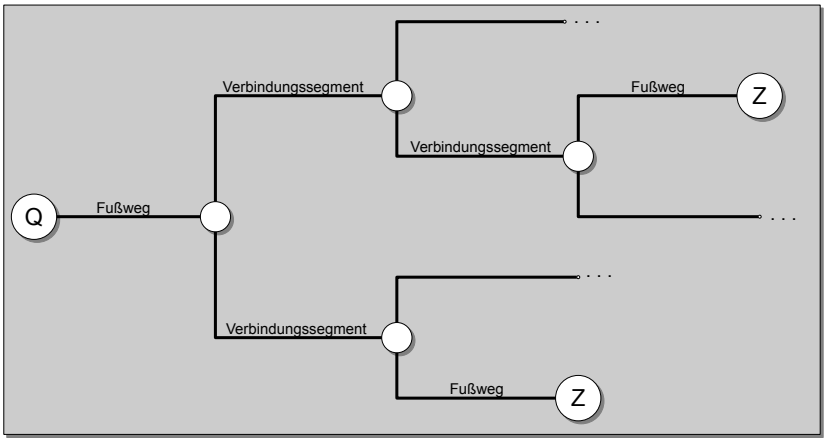


Abbildung 6.4: Grundstruktur eines Verbindungsbaums für Quelle Q bis Ziel Z (nach [62]).

Bei der nachfolgenden *Verbindungswahl* wird der bisher gewonnene Suchbaum neu evaluiert. Hierbei greifen strengere, benutzerdefinierte Bindungen für die Verbindungen. Auf diese Weise werden in diesem Schritt weitere Verbindungen, die den Bedürfnissen des Nutzers nicht entsprechen, nachträglich aus dem Suchbaum entfernt.

In der abschließenden *Fahrtenaufteilung* wird die Nachfrage einer Quelle-Ziel-Beziehung gemäss eines Aufteilungsmodell auf die noch verbliebenen Verbindungen verteilt. Die so belasteten Verbindungen werden nun auf das Verkehrsnetz abgebildet, wobei man so sukzessive das vollständige belastete Verkehrsnetz erhält.

6.3 Parallelisierung der fahrplanfeinen Umlegung

Das in Abschnitt 6.2 beschriebene Verfahren zur fahrplanfeinen Umlegung der Nachfrage des Öffentlichen Verkehrs ist eines der genauesten, realistischsten und schnellsten Verfahren. Allerdings besteht auch bei diesem das Problem, dass die Rechenzeit bei größeren Verkehrsnetzen stark ansteigt. Daher wurden im Rahmen dieser Arbeit Strategien zur Parallelisierung entwickelt und implementiert [63].

Abb. 6.5 zeigt die Rechenzeit für verschiedene Szenarien der fahrplanfeinen Umlegung. Simuliert wurde dabei ein Zeitraum von 24h. Man erkennt, dass man bei der Rechenzeit im sequentiellen Fall schnell in den Stundenbereich gelangt. Dies ist zum Teil für die Einsatzgebiete beispielsweise in Ingenieurbüros häufig nicht akzeptabel. Dies führt dazu, dass gegenwärtig nur Netze kleiner und mittlerer Größe genutzt werden. Für realistischere und genauere Simulationsergebnisse ist es allerdings erforderlich, dass entweder (1) größere

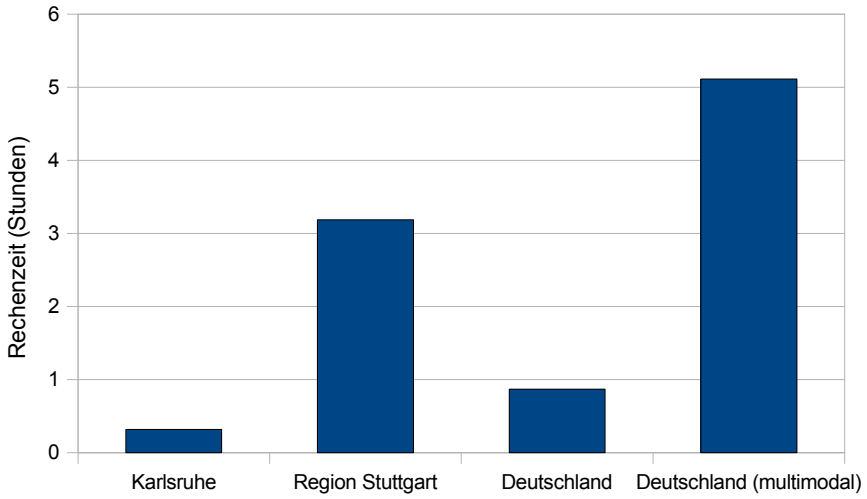


Abbildung 6.5: Rechenzeitverteilung bei Ausführung der fahrplanfeinen Umlegung für den Öffentlichen Verkehr bei verschiedenen Verkehrsnetzen für einen Umlegungszeitraum von 24h.

oder (2) detaillierte Netze betrachtet werden. Anforderung (1) ist notwendig, wenn man genauer den Einfluss eines größeren Gebiets betrachten möchte. So ist es häufig von Bedeutung, auch den einfließenden Verkehr in ein Stadtgebiet zu betrachten und nicht nur das Stadtgebiet selbst. Nutzt man – wie in (2) gefordert – detailliertere Verkehrsnetze, so lässt sich dadurch präziser die Verkehrsnachfrage und das Verkehrsverhalten modellieren: Mehr Verkehrsbezirke im Netz erlauben eine genauere Aufschlüsselung der Verkehrsnachfrage und damit der Fahrzeugfahrten. Im Extremfall kann man ein gesamtes Stadtgebiet oder Stadtviertel als einen einzelnen Bezirk realisieren. Dementsprechend würde der gesamte Verkehr zusammengefasst betrachtet und in das System eingespeist werden. Modelliert man mehrere Bezirke, so lässt sich dadurch detaillierter die Struktur des Viertels mit jeweils unterschiedlichem Verkehrsverhalten beschreiben.

Es ist allerdings offensichtlich, dass diese Anforderungen den Rechenzeit- und Speicherbedarf steigen lassen. Will man also diese Verkehrsnetze simulieren, so bedarf es neuer Ansatzpunkte. Ein Ansatz, der hier weiter verfolgt werden soll, ist die Parallelisierung des beschriebenen sequentiellen Algorithmus, um dadurch die Ressourcen vieler Computer nutzen zu können [63].

6.3.1 Auswahl einer geeigneten Parallelisierungsstrategie

Listing 6.1 gibt in vereinfachter Form die Grundstruktur des in Abb. 6.1 beschriebenen Algorithmus wider.

```
(0) Lade Verkehrsnetz
(1) Lade Aktivitaeten
(2) Vorverarbeitung
for ( n = 0; n < number_of_actvities; n += 1 )
   (3) Fuehre Routensuche fuer jedes QZ-Paar durch
   (4) Selektiere und Entferne Verbindungen durch die
       Verbindungswahl
   (5) Teile die Verkehrsnachfrage auf die gefunden
       Verbindungen auf
end for
(6) Fasse Ergebnisse aller QZ-Paare zusammen und ermittle
    Gesamtbelastung des Verkehrsnetzes
```

Listing 6.1: Vereinfachter Algorithmus der fahrplanfeinen Umlegung

Die Operationen `(0)` bis `(2)` sind im Folgenden als Vorgang des Parsens zusammengefasst, da in ihnen das Einlesen der entsprechende Dateien erfolgt, die Netzstrukturen aufgebaut und initialisiert werden und beides losgelöst vom eigentlichen Algorithmus ist.

Die Operationen `(3)` bis `(5)` werden zunächst, vereinfachend als Berechnungsvorgang zusammengefasst. Hier erfolgt die eigentliche Rechenarbeit im Algorithmus. Operation `(6)` dient der Auf- und Nachbereitung der Simulationsergebnisse.

Vor Beginn der Parallelisierung sollen zunächst die einzelnen Komponenten des Algorithmus auf ihre Laufzeit untersucht werden, um so die rechenintensivsten Teile zu identifizieren. Dies ist für die bereits genannten Szenarien in Abb. 6.6 dargestellt.

Wie zu erwarten war, nimmt die *Berechnung* (Operationen `(3)` bis `(5)`) den Hauptteil der Rechenzeit in Anspruch. Darin erweist sich wiederum die Berechnung der Routen (Verbindungen) als am teuersten. Sie kostet im Schnitt zwischen 80 und 90% der Zeit. Dieser Teil bietet sich daher zunächst zur Parallelisierung an.

Bei der Betrachtung der Abhängigkeiten innerhalb des Algorithmus erkennt man, dass für jeden Quellknoten ein eigener Verbindungsbaum aufgebaut wird. Die Berechnung der einzelnen Verbindungsbäume ist unabhängig voneinander. Es werden lediglich die Informationen aus dem Vorverarbeitungsschritt benötigt. Dies sind die dort berechneten Verbindungssegmente. Dies motiviert eine einfache Parallelisierung der Routensuche: die zu berechnenden Verbindungsbäume müssen lediglich auf die verfügbaren Rechenknoten verteilt werden.

Die weiteren Teile des Algorithmus, die Verbindungswahl und die Fahrtenaufteilung (`(4)` und `(5)`) benötigen ebenfalls lediglich ihren jeweiligen Verbindungsbaum, der in der Rou-

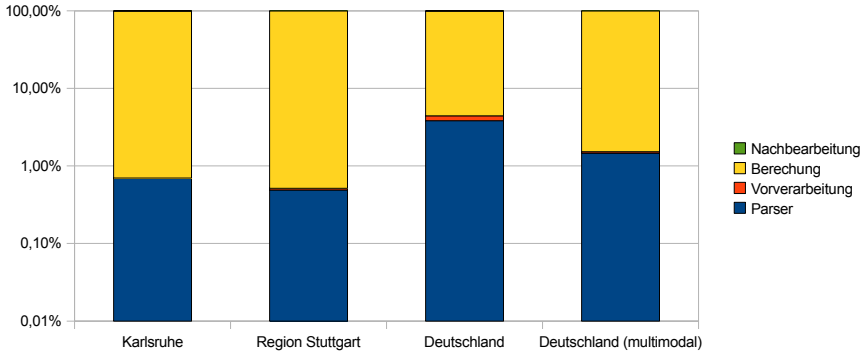


Abbildung 6.6: Rechenzeitverteilung bei Ausführung der fahrplanfeinen Umlegung für den Öffentlichen Verkehr bei verschiedenen Verkehrsnetzen.

tensuche erstellt wurde. Erst in Schritt `(6)` müssen alle Teilergebnisse, das heißt die Teilbelastungen der Verbindungen zu Gesamtbelastungen zusammengeführt werden.

Diese ersten Überlegungen zur Parallelisierung der fahrplanfeinen Umlegungen führen zu dem in Abb. 6.7 dargestellten parallelen Algorithmus, der im Rahmen dieser Arbeit entstand und auch in [63] beschrieben ist.

Hierbei laufen lediglich die Initialisierungen (Laden des Netzes und der Nachfragedaten) und die Vorverarbeitung sequentiell auf einem Rechenknoten ab. Am Ende dieser Schritte werden die aus der Vorverarbeitung gewonnenen Verbindungssegmente sowie die Nachfragedaten (QZ-Paare) an die restlichen Rechenknoten kommuniziert. Eine weitere Kommunikation erfolgt erst bei der Übertragung der Gesamtergebnisse. Die Rechenknoten empfangen die die Verbindungssegmente und ihre Arbeitsaufträge in Gestalt von QZ-Paaren. Für jedes dieser Paare werden nun auf einem Rechenknoten die Schritte der fahrplanfeinen Umlegung (Verbindungssuche, -wahl und Fahrtenaufteilung) durchgeführt. Die Zwischenergebnisse in Form belasteter Verbindungen werden temporär auf den Rechenknoten gespeichert. Sind keine weiteren Arbeitsaufträge mehr vorhanden, werden die Zwischenergebnisse aller Rechenknoten in einem Postprocessingschritt zusammengeführt und daraus die Gesamtbelastung des ÖV-Netzes berechnet.

Neben der Reduzierung der Rechenzeit ist auch eine Reduzierung des Speicherbedarfs Ziel einer Parallelisierung. Im vorgestellten Ansatz sind auf jedem Rechenknoten die Verbindungssegmente etc. repliziert. Allerdings erfordert der Branch & Bound Algorithmus die Existenz aller notwendigen Verbindungssegmente. Der Speicherbedarf ließe sich also nur durch einen erhöhten Kommunikationsaufwand zwischen den Rechenknoten vermindern, da immer wieder die notwendigen Teile der Gesamtmenge der Verbindungssegmente angefordert werden müssten.

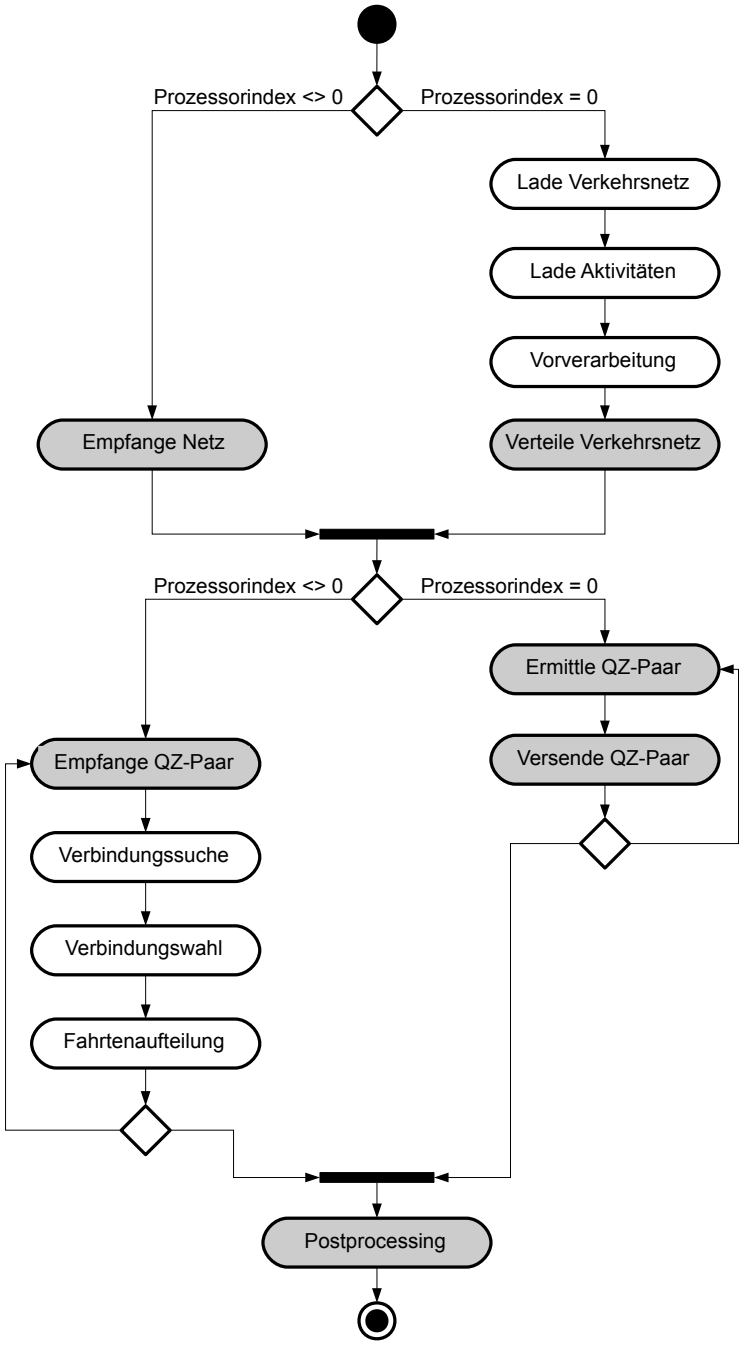


Abbildung 6.7: Ablauf der parallelen fahrplanfeinen Umlegung des Öffentlichen Verkehrs. Neben der Aufteilung auf mehrere Rechenknoten sind bei der parallelen Umlegung einige Schritte (grau hinterlegt) neu gegenüber dem sequentiellen Ablauf in Abb. 6.2 hinzugekommen.

Eine weitere Möglichkeit zur Effizienzsteigerung ist die Wiederverwendung von Teilverbindungsbäumen in den Verbindungsbäumen unterschiedlicher Quellknoten. Dieser würde den Aufwand des Aufbaus der Bäume reduzieren. Allerdings kann ein Teilverbindungsbaum nur genau dann wiederverwendet werden, wenn die Abfahrtszeit in der neuen Verbindung genau der durch den bisherigen Baum vorgegebenen entspricht. Andernfalls fehlen gültige Verbindungen. Außerdem wäre es notwendig, permanent eine global verfügbare Liste der vorhandenen Teilverbindungsbäume vorzuhalten. Dies würde zu einem erhöhten Kommunikationsaufwand führen.

Aus diesen Gründen erweist sich die vorgestellte Parallelisierung nach Abb. 6.7 als am besten geeignet, was sich in den Ergebnissen in Kapitel 9 zeigen wird.

6.3.2 Überlegungen zur Lastverteilung

Um einen wirklich effizienten parallelen Algorithmus zu erhalten, bedarf es allerdings noch weiterer Überlegungen. Ein wichtiger, verbleibender Punkt ist hierbei die Wahl einer geeigneten Lastverteilungsstrategie. Die Berechnungszeit einzelner Verbindungsbäume kann stark variieren. Dies ist unter anderem abhängig von der Tiefe der zu berechnenden Verbindungsbäume. Eine gleichmäßige Verteilung der Rechenlast ist auch erstrebenswert. Es sind dabei unterschiedliche Strategien vorstellbar, von denen die folgenden näher betrachtet werden sollen:

I. Statische Aufteilung der Nachfragedaten auf die einzelnen Rechenknoten

II. Statische Aufteilung der Nachfragedaten auf die einzelnen Rechenknoten unter Verwendung von Aufwandsindikatoren (beispielsweise die Anzahl der am Quellknoten abgehenden Verbindungen)

III. Dynamische Verteilung der Nachfragedaten durch einen zentralen Masterknoten

Strategie I verlangt die einmalige Verteilung der möglichen Quellknoten auf die Rechenknoten. Ist die Anzahl der zu verteilenden Quellknoten deutlich höher als die Anzahl der verfügbaren Rechenknoten, so kann man aufgrund des statistischen Durchschnitts davon ausgehen, wenn die Knoten zufällig verteilt werden, dass jeder Rechenknoten in etwa die gleiche Last bekommt. Je kleiner allerdings die Nachfragedaten sind, d.h. je weniger Verbindungsbäume berechnet werden müssen, desto mehr fallen die Unterschiede in der benötigten Rechenzeit ins Gewicht.

Bei Strategie II bedarf es zusätzlich einer Berechnung von Aufwandsindikatoren. Diese können auf vielfältige Weise gewählt werden. Ein Beispiel ist die Anzahl der am Quellknoten abgehenden Verbindungen. Man betrachtet also die Breite des Verbindungsbaums auf Tiefe 1. Anschließend muss die Nachfrage gemäß dieser Indikatoren neu sortiert und an die Rechenknoten verteilt werden. Bei dieser Strategie gilt die geeignete Wahl der Indikatoren als problematisch.

Die dynamische Verteilung nach Strategie III setzt auf das Master-Slave-Prinzip, d.h. es gibt einen aktiven Masterrechenknoten, der Arbeitsaufträge an die anderen Knoten (Slaves) vergibt. Die gesamte Nachfrage wird zunächst auf dem Masterknoten vorgehalten und dann schrittweise verteilt. Hat ein Rechenknoten seinen Arbeitsauftrag abgearbeitet, fordert er einen neuen Auftrag, also ein neues OD-Paar an. Auf diese Weise wird gewährleistet, dass die Last gleichmäßiger verteilt wird: Schnelle Rechenknoten oder Knoten mit kleinen Aufträgen arbeiten mehr Aufträge ab als andere.

In Abb. 6.7 ist die gewählte Lastverteilungsstrategie angedeutet, die auf dem Master-Slave-Prinzip beruht. Eine Entscheidung fiel zu Gunsten von Strategie III, da sie am flexibelsten auf unterschiedliche Szenarien (Verkehrsnetze, Verkehrsnachfrage und Rechnerinfrastruktur) reagieren kann. Allerdings kann es bei einer hohen Anzahl von Rechenknoten zu Engpässen am Masterknoten kommen. Wie in Kapitel 9 gezeigt werden wird, erweist sich dies bei den durchgeführten Simulationen noch nicht als Problem.

6.4 Zusammenfassung

In diesem Kapitel wurde eine Möglichkeit der makroskopischen Simulation von Öffentlichem Verkehr gezeigt. Die fahrplanfeine Umlegung, die auch in kommerziellen Produkten zum Einsatz kommt [135], erwies sich als ausgereifte, realistische Variante der Simulation. Ausgehend von einem vorhandenen Algorithmus dieses Verfahrens wurde eine Parallelisierungsstrategie mit geeigneter Lastverteilung entwickelt, welche erstmalig die Simulation detaillierter und großer Verkehrsnetze und Simulationsszenarien in vertretbarer Zeit erlaubt (siehe Kapitel 9).

7 Makroskopische Simulation des motorisierten Individualverkehrs

In Kapitel 6 wurden Möglichkeiten der makroskopischen Simulation des Öffentlichen Verkehrs aufgezeigt. Eine Simulation des Individualverkehrs kann auf ähnliche Weise erfolgen. Dieses Kapitel beschäftigt sich mit Umlegungsverfahren zur makroskopischen Simulation des Individualverkehrs. Es beginnt zunächst mit einer kurzen Vorstellung einer Auswahl dieser Umlegungsverfahren. Der Schwerpunkt liegt auf einer stochastischen Umlegung. Nach einer kurzen Beschreibung des sequentiellen Verfahrens, wendet sich das Kapitel verschiedenen Parallelisierungsstrategien hierfür zu, die im Rahmen dieser Arbeit entwickelt und implementiert wurden.

7.1 Umlegungsverfahren für den Individualverkehr

Man unterscheidet primär zwei verschiedene Arten von Umlegungsverfahren für den motorisierten Individualverkehr: *statische* und *dynamische Umlegungen*. Man spricht von statischer Umlegung, wenn der betrachtete Fluss auf den einzelnen Strecken unabhängig von der Zeit ist. Bei den dynamischen Verfahren wird explizit die Zeitachse in das Modell mit einbezogen. Hierbei wird der gesamte Umlegungszeitraum in einzelne Intervalle fester oder variabler Länge unterteilt. Anschließend werden die Belastungen und Widerstände im Verkehrsnetz unabhängig voneinander für jedes Intervall berechnet. Die Nachfrage (oder auch die Fahrten) wird dann auf dasjenige Intervall, in welchem der Abfahrtszeitpunkt liegt, gemäß eines der in Kapitel 2 genannten Aufteilungsmodelle verteilt. Damit ergibt sich eine Routenwahl und -belastung, die tageszeitlichen Schwankungen unterworfen ist.

Abb. 7.1 zeigt eine Übersicht über verschiedene Methoden der Umlegung für den motorisierten Individualverkehr. Exemplarisch sollen einige Verfahren zur statischen Umlegung skizziert werden. Die dynamischen Verfahren sind nicht Gegenstand dieser Arbeit und werden daher nicht weiter betrachtet. Ein Überblick über beide Umlegungsklassen ist in

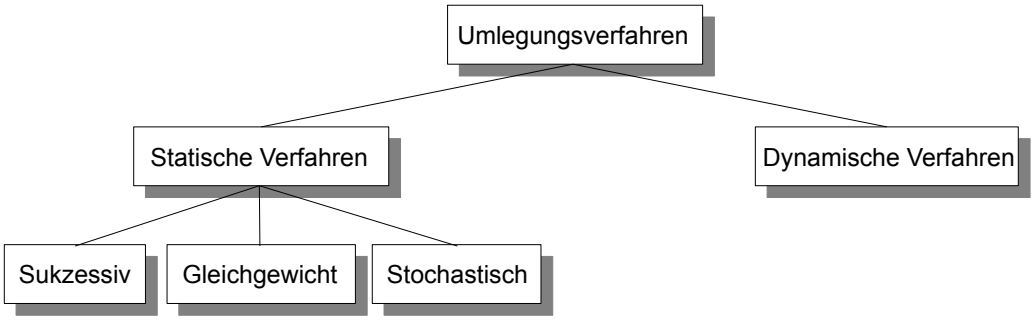


Abbildung 7.1: Auswahl einer Übersicht über Umlegungsverfahren für den motorisierten Individualverkehr.

[135, 67] und in [50] sowie in [171] zu finden.

7.1.1 Sukzessivverfahren

Beim Sukzessivverfahren wird schrittweise das zunächst leere Verkehrsnetz immer weiter gefüllt, indem neue Fahrzeuge hinzugefügt werden. Am Anfang werden für das leere Verkehrsnetz die Widerstände der einzelnen Netzelemente ermittelt.

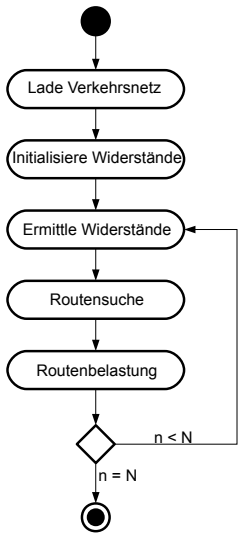


Abbildung 7.2: Ablauf der sukzessiven Umlegung für den motorisierten Individualverkehr.

Auf Grundlage dieser Widerstände wird dann die Routensuche und die Routenbelastung durchgeführt. Mit Hilfe der ermittelten Belastungen müssen die Widerstände der Netzele-

mente angepasst werden. Dies geschieht bei der *Ermittlung der Widerstände*. In dem nun neu belasteten Netz beginnt die Berechnung von vorne. Das Verfahren bricht ab, sobald eine vom Benutzer vorgegebene maximale Anzahl von Iterationen N erreicht wird. Der Ablauf dieses Verfahrens ist in Abb. 7.2 dargestellt. Ein großer Nachteil dieses Verfahrens ist das Abbruchkriterium. So bricht das Verfahren nach n Iterationen ab, ohne dass eine Qualitätskontrolle des Ergebnisses stattfindet.

7.1.2 Gleichgewichtsverfahren

Die Gleichgewichtsverfahren verfolgen einen etwas anderen Ansatz. Sie versuchen, ausgehend von einer Initiallösung, die Nachfrage nach dem Ersten Wardrop'schen Prinzip [166], dass in Kapitel 2 vorgestellt wurde, aufzuteilen. Dabei ist ein Verkehrsteilnehmer bestrebt, seine Route so zu wählen, dass der Widerstand auf allen anderen Alternativrouten gleich ist und dementsprechend ein Wechsel auf eine andere Route keine Verbesserung für den Verkehrsteilnehmer hinsichtlich seiner Reisezeit bringen würde.

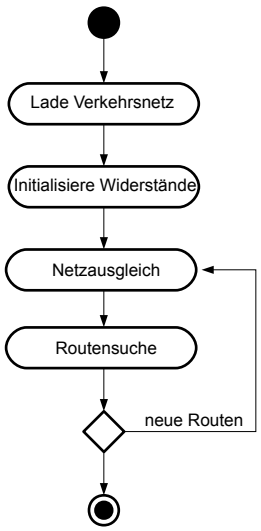


Abbildung 7.3: Ablauf des Gleichgewichtsverfahren für den motorisierten Individualverkehr.

Abb. 7.3 zeigt den schematischen Ablauf eines Gleichgewichtsverfahrens. Bei diesem Verfahren ist der Schritt *Netzausgleich* ebenfalls ein iterativer Prozess, in welchem immer paarweise je zwei Routen eines QZ-Paares miteinander verglichen werden. Es wird dabei versucht, durch eine Verschiebung von Fahrzeugfahrten ein Gleichgewicht herzustellen. In der äußeren Schleife wird so lange iteriert bis keine neuen Routen mehr gefunden werden. Dann ist ein stabiler, ausgeglichener Netzzustand erreicht. Man spricht von einem Gleichgewicht.

7.1.3 Ein Stochastisches Umlegungsverfahren für den Individualverkehr

Neben den im vorherigen Abschnitt vorgestellten Umlegungsverfahren existiert noch eine weitere, wichtige Gruppe von Verfahren: die stochastischen Umlegungen. Bei dieser Art der Umlegung kommt eine Wahrscheinlichkeitskomponente ins Spiel. Die Stochastik wird lediglich genutzt, um bei der Routensuche Alternativen zu erzeugen. Dies soll das individuelle Entscheidungsverhalten der Verkehrsteilnehmer simulieren. Jeder Verkehrsteilnehmer besitzt individuelle Vorlieben bei einer Routenwahl. Beispielsweise kann es durchaus möglich sein, dass ein Fahrer nicht immer den für ihn an sich günstigsten Weg zur Arbeit wählt, sondern aus bestimmten Gründen lieber einen kurzen Umweg in Kauf nimmt oder über keine vollständige Kenntnis der Verkehrssituation verfügt.

Die Generierung der Alternativen wird durch eine Variation der Widerstände aller Netzelemente erreicht. Bei einer erneuten Routensuche werden nun weitere Routen gefunden, die zuvor aufgrund ihrer Kosten nicht ausgewählt wurden.

In Abb. 7.4 ist in vereinfachter Form der Ablauf einer stochastischen Umlegung des motorisierten Individualverkehrs skizziert, welcher ausgereift und erprobt auch in kommerzieller Simulationssoftware Anwendung findet [135].

Das Verfahren gliedert sich in zwei Schleifen: eine innere und eine äußere Schleife. Nach dem Laden der Verkehrsnetzstruktur, der Nachfragedaten und der Initialisierung aller Netzelemente erfolgt zunächst in einer äußeren Schleife die Routensuche. Dabei werden, wie bereits geschildert, die Widerstände aller Netzelemente mehrfach gemäß einer Wahrscheinlichkeitsverteilung (beispielsweise Normalverteilung) variiert und anschließend die Routenberechnung neu angestoßen. Auf diese Weise werden für jedes Quelle-Ziel-Paar der Nachfragedaten eine Menge an Alternativrouten berechnet.

Dieses Verfahren der mehrfachen Routenberechnung ist ähnlich der Suche nach den k-kürzesten Wegen in einem Graphen [23, 132, 144]. Bei der Berechnung der k-kürzesten Wege taucht allerdings das Problem der Ähnlichkeit der Wegalternativen auf: Wege können gemeinsame Teilabschnitte besitzen. Bei der stochastischen Umlegung führen diese Routen zu einem Anstieg der Gesamtroutenmenge, ohne dass sie einen größeren Einfluss auf die Belastung des Verkehrsnetzes haben werden. Wünschenswert sind daher Wege bzw. Routen, die sich mindestens um einen vom Benutzer definierten Wert unterscheiden.

In dem hier betrachteten Verfahren wird dieses Problem durch einen *Umwegetest* gelöst. Dies geschieht folgendermaßen: Betrachtet man zwei Routen R_1 und R_2, die sich lediglich in den Teilstrecken U_1 und U_2 unterscheiden, so wird R_2 verworfen, wenn die Länge von U_2 länger ist als diejenige von U_1 multipliziert mit einem individuell einstellbaren Faktor.

Nach der erfolgreichen Routensuche betritt das Verfahren die innere Schleife, in welcher die Routenwahl und die Routenbelastung stattfinden. Diese Aufteilung der Nachfrage erfolgt nach einem der in Kapitel 2 vorgestellten Aufteilungsmodelle, beispielsweise Kirchhoff oder Logit. Die eigentliche Aufteilung der Nachfrage erfolgt bei der *Routenwahl*. Gemäß

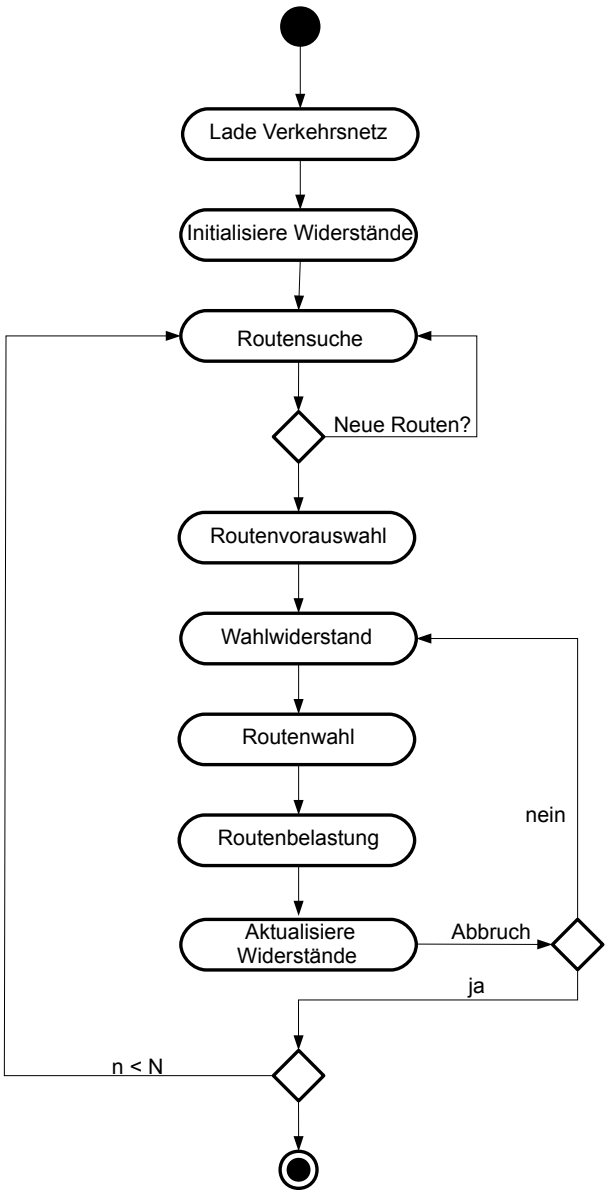


Abbildung 7.4: Ablauf der stochastischen Umlegung für den motorisierten Individualverkehr.

der dort ermittelten Aufteilung werden anschließend die *Routen belastet*. Bei dieser Aufteilung ist allerdings ebenfalls auf die Ähnlichkeit der Routen zu achten [135]. Verwendet man den C-Logit-Ansatz[1], so kann man den von Cascetta [29, 31] vorgeschlagenen *Commonality Faktor* $C_{i,j}$ verwenden, der für jedes mögliche Routenpaar berechnet werden muss. Dieser gibt die Ähnlichkeit zweier Routen i und j an.

$$C_{i,j} = \frac{t_{i,j}}{\sqrt{t_i \cdot t_j}} \tag{7.1}$$

Dabei ist $t_{i,j}$ die Zeit auf den gemeinsamen Abschnitten der beiden Routen und t_i die Zeit auf Route i. Der Wert von $C_{i,j}$ bewegt sich folglich innerhalb des Intervalls $[0,1]$ und nimmt bei Identität der beiden Routen den Wert 1 an, bzw. den Wert 0, wenn die beiden Routen keine gemeinsamen Abschnitte haben.

Um die Aufteilung durchführen zu können, muss man mittels des Commonality Faktors die *Eigenständigkeit* E_i der Route i zu allen anderen Routen j berechnen. Je näher der Wert der Eigenständigkeit einer Route bei 1 liegt, desto weniger hat diese Route mit den anderen gemeinsam.

$$E_i = \frac{1}{\sum_j C_{i,j}} \tag{7.2}$$

Dann ergibt sich der prozentuale Anteil P_i der Nachfrage für die Route i bei dem Logit-Aufteilungsmodell als

$$P_i = \frac{e^{w_i} \cdot E_i}{\sum_{j=1}^{N} (e^{w_j} \cdot E_j)}, \tag{7.3}$$

Wobei w_i und w_j die Widerstände der jeweiligen Routen sind. Im Kirchhoff-Aufteilungsmodell berechnet sich P_i zu

$$P_i = \frac{w_i \cdot E_i}{\sum_{j=1}^{N} (w_j \cdot E_j))}. \tag{7.4}$$

Nach der Aufteilung der Fahrten – der Routenbelastung – wird die innere Schleife solange wiederholt bis die Belastungen unter einen vordefinierten Schwellwert konvergieren. Anschließend, nach Verlassen der inneren Schleife, werden die Widerstände aller Netzelemente gemäß der Routenbelastungen aktualisiert. Das auf diese Weise aktualisierte und veränderte Verkehrsnetz weist nun andere Charakteristika auf als das aus der vorherigen Iteration der äußeren Schleife. Auf Grund der neuen Widerstände der Netzelemente können sich neue Routen bei der Routensuche ergeben, das heißt Fahrer würden aufgrund neuer Verkehrsauslastungen andere Routen für ihre Fahrt wählen. Dementsprechend wird eine neue Iteration der äußeren Schleife durchgeführt. Ein Abbruch erfolgt, wenn (1) keine weiteren Routen mehr gefunden werden oder (2) eine vordefinierte Anzahl von Iterationen erreicht ist.

[1] Der Logit-Ansatz hat einige Probleme bei der Aufteilung der Nachfrage mit Pfaden, die Teilstrecken gemeinsam haben. Der C-Logit-Ansatz löst dieses Problem.

7.2 Parallelisierung der stochastischen Umlegung

Wie aus der Beschreibung des Ablaufs der stochastischen Umlegung im vorherigen Abschnitt ersichtlich ist, steigt der Rechenaufwand und der Speicherplatzbedarf bei diesem Verfahren bei wachsender Größe der Verkehrsnetze beziehungsweise der Nachfragedaten stark an. Je größer das Verkehrsnetz und desto mehr Quelle-Ziel-Paare existieren, umso aufwändiger gestaltet sich die Routensuche und desto mehr Routen müssen im Speicher gehalten werden.

Dennoch gilt wie in den vorherigen Kapiteln schon, dass man für realistischere Simulationen großer Verkehrsnetze mit detaillierter Nachfrage berechnen möchte. Hinzu kommen Simulationen möglicher, noch größerer Anwendungsszenarien, die heute noch nicht denkbar, aber wünschenswert sind. Dies könnten beispielsweise detailgenaue Simulationen des gesamten Straßenverkehrs eines Landes oder der EU sein.

Eine Möglichkeit, diese Wünsche zu befriedigen, besteht in einer Parallelisierung des bestehenden sequentiellen Verfahrens. Dabei gilt es zunächst die Laufzeiten der einzelnen Teile des Algorithmus zu ermitteln und auf diese Weise die rechenintensivsten zu bestimmen, um daraus eine geeignete Parallelisierungsstrategie abzuleiten.

Listing 7.1 zeigt den vereinfachten Algorithmus der stochastischen Umlegung wie er schon in Abb. 7.4 skizziert wurde.

```
(0) Initialisiere Widerstand aller Netzelemente
for ( n = 0; n < N; n += 1 )
   (1) Berechne Routen fuer alle Quelle-Ziel-Paare
   (2) if Anzahl neuer kuerzester Routen = 0 then break
   (3) Routenvorauswahl
   for ( m = 0; m < M; m += 1 )
      (4) Berechne Widerstand der Routen
      (5) Teile Nachfrage auf Alternativrouten auf
      (6) Aktualisiere Widerstaende der Netzelemente
      (7) if Abweichung der Belastungen der Netzelemente
             aus letztem Schritt zu klein then break
   end for
end for
```

Listing 7.1: Vereinfachter Algorithmus der stochastischen Umlegung.

Bei der Routensuche (Operation `(1)`) werden für jedes Quelle-Ziel-Paar der Fahrtenmatrix mehrere Routen berechnet. Die Routensuche erfolgt dabei mit einem Algorithmus zur Kurzwegsuche, der die Besonderheiten eines Verkehrsnetzes berücksichtigt (siehe Kapitel 2). Nachdem eine Route berechnet wurde, werden die Widerstände der Netzelemente variiert und eine neue Routensuche angestoßen. Auf diese Weise erhält man mehrere Alternativrouten.

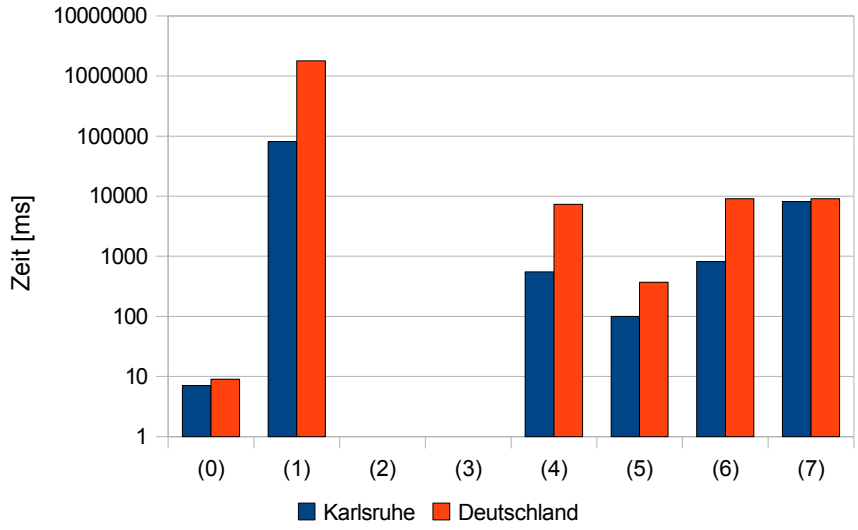


Abbildung 7.5: Rechenzeitverteilung bei Ausführung der stochastischen Umlegung für den motorisierten Individualverkehr Verkehr bei verschiedenen Verkehrsnetzen. Die Bezeichner der x-Achse geben, die Schritte des Algorithmus wieder.

Schritt `(2)` stellt lediglich das Abbruchkriterium für die äußere Schleife dar, d.h. wenn in Schritt `(1)` keine neuen Routen gefunden werden, wird die Ausführung beendet.

In der in Schritt `(3)` stattfindenden Routenvorauswahl werden all diejenigen Routen verworfen, deren bisheriger Widerstand im Vergleich zu anderen Routen des selben Quelle-Ziel-Paars deutlich höher ist. In Schritt `(4)` wird der neue Widerstand der Routen auf die Summe der Widerstände der überfahrenen Netzelemente gesetzt. In Schritt `(5)` sind die Routenwahl und die Routenbelastung zusammengefasst: Die Nachfrage jedes Quelle-Ziel-Paars wird gemäß eines Aufteilungsmodells auf die gefundenen Alternativrouten verteilt und die Belastung der Routen wird berechnet. Die Widerstände der Netzelemente werden gemäß der Routenbelastungen in Schritt `(6)` aktualisiert.

Schließlich markiert Schritt `(7)` das Abbruchkriterium der inneren Schleife. Sie wird abgebrochen, wenn sich die Belastungen der Netzelemente und der Routen im Vergleich zur letzten Iteration nicht mehr signifikant geändert haben.

Betrachtet man nun unter diesen Aspekten die Laufzeitverteilung bei der Berechnung der Umlegung für verschiedene Szenarien, so ergibt sich das in Abb. 7.5 dargestellte Ergebnis. Dabei wird deutlich, dass in beiden Fällen, dem relativ kleinen Verkehrsnetz von Karlsruhe und dem deutlich größeren Verkehrsnetz Deutschlands, die Routensuche (Schritt `(1)`), den deutlich höchsten Rechenaufwand in sich birgt. Die Schritte `(2)` und `(3)` sind vernachlässigbar schnell und laufen in praktisch nicht messbarer Zeit ab.

Diese Ergebnisse motivieren eine mögliche Beschleunigung der Routensuche. Im Rahmen

	Verkehrsnetz		Routen	
	Lesen	Schreiben	Lesen	Schreiben
(0)	√	√	–	–
(1)	√	–	–	√
(2)	–	–	–	–
(3)	–	–	√	√
(4)	√	–	√	√
(5)	–	–	√	√
(6)	√	√	√	–
(7)	√	–	√	√

Tabelle 7.1: Zugriff auf die Datenstrukturen in den Schritten des Algorithmus.

dieser Arbeit entstanden zwei Strategien zur Parallelisierung dieses Umlegungsverfahrens, die sowohl die Minimierung der Rechenzeit als auch des Speicherbedarfs zum Ziel hatten. Beide Strategien wurden im Rahmen dieser Arbeit entwickelt, implementiert und auch in [91] weiter untersucht. Im ersten Verfahren erfolgt eine *Aufteilung der Fahrtenmatrix*, also der QZ-Paare. Das zweite behandelt Methoden der Verkehrsnetzpartitionierung und daran angepasster Suchstrategien, das heißt es wird eine räumliche Aufteilung des Verkehrsnetzes untersucht.

7.2.1 Aufteilung der Fahrtenmatrix

Betrachtet man die wichtigsten Datenstrukturen, die für die Parallelisierung notwendig sind, so sind dies das Verkehrsnetz, die Nachfragedaten in Form der Fahrtenmatrix und die berechneten Routen. Jeder dieser Strukturen sind Attribute zugeordnet, die sich in *statische* (Maximalgeschwindigkeit einer Strecke, ...) und *dynamische Attribute* (Auslastung einer Strecke, Widerstand eines Netzelements, ...) aufteilen lassen. Die statischen Attribute, die zum großen Teil den Netzelementen zuzuordnen sind, bleiben während der gesamten Simulation unverändert, während sich die dynamischen im Laufe der Simulation ändern können.

Nicht in jedem Schritt der Simulation wird Zugriff auf alle Datenstrukturen benötigt. Wichtige Elemente sind das Verkehrsnetz und die berechneten Routen. Die Fahrtenmatrix kann hier vernachlässigt werden, da auf sie stets nur lesend zugegriffen wird. Um eine geeignete Parallelisierungsstruktur zu entwickeln, bedarf es einer Analyse auf welche dieser Strukturen wann und wie zugegriffen wird (siehe Tabelle 7.1).

Daraus lässt sich eine erste Parallelisierungsstrategie ableiten, die das Aufteilen der Fahrtenmatrix, das heißt eine parallele Bearbeitung der Quelle-Ziel-Paare durch mehrere Rechenknoten vorsieht. Die umgesetzte Strategie ist in Abb. 7.6 dargestellt.

Die parallelisierte Umlegung läuft dabei in den folgenden Schritten ab. Zunächst werden das Verkehrsnetz und die Fahrtenmatrix auf einem Rechenknoten (beispielsweise demjenigen mit der Prozessor-ID 0) geladen und initialisiert. Es wird dabei Schritt `(0)` des Al-

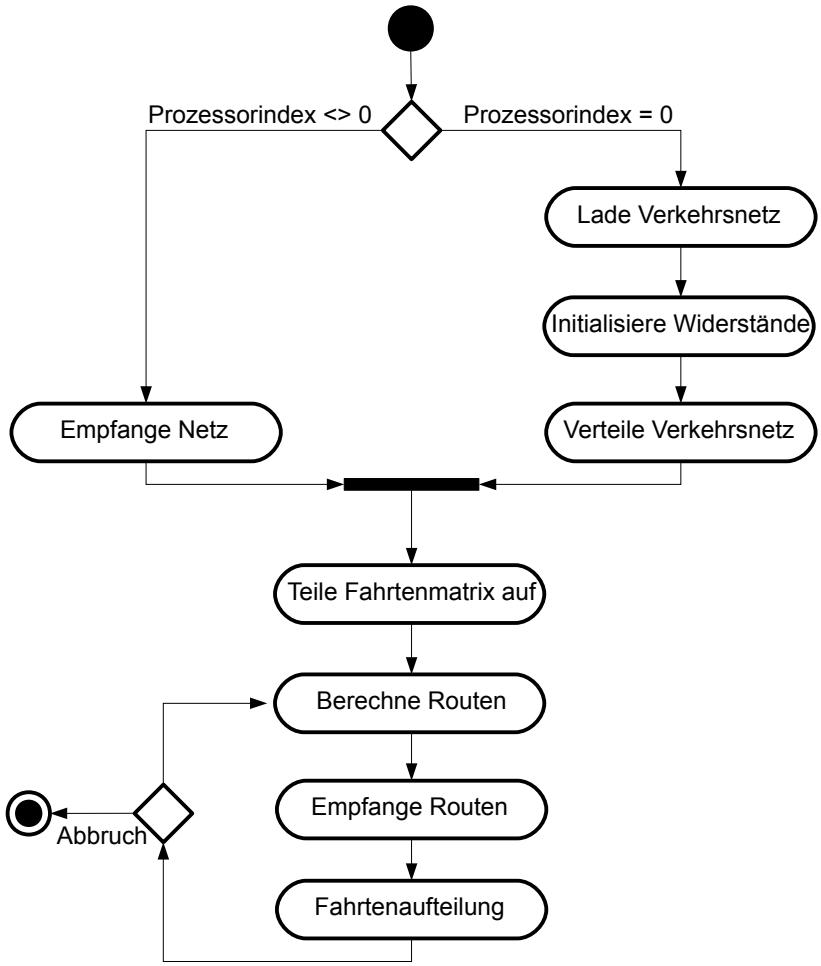


Abbildung 7.6: Ablauf der parallelisierten stochastischen Umlegung bei Anwendung der Aufteilung der Fahrtenmatrix.

gorithmus ausgeführt. Anschließend wird das gesamte Verkehrsnetz mit den berechneten (initialisierten) Widerständen auf alle anderen Rechenknoten verteilt, was im Schritt *Verteile Verkehrsnetz* geschieht.

Als nächstes wird die Fahrtenmatrix auf die Rechenknoten aufgeteilt. Dies kann entweder statisch oder dynamisch erfolgen – analog zu den Methoden, die in Abschnitt 6.3.2 diskutiert wurden. Eine möglichst gleichmäßige Verteilung ist auch hier wünschenswert.

Nun wird die Routensuche (Schritt `(1)`) auf allen beteiligten Rechenknoten gleichzeitig durchgeführt. Jeder Rechenknoten berechnet dabei die Routen für seinen Teil der Fahrtenmatrix. Nach der Berechnung müssen diese Routen synchronisiert werden, d.h. alle Routen müssen allen Rechenknoten bekannt sein.

Abschließend wird auf jedem Rechenknoten die innere Schleife des Algorithmus ausgeführt und dabei die Fahrtenaufteilung vorgenommen.

Zur Lastverteilung wurde erneut eine Master-Slave-Methode gewählt – wie es in Abb. 7.6 bereits angedeutet ist. Eine statische Lastverteilung ist bei dieser Art der Umlegung noch etwas kritischer als bei der fahrplanfeinen des Öffentlichen Verkehrs.

Bei dem gerade beschriebenen Verfahren zur Parallelisierung der Umlegung wird gezielt nur die Routensuche beschleunigt, da jeder Rechenknoten nur Routen für einen Teil der Quelle-Ziel-Paare berechnen muss. Alle anderen Schritte werden redundant ausgeführt. Allerdings adressiert diese Parallelisierungsstrategie noch nicht das Problem des Speicherplatzverbrauchs durch das Verkehrsnetz und die Vorhaltung der berechneten Routen.

7.2.2 Partitionierung des Verkehrsnetzes

Eine weitere Möglichkeit, die stochastische Umlegung des Individualverkehrs zu beschleunigen, besteht darin, eben jenen Speicherplatzverbrauch und seine Ursachen zu betrachten. Der Hauptspeicher ist, ebenso wie die Laufzeit, ein limitierender Faktor: Er begrenzt die effiziente Berechnung von größeren Szenarien mit detaillierten Nachfragedaten.

Wenn das Verkehrsnetz jedoch nicht komplett im Hauptspeicher gehalten werden muss, kann man dadurch größere Szenarien simulieren. Dazu müssen die Verkehrsnetze auf mehrere Rechenknoten aufgeteilt werden. Eine solche Partitionierung der Daten ist in anderen Anwendungsgebieten der numerischen Simulation bereits üblich [59].

Teilt man das Verkehrsnetz auf, so führt jeder Rechenknoten die stochastische Umlegung nur noch auf einem Teil des Netzes aus. Es entstehen zwangsläufig Verbindungskanten zwischen den einzelnen Partitionen, da ein Verkehrsnetz ein zusammenhängender Graph ist. Diese Verbindungskanten (Schnittkanten) sind Orte, an denen Nachrichtenaustausch zwischen den einzelnen, benachbarten Partitionen durchgeführt werden muss. Dies ist insbesondere bei der Routensuche der Fall, wenn sich Start und Ziel in unterschiedlichen Partitionen befinden.

Daraus lässt sich die zweite Strategie ableiten. Geht man davon aus, dass das Verkehrsnetz auf irgendeine Weise bereits partitioniert wurde, so gestaltet sich der Ablauf dieser Strategie wie folgt.

Es existiert kein Rechenknoten, der die Berechnung als Master zentral steuert. Alle Rechenknoten sind bei dieser Parallelisierung gleichberechtigt. Alle Schritte der Umlegung müssen zwangsläufig parallel ausgeführt werden, da kein Rechenknoten Zugriff auf die gesamten Datenstrukturen besitzt, sondern immer nur auf ausgewählte Partitionen davon. Es wird durch eine Synchronisation gewährleistet, dass sich zu jedem Zeitpunkt alle Rechenknoten im gleichen Schritt der Umlegung befinden.

Jeder Rechenknoten speichert einen Teil des gesamten Verkehrsnetzes sowie alle in Schritt `(1)` gefundenen Routen und die Fahrtenmatrix. Die Schritte `(3)` und `(5)`, die lediglich die Routenbehandlung betreffen, werden in dieser gewählten Strategie redundant ausgeführt. In diesen Schritten findet die Routenvorauswahl und die Aufteilung der Nachfrage auf die Alternativrouten statt. Da diese Schritte nur einen recht geringen Anteil an der Gesamtlaufzeit des Algorithmus haben, wird in der hier vorgestellten Strategie auf deren Parallelisierung und damit auf eine Aufteilung der Fahrtenmatrix und der Routen verzichtet.

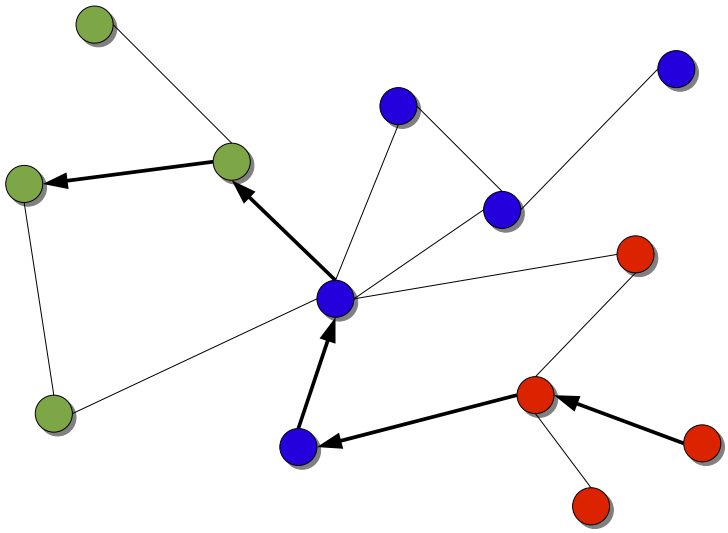


Abbildung 7.7: Eine Route erstreckt sich über drei verschiedene Partitionen.

In Schritt `(4)` wird der Widerstand jeder Route berechnet. Dazu müssen wie im sequentiellen Fall die Einzelwiderstände aller überfahrenen Netzelemente summiert werden. Dieser Schritt erfordert aufgrund der partitionierten Daten eine Kommunikation der beteiligten Rechenknoten: Eine Route kann sich über mehrere Partitionen erstrecken. Dies ist in Abb. 7.7 veranschaulicht, wo sich eine Route über drei Partitionen verteilt.

Fasst man dies alles zusammen, so ergibt sich folgender Ablauf (siehe Abb. 7.8). Als erstes erfolgt in Schritt `(0)` die *Initialisierung der Widerstände*. Jeder Rechenknoten initialisiert

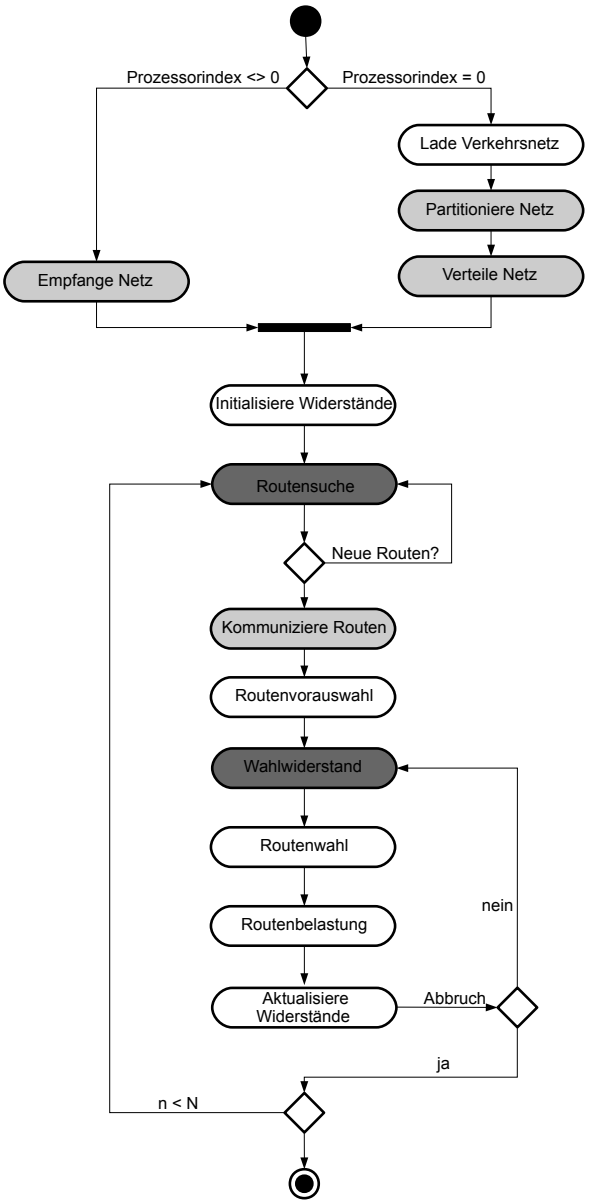


Abbildung 7.8: Ablauf der parallelen stochastischen Umlegung bei partitioniertem Verkehrsnetz. Die grauen Schritte sind gegenüber der sequentiellen Variante neu hinzugekommen beziehungsweise wurden verändert. Die dunkelgrauen Schritte laufen verteilt über alle beteiligten Rechenknoten ab.

dabei die Widerstände der Netzelemente, die zu seiner Partition gehören. In Schritt `(1)` erfolgt die *Routensuche*, die in Form einer verteilten Kurzwegsuche durchgeführt wird (siehe Abschnitt 7.2.4). Nach erfolgter Routensuche werden die gefundenen Routen an alle Rechenknoten kommuniziert.

Das *Abbruchkriterium der äußeren Schleife* in Schritt `(2)` kann lokal von jedem Rechenknoten durchgeführt werden, der dabei eine lokale Entscheidung trifft, ob er neue Routen gefunden hat oder nicht. Wie bereits weiter oben erwähnt wird Schritt `(3)`, die *Routenvorauswahl*, von allen Rechnern redundant durchgeführt. Dieser Schritt läuft auf jedem Rechenknoten deterministisch ab. In der Phase des *Wahlwiderstands* (Schritt `(4)`) wird der Widerstand jeder Route über eine verteilte Aggregation der überfahrenen Netzelemente berechnet.

Auch Schritt `(5)`, die *Routenbelastung*, wird von jedem Rechenknoten redundant ausgeführt. Jeder Rechenknoten aktualisiert in Schritt `(6)` die Widerstände seiner lokalen Netzelemente selbst. Alle Rechnerknoten ermitteln für ihre Netzelemente und Routen in Schritt `(7)`, ob das *Abbruchkriterium der inneren Schleife* erfüllt ist.

Nach der Betrachtung des Grundablaufs dieser Parallelisierungsstrategie, die die Durchführung der Partitionierung und der verteilten Kurzwegsuche ausgeklammert hatte, werden im Folgenden verschiedene Techniken der Partitionierung der Verkehrsnetze betrachtet, die im Rahmen dieser Arbeit analysiert und angewendet wurden. Abschließend endet die Vorstellung dieser Strategie mit einer Beschreibung der Verfahren zur verteilten Kurzwegsuche.

Rekursive Koordinatenbisektion

Ziel der Partitionierung ist es, das Verkehrsnetz – also einen Graphen – in p ungefähr gleich große Partitionen aufzuteilen. Es ist dabei wichtig, die Anzahl der Schnittkanten – die Kanten, die die Partitionen miteinander verbinden – möglichst minimal zu halten, denn an diesen Kanten müssen bei der Umlegung Daten ausgetauscht werden. Den Aufwand der Kommunikation gilt es dabei so gering wie möglich zu halten, was mit der Zahl der Schnittkanten korreliert.

Abb. 7.9 veranschaulicht an einem einfachen Beispiel, wie die Wahl der Partitionen die Zahl der Schnittkanten beeinflussen kann. Dort ist derselbe Graph auf zwei unterschiedliche Weisen partitioniert. Links ist durch eine geschickte Wahl der Partitionen die Anzahl der Schnittkanten minimal.

Wie schon in Kapitel 2 angesprochen, ist die Berechnung der Partitionierung eines Graphen mit minimaler Schnittkantenzahl allgemein ein NP-vollständiges Problem. Es existieren eine Vielzahl von Algorithmen, die in Näherung gute Ergebnisse bringen. In [46] und [55] wird jeweils ein Überblick über bestehende Verfahren gegeben.

Eine einfache und schnelle Möglichkeit besteht darin, den Graphen rekursiv immer wieder in zwei Teile aufzuteilen bis die gewünschte Anzahl an Partitionen p erreicht ist. Dies setzt

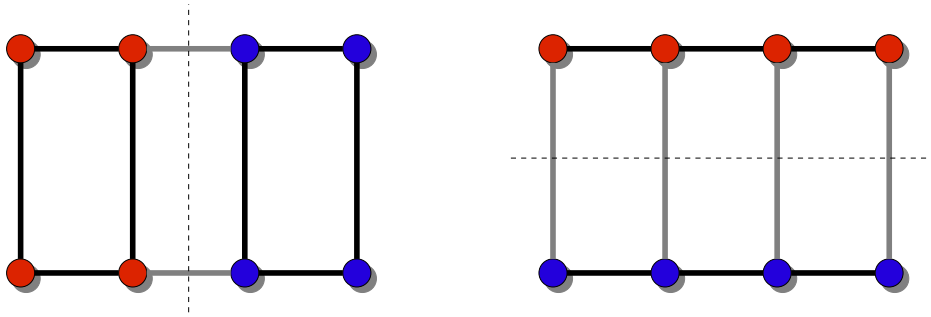


Abbildung 7.9: Die Zahl der Schnittkanten ist abhängig von der Wahl der Partitionen: gut gewählte Partitionierung mit zwei Schnittkanten (links) und eine schlecht gewählte Partitionierung mit vier Schnittkanten (rechts).

allerdings voraus, dass die angestrebte Menge der Partitionen einer Zweierpotenz entspricht ($p = 2^n$ mit $n \in \mathbb{N}$). Das Vorgehen ist in Abb. 7.10 veranschaulicht.

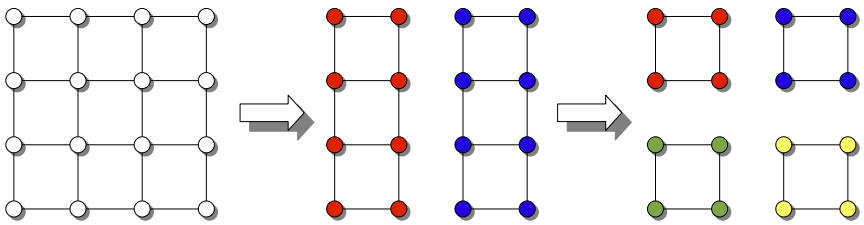


Abbildung 7.10: Grundprinzip der rekursive Koordinatenbisektion.

Dieses Verfahren wird, wenn man geometrische beziehungsweise geographische Informationen hinzunimmt, auch als *rekursive Koordinatenbisektion* bezeichnet und verfolgt eine *Divide-and-Conquer*-Strategie. Verkehrsnetze enthalten geographische Positionen der Knoten in Form von x- und y-Koordinaten. Die Bisektion entscheidet sich für eine Koordinatenachse. Orthogonal zu dieser Achse werden die Knoten des Verkehrsnetzes in zwei gleichgroße Mengen aufteilt. Hierfür berechnet man den Median der Koordinatenwerte aller Knoten. Damit wird die Knotenmenge halbiert [10].

Allerdings entsteht ein Problem, wenn man stets die gleiche Koordinatenachse für die Bisektion wählt. Diese Methode führt oft zu einer sehr hohen Schnittkantenzahl, da der Graph in sehr dünne Streifen mit langen Grenzen aufgeteilt wird. Dieses Problem lässt sich umgehen, indem man in jedem Schritt diejenige Achse auswählt, die zur geringeren Anzahl von Schnittkanten führt. Dies erfordert zusätzlichen Berechnungsaufwand, den man dadurch vermeiden kann, dass stets abwechselnd die beiden Achsen gewählt werden. Im Mittel erreicht man dadurch gute Ergebnisse, in Einzelfällen aber auch weit vom Optimum entfernte Lösungen. Diese Ungenauigkeit steht jedoch eine hohen Ausführungs- und Berechnungs-

Abbildung 7.11: Vier Partitionen nach der Anwendung der rekursiven Koordinatenbisektion auf das Verkehrsnetz von Karlsruhe.

geschwindigkeit gegenüber [153].

Die Abbildungen 7.11 und 7.12 zeigen die Anwendung der rekursiven Koordinatenbisektion auf das Verkehrsnetz von Karlsruhe, wobei Netze mit vier beziehungsweise 128 Partitionen entstehen.

Diese vorgestellte sequentielle Variante der rekursiven Koordinatenbisektion kann für das Framework eingesetzt werden. Allerdings hilft sie nicht bei der Lösung des Problems mit sehr großen Verkehrsnetzen, die nicht mehr vollständig in den Hauptspeicher passen. Denn mindestens ein Rechenknoten muss das vollständige Netz vorhalten, um die Bisektion durchzuführen.

Dies motiviert den Einsatz einer parallelen beziehungsweise verteilten Variante dieses Verfahrens, bei der es nicht erforderlich ist, dass ein Rechenknoten das gesamte Netz vorhalten muss. Bei der Umsetzung einer solchen Parallelisierung hilft es, dass die Bisektion für die Partitionierung nicht alle Informationen eines Verkehrsnetzes benötigt. Es sind lediglich die Knoten mit ihren geographischen Koordinaten von Interesse. Die Strecken und Abbiegebeziehungen können für die Partitionierung zunächst vernachlässigt werden. Es genügt, diese nach erfolgter Partitionierung an die entsprechenden Rechenknoten nachträglich zu kommunizieren.

Abb. 7.13 zeigt den Ablauf der parallelen rekursiven Koordinatenbisektion. Als erstes findet in einer *Initialisierungsphase* das Einlesen aller Knoten und ihrer Koordinaten statt. Diese Knoten werden zufällig auf die beteiligten Rechenknoten verteilt. Es wird versucht, die

Abbildung 7.12: 128 Partitionen nach der Anwendung der rekursiven Koordinatenbisektion auf das Verkehrsnetz von Karlsruhe.

Anzahl der Knoten auf den Rechenknoten möglichst gleich zu halten. Das Ergebnis dieses Schritts ist in Abb. 7.13 (1) dargestellt.

Anschließend erfolgt die *Berechnung des Medians* der Koordinatenwerte aller Knoten (Abb. 7.13 (2)). Für die Berechnung des Median wird ein paralleler Algorithmus verwendet, der den Median der über alle Rechenknoten verteilten Koordinatenwerte berechnet. Der von Bader und Joseph in [10] vorgestellte Algorithmus zur Bestimmung des i-kleinsten Elements aus einer Menge von n Elementen ermöglicht dies und gewährleistet zusätzlich, dass die Kommunikation zwischen dem Rechenknoten nahezu minimal gehalten wird. Dieser Algorithmus wendet eine Divide-and-Conquer-Technik an, wobei die Menge der n Elemente in zwei Teile aufgeteilt wird (vgl. Quickselect, eine Quicksortvariante zur Mediansuche [37]). Eine Teilmenge enthält alle Elemente, die kleiner als ein zuvor gewähltes Pivotelement sind. Die andere setzt sich dementsprechend aus den Elementen zusammen, die größer dem Pivotelement sind. Nun wird untersucht, wie groß die beiden Teilmengen sind: Gilt $i \leq t$ bei einer der Mengen mit t Elementen, so wird in dieser weitergesucht, andernfalls wird in der anderen nach dem i' größten Element mit $i' = i - t$ gesucht denn man war ursprünglich auf der Suche nach dem i-ten Element. Kritisch für die Laufzeit ist die Wahl des Pivotelements. Dieses wird bestimmt, indem zunächst der Median der lokalen Elemente auf einem Rechenknoten bestimmt wird und anschließend der Median der Mediane der Rechenknoten ermittelt (vgl. BFPRT-Algorithmus [20, 37]). Dieser wird als Pivotelement verwendet.

Mit Hilfe des Medians werden nun die Rechenknoten, wie in Abb. 7.13 (3) dargestellt, in *zwei Partitionen aufgeteilt*. Als nächstes tauschen die Rechenknoten der beiden Partitio-

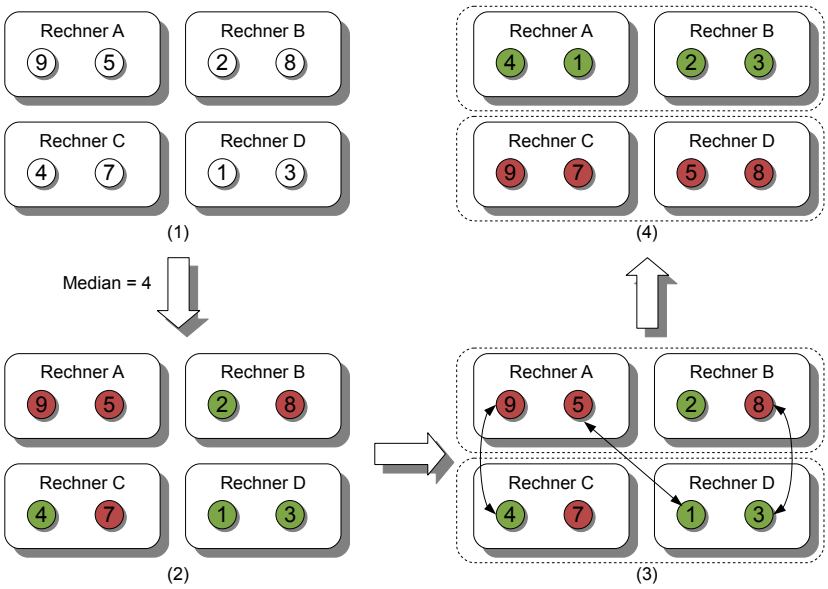


Abbildung 7.13: Ablauf der vier Schritte der parallelen rekursiven Koordinatenbisektion. Die Zahlen in den Knoten bezeichnen jeweils die Koordinatenwerte, die zur Partitionierung herangezogen werden. Die Farben geben die Zugehörigkeit zu einer Partitionen an.

nen solange Knoten aus, bis die erste Partition alle Knoten mit Koordinaten unterhalb des Medians und die andere oberhalb des Medians enthält (Abb. 7.13 (4)). Dies wird solange rekursiv fortgesetzt, bis die gewünschte Zahl der Partitionen erreicht ist.

Multilevelpartitionierung

Das im vorherigen Abschnitt vorgestellte Verfahren zur Partitionierung der Verkehrsnetze mittels rekursiver Koordinatenbisektion ist ein ausgesprochen schnelles Verfahren. Wie aber bereits angedeutet, bringt es auch den Nachteil mit sich, dass es nicht immer das beste Ergebnis liefert. Die Verwendung der geographischen Informationen zur Partitionierung gewährleistet nicht automatisch eine minimale Anzahl von Schnittkanten.

Eine Möglichkeit, diese Probleme zu überwinden, besteht in der Verwendung anderer Arten von Algorithmen. Dazu gehören die so genannten *Multilevelpartitionierungen*. Ein Überblick dazu und eine theoretische Evaluierung gibt [92].

Die Grundidee hinter diesen Verfahren ist, die Verkleinerung des Graphen durch Knotenverschmelzungen. Durch das Verschmelzen werden stark vernetzte Teile des Graphen zusammengefasst und können daher nicht zerteilt werden. Der dadurch entstandene Graph wird anschließend partitioniert (beispielsweise mittels Bisektion). Anschließend wird die

Verschmelzung für die Partitionen wieder rückgängig gemacht. Die drei Phasen dieser Verfahren sind also *Vergröberung*, *Partitionierung* und *Verfeinerung*. Dieses Vorgehen ist in Abb. 7.14 veranschaulicht.

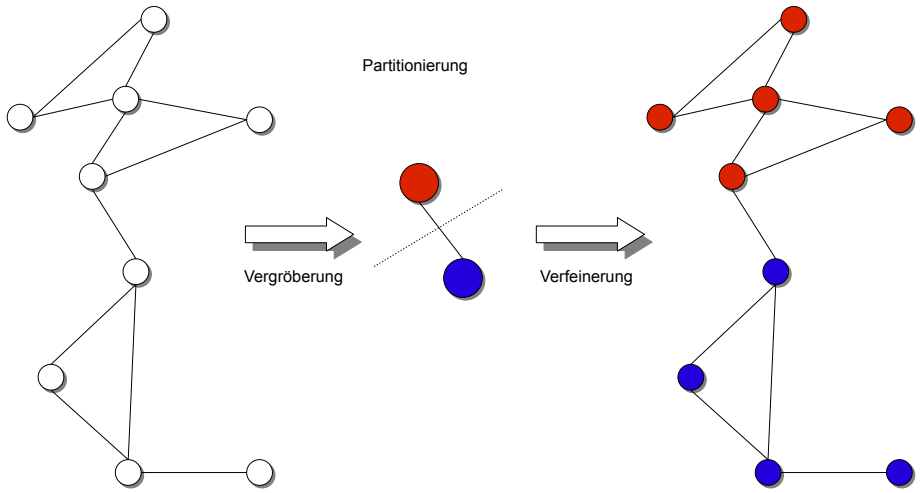


Abbildung 7.14: Die drei Phasen einer Multilevel Partitionierung: Zunächst wird der Ursprungsgraph durch Verschmelzungen von Knoten vergröbert. Der grobe Graph wird anschließend in einem weiteren Schritt partitioniert. Abschließend wird der partitionierte grobe Graph wieder verfeinert.

Betrachtet man die Vorgänge in den drei Phasen etwas genauer, so ergibt sich folgendes Bild. Geht man von einem initialen gewichteten Graphen $G_0 = (V_0, E_0)$ aus, so ist das Ziel der *Vergröberung*, eine Folge von kleineren Graphen zu erzeugen, von dem jeder weniger Knoten besitzt. Zu diesem Zweck wird eine Menge von Knoten V_i^v aus G_i zu einem neuen Knoten v (*Multinode*) im nächstgröberen Graphen G_{i+1} verschmolzen. Dabei ergibt sich das Gewicht des Knoten v aus der Summe der Einzelgewichte der V_i^v und die Kanten von v entsprechen der Vereinigung der Kanten von V_i^v. Zwei Knoten werden dabei miteinander verschmolzen, wenn sie benachbart sind. Dies erfolgt über die Bestimmung eines *Matching* des Graphen. Darunter versteht man die Menge von Kanten, von denen jeweils zwei nicht in einen gemeinsamen Knoten eintreffen. Dementsprechend gewinnt man den Graphen G_{i+1} der nächstgröberen Ebene aus dem Graphen G_i, indem man ein Matching von G_i bestimmt und die Knoten, die dem Matching entsprechen, zu Multinodes zusammenfasst.

In der zweiten Phase, der *Partitionierung*, wird eine Bisektion P_m mit einer minimalen Anzahl von Schnittkanten des groben Graphen $G_m = (V_m, E_m)$ angestrebt. Dabei sollte jede Partition ungefähr die Hälfte der Knoten aus dem Ursprungsgraphen enthalten. Da G_m in der Regel nur noch wenige Knoten enthält, können für die Bisektion andere Verfahren verwendet werden, beispielsweise die im vorherigen Abschnitt beschriebene Koordinatenbisektion. Durch die geringe Größe von G_m kann dieser Schritt sehr schnell ausgeführt werden.

In der dritten und letzten Phase, der *Verfeinerung*, gilt es, den ursprünglichen Graphen wieder zurückzugewinnen. Dazu wird jede Partition auf den Originalgraphen abgebildet, in dem man schrittweise durch die Graphenfolge $G_{m-1}, G_{m-2}, \ldots, G_1$ geht und die Multinodes schrittweise wieder auflöst.

In [93] untersuchen Karypis und Kumar verschiedene Algorithmen für diese drei Phasen und messen die Qualität und die Geschwindigkeit der verschiedenen Wahlmöglichkeiten. Ihre Untersuchungen führten zur Implementierung einer Bibliothek, METIS, die die effizientesten Verfahren beinhaltet.

In [94] wurde METIS mittels MPI effizient als Multilevelpartitionierung für Parallelrechner umgesetzt. Das Ergebnis ist ParMETIS, welches im Rahmen dieser Arbeit für die stochastische Umlegung eingesetzt und in das Framework aufgenommen wurde.

Abbildung 7.15: Vier Partitionen nach der Anwendung der Multilevelpartitionierung mittels ParMETIS auf das Verkehrsnetz von Karlsruhe.

Die Abbildungen 7.15 und 7.16 zeigen die Partitionierung des Verkehrsnetzes von Karlsruhe mit vier Partitionen und 128 Partitionen. Die Laufzeitergebnisse, eine Betrachtung der Schnittkantenzahl und ihr Vergleich mit der rekursiven Koordinatenbisektion sind in Kapitel 9 zu finden.

Abbildung 7.16: 128 Partitionen nach der Anwendung der Multilevelpartitionierung mittels ParMETIS auf das Verkehrsnetz von Karlsruhe.

7.2.3 Besonderheiten bei der Speicherung partitionierter Verkehrsnetze

In den vorherigen Abschnitten wurde gezeigt, wie Verkehrsnetze partitioniert werden können. Durch diese Partitionierungen wird jeder Knoten des Verkehrsnetzes auf eine Partition und damit auch auf einen eindeutigen Rechenknoten abgebildet. Wie bereits erwähnt, müssen nach erfolgter Partitionierung die restlichen Attribute des Verkehrsnetzes wie Abbiegebeziehungen und detaillierte Strecken gemäß der Partitionierung kommuniziert werden.

Um das Problem der Schnittkanten zu lösen, werden wie im mikroskopischen Fall *Geisterknoten* eingeführt, die dafür sorgen, dass das Verkehrsnetz nach wie vor zusammenhängend bleibt. Diese Geisterknoten werden auf beiden beteiligten Partitionen einer Schnittkante repliziert und sind Orte der Kommunikation zwischen diesen. Das Grundprinzip für den makroskopischen Fall ist in Abb. 7.17 dargestellt.

7.2.4 Kurzwegsuche in partitionierten Netzen

Bei der beschriebenen Partitionierung des Verkehrsnetzes hat kein Rechenknoten direkten Zugriff auf das vollständige Verkehrsnetz. Will man nun eine Route zu einem Quelle-Ziel-Paar aus der Fahrtenmatrix berechnen, bei der sich die Quelle und das Ziel nicht in derselben Partition befinden, so ist eine Kommunikation der beteiligten Rechenknoten notwendig. Möchte man für die Routenberechnung den klassischen Algorithmus von Dijkstra [44] ver-

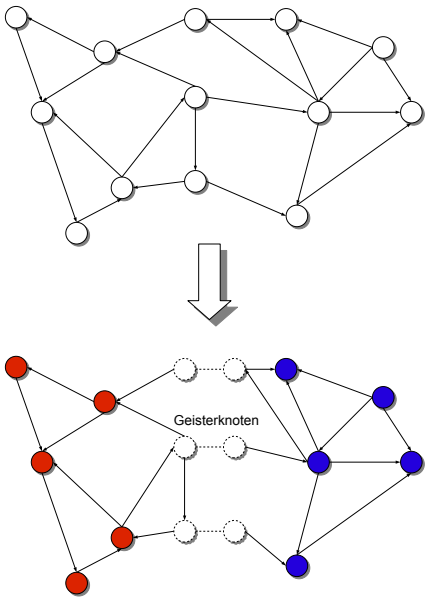


Abbildung 7.17: Verknüpfung zweier Partitionen über Geisterknoten (gestrichelt).

wenden, so müssen in der Regel eine Vielzahl von Knoten besucht werden, so dass im schlimmsten Knoten alle Rechenknoten miteinander kommunizieren müssen.

```
(0) Lege Quellknoten in Liste ab
(1) while (Liste nicht leer) do
   (2) Entferne Minimum aus Liste
   (3) for alle ausgehenden Kanten von Minimum do
       (4) Relaxiere Knoten
       end for
   end while
```

Listing 7.2: Algorithmus von Dijkstra

Listing 7.2 zeigt die Grundstruktur des Algorithmus von Dijkstra, der im Folgenden noch etwas genauer betrachtet werden soll. Bei diesem Algorithmus ist es für die Laufzeit wichtig, dass die Knoten des Verkehrsnetzes in der richtigen Reihenfolge aus der Liste (Schritt `(0)`) entfernt werden. Zu diesem Zwecke wird eine *verteilte Prioritätswarteschlange* verwendet. Sie bietet die für Warteschlangen klassischen `push` und `pop`-Operationen. Die Operationen auf der Warteschlange werden verteilt durchgeführt. Das Ermitteln des Minimums in Schritt `(2)` geschieht über eine `extractMin`-Operation, die den kleinsten Wert aus allen Rechenknoten zurückgibt. In Schritt `(4)` können beispielsweise von jedem Rechenknoten über die `push`-Operation der Warteschlange Knoten in diese eingefügt werden.

Dabei wird in Schritt `(4)` – analog zum klassischen Dijkstra-Algorithmus – jede ausge-

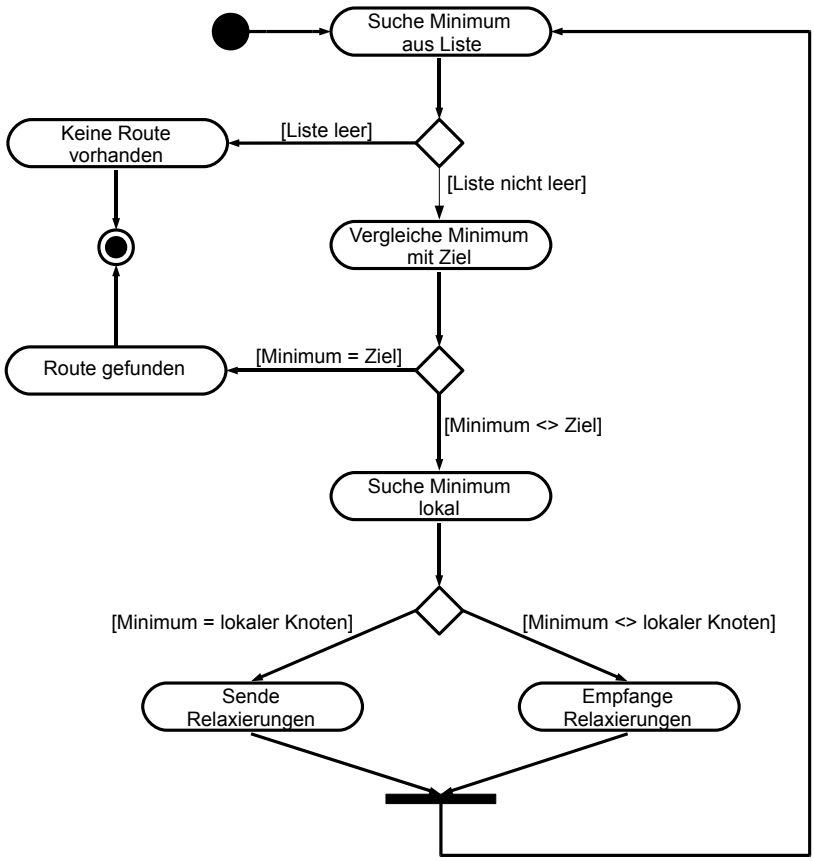


Abbildung 7.18: Ablauf der verteilten Routensuche in partitionierten Verkehrsnetzen.

hende Kante $e = (i, j)$ des in Schritt (3) aus der Liste entfernten Knotens i, welcher das minimale Gewicht hat, untersucht. Ergibt sich dabei eine kürzere Distanz zum Startknoten als bisher, so wird die Distanz des Knotens j aktualisiert beziehungsweise relaxiert.

Hierfür bedarf es allerdings einer Fallunterscheidung. Es können zwei Situationen auftreten. Zum einen kann es sich bei der Kante e um eine *lokale* Kante handeln, das heißt Knoten j befindet sich in derselben Partition auf Rechenknoten P, in der sich auch Knoten i befindet. In diesem Fall wird die Relaxierung lokal durchgeführt und es ist keine Kommunikation mit den anderen Rechenknoten notwendig.

Zum anderen kann es sich bei der betrachteten Kante um eine Schnittkante handeln, d.h. die Knoten i und j befinden sich in unterschiedlichen Partitionen und damit auch auf unterschiedlichen Rechenknoten. Rechenknoten P ist dementsprechend der Knoten j nicht bekannt und muss daher denjenigen Rechenknoten informieren, der diesen verwaltet, damit dieser die Relaxierung durchführt. Es findet demzufolge eine *entfernte* Relaxierung statt.

Der Ablauf dieser Routensuche in einem partitionierten Verkehrsnetz ist dabei Abb. 7.18 zu entnehmen.

In der beschriebenen Form kann der Algorithmus allerdings immer nur eine Route gleichzeitig berechnen, da alle Rechenknoten synchron arbeiten müssen. Der vorgestellte Algorithmus ist in der Lage Routen in verteilten Netzen zu suchen. Darunter leidet aber etwas die Rechengeschwindigkeit.

Als eine weitere Möglichkeit der Berechnung der Routen kann man anstelle des Algorithmus von Dijkstra, der zu den so genannten *label-setting*-Algorithmen gehört, Algorithmen der *label-correcting*-Klasse bemühen. Zu letzterer gehört der Algorithmus von Bellman und Ford [18, 58]. Dafür hebt man die Bedingung auf, dass immer derjenige Knoten mit dem kürzesten Abstand zum Quellknoten aus der Liste entfernt wird. Auf diese Weise können die Kanten in beliebiger Reihenfolge relaxiert werden.

Bei diesem Verfahren besitzt jeder Rechenknoten eine eigene, lokale Liste, in der er seine lokalen Knoten verwaltet. Eine Synchronisation der Rechenknoten erfolgt nicht mehr strikt vor jeder Extraktion aus der Liste. Jeder Rechenknoten ermittelt die Distanzen aller lokalen Knoten und kommuniziert die Relaxierungen für nicht lokale Knoten an die dafür zuständigen Rechenknoten. Dies wird solange durchgeführt bis seine eigene lokale Liste leer ist. Allerdings muss das Abbruchkriterium angepasst werden. Es reicht nicht aus, die Extraktion des Zielknotens dafür zu verwenden, da zu beliebigen Zeitpunkten neue Knoten in die Liste aufgenommen werden können und sich so die Suchinformation ändern kann. Daher terminiert der Algorithmus erst dann, wenn die lokalen Listen aller Rechenknoten leer sind und keine noch nicht empfangenen Nachrichten mit Relaxierungsinformationen mehr existieren. In [88] haben Hribar et al gezeigt, dass die label-correcting Algorithmen ein schlechteres Laufzeitverhalten besitzen als die label-setting Methoden, da sie mehr Iterationen bis zum Abbruch benötigen. Aus diesem Grund werden diese label-correcting Verfahren im Rahmen dieser Arbeit nicht weiter betrachtet.

7.2.5 Parallelisierung der Kurzwegsuche für partitionierte Netze

Wie im vorherigen Abschnitt gezeigt, besitzt die im Rahmen dieser Arbeit entwickelte Kurzwegsuche, die auf einer Idee aus [88] basiert, noch einige Nachteile: Lange Leerlaufzeiten der Rechenknoten und keine Möglichkeit mehrere Routen gleichzeitig zu berechnen. Diese Nachteile lassen sich durch eine Parallelisierung des Algorithmus reduzieren.

Zu diesem Zweck erhält jeder Rechenknoten eine lokale Liste mit Knoten, die bereits relaxiert wurde, deren endgültige Distanz zum Startknoten aber noch nicht feststeht. Es gilt dabei aber zu beachten, dass es sich bei dem entwickelten Algorithmus um einen handelt, der das label-setting nutzt. Dementsprechend ist die Reihenfolge, in der Knoten extrahiert werden, essentiell. Eine parallele Extraktion mehrerer Knoten würde die Abbrucherkennung erschweren. Daher wird ein etwas anderer Ansatz verfolgt.

Es wird im Folgenden davon ausgegangen, dass immer der Knoten mit den niedrigsten Kosten aller Listen extrahiert wird. Ein guter Ansatzpunkt zur Effizienzmaximierung ist es, die Leerlaufzeit der einzelnen Rechenknoten zu minimieren, das heißt die Zeiten, in denen die Rechenknoten nicht gerade mit der Routensuche beschäftigt sind. Hierfür wird zunächst

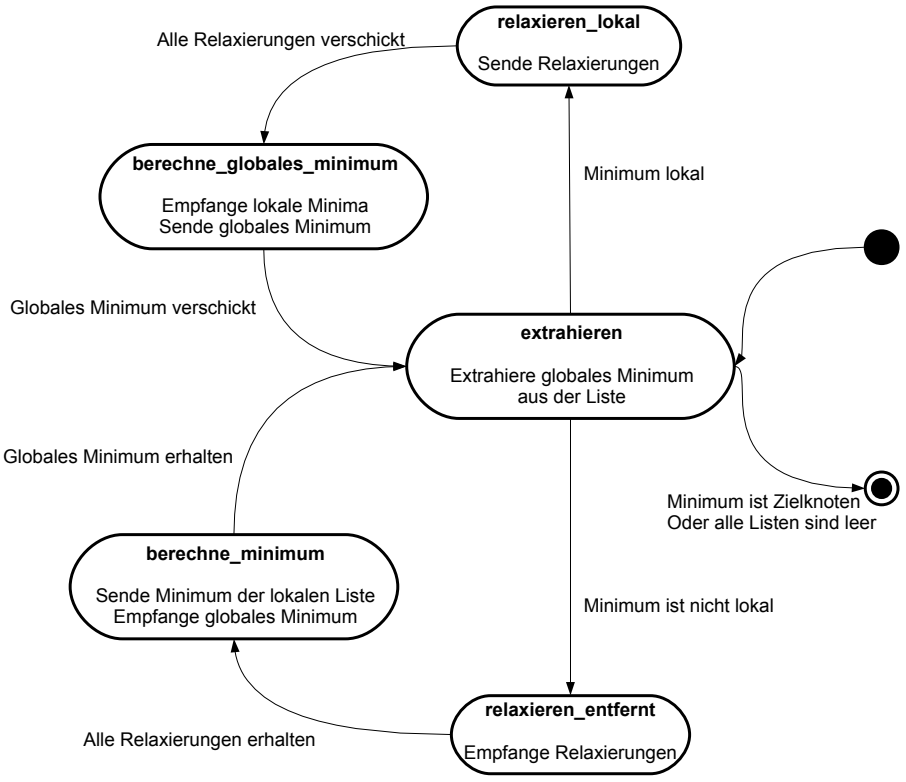


Abbildung 7.19: Zustandsautomat der parallelen Kurzwegsuche: Jeder Rechenknoten arbeitet gemäß dieses Automaten.

der Zustandsautomat jedes Rechenknoten zur Kurzwegsuche betrachtet, wie er in Abb. 7.19 dargestellt ist.

Am Beginn jeder Iteration befindet sich jeder Rechenknoten im Zustand `extrahieren`. Es wird dann derjenige Rechenknoten *P* ermittelt, dessen Liste den Knoten mit der kürzesten Distanz zum Quellknoten besitzt. Es wird also das globale Minimum gesucht. Dieser Knoten wird aus der lokalen Liste extrahiert und *P* wechselt in den Zustand `relaxieren_lokal`. Die anderen Rechenknoten wechseln in `relaxieren_entfernt`. *P* führt nun lokal die Relaxierungen durch und verschickt für die nicht lokal vorliegenden Knoten – also Knoten, die nicht in der Partition des Netzes liegen, die *P* verwaltet – und die relaxiert werden sollen Nachrichten an die entsprechenden Rechenknoten. Da sich die Entfernungen der Knoten durch die Relaxierungen in allen Partitionen geändert haben können, muss das globale Minimum aller Listen neu berechnet werden. Dazu sammelt *P* alle lokalen Minima der beteiligten Rechenknoten ein und ermittelt daraus das neue globale Minimum. Der Automat befindet sich dabei im Zustand `berechne_globales_minimum` bzw. `berechne_minimum`. Abschließend wechseln alle Rechenknoten am Ende einer Iteration zurück in den Anfangszustand `extrahieren`.

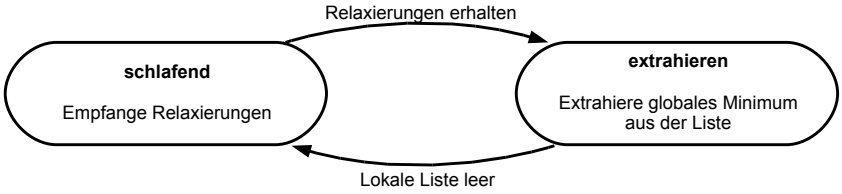


Abbildung 7.20: Erweiterung des Zustandsautomaten der parallelen Kurzwegsuche zur Minimierung der Leerlaufzeiten der einzelnen Rechenknoten.

Erhält ein Rechenknoten, der sich im Zustand `relaxieren_entfernt` befindet, keine Relaxierungen, kann sich zwangsläufig sein lokales Minimum nicht geändert haben, denn es hat sich keine Veränderung in seiner lokalen Liste ergeben. Dieser Rechenknoten ist dementsprechend nicht bei der Berechnung des globalen Minimums beteiligt. Der Rechenknoten ist also in dieser Phase inaktiv.

Um die Leerlaufzeiten der einzelnen Rechenknoten zu minimieren, muss der Zustandsautomat aus Abb. 7.19 um einen Zustand `schlafend` erweitert werden (siehe Abb. 7.20). Indem Rechenknoten diesen Zustand nutzen, können sie bei der Berechnung einer Route aussetzen. Erhält ein Rechenknoten im Zustand `relaxieren_entfernt` keine Nachrichten über Relaxierungen, tritt er in diesen Zustand über. Eine Berechnung des lokalen Minimums ist, wie gezeigt wurde, nicht notwendig. Er verweilt so lange in diesem Zustand bis er eine Relaxierung oder das Terminierungssignal der Routensuche erhält.

Unter Ausnutzung dieses Verhaltens können die nicht genutzten Rechenknoten zur Berechnung weiterer Routen herangezogen werden. Hierfür wird die Suche für n Einträge der Fahrtenmatrix auf allen Rechnern gleichzeitig n mal gestartet und abwechselnd ausgeführt. Zwei Routen können genau dann vollständig parallel berechnet werden, wenn sich die Mengen der Partitionen aus denen die Suchen Knoten betrachten sich nicht überschneiden.

Abb. 7.21 zeigt den Suchraum und die beteiligten Partitionen bei der Berechnung einer Route (blau Linie). Der Suchraum besteht lediglich aus den rot eingefärbten Strecken und Knoten. Die grauen Partitionen sind zu keinem Zeitpunkt an der Berechnung der Route beteiligt. Die sie verwaltenden Rechenknoten können wie oben beschrieben zur Berechnung weiterer Routen herangezogen werden.

Wie man sich leicht veranschaulichen kann, ist der hier vorgestellte Algorithmus besonders effektiv, wenn man anstelle der klassischen Dijkstra-Suche eine zielgerichtete, optimierte Kurzwegsuche, wie sie beispielsweise in Kapitel 2 vorgestellt wurde, verwendet. Diese sorgt dafür, dass der betrachtete Suchraum möglichst klein gehalten wird und dementsprechend möglichst wenig Partitionen besucht werden müssen.

Abbildung 7.21: An der Suche einer Route beteiligte Partitionen im Karlsruher Verkehrsnetz: Die gefundene Route ist blau markiert, die roten Strecken stellen den betrachteten Suchbaum dar.

7.3 Zusammenfassung

In diesem Kapitel wurden verschiedene Möglichkeiten der Durchführung der Simulation des makroskopischen motorisierten Individualverkehrs aufgezeigt. Unter den verschiedenen Möglichkeiten wurde ein Verfahren zur stochastischen Umlegung für eine genauere Analyse herangezogen. Diese stochastische Umlegung wurde auf ihr Verhalten hin untersucht, um geeignete Parallelisierungsstrategien zu erarbeiten. Dabei wurden zwei Strategien entwickelt. Die erste nutzt die einfache Aufteilung der Fahrtenmatrix, die zweite bedient sich der Möglichkeiten der Verkehrsnetzpartitionierung. Für letztere wurden verschiedene Möglichkeiten der Partitionierung aufgezeigt und eine parallele Kurzwegsuche entwickelt, die auf verteilten Verkehrsnetzen arbeitet. Wie in Kapitel 9 gezeigt werden wird, führen beide Strategien zu einer deutlich beschleunigten Berechnung dieser Simulation.

8 Hybride Verkehrssimulationen

In den vorhergehenden Kapiteln 5, 6 und 7 wurden die verschiedenen Verkehrsmodelle zur Simulation des Verkehrsflusses vorgestellt und geeignete Parallelisierungsstrategien dazu erarbeitet. Interessant sind aber auch Ansätze, die diese Simulationsarten kombinieren oder um neue Komponenten erweitern. Im Folgenden werden diese Kombinationen als *hybride* Simulationen bezeichnet.

In diesem Kapitel wird zunächst eine Kopplung einer mikroskopischen Simulation, basierend auf dem in Kapitel 5 vorgestellten Modell, sowie einer stochastischen Umlegung des motorisierten Individualverkehrs (siehe Kapitel 7) vorgestellt. Anschließend wird eine einfache, mikroskopische Fußgängersimulation zu obiger mikroskopischen Verkehrssimulation hinzugefügt.

8.1 Kombinierte Mikro- und Makrosimulation

Wie bereits erörtert sind Simulationen des Verkehrsflusses in großen und detaillierten Netzen wünschenswert und zum Teil auch unabdingbar. Für diese Simulationen wurden in den vorhergehenden Kapiteln schon Strategien aufgezeigt, die dies in vertretbarer Rechenzeit ermöglichen. Die Ergebnisse hierzu sind in Kapitel 9 zu finden.

Mit folgender Überlegung lässt sich allerdings noch weiter Rechenzeit einsparen, und sie ermöglicht neue Perspektiven auf das Verkehrsgeschehen. Betrachtet man ein heterogenes Verkehrsnetz, also ein Netz, dass beispielsweise nicht nur aus rein städtischen Straßen besteht beziehungsweise ein reines Autobahnnetz ist, sondern sich aus verschiedensten Strukturen zusammensetzt (wie das Straßennetz Deutschlands), so bietet sich hier folgender Ansatzpunkt.

Eine sehr große mikroskopische Simulation ist sehr rechenintensiv und enthält an vielen Stellen Informationen, die nicht komplett benötigt werden. Hingegen können makroskopi-

sche Simulationen, wie die im letzten Kapitel vorgestellte stochastische Simulation, schneller sein, aber nicht in allen Situationen die notwendigen Detailinformationen enthalten, die man bräuchte.

Meist ist in solchen Netzen nicht immer an allen Stellen eine hohe Detailauflösung interessant und notwendig. Dies ist abhängig von den gewünschten Ergebnissen. So kann es in einem heterogenen Verkehrsnetz durchaus lediglich von Interesse sein, den städtischen Verkehr in Ballungsgebieten detailliert zu betrachten und den Verkehr zwischen diesen Zentren, der in der Regel über Autobahnen oder Bundesstraßen fließt, in einer gröberen Auflösung zu betrachten. Als ein weiteres Beispiel seien die detaillierten Betrachtungen von Autobahnen genannt. Hier bietet es sich an, den Verkehr an kritischen Stellen wie etwa Auf- und Abfahrten detailliert zu simulieren und den restlichen, doch eher eintönigen Teil nur noch grob zu betrachten.

Für die Detailbetrachtungen kommt eine mikroskopische Simulation in Betracht, für die gröbere eine makroskopische. Es lohnt sich über eine Kopplung beider Simulationsarten nachzudenken. Zunächst werden im folgenden Abschnitt bereits bestehende Kopplungen zwischen mikroskopischer und makroskopischer Simulation diskutiert, bevor dann die im Rahmen dieser Arbeit entstandene Kopplung zwischen einer heterogenen mikroskopischen Simulation basierend auf zellulären Automaten und einer stochastischen Umlegung aufgezeigt wird.

8.1.1 Existierende Kopplungen

Es existieren mittlerweile zahlreiche Kopplungen von Verkehrssimulationen. Burghout gibt in [24] einen guten Überblick über bestehende Kopplungen. Im Folgenden soll eine Beschränkung auf diejenigen Kopplungen erfolgen, die von ihrer Struktur ähnlich zu der hier entworfenen sind. Montero et al beschreiben in [118] eine Kopplung einer makroskopischen mit einer mikroskopischen Simulation. Als makroskopische Komponente setzen sie die Software EMME/2[1], ein Werkzeug zur Berechnung von Verkehrsbelastungen und -nachfrage, ein. Für den mikroskopischen Teil nutzen sie den Simulator AIMSUN2 [16], der sowohl die Simulation städtischen als auch außerstädtischen Verkehrs erlaubt. Anders als bei dem im Rahmen dieser Arbeit entwickelten Ansatz erfolgt allerdings nur ein Informationsaustausch von der makroskopischen zur mikroskopischen Simulation. Eine beliebige Aufteilung der Verkehrsnetze in mikroskopische und makroskopische Teile ist mit diesem Verfahren nicht möglich.

In [51] wird eine Kopplung der beiden Softwarewerkzeuge VISUM und VISSIM der PTV AG vorgestellt. Diese Kopplung unterstützt einen beidseitigen Informationsaustausch. Es wird versucht mit dieser Kopplung die Genauigkeit der makroskopischen Simulation zu erhöhen, indem an bestimmten Stellen eine mikroskopische Simulation zum Einsatz kommt. Die makroskopische Simulation, VISUM, liefert dabei die Daten der Verkehrsinfrastruktur und der Verkehrsnachfrage. Diese werden als Ausgangsdaten für die mikroskopische

[1] http://www.inro.ca/en/products/emme/index.php

Simulation mittels VISSIM genutzt, welche dann den eigentlichen Verkehrsfluss simuliert.

Burghout und Koutsopoulos [25] stellen eine Kopplung einer mikroskopischen Simulation mit einer mesoskopischen vor. Bei einer mesoskopischen Simulation handelt es sich, vereinfacht gesprochen, um einen Zwischenlösung zwischen einer rein mikroskopischen und einer rein makroskopischen Betrachtung. In [25] werden Teile des Verkehrsnetzes mikroskopisch und andere mesoskopisch simuliert. Die dort beschriebene Kopplung kann aber nicht zur Kopplung mit einem rein makroskopischen Teil verwendet werden.

Flötteröd und Nagel [56] beschreiben eine schnelle Kopplung zwischen einer makroskopischen Simulation, basierend auf einer strömungsmechanischen Betrachtung, und von mikroskopischen Fahrerverhalten. Auf Grund der strömungsmechanischen Natur der makroskopischen Komponenten scheidet diese Kopplung allerdings für die im Rahmen dieser Arbeit zu entwickelnde Kopplung aus. Das selbe gilt für die in [82] vorgestellte Kopplung.

8.1.2 Problem einer Kopplung

Ein wesentliches Problem bei der Kopplung der in Kapitel 5 vorgestellten mikroskopischen Simulation mit einer stochastischen Umlegung liegt in der unterschiedlichen Betrachtung der Zeit in beiden Simulationsmodellen. So handelt es sich bei der mikroskopischen Simulation um eine dynamische, zeitabhängige Betrachtung des Verkehrsgeschehens, wohingegen die stochastische Umlegung – wie in den Abschnitten 7.1 und 7.1.3 diskutiert – eine statische Umlegung, das heißt eine Umlegung unter Vernachlässigung der Zeitachse darstellt. Dennoch müssen beide zum Zwecke einer Kopplung aufeinander abgebildet werden.

8.1.3 OD-Ansatz

Der erste betrachtete Ansatz zur Kopplung beider Simulationen ist der so genannte *OD-Ansatz*. Hier werden beide Simulationsprogramme über ein externes Programm, den *Koppler*, miteinander verknüpft. Dieses Programm übernimmt die Aufgaben des Datenaustauschs und der Synchronisation der beiden Simulation. Dies ist in Abb. 8.1 schematisiert.

Beide Simulationen, die makroskopische Umlegung und die mikroskopische, nutzen als Eingabedaten für die Nachfrage je eine QZ-Matrix und verwenden diese wie in den Abschnitten 5.2 und 7.1.3 beschrieben. Es liegt nun nahe diese Matrizen als Austauschdatenformat heranzuziehen. Dazu werden entsprechend generierte QZ-Matrizen nach einer bestimmten Anzahl von Zeitschritten zwischen den Simulationen ausgetauscht.

Bevor die Kopplung beginnen kann, muss zunächst das Verkehrsnetz unter den beiden Simulationen aufgeteilt werden. Dazu gilt es, diejenigen Komponenten zu identifizieren, die sich am besten für eine mikroskopische und diejenigen für eine makroskopische Simulation eignen. Dies ist abhängig von den gewünschten Simulationsergebnissen und -schwerpunkten.

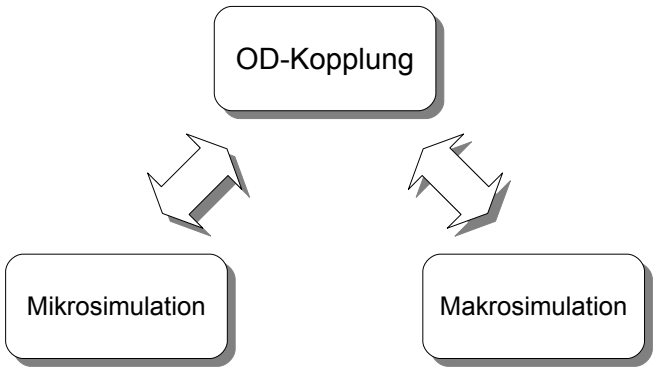


Abbildung 8.1: Kopplung von mikroskopischen und makroskopischen Simulationen über ein separates Programm beim OD-Ansatz.

Im Folgenden sei angenommen, dass das Ziel ist, Straßen mit hoher zulässiger Maximalgeschwindigkeit makroskopisch zu simulieren und die restlichen mikroskopisch. Hierfür werden die Merkmale der einzelnen Strecken des Verkehrsnetzes genutzt: Maximalgeschwindigkeit bei freiem Fluss, Anzahl der Fahrstreifen, Kapazität und Straßentyp.

Die interessantesten Merkmale sind hierbei die Maximalgeschwindigkeit und der Straßentyp. Steht der Straßentyp nicht zur Verfügung, so lässt sich in guter Näherung dieser über die Maximalgeschwindigkeit bestimmen.

Auf diese Weise gewinnt man zwei Teilnetze, die dann zusammen mit initialen Nachfragedaten durch den Koppler an die jeweiligen Simulationen weitergeleitet werden. Dies ist exemplarisch in Abb. 8.2 dargestellt. Haben die beiden Simulationen diese Daten erhalten, beginnt deren Simulationsablauf. Es ist nun wichtig die beiden Austauschrichtungen und damit auch das Verhalten der Teilsimulationen zu betrachten.

Von der Makrosimulation zur Mikrosimulation

Wie bereits gezeigt wurde, handelt es sich bei einer stochastischen Umlegung um ein statisches Verfahren, das nur einen globalen Simulationszeitraum betrachtet. Dieses Problem ist dadurch gelöst, dass der Umlegungszeitraum verkürzt wird, d.h. der globale Simulationszeitraum der Gesamtsimulation wird in kleinere Zeitscheiben zerlegt. Beispielsweise wird ein Gesamtsimulationszeitraum von 24 Stunden in 48 Zeitscheiben der Länge 30 Minuten zerlegt. Die beiden Simulationen werden nun parallel ausgeführt.

Am Ende jeder Zeitscheibe erfolgt die Synchronisation mit der mikroskopischen Simulation. Dabei müssen die Informationen zur Fortführung der Simulation auf der mikroskopischen Seite übertragen werden. Die Schnittkanten (Schnittstrecken) sind die Orte, wo diese Übergabe erfolgen muss.

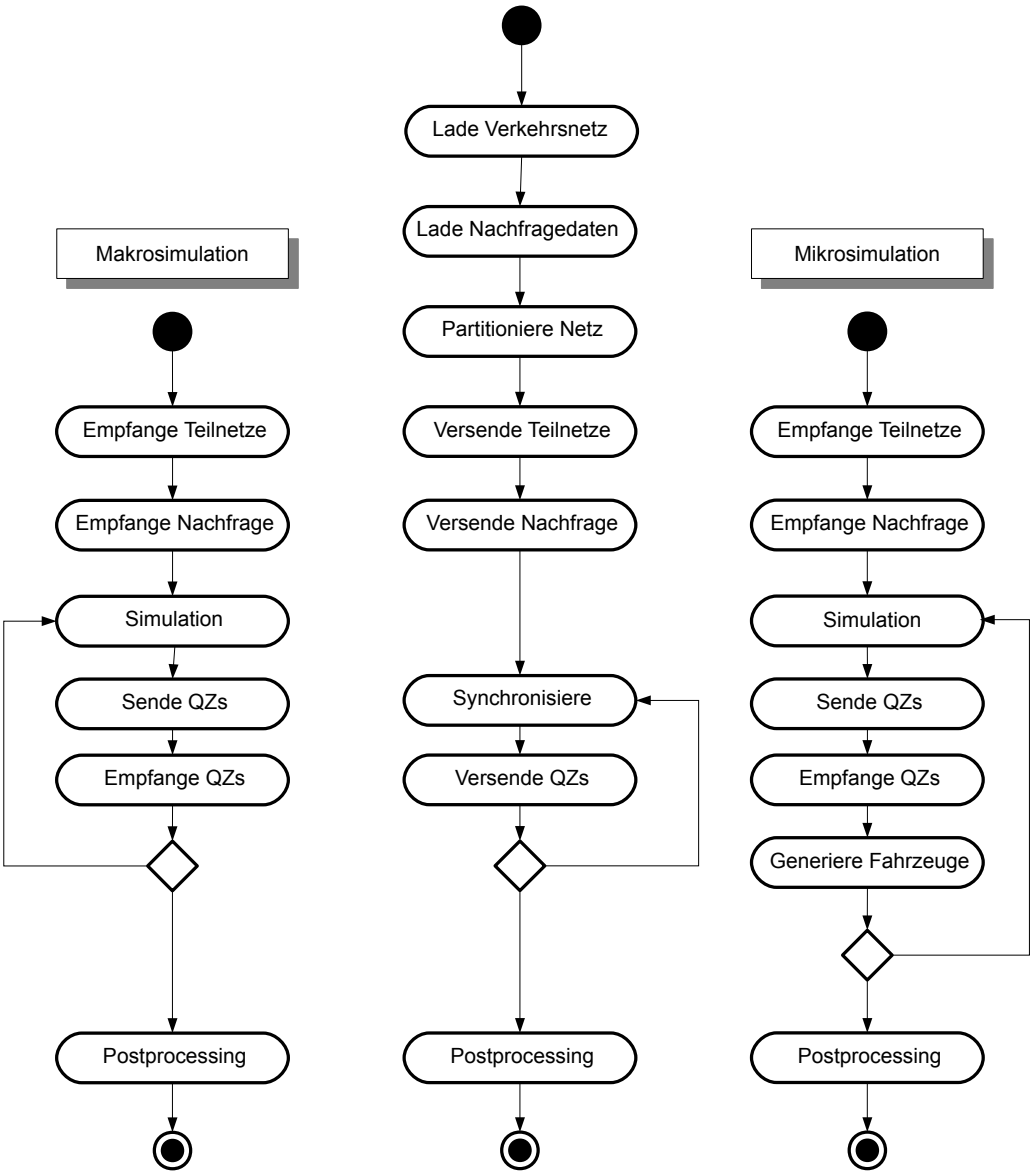


Abbildung 8.2: Ablauf des OD-Ansatzes.

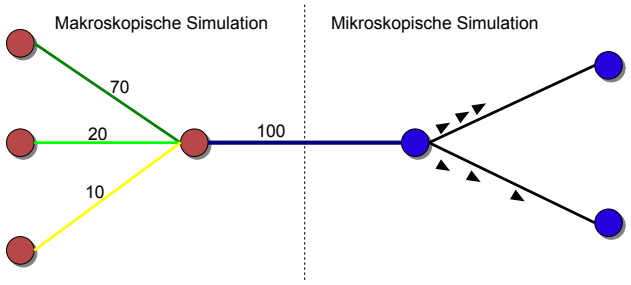


Abbildung 8.3: Generierung von Fahrzeugen an der Schnittstelle zwischen makroskopischer und mikroskopischer Simulation.

Ist die Umlegung einer Zeitscheibe abgeschlossen, so liegt für den makroskopischen Teil ein belastetes Verkehrsnetz vor (jeder Route und damit jeder Strecke der Route sind eine gewisse Anzahl von Fahrzeugfahrten zugewiesen). Aus diesen Belastungen lassen sich spezielle QZ-Paare generieren. Quelle und Ziel der Fahrzeugfahrten sind der makroskopischen Simulation ja bereits bekannt. Es müssen nun an die mikroskopische Simulation lediglich die jeweiligen Paare von Ziel und Fahrzeugfahrten auf einer Schnittstrecke übergeben werden. Damit verfügt die Mikrosimulation über die Informationen, die sie benötigt, um innerhalb der nächsten Zeitscheibe individuelle Fahrzeuge zu generieren (siehe Abb. 8.3). Dies geschieht analog zur klassischen mikroskopischen Simulation wie sie in Kapitel 5 beschrieben wurde.

Von der Mikrosimulation zur Makrosimulation

Die Übertragung der Daten von der mikroskopischen zur makroskopischen Simulation gestaltet sich etwas einfacher. Die Makrosimulation benötigt als Eingabedaten eine QZ-Matrix für die nächste Zeitscheibe. An den Schnittknoten wird dazu die Anzahl der ankommenden individuellen Fahrzeuge gezählt und gemäß ihrer Quelle und ihres Ziels aufsummiert (siehe Abb. 8.4). Aus diesen gewonnen Daten lässt sich die QZ-Matrix für den betrachteten Schnittknoten erstellen.

Diese QZ-Paare werden an die Makrosimulation übergeben. Alle übergebenen QZ-Matrizen stellen die Eingabedaten für die Nachfrage der nächsten Zeitscheibe dar.

8.1.4 Integrierter Ansatz

Der zweite Ansatz, der *integrierte Ansatz*, verfolgt eine andere Strategie. Bei ihm wird kein externes Programm zur Kopplung verwendet im Gegensatz zum vorherigen Ansatz, sondern vielmehr übernimmt die mikroskopische Simulation vollständig die Kontrolle über die Kopplung.

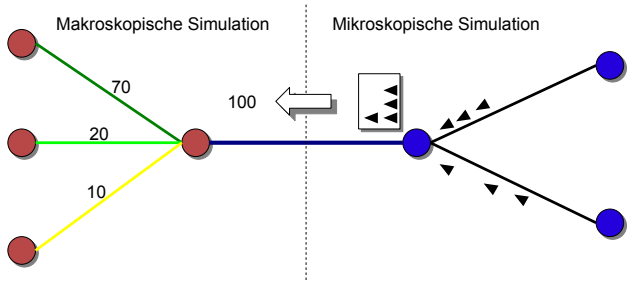


Abbildung 8.4: Generierung von Fahrzeugfahrten an der Schnittstelle zwischen makroskopischer und mikroskopischer Simulation.

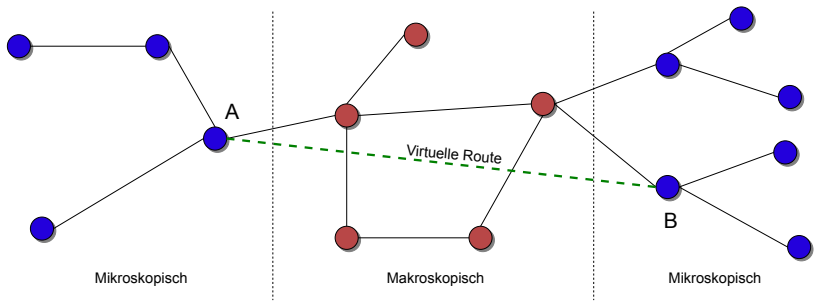


Abbildung 8.5: Ein in mikroskopische und makroskopische Teile aufgeteiltes Verkehrsnetz: Eine virtuelle Route überbrückt den makroskopischen Teil.

Zu diesem Zweck muss allerdings der mikroskopische Simulator modifiziert werden. Zur Überbrückung des makroskopischen Teilnetzes werden jeweils *Virtuelle Routen* eingeführt, deren Endknoten durch den jeweiligen letzten beziehungsweise nächsten mikroskopischen Knoten gegeben sind. Dies ist in Abb. 8.5 dargestellt. Dort wird eine virtuelle Route (grün) herangezogen, um die beiden mikroskopischen Teilnetze direkt miteinander zu verbinden. Diese virtuellen Routen speichern wie die normalen Routen alle notwendigen Informationen, beispielsweise die Anzahl der Fahrzeuge, die auf ihnen fahren, ab.

Anders als beim OD-Ansatz übermittelt die makroskopische Simulation an den Schnittstellen nicht mehr die Anzahl der Fahrzeuge, die sie verlassen, sondern die durchschnittliche Geschwindigkeit. Dieser Wert ist wichtig für die weitere Behandlung in der mikroskopischen Simulation.

Wenn nun ein Fahrzeug den mikroskopischen Teil eines Verkehrsnetzes am Knoten A verlässt und in den makroskopischen eintritt, so kann seine Austrittszeit aus dem makroskopischen Teil hinein in einen neuen mikroskopischen berechnet werden. Diese Verweildauer ergibt sich aus der Gesamtlänge der Strecken im makroskopischen Teil und der durchschnittlichen Geschwindigkeit, die der virtuellen Strecke zugewiesen wurde.

Diese Verweildauer wird dazu herangezogen, in der entsprechenden Zeitscheibe am Austrittsknoten B die Fahrzeuge auftreten zu lassen, die bei A im makroskopischen Teil verschwunden sind.

Die makroskopische Simulation benötigt als Eingabedaten wiederum – wie im OD-Ansatz – QZ-Matrizen. Hierfür werden die Informationen der virtuellen Route herangezogen: Quelle und Ziel der makroskopischen Simulation sind durch die Endpunkte der virtuellen Route definiert, und zusätzlich hält die virtuelle Route noch die Anzahl der Fahrzeuge vor, die sich auf ihr befinden. Aus diesen drei Informationen wird die Eingabe-QZ erstellt und an die makroskopische Simulation übergeben.

Der Ablauf des integrierten Ansatzes ist in Abb. 8.6 dargestellt. Nachdem die mikroskopische Simulation das Verkehrsnetz und die initialen Nachfragedaten geladen hat, teilt sie das Gesamtverkehrsnetz in einen mikroskopischen und einen makroskopischen Teil auf. Zur Überbrückung der jeweiligen makroskopischen Abschnitte der berechneten Routen werden die virtuellen Routen bestimmt.

Anschließend wird der Teil des Verkehrsnetzes für welches die makroskopische Simulation zuständig ist, an diese zusammen mit der Initialnachfrage kommuniziert. Nun beginnen beide Simulationen mit ihren Berechnungen. Ist die mikroskopische Simulation fertig, sendet sie eine Synchronisierungsnachricht an die makroskopische Simulation. Diese wartet auf die Nachricht für den Fall, dass sie mit ihren Berechnungen schon früher fertig war. Sie ermittelt nun die durchschnittlichen Geschwindigkeiten und sendet diese an die mikroskopische Simulation, welche im Gegenzug die QZs für die virtuellen Routen und die makroskopische Simulation übermittelt. Anschließend beginnen die Simulation mit der Berechnung der nächsten Zeitscheibe. Abschließend werden die Ergebnisse zusammen geführt.

Für die Simulationsergebnisse dieses Ansatzes und die darauf beruhende Bewertung sei auf

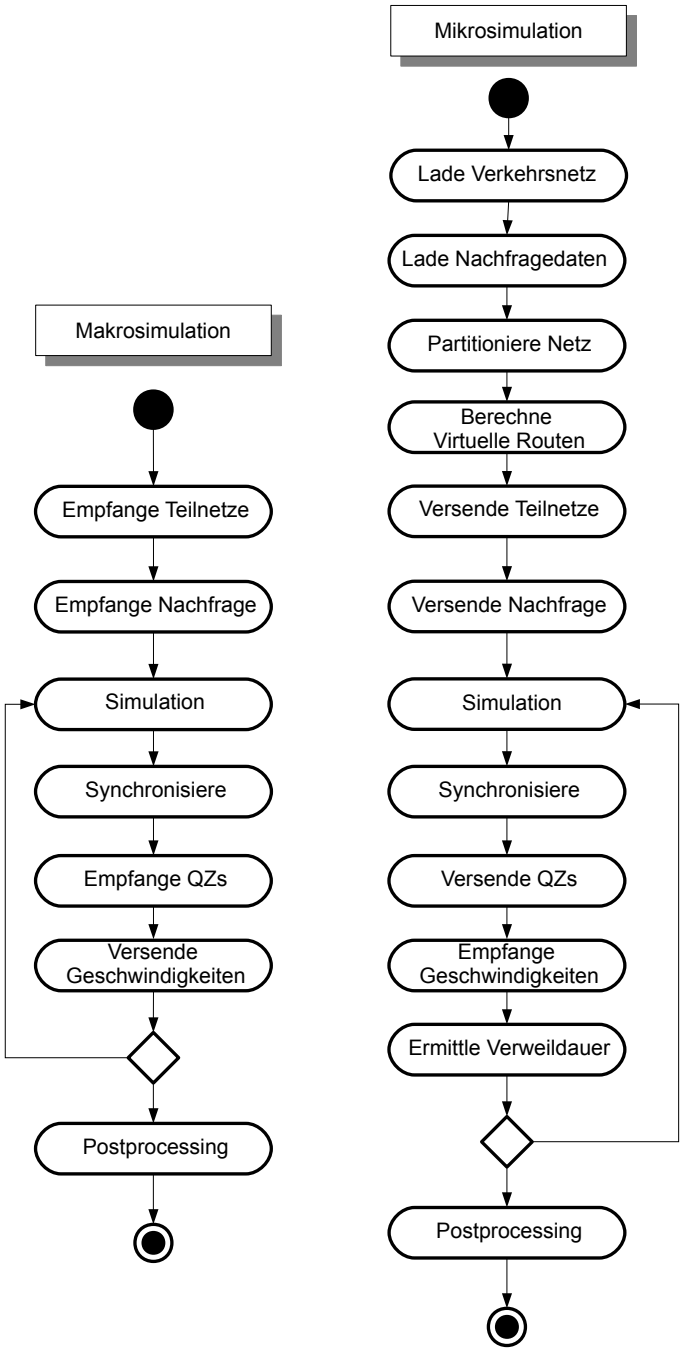


Abbildung 8.6: Ablauf des integrierten Ansatzes.

Kapitel 9 verwiesen.

8.2 Kombinierte Verkehrs- und Fußgängersimulation

Eine zusätzliche Erweiterung einer mikroskopischen Simulation besteht in einer Kombination von dieser mit einer mikroskopischen Fußgängersimulation. In unseren alltäglichen Verkehrssystemen bewegen sich nicht nur Fahrzeuge verschiedenster Ausprägung, sondern natürlich auch eine Vielzahl von Fußgängern. Letztere interagieren auf vielfältige Weise mit dem übrigen Verkehr, beispielsweise beim Überqueren einer Straße, an Haltestellen beim Ein- und Aussteigen in Verkehrsmittel des öffentlichen Verkehrs oder auf Parkplätzen [9]. Will man den Realitätsgrad einer mikroskopischen Verkehrssimulation erhöhen und neue Perspektiven der Verkehrsplanung ermöglichen, so bedarf es der Betrachtung und Integration von Fußgängerbewegungen und Bewegungsprofilen in eine mikroskopische Verkehrssimulation.

Dies ist Gegenstand dieses Abschnitts. Zunächst werden bestehende Fußgängersimulationsmodelle vorgestellt und bezüglich ihrer Brauchbarkeit für die hier gewünschten Ziele diskutiert. Anschließend erfolgt die Vorstellung eines einfachen Fußgängersimulationsmodells, welches in das in dieser Arbeit entstandene Framework integriert wurde.

8.2.1 Bestehende Fußgängersimulationsmodelle

Die mikroskopische Simulation von Fußgängern kann man grob in drei Typen aufteilen. Im ersten Fall folgt die Simulation einem Social-Force-Modell [83], der zweite setzt Warteschlangensysteme zur Simulation ein [127]. Der letzte Typ greift auf die bereits bekannte Technologie der zellulären Automaten zurück [97].

Das *Social-Force-Modell* ist ein kontinuierliches Simulationsmodell, das sich bei der Simulation an Elementen der Strömungssimulation und Gaskinetik orientiert. In ihm werden die Fußgänger als Mehrteilchensystem aufgefasst. Die Bewegung der Fußgänger ergibt sich dabei aus der Grundidee, dass zu jedem Zeitpunkt auf jeden Fußgänger im System eine bestimmte Kraft wirkt. Die Stärke und Art der Kraft wird dabei aus dem Sozialverhalten der Menschen abgeleitet. Dieses System arbeitet wie in [26] und [83] beschrieben auf einem System gekoppelter Differentialgleichungen, welches es zu lösen gilt. Die Schrittweiten bei der numerischen Lösung und Integration sind dabei sehr klein zu wählen, was sich deutlich im Rechenaufwand niederschlägt. Zusätzlich ist das Social-Force-Modell auch bei der Verwendung der Geometrien des Verkehrsnetzes anspruchsvoll. Sie muss vektorisiert vorliegen, was die Flexibilität einschränkt und den Rechenaufwand erhöht [148].

Dieser erhöhte Rechenaufwand der Social-Force-Modellklasse schränkt ihre Einsatzmöglichkeiten in großräumigen mikroskopischen Simulationen ein. Will man die Fußgängerströme in großräumigeren Gebieten, wie beispielsweise ganzen Städten, simulieren, so scheiden diese Modelle in der Regal aus.

Die Modelle, die auf *zellulären Automaten* beruhen, sind hierfür besser geeignet. Dies ist im Wesentlichen aus denselben Gründen wie bei der mikroskopischen Verkehrssimulation der Fall. Bei den meisten CA-Modellen ist es möglich mehrere tausend bis zehntausend Personen in vertretbarer Zeit zu simulieren. Im Folgenden werden einige CA-Modelle mit ihren jeweiligen Hauptcharakteristika vorgestellt.

Die Modelle unterscheiden sich dabei meist nur in einigen wenigen Punkten. Im Modell, welches in [27], [28] und [146] entwickelt wurde, wird über die zu Grunde liegende Geometrie ein zweidimensionales Gitter gelegt. Dieses Gitter, welches für die Simulation verwendet wird, ist im Gegensatz zu den Social-Force-Modellen nach der Generierung unabhängig von der Geometrie, das heißt das Gitter lässt sich auch im Nachhinein noch gut anpassen und verändern.

In diesem Modell besteht das Gitter aus quadratischen Zellen mit einer Kantenlänge von 40 cm. Dies entspricht dem Platz, den ein stehender Mensch durchschnittlich benötigt. Ähnlich dem CA-Modell für den Straßenverkehr kann eine Zelle genau einen von zwei Zuständen einnehmen: Entweder ist sie belegt (Zustandswert 1) oder sie ist frei (Zustandswert 2). Dabei macht es keinen Unterschied, ob eine Zelle durch eine Person, eine Wand oder ein anderes Hindernis belegt ist.

Bei den Nachbarschaftsbeziehungen einer Zelle wird die Moore-Nachbarschaft herangezogen (siehe Abschnitt 2.2), das heißt die acht direkt umliegenden Zellen. Jeder Zelle ist eine 3×3 Matrix U mit Übergangswahrscheinlichkeiten zugeordnet, die dafür herangezogen werden, wohin sich eine Person im nächsten Simulationsschritt bewegt. Die einzelnen Einträge $u_{i,j}$ der Matrix werden dabei folgendermaßen bestimmt:

$$u_{i,j} = N \cdot M_{i,j} \cdot S_{i,j} \cdot D_{i,j} \cdot (1 - n_{i,j}) \tag{8.1}$$

Dabei entspricht $n_{i,j}$ der Wahrscheinlichkeit, ob eine Zelle belegt ist oder nicht. Ist eine Zelle belegt, so berechnet sich der entsprechende Eintrag $u_{i,j}$ zu 0. Dies verhindert, dass sich eine Person in diese Zelle bewegen kann. Der Normierungsfaktor N gewährleistet, dass die Summe aller Übergangswahrscheinlichkeiten 1 ist ($\sum_U u_{i,j} = 1$). Der Faktor $M_{i,j}$ ist ein Eintrag aus der Bewegungsmatrix M. Diese Bewegungsmatrix gibt die Richtungen an, in die sich eine Person bewegen möchte. Dementsprechend sind die Einträge in Richtung des Ziels in der Regel höher als die der Gegenrichtung.

In diesem Modell werden Bewegungen der Personen durch Potentialfelder bestimmt. Es wird dabei zwischen einem statischen Potentialfeld S und dynamischen Potentialfeldern D unterschieden. Das Potentialfeld S wird dabei über das gesamte Gitter gelegt und wird dafür genutzt, das Wissen von Personen und ihrer Bewegungsrichtungen zu simulieren. Dabei wird der Zielzelle im Gitter der Wert 1 zugewiesen. Je weiter sich die Zellen von diesem Ziel befinden, desto geringer ist die Stärke des Potentialfeldes. $S_{i,j}$ entspricht dabei dem Wert des Potentialfeldes in Zelle (i, j).

Die dynamischen Potentialfelder D können in diesem Modell unter anderem dazu herangezogen werden, Gruppendynamiken zu simulieren: Verlässt eine Person, die zu einer Gruppe gehört, eine Zelle (i, j), so wird der Wert des dynamischen Potentialfelds $D_{i,j}$ an dieser

Stelle um einen Wert $\Delta d_{i,j}$ erhöht. Ein Teil dieser Erhöhung wird dabei auch an die Nachbarzellen weitergegeben, um so eine Folgeroute zu ermöglichen.

Keßel entwickelte 2002 ein Modell, dass sich in der grundlegenden Bewegungsmöglichkeit von dem zuvor genannten unterscheidet [97]. Auch hier wird den Bewegungen ein zweidimensionales Gitter aus zellulären Automaten zu Grunde gelegt. Allerdings sind in diesem Modell keine schrägen Bewegungen möglich, sondern nur Bewegungen in Zellen, die sich aus der von-Neumann-Nachbarschaft (siehe Abschnitt 2.2) ergeben. Jede Person erhält einen Driftwert, der ihren Vorwärtsdrang beschreibt. In diesem Modell wird vor der Durchführung der Bewegung die Anzahl der freien Nachbarschaftszellen bestimmt. Für alle freien Zellen wird eine Bewegungswahrscheinlichkeit bestimmt und schließlich gemäß dieser Wahrscheinlichkeiten fortgeschritten. Zur Bestimmung der Bewegungsrichtungen werden hierfür keine Potentialfelder benötigt.

Alle vorgenannten Modelle haben allerdings das Problem, dass sie lediglich eine konstante Bewegungsgeschwindigkeit erlauben. Klüpfel et al entwickelte ein ähnliches Modell, allerdings um die Möglichkeit unterschiedlicher Bewegungsgeschwindigkeiten erweitert [101]. Im Jahr 2006 wurde dieses Modell um die Möglichkeit von Diagonalbewegungen (Moore-Nachbarschaft) und um Potentialfelder erweitert [99, 100].

Blue und Adler entwickelten ebenfalls ein Modell, welches verschiedene Geschwindigkeiten der Personen erlaubt [19]. Sie erlauben vier unterschiedliche Geschwindigkeiten, was unterschiedliche Personengruppen reflektieren soll. Zusätzlich ist es in ihrem Modell für Personen möglich, vorausschauend zu handeln. In den anderen Modellen fielen Bewegungsentscheidungen erst bei lokal auftreffenden Problemen, beispielsweise wenn eine Person schon direkt vor einem Hindernis steht. Dieses Vorausschauen ermöglicht gezielte Spurwechsel zum Beibehalten der eigenen Wunschgeschwindigkeit und gezieltes Ausweichen von Hindernissen.

Neben verschiedenen Variationen der Bewegungsmöglichkeiten und Geschwindigkeiten besteht auch die Möglichkeit, die Zielbestimmung anders umzusetzen. In den vorgenannten Modellen geschieht dies – wie gezeigt – in der Regel unter Ausnutzung von Potentialfeldern.

Mundani et al verfolgen dabei einen anderen Ansatz [120]. Anstelle von Potentialfeldern wird die Route durch das Gitter über eine Kurzwegsuche zu Beginn der Simulation ermöglicht. Dem Modell liegt wieder ein zweidimensionaler zellulärer Automat zugrunde, der aus einem CAD-Modell abgeleitet wird, und ist daher vor allem für Personenbewegungen innerhalb von Gebäuden geeignet. Die Personen werden anschließend im Gebäude platziert und ihre Routen zu den gewünschten Zielen (beispielsweise zu den Ausgängen) mittels des Kurzwegsuchealgorithmus von Dijkstra ermittelt. Die Bewegungen der Personen orientieren sich an der von-Neumann-Nachbarschaft der Zellen.

Die genannten Modelle werden vor allem zur Simulation von Personenevakuierungen, beispielsweise aus Passagierschiffen [101], eingesetzt. Dementsprechend sind sie in einigen Bereichen etwas eingeschränkt: Die Personen sind meist nicht individuell identifizierbar, was für eine Evakuierungssimulation auch nicht notwendig ist und die Personen bewegen

sich in Richtung eines gewissen Ziels. Eine Abarbeitung von Aktivitätsplänen wie dies bei der mikroskopischen Simulation der Fall ist, ist bei diesen Modellen nicht notwendig oder nur sehr rudimentär vorgesehen. Anpassungen für die im Rahmen dieser Arbeit erforderlichen Modelle werden im Folgenden Abschnitt näher betrachtet.

8.2.2 Modell für Fußgängerbewegungen

Bei dem im Rahmen dieser Arbeit verwendeten Modell handelt es sich um ein einfaches Zellularautomatenmodell, welches die speziellen Anforderungen an die kombinierte mikroskopische Verkehrs- und Fußgängersimulation erfüllt: Dazu gehören eine Behandlung von Aktivitätsplänen, eine damit einhergehende effiziente Routenberechnung und eine Interaktion mit den Verkehrsmitteln der mikroskopischen Verkehrssimulation.

In diesem Modell wird aus Informationen des Verkehrsnetzes der Graph der Fußgängersimulation gewonnen. Dabei werden zweidimensionale zelluläre Automaten erstellt, die die Fußwege darstellen. Es bestehen dabei zwei Möglichkeiten, diese zu generieren: (1) manuell durch Definition entsprechender Strecken mit speziellem Typ (`Fussweg`) im Verkehrsnetz oder (2) automatisch, wobei parallel zu den schon existierenden Strecken des Verkehrsnetzes Fußwege erzeugt werden. Es werden dabei jeweils zelluläre Automaten zwischen je zwei Knoten des Netzes erzeugt. Die dabei entstehenden Automaten werden dabei anschließend miteinander verknüpft. Auf diese Weise entsteht ein Netz zellulärer Automaten, das die Fußwege darstellt.

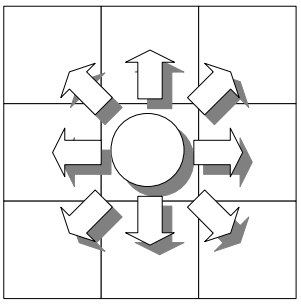


Abbildung 8.7: Bewegungsmöglichkeiten eines Fußgängers unter Berücksichtigung der Moore-Nachbarschaft in dem zweidimensionalen zellulären Automaten des Modells.

Die einzelnen Zellen der Automaten sind, wie in den obigen Modellen, Quadrate mit einer Kantenlänge von 40 cm. Für die Bewegungsmöglichkeiten wird die Moore-Nachbarschaft betrachtet, ähnlich wie dies in [26] der Fall ist. Dabei ergeben sich für einen Fußgänger in jedem Zeitschritt theoretisch neun mögliche Zellen in die er sich begeben kann (inklusive der Möglichkeit stehen zu bleiben). Dies ist in Abb. 8.7 dargestellt.

Analog zu [26] und [27] wird die maximale Geschwindigkeit der einzelnen Fußgänger auf $v_{max} = 1$ beschränkt, das heißt in jedem Zeitschritt können sich die Individuen maximal um

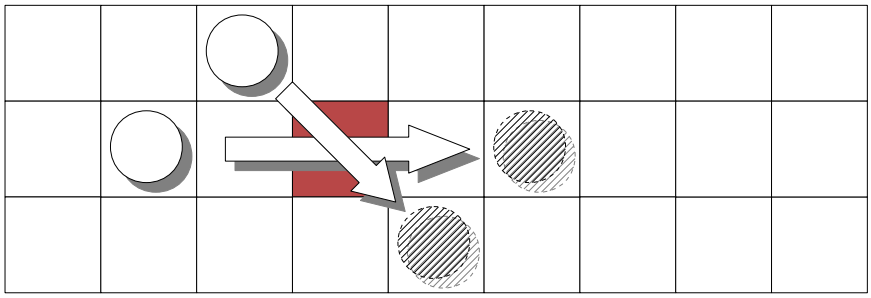


Abbildung 8.8: Mehrdeutigkeit bei Geschwindigkeiten der Fußgänger, die größer als eine Zelle pro Zeitschritt betragen. In der roten Zelle kreuzen sich die Wege der beiden Personen.

eine Zelle weiterbewegen. Wenn sich die Fußgänger mit höheren Geschwindigkeiten bewegen würden, so könnten, abhängig von der Art des Fortbewegens, Probleme auftreten. Ein mögliches Problem ist in Abb. 8.8 skizziert. Verwendet man für die Bewegung der Personen ein paralleles Update – wie dies bei der mikroskopischen Verkehrssimulation der Fall ist (siehe Abschnitt 2.2) – so kann es passieren, dass sich die Wege zweier Personen in einer Zelle kreuzen (rote Zelle in Abb. 8.8). Eine Lösung dieser Problematiken ist rechenintensiv und häufig nicht befriedigend durchzuführen. Eine Beschränkung auf eine Geschwindigkeit von einer Zelle pro Zeitschritt ist aber in der Regel ausreichend [146].

Person ID	Source	Start	Destination	Time	Destination	Time	...
0	10	9:00	40	12:00	37	16:15	...
1	12	8:30	23	10:00	15	14:00	...
2	15	11:15	20	13:30	25	15:45	...
⋮	⋮	⋮	⋮	⋮			

Abbildung 8.9: Aktivitätsliste der in der Simulation beteiligten Fußgänger.

Für die Bewegungen liegt ebenfalls für jede belegte Zelle eine 3×3-Matrix mit Übergangswahrscheinlichkeiten vor. Die Bewegungsrichtung der Fußgänger wird allerdings durch deren Aktivitätspläne bestimmt. Jeder Person kann ein Aktivitätsplan zugeordnet sein, das heißt eine Liste mit Zielen, die diese Person während des Simulationszeitraums besuchen will (siehe Abb. 8.9). Dabei ist jeder Person, die individuell durch eine `ID` identifiziert werden kann, ein Quellknoten (`Source`) und eine Uhrzeit zugeordnet, wann sie ihre Reise beginnt. Anschließend folgt eine Liste von Paaren mit Ziel (`Destination`) und Abreisezeitpunkt (`Time`).

So beginnt Person 0 ihre Reise an Knoten 10 und verlässt diesen um 9:00 Uhr in Richtung Knoten 40. Diesen wünscht sie um 12:00 wieder zu verlassen, um zu Knoten 37 zu gelan-

gen, usw. Kommt die Person dabei zu früh an einem Knoten an, so wartet sie an diesem bis der Abreisezeitpunkt erreicht ist.

Zur Routenfindung wird auf die Verwendung von Potentialfeldern verzichtet, da sich bei einer Vielzahl von Zielen zahlreiche Potentialfelder überlagern würden. Vielmehr erfolgt die Routenberechnung über Dijkstras Algorithmus zur Kurzwegsuche (vgl. [120]). Unter Verwendung von Schwarmtechnologie-Techniken kann dieses Verfahren auch zum dynamischen Re-Routing genutzt werden, wenn sich für Personen zu lange Wartezeiten ergeben (beispielsweise zu viele Fußgänger verstopfen einen Weg, oder ein Weg ist anderweitig blockiert) [86, 128, 157].

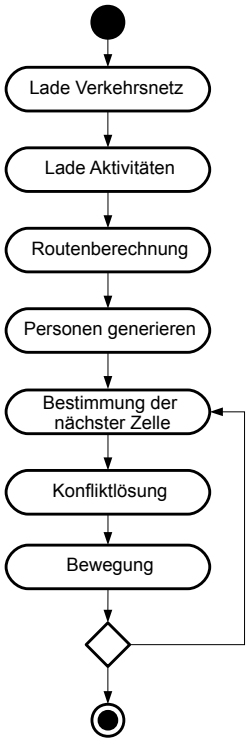


Abbildung 8.10: Ablauf der Fußgängersimulation.

Abb. 8.10 zeigt den Ablauf der Fußgängersimulation. Zunächst werden dabei die Initialdaten, wie Verkehrsnetz und Aktivitäten, geladen. Entsprechend der Aktivitäten erfolgen anschließend die Routenberechnung. Dabei wird auf eine mögliche Wiederverwendbarkeit von Teilrouten oder Routen geachtet. Nach der Generierung der Personen beginnt die eigentliche Simulation. Dabei handelt es sich um einen 3-phasigen Prozess, bei dem jede Phase für alle Fußgänger parallel durchgeführt wird.

1. *Bestimmung der nächsten Zelle*: Entsprechend der Moore-Nachbarschaft und der Bewegungsrichtung des Fußgängers wird diejenige freie Zelle ermittelt, in die sich der Fußgänger als nächstes bewegen wird. In diesen Bereich fällt auch gegebenenfalls

ein Re-Routing, wenn die Wartezeiten der Person zu lange sind oder der Fußweg blockiert ist.

2. *Konfliktlösung*: Durch das parallele Bearbeiten von Phase 1, kann es passieren, dass sich nach einer Bewegung mehrere Personen dieselbe Zelle teilen würden, was nach dem Modell nicht gestattet ist. Dementsprechend erfolgt in dieser Phase eine Auflösung dieser Konflikte. Mit gewissen Wahrscheinlichkeiten wählen einige der beteiligten Personen eine alternative Zelle oder bleiben auf ihrer bisherigen Zelle stehen.
3. *Bewegen*: Alle Personen werden auf die in den ersten beiden Phasen ermittelten Zellen bewegt.

Nach Ende der dritten Phase wird überprüft, ob das Ende der Simulation erreicht wurde. Ist dies nicht der Fall, so beginnt die Simulation erneut mit Phase 1.

8.2.3 Interaktion mit der mikroskopischen Simulation

Nachdem im vorherigen Abschnitt das in dieser Arbeit verwendete Modell beschrieben wurde, wird nun die Interaktion der Fußgängersimulation mit der mikroskopischen Verkehrssimulation beschrieben. Um das Framework so flexibel wie möglich zu halten, fiel die Entscheidung, beide Simulationen als eigenständige Programme zu erhalten und miteinander zu koppeln (ähnlich wie es beim OD-Ansatz der Kopplung von mikroskopischer und makroskopischer Simulation in Abschnitt 8.1.3 geschehen ist). Diese Kopplung ist ebenfalls in Abb. 8.11 skizziert.

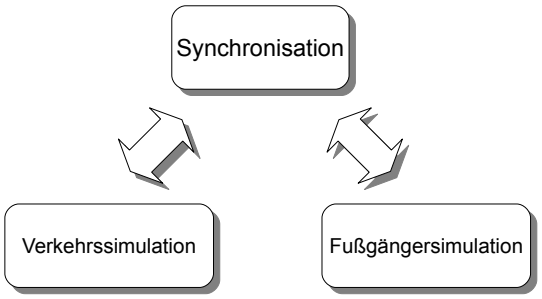


Abbildung 8.11: Kopplung der mikroskopischen Fußgängersimulation und der mikroskopischen Verkehrssimulation. Eine zentrale Instanz synchronisiert dabei das Fortschreiten der beiden Simulationen entsprechend ihrer Uhren.

Dabei sorgt eine zentrale Instanz (*Synchronisation*) für die Synchronisation der beiden Simulationszeiten und übernimmt die Kontrolle über den Datenaustausch. Allerdings ist nur eine Art von Interaktion vorgesehen. Eine Beeinflussung des Straßenverkehrs beim Überqueren von Straßen durch Fußgänger ist nicht vorgesehen. Vielmehr „überspringen“ die Fußgänger einfach eine Straße, wenn sie diese Überqueren müssen. Für dieses Überspringen erhalten sie einen einfachen Zeitzuschlag, das heißt ihre Reisezeit und damit ihr neues

Auftauchen auf der anderen Straßenseite wird um eine Zeit Δt verzögert.

Eine direkte Interaktion besteht allerdings bei er Kopplung von Fußgängern und dem mikroskopischen Öffentlichen Verkehr. Wie in Abschnitt 5.1.2 dargestellt, existieren in der Mikrosimulation zwei Möglichkeiten Haltestellen zu modellieren. Die hier interessanteste ist die Verwendung einer Warteschlange. Die Haltestellen sind eindeutig als Knoten im Verkehrsnetz identifizierbar. Sie können also über als Ziele von Aktivitäten bei Fußgängern angeben werden.

Geschieht dies und kommt ein Fußgänger an einer Haltestelle an, so wird er in die Warteschlange der Haltestelle eingehängt und wartet auf das Ankommen eines Busses. Die Behandlung erfolgt dabei gemäß Abschnitt 5.1.2. Das nächste Ziel wird dann über den Bus erreicht.

8.3 Zusammenfassung

In diesem Kapitel wurden zwei mögliche Erweiterungen des Frameworks gezeigt. Zum einen eine Kopplung zwischen einer mikroskopischen Verkehrssimulation basierend auf einem Zellularautomatenmodell und einer stochastischen Umlegung. In diesem Fall wird ein bestehendes Verkehrsnetz in einen mikroskopischen und einen makroskopischen Teil aufgeteilt, die dann den jeweiligen Simulationen übergeben werden. Es wurden zwei Ansätze zur Kopplung vorgestellt, den OD-Ansatz und einen integrierten Ansatz. Wie die Ergebnisse in Kapitel 9 zeigen, ist die Qualität der Kopplung ausreichend, aber noch verbesserungswürdig.

Als zweites wurden eine mikroskopische Fußgänger- und Verkehrssimulation gekoppelt. Anders als in anderen Ansätzen wurde hier auf eine direkte Integration der Fußgängersimulation in die mikroskopische Simulation verzichtet, diese wurde vielmehr als eigenständige Simulation beibehalten. Dies erlaubt eine flexible Erweiterbarkeit des Frameworks bzw. den Austausch einzelner Komponenten.

9 Ergebnisse

In den vorherigen Kapiteln wurde das im Rahmen dieser Arbeit entstandene Framework zur Verkehrssimulation mit seinen Konzepten und Implementierungen vorgestellt. Um eine Bewertung dieses Frameworks vornehmen zu können, bedarf es allerdings noch verschiedener Analysen und Evaluierungen, insbesondere zu dem Verhalten der vorgestellten Parallelisierungsstrategien.

Das Kapitel beginnt zunächst mit einer Beschreibung der simulierten Szenarien und der Hardware, auf der die Berechnungen durchgeführt wurden. Daran schließt sich ein kurze Zusammenstellung der Ergebnisse zur Bewertung der verkehrssimulationsspezifischen Aspekte an. Das Kapitel endet mit einer detaillierten Analyse der Leistungsmerkmale der vorgestellten Simulationen.

9.1 Szenarien

Für die Evaluation des im Rahmen dieser Arbeit entstandenen Frameworks gilt es einige Testszenarien zu definieren. Es bietet sich an, einige der in Kapitel 2 genannten Verkehrsnetze dafür heranzuziehen. Eine sinnvolle Auswahl ergibt sich, wenn man von der Grundstruktur her möglichst unterschiedliche Verkehrsnetze verwendet, um ein breites Spektrum an Fällen abzudecken. Tabelle 9.1 zeigt die gewählten Verkehrsnetze, die zum Teil von der PTV AG zur Verfügung gestellt wurden beziehungsweise im Rahmen des Projekts „Multimodale Simulation des Verkehrsablaufs in großen Netzen“ der Landesstiftung Baden-Württemberg entstanden.

Das Netz *Karlsruhe* ist relativ klein und reagiert damit potentiell sensibel auf aufwändige Partitionierungsverfahren oder Parallelisierungsstrategien. Zum anderen handelt es sich dabei um ein reines Stadtnetz, was zu einem kompakten Straßennetz mit vielen Kreuzungen führt, die sich vor allem bei der mikroskopischen Simulation bemerkbar machen dürften. Demgegenüber steht das Verkehrsnetz *Deutschland*, welches auf der einen Seite schon

	Knoten	Strecken	Bezirke	Anbindungen	Abbiege-beziehungen
Karlsruhe	8.355	23.347	725	5.601	72.453
Region Stuttgart	149.652	369.009	784	14.787	1.030.269
Deutschland	109.842	758.139	6.928	55.347	643.382
DB-Netz (ÖV)	14.935	32.509	996	1.935	76.493
New York	264.346	733.846	812	26.923	–

Tabelle 9.1: Kennzahlen einiger ausgewählter Verkehrsnetze

deutlich größer als das von Karlsruhe ist. Auf der anderen Seite ist es ein von der Struktur her gröberes Netz, dass vor allem Straßen höherer Hierarchiestufen (wie etwa Autobahnen und Bundesstraßen) und keinerlei innerstädtische Bereiche enthält. Als Szenario zwischen diesen beiden Extrema ist das Verkehrsnetz *Region Stuttgart* anzusehen, welches neben einem größeren außerörtlichen Anteil an Straßen auch zahlreiche Stadtgebiete umfasst. Auf diese Weise ist es durch eine sehr heterogene Struktur geprägt. Schließlich dient das Stadtnetz *New York* als Testszenario für die Umlegung des motorisierten Individualverkehrs. Auf Grund der fehlenden Abbiegebeziehungen wird es für die mikroskopische Verkehrssimulation ignoriert. Speziell für die makroskopische Simulation des Öffentlichen Verkehrs steht das Schienennetz der Deutschen Bahn AG (*DB-Netz*) zur Verfügung.

9.2 Testumgebung

Die im vorherigen Abschnitt vorgestellten Testszenarien wurden auf einem nachrichtengekoppelten Multiprozessorsystem – einem Cluster aus 32 Knoten – ausgeführt. Jeder dieser Knoten des Clusters setzt sich dabei aus je vier AMD Opteron 850 Prozessoren mit jeweils einer Taktfrequenz von 2,4 GHz und acht Gigabyte Hauptspeicher zusammen. Mit dem Festplattenspeicher von 3,4 Terabyte sind die Knoten jeweils über eine Gigabit Ethernet Verbindung verbunden. Die Kommunikation der einzelnen Prozessoren geschieht über einen Infiniband Switch, welcher 96 Ports mit jeweils vier Kanälen besitzt.

Zur Entwicklung der parallelen Algorithmen für nachrichtengekoppelte Systeme wurde die freie MPI-Implementierung [115] MPICH2[1] für C/C++ verwendet.

9.3 Evaluierung der Verkehrssimulationen

Ein wichtiges Kriterium, mit dessen Hilfe man versucht, eine Simulation (v.a. eine mikroskopische Simulation) zu evaluieren, ist, das klassische Fundamentaldiagramm (siehe

[1] `http://www.mcs.anl.gov/research/projects/mpich2/`

Kapitel 2) nachzubilden. Abb. 9.1 zeigt das aus dem in Kapitel 5 vorgestellten mikroskopischen Simulator mit heterogenem Verkehr gewonnene Fundamentaldiagramm. Dabei sind die wichtigen Bereiche identifizierbar, die auch in Fundamentaldiagrammen, die aus realen Verkehrsdaten ermittelt wurden, zu finden sind. Dazu gehören der Freiflussbereich, der Bereich des gebundenen Verkehrs und der Staubereich. So ist im ersten Teil des Diagramms, die Region des freien Flusses erkennbar, das heißt hier bewegen sich die Fahrzeuge noch ungehindert voneinander auf der Straße. Nach einer kurzen Phase des gebundenen Verkehrsflusses folgt die Stausituation. Durch die Heterogenität der Verkehrsteilnehmer und deren unterschiedliches Verhalten kommt eine breitere Streuung des hinteren Bereichs des Diagramms zustande. Ähnliche Ergebnisse sind auch in [126] zu finden.

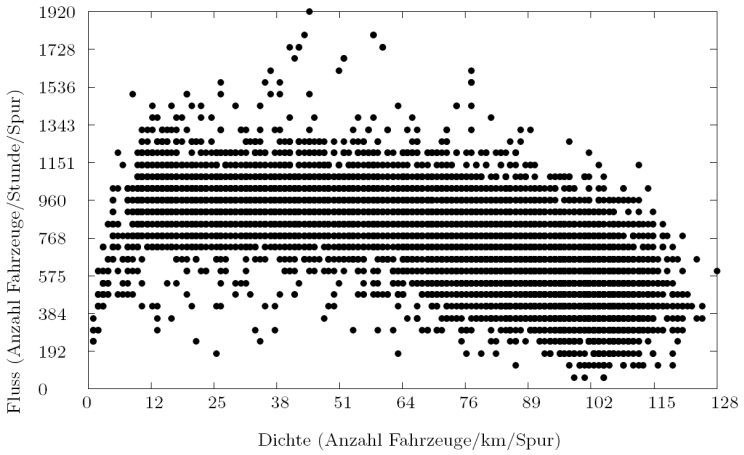


Abbildung 9.1: Fundamentaldiagramm des heterogenen, mikroskopischen Verkehrs.

Abb. 9.2 zeigt ein Weg-Zeit-Diagramm für eine typische Strecke innerhalb der heterogenen Verkehrssimulation. Es wird dabei lediglich ein Fahrstreifen betrachtet. Die Größe der Dreiecke im Diagramm gibt den Fahrzeugtyp wieder: ein kleines Dreieck stellt Zweiräder dar, ein großes normale Pkw. Die Farben geben die Geschwindigkeit der einzelnen Fahrzeuge wieder, wobei die Farbe blau dem Stillstand entspricht und rot der maximalen Geschwindigkeit der Strecke. Die durchschnittliche Fahrzeugdichte auf der Strecke entspricht 15%, etwa 20% der Fahrzeuge trödeln (sowohl bei Zweirädern als auch bei den Pkw). Wie gut zu erkennen ist, kommt es auch hier zu den charakteristischen Stop-and-Go-Wellen wie sie für solche Diagramme und die Realität typisch sind.

Als Beispiel einer makroskopischen Umlegung des motorisierten Individualverkehrs wurde das Verkehrsnetz der Stadt New York gewählt. Abb. 9.3 zeigt das Ergebnis einer stochastischen Umlegung für einen Zeitraum von 24h mit einer zufällig generierten Fahrtenmatrix. Die unterschiedlichen Farben der Strecken geben ihre Belastungen wieder (hohe Streckenbelastung rot, geringe Streckenbelastung grün).

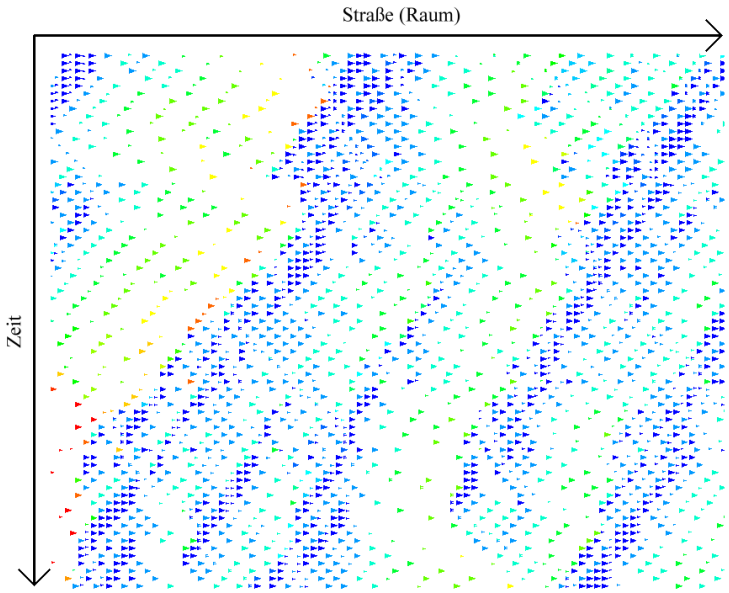


Abbildung 9.2: Weg-Zeit-Diagramm zum heterogenen, einstreifigen, mikroskopischen Verkehr. Kleine Dreiecke stellen Zweiräder dar, größere Pkw. Die Farben geben die Geschwindigkeiten an, wobei blau einem Stillstand entspricht und rot der Maximalgeschwindigkeit.

Abbildung 9.3: Umlegungsergebnis für das New Yorker Stadtnetz in einem Simulationszeitraum von 24h an einem Werktag.

9.3.1 Kopplung der makroskopischen und mikroskopischen Simulation

Die Kopplung zwischen der stochastischen Umlegung und der vorgestellten mikroskopischen Simulation soll im Folgenden exemplarisch am Beispiel des Karlsruher Netzes gezeigt werden (siehe Abb. 9.4 und 9.5). Hierfür wurde das Verkehrsnetz in mikro- und makroskopische Komponenten aufgeteilt. Die Strecken wurden gemäß ihrer Maximalgeschwindigkeit entweder dem mikroskopischen oder dem makroskopischen Teil zugeordnet. Als Grenzwert für die Trennung wurde dabei eine Geschwindigkeit von 40 km/h und eine Länge der Zeitscheiben von 30 min gewählt. Farblich dargestellt sind die Belastungen im mikroskopisch modellierten Teil des Verkehrsnetzes. Die makroskopischen Teile des Verkehrsnetzes sind dabei nicht visualisiert.

Abbildung 9.4: Gekoppelte Simulation des Karlsruher Verkehrsnetzes unter Verwendung einer Split-Geschwindigkeit von 40 km/h und einer Länge der Zeitscheiben von 30min: Integrierter Ansatz (links) und mikroskopische Referenzsimulation (rechts).

Abbildung 9.5: Gekoppelte Simulation des Karlsruher Verkehrsnetzes unter Verwendung einer Split-Geschwindigkeit von 40 km/h und einer Länge der Zeitscheiben von 30min: OD-Ansatz (links) und mikroskopische Referenzsimulation (rechts).

In den Abb. 9.4 (rechts) und 9.5 (rechts) ist das Ergebnis einer rein mikroskopischen Si-

mulation (mittels dem Basismodell von Nagel und Schreckenberg) dargestellt und dient als Vergleichswert für die Kopplungen. Der integrierte Ansatz weist eine gute Übereinstimmung auf. Im Durchschnitt weichen die Belastungen um lediglich 3% vom Referenzwert der mikroskopischen Simulation ab. Die Abweichung ist beim OD-Ansatz mit knapp 10% deutlich größer. Dies liegt unter anderem daran, dass es beim OD-Ansatz zu Abweichungen durch die Weitergabe der QZ-Matrizen kommt. Der OD-Ansatz ist dann möglichst genau, wenn die Durchquerung des makroskopischen Teils ungefähr der Dauer einer Zeitscheibe entspricht.

9.4 Leistungsdaten des Frameworks

In einem nächsten Schritt werden die Leistungsdaten des Frameworks und damit seiner Einzelkomponenten evaluiert. Zunächst sollen die Daten der mikroskopischen Simulation diskutiert werden, bevor eine Zuwendung zur makroskopischen Simulation stattfindet.

9.4.1 Mikroskopische Simulation

Im Rahmen der mikroskopischen Simulation entstand ein Routensuchalgorithmus, der auf der klassischen Kurzwegsuche von Dijkstra aufsetzt und diesen um Konzepte der Hierarchie und Abbiegebeziehungen erweiterte (siehe Kapitel 5). Abb. 9.6 zeigt die Hierarchieebenen des Stadtnetzes von Karlsruhe. Die Hierarchiestufen steigen dabei von Stufe 0 (blau) bis hin zu Stufe 5 (rot) an und orientieren sich an den jeweiligen Straßentypen, die im Netz vertreten sind. In den Abbildungen 9.7 und 9.8 sind exemplarisch drei Suchbäume dargestellt, die drei Routen innerhalb des Netzes berechnen. Die Quell- und Zielknoten der jeweiligen Routen sind durch rote Pfeile markiert. Die rot gefärbten Kanten geben dabei die im Laufe der Suche betrachteten Strecken – den Suchbaum – wieder. Wie in Abb. 9.7 zu erkennen ist, gelangt die Routensuche im unteren Teil in einen Bereich des Verkehrsnetzes, in dem sich lediglich Strecken niederer und ähnlicher Hierarchiestufe befinden. Es findet an dieser Stelle eine Verbreiterung des Suchbaums statt. In dieser Situation verhält sich der entwickelte Algorithmus vergleichbar des klassischen Dijkstra.

Anders sieht die Situation in Abb. 9.8 aus. Die Routensuche bewegt sich in erster Linie auf Strecken höherer bzw. hoher Hierarchiestufen (Strecken mit hoher Maximalgeschwindigkeit). Die Suchbäume sind deutlich kompakter und damit der Suchraum kleiner. Dies führt zu einer deutlichen Beschleunigung der Routensuche.

Betrachtet man nun das Laufzeitverhalten der mikroskopischen Simulation für die verschiedenen Testszenarien, so kann man die Unterschiede der verschiedenen Verkehrsnetze erkennen. In allen Szenarien wurde ein Zeitraum von einer Stunde simuliert. Abb. 9.9 zeigt die Speedup-Werte des Straßennetzes von Karlsruhe, Abb. 9.10 diejenigen des Straßennetzes von Deutschland und Abb. 9.11 die vom Verkehrsnetz der Region Stuttgart.

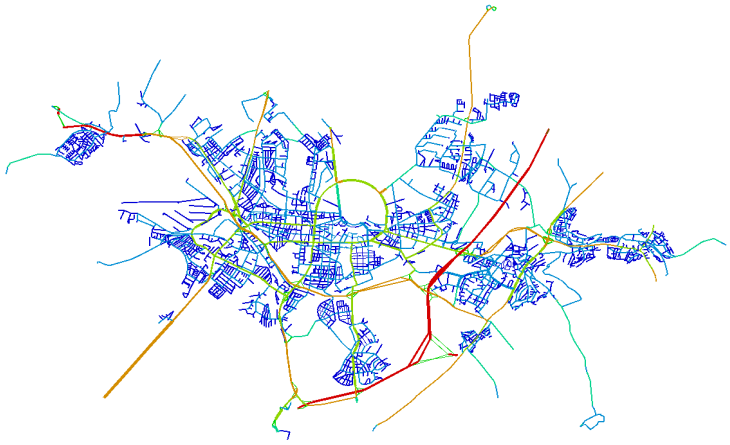


Abbildung 9.6: Hierarchiestufen des Straßennetzes der Stadt Karlsruhe.

Abbildung 9.7: Suchbaum für die Ermittlung einer Route im Stadtnetz von Karlsruhe mittels des entwickelten Hierarchie-basierten Dijkstras. Die Pfeile geben den Quell- beziehungsweise den Zielpunkt an. Die rot markierten Strecken wurden bei der Routensuche betrachtet und dem Suchbaum hinzugefügt.

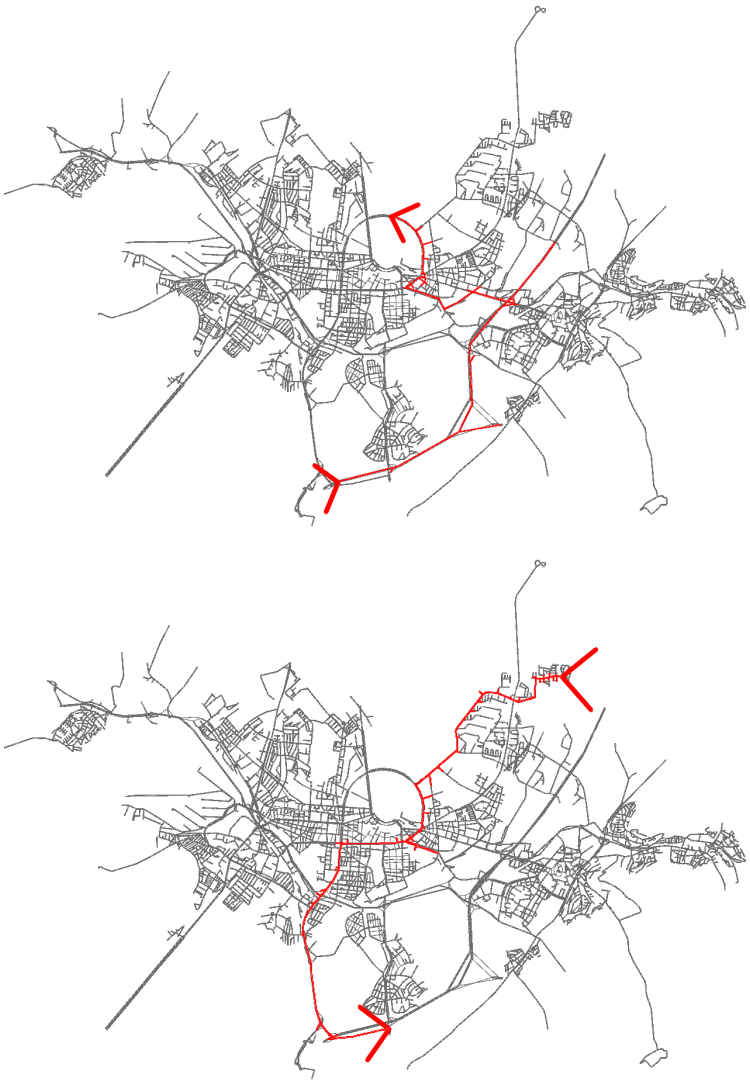


Abbildung 9.8: Weitere Suchbäume für die Ermittlung einer Route in Karlsruhe mittels des entwickelten Hierarchie-basierten Dijkstras. Deutlich ist hier die Bevorzugung der höheren Stufen zu erkennen, was den Suchbaum reduziert.

Es werden dabei jeweils die beiden in Kapitel 2 beschrieben Lastverteilungsstrategien – statisch und dynamisch – gegenübergestellt. Die Ausführungszeit der Simulation im sequentiellen Fall der Simulation des Deutschlandnetzes beträgt etwa 3,3 Stunden, bei dem Stadtnetz von Karlsruhe etwa 0,3 Stunden und schließlich bei dem Szenario der Region Stuttgart ca. 2,6 Stunden.

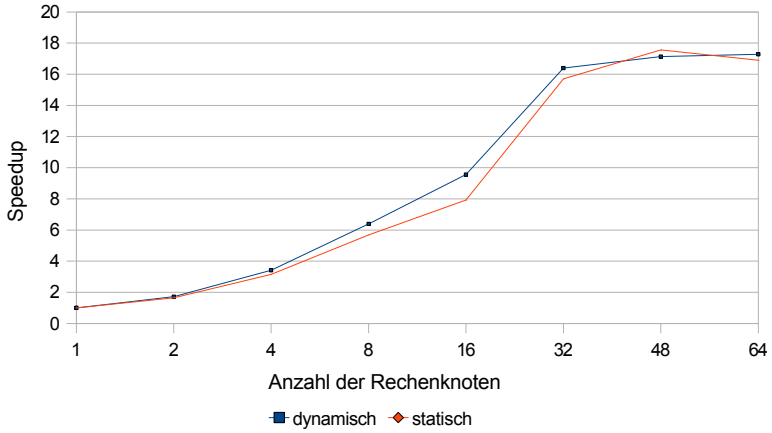


Abbildung 9.9: Speedup für die parallele mikroskopische Simulation des Karlsruher Stadtnetzes (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

Im Falle des Karlsruher Testszenarios sind zwei Dinge erkennbar. Erstens, die dynamische Lastverteilung ist der statischen meist leicht überlegen. Zweitens, die Speedup-Werte erreichen relativ schnell bei etwa 32 Rechenknoten eine Sättigung. Die Ursache für letzteres ist darin zu suchen, dass die Laufzeiten des parallelen Teils des Programms bei steigender Anzahl der Rechenknoten immer weiter sinkt. Wohingegen der sequentielle Teil (Einlesen des Netzes, der Nachfragedaten, Initialisierung, ...) stabil bleibt. Die parallele Ausführungszeit konvergiert nun gegen diesen sequentiellen Sockel, wodurch hier ab ca. 32 Rechenknoten nur noch marginale Verbesserungen erzielt werden können. Zusätzlich erhöht sich bei steigender Rechenknotenanzahl auch der Kommunikations- und Synchronisationsaufwand, welcher schließlich den Vorteil der parallelen Ausführung auffrisst.

In den beiden weiteren Testszenarien ist die statische Lastverteilung bis etwa 16 Rechenknoten der dynamischen leicht überlegen. Dies liegt vor allem daran, dass die Zusatzkosten der dynamischen Lastverteilung (Repartitionierung, ...) relativ hoch sind, sich ihre Vorteile aber erst bei einer höheren Anzahl von Rechenknoten bemerkbar machen. In allen Fällen ist aber festzuhalten, dass die niedrigen Speedup-Werte zum Teil auf die Wartezeiten bei den Kommunikations- und Synchronisationsschritten zurückzuführen sind. Vor einer Datenkommunikation müssen zunächst alle Rechenknoten synchronisiert werden. Angesichts dieser Werte erscheinen andere Kommunikationsstrategien als interessant.

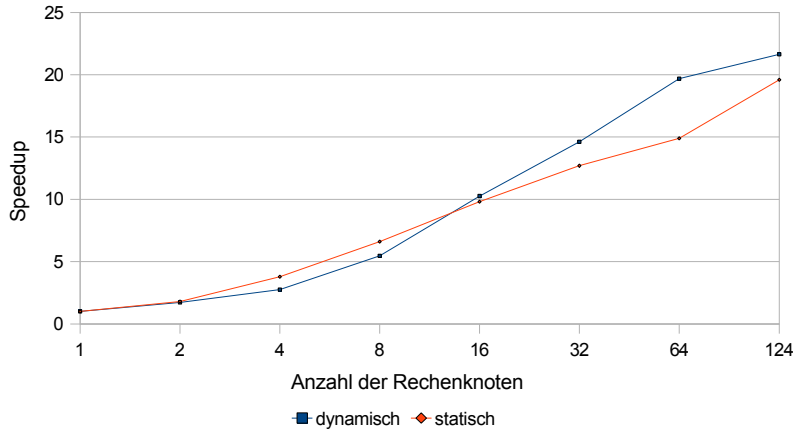


Abbildung 9.10: Speedup für die parallele mikroskopische Simulation des Deutschlandnetzes (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

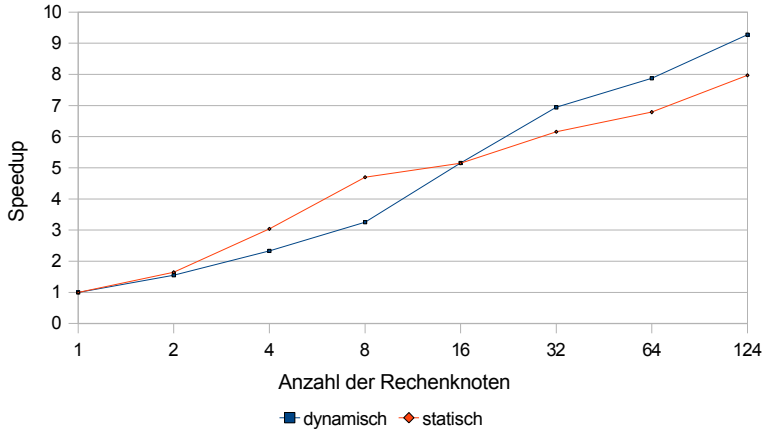


Abbildung 9.11: Speedup für die parallele mikroskopische Simulation des Verkehrsnetzes der Region Stuttgart (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

Beim Netz der Region Stuttgart bleibt festzuhalten, dass die Speedup-Werte dort besonders niedrig sind. Dies ist vor allem auf die Struktur des Verkehrsnetzes selbst zurückzuführen. Durch diese ergeben sich zwangsläufig bei einer Partitionierung zahlreiche Schnittkanten. Der Anteil der Kommunikation ist bei diesem Verkehrsnetz daher besonders hoch.

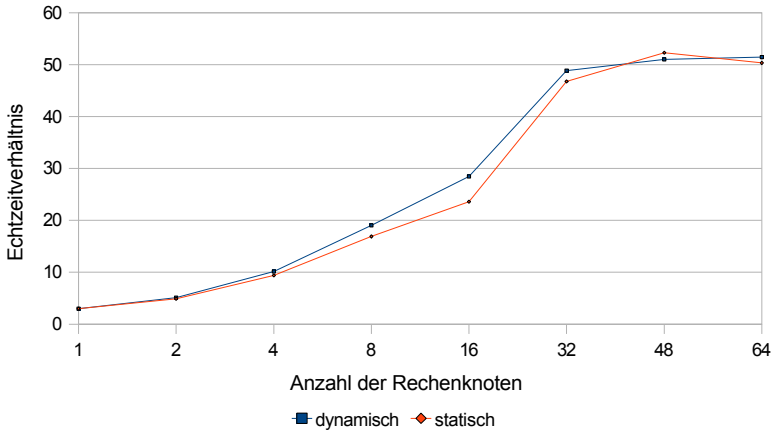


Abbildung 9.12: Echtzeitverhältnis für die parallele mikroskopische Simulation des Karlsruher Stadtnetzes (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

Diese Punkte schlagen sich auch in den Werten der Echtzeitverhältnisse nieder. Man kann bei den Karlsruher Werten erkennen, dass dies das einzige Szenario ist, bei dem das Echtzeitverhältnis schon im sequentiellen Fall größer als eins ist, d.h. die Simulation benötigt weniger Zeit als durch den Simulationszeitraum abgedeckt. Bei den beiden anderen Szenarien gelingt dies erst bei etwa vier Rechenknoten. Die Werte des Echtzeitverhältnisses für das Karlsruher Netz sind allerdings mit anderen Arbeiten kompetitiv. Hinzu kommt, dass bei anderen Arbeiten (siehe Kapitel 2) häufig nur homogener Verkehr betrachtet wird, in dieser Arbeit allerdings selbst die Simulation heterogenen Verkehrs schneller als Realzeit möglich ist. Zusätzlich ist die Größe der Verkehrsnetze in dieser Arbeit oft umfangreicher als in verwandten Arbeiten. Beispielsweise wird in [154] ein Szenario untersucht, das die Region Dallas umfasst und mit etwa 25.000 Strecken und 9.900 Knoten modelliert ist und für diesen Fall vergleichbare Werte wie bei Karlsruhe liefern.

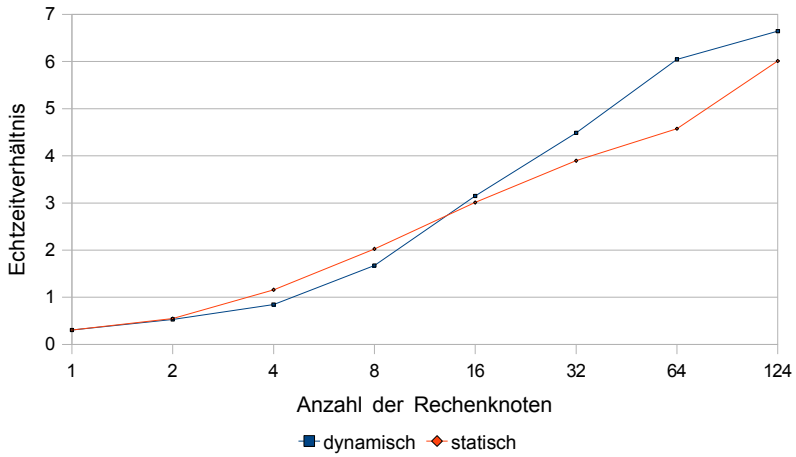


Abbildung 9.13: Echtzeitverhältnis für die parallele mikroskopische Simulation des Deutschlandnetzes (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

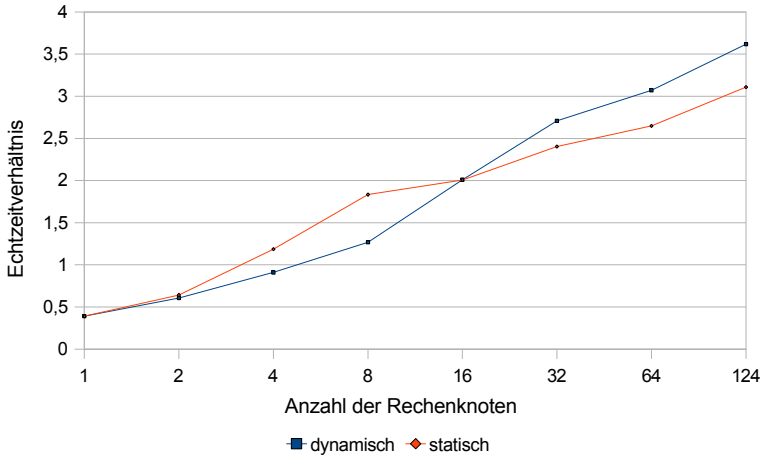


Abbildung 9.14: Echtzeitverhältnis für die parallele mikroskopische Simulation des Verkehrsnetzes der Region Stuttgart (statische und dynamische Lastverteilung). Simuliert wurde ein Zeitraum von einer Stunde.

9.4.2 Makroskopische Simulation

Bei der makroskopischen Simulation gilt es zunächst, die beiden Typen – ÖV und IV – zu unterscheiden. Zunächst erfolgt eine Betrachtung des makroskopischen Öffentlichen Verkehrs. Hierbei ist vor allem eine Analyse des Laufzeitverhaltens von Interesse. Daran schließt sich die Simulation des motorisierten Individualverkehrs an. Dabei werden die verschiedenen Strategien der Parallelisierung näher betrachtet und bewertet.

Simulation des Öffentlichen Verkehrs

Wie in Kapitel 6 dargelegt, findet bei der Parallelisierungsstrategie des ÖV eine Aufteilung der Quelle-Ziel-Matrix und eine dynamische Lastverteilung nach dem Master-Slave-Prinzip statt. Abb. 9.15 zeigt dabei die Speedup-Werte für die Umlegung der verschiedenen Testszenarien für einen Zeitraum von 24h.

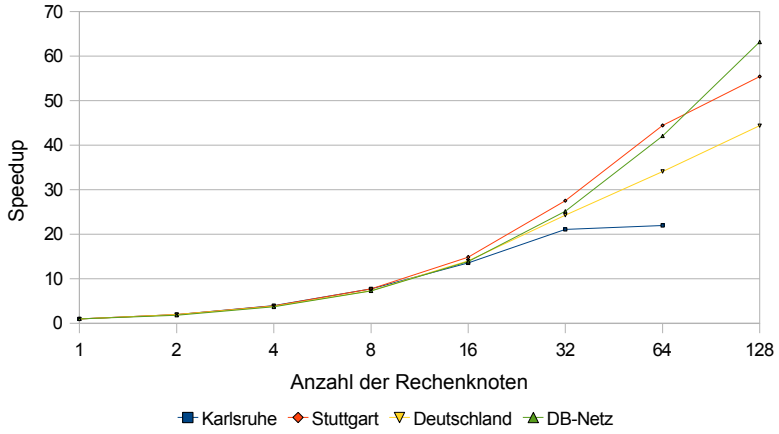


Abbildung 9.15: Speedup der Umlegung des Öffentlichen Verkehrs in den verschiedenen Verkehrsnetzen für einen Umlegungszeitraum von 24h.

Die sequentiellen Ausführungszeiten für die verschiedenen Szenarien sind dabei für Karlsruhe etwa 0,3 Stunden, bei der Region Stuttgart ca. 3,2 Stunden, sowie für das Deutschlandnetz und das Schienennetz der Deutschen Bahn (DB-Netz) 6,5 bzw. 7 Stunden. Im Karlsruher Netz ist eine Sättigung ab etwa 32 Knoten zu erkennen. Dies liegt wieder daran, dass bei steigender Anzahl von Rechenknoten, der Anteil des parallelen Teils (die eigentliche Berechnung) an der Gesamtlaufzeit stetig sinkt, während der sequentielle Teil (Initialisierungen, . . .) konstant bleibt.

Die Echtzeitverhältnisse sind (vor allem beim Karlsruher Netz) sehr hoch. Vergleichbare Werte existieren für die im Rahmen dieser Arbeit entstandene Parallelisierungsstrategie der fahrplanfeinen Umlegung in der Literatur nicht.

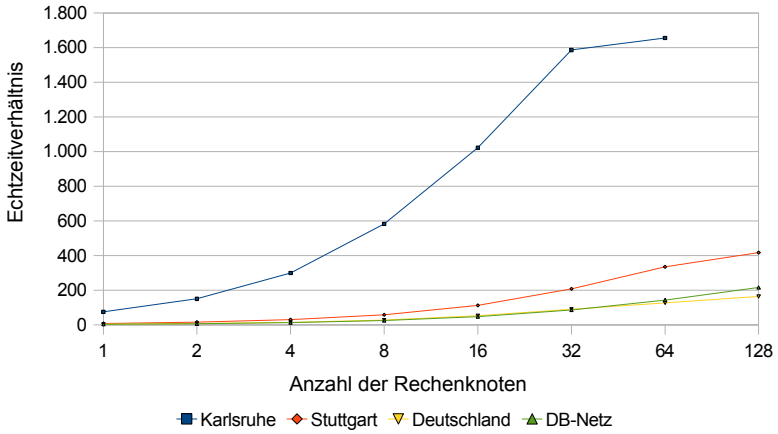


Abbildung 9.16: Echtzeitverhältnis der Umlegung des Öffentlichen Verkehrs in den verschiedenen Verkehrsnetzen für einen Umlegungszeitraum von 24h.

Simulation des Individualverkehrs

Bei der Betrachtung der Leistungsdaten der makroskopischen Simulation des Individualverkehrs gilt es zusätzliche Aspekte zu beachten. Zum einen wurden im Rahmen dieser Arbeit zwei Parallelisierungsstrategien entwickelt. Zum anderen ist es vor allem bei der zweiten vorgestellten Strategie, der Partitionierung der Verkehrsnetze, von entscheidender Bedeutung, dieses Verfahren in seinen einzelnen Aspekten im Detail zu betrachten. Daher wird diese zweite Strategie zunächst betrachtet.

Die Partitionierung des Verkehrsnetzes – sei es mittels rekursiver Koordinatenbisektion oder der Multilevelpartitionierung durch ParMETIS – lag bei allen untersuchten Szenarien im Sekundenbereich und fiel im Vergleich mit der kostspieligen Routensuche nicht weiter ins Gewicht. Dennoch ist die gewählte Partitionierungsstrategie von entscheidender Bedeutung für die Berechnungszeit, die die Routensuche benötigt. Je höher die Schnittkantenzahl ist, desto höher ist der Kommunikations- und Synchronisationsaufwand. Abb. 9.17 (oben) zeigt die Schnittkantenzahl bei der Partitionierung des Karlsruher Netzes bei Verwendung der beiden behandelten Partitionierungsmethoden. Analog sind die Schnittkanten für das Stuttgarter Netz in Abb. 9.17 (unten) dargestellt. In beiden Fällen ist erkennbar, dass ParMETIS die deutlich niedrigere Schnittkantenzahl liefert. Besonders offensichtlich ist dies beim Stuttgarter Verkehrsnetz, da dieses durch seine Struktur zwangsläufig eine sehr hohe Schnittkantenzahl verursacht.

Wie bereits erwähnt, hat die Anzahl der Schnittkanten einen großen Einfluss auf das weitere Laufzeitverhalten der Routensuche. Besonders beim arc flag Verfahren macht sich das bemerkbar, da hier für jede Schnittkante der Suchalgorithmus mindestens einmal angestoßen werden muss [116]. Aus diesem Grund wird im Weiteren nur noch die Partitionierung

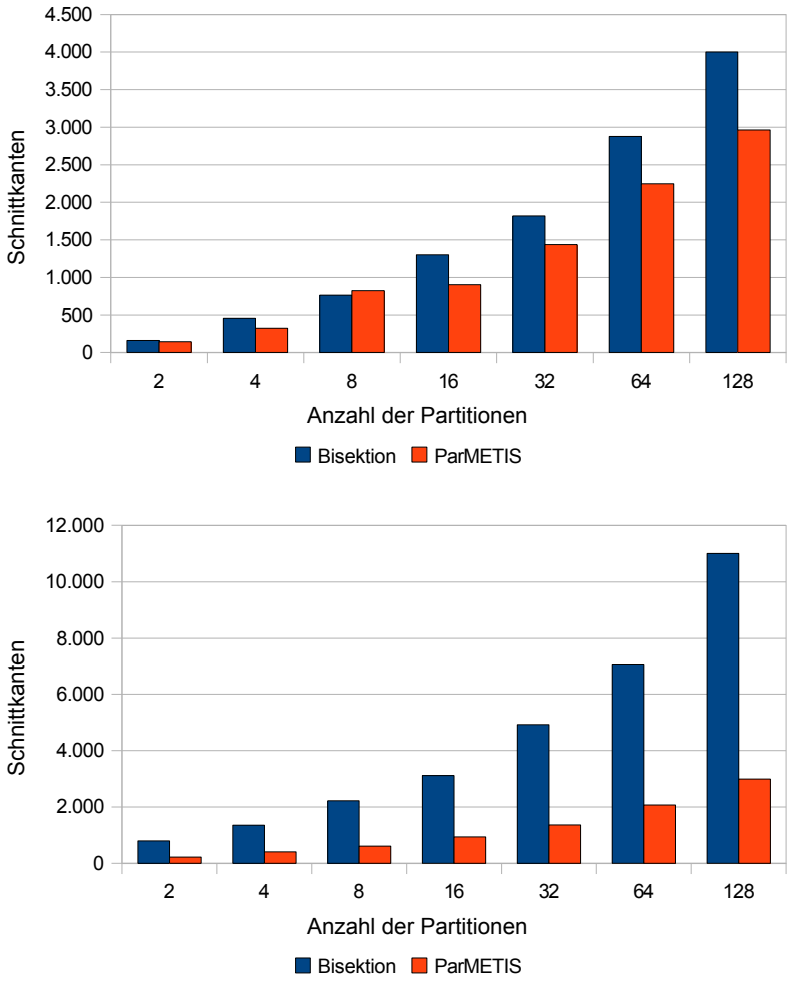


Abbildung 9.17: Anzahl der Schnittkanten bei der Anwendung der rekursiven Koordinatenbisektion (blau) und der Verwendung der Multilevelpartitionierung mittels ParMETIS (orange) am Beispiel des Stadtnetzes von Karlsruhe (oben) und des Verkehrsnetzes der Region Stuttgart (unten).

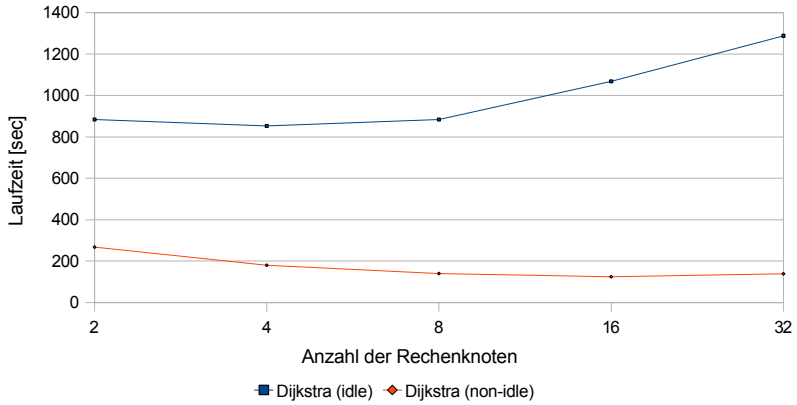


Abbildung 9.18: Vergleich der Routenberechnung zwischen dem parallelen Dijkstra (blau) und dem erweiterten parallelen Dijkstra (orange), der mehrere Routen zeitgleich zu berechnen versucht, im Verkehrsnetz Deutschlands.

mittels ParMETIS betrachtet.

Nach der Betrachtung der Partitionierungscharakteristika ist es sinnvoll, sich der Routensuche zuzuwenden. In Abschnitt 7.2.5 wurde ein paralleler Dijkstra entwickelt, der zunächst lediglich eine Route zeitgleich berechnen konnte. Dieses Verfahren wurde schließlich durch die Ausnutzung von *idle*-Zeiten dahingehend erweitert, dass es die Berechnung mehrerer Routen parallel erlaubte. In Abb. 9.18 sind die Laufzeitverhalten dieser beiden Verfahren gegenübergestellt. In beiden Fällen wurden jeweils 1.000 Routen im Verkehrsnetz von Deutschland berechnet. Deutlich sind die Vorteile des erweiterten Verfahrens erkennbar. Der Laufzeitanstieg des „einfachen“ lässt sich darauf zurückführen, dass sich dabei nur der Synchronisations- und Kommunikationsaufwand erhöht, wenn neue Rechenknoten hinzugefügt werden. Es bleibt dabei, dass immer nur eine Route berechnet wird. Beim erweiterten Dijkstra steigt zwischen 16 und 32 Rechenknoten die Laufzeit wieder leicht an, was auf den erhöhten Synchronisationsaufwand zurückzuführen ist.

Vergleicht man nun diesen erweiterten Dijkstra mit einer optimierten Suche wie dem arc flag Verfahren, so ergibt sich das in Abb. 9.19 gezeigte Laufzeitverhalten. Aus diesem Verhalten lässt sich ableiten, dass sich der Einsatz des arc flag Verfahrens ab einer gewissen Größe der Fahrtenmatrix (hier: 1.000 Quelle-Ziel-Paare) lohnt. Erst hier dauert die Vorberechnungsphase kürzer als die eigentliche Routensuche. Allerdings ist der Einsatz auch stark von der Netzstruktur abhängig. Die Vorberechnungsphase dauert unter Verwendung von 32 Rechenknoten im Netz von Deutschland etwa drei Minuten, in dem der Region Stuttgart allerdings fast acht Stunden. Hinzu kommt, dass bei dieser stochastischen Umlegung in jeder Iteration der äußeren Schleife die Suche in einem neu gewichteten Verkehrsnetz durchgeführt werden muss (siehe Abb. 7.4). Hierfür muss dann auch die Vorberechnungsphase des arc flag Verfahrens durchgeführt werden.

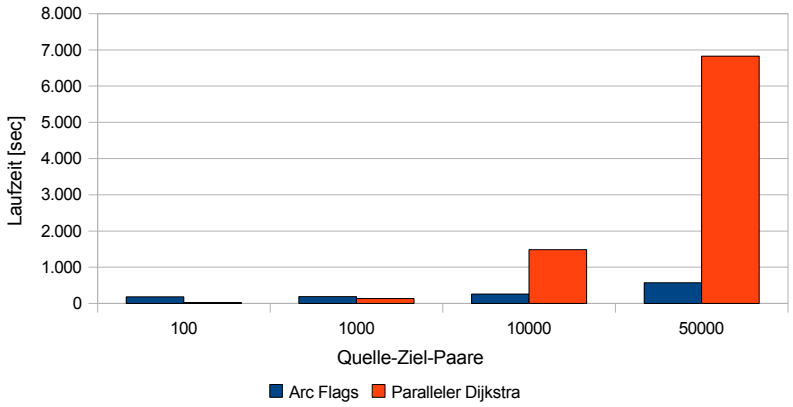


Abbildung 9.19: Vergleich der Routenberechnung zwischen dem parallelen Dijkstra und dem arc flag Verfahren unter Verwendung von 32 Rechenknoten im Verkehrsnetz Deutschlands.

Betrachtet man nun die erste Parallelisierungsstrategie für den Individualverkehr, so findet dort eine Aufteilung der Fahrtenmatrix statt. Das Verkehrsnetz liegt repliziert auf jedem Rechenknoten vor und auf jedem dieser Knoten wird anschließend eine sequentielle Kurzwegsuche durchgeführt. Deshalb wird hier auf eine Betrachtung der Laufzeit dieser Verfahren verzichtet, und es sei auf die entsprechende Literatur verwiesen [37].

Vergleicht man den Speedup beider Strategien, wie in Abb. 9.20 für das Netz von Karlsruhe und in Abb. 9.21 für das Verkehrsnetz von Deutschland dargestellt, so erkennt man, dass die erste Strategie – also das Aufteilen der Fahrtenmatrix – nahezu lineares Verhalten zeigt. Bei der zweiten Strategie, die sich die Partitionierung des Verkehrsnetzes zunutze macht, ist vor allem der Synchronisationsaufwand für die Kurzwegsuche recht teuer. Die Kosten hierfür steigen mit wachsender Anzahl von Rechenknoten, wodurch die anfänglichen Vorteile gegenüber der ersten Strategie schließlich aufgefressen werden. Dasselbe gilt bei auch für die Suche mit dem arc flag Verfahren.

Abb. 9.22 zeigt exemplarisch den Speicherverbrauch der Netze von Karlsruhe und Stuttgart pro Rechenknoten. Dabei werden die beiden Strategien (Aufteilung der Fahrtenmatrix, Partitionierung des Verkehrsnetzes) dargestellt, wobei bei der Partitionierung kein signifikanter Unterschied beim Speicherverbrauch zwischen dem parallelen Dijkstra und dem arc flag Verfahren besteht. Als Referenzwert ist zusätzlich der Speicherverbrauch im sequentiellen Fall angegeben. Zunächst sinkt der Speicherverbrauch bei der Partitionierung der Netze, da jeder Rechenknoten einen kleineren Teil des Netzes im Speicher halten muss. Allerdings wächst der Speicherverbrauch ab dem Einsatz einer gewissen Anzahl von Rechenknoten wieder an. Die zusätzlichen Datenstrukturen, die zur Verwaltung der Partitionen benötigt werden, sind dafür verantwortlich und wiegen die Vorteile der Partitionierung ab einer gewissen Größe der Partitionen auf.

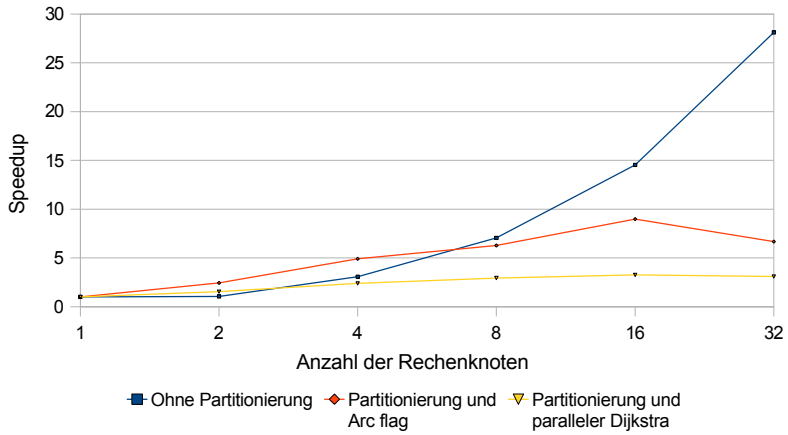


Abbildung 9.20: Speedup der makroskopischen Simulation des Individualverkehrs im Stadtnetz von Karlsruhe bei Anwendung der unterschiedlichen Strategien: Aufteilung der Fahrtenmatrix (blau), Partitionierung des Verkehrsnetzes und dem vorgestellten parallelen Dijkstra (gelb), sowie der Partitionierung in Kombination mit der arc flag Suche (orange).

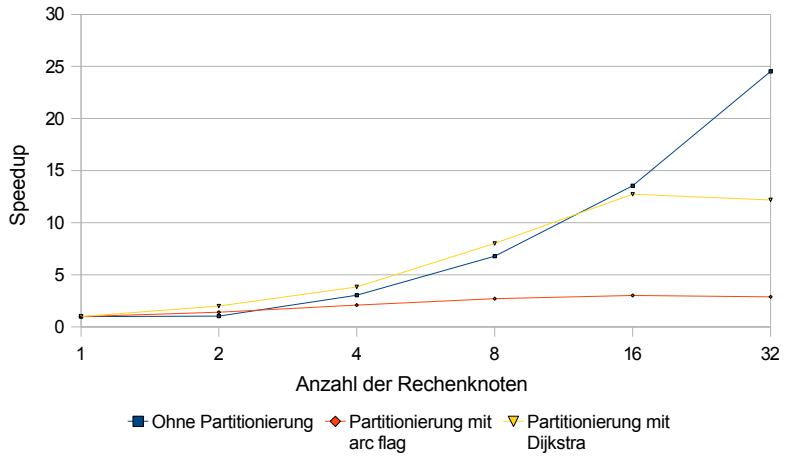


Abbildung 9.21: Speedup der makroskopischen Simulation des Individualverkehrs im Deutschlandnetz bei Anwendung der unterschiedlichen Strategien: Aufteilung der Fahrtenmatrix (blau), Partitionierung des Verkehrsnetzes und dem vorgestellten parallelen Dijkstra (gelb), sowie der Partitionierung in Kombination mit der arc flag Suche (orange).

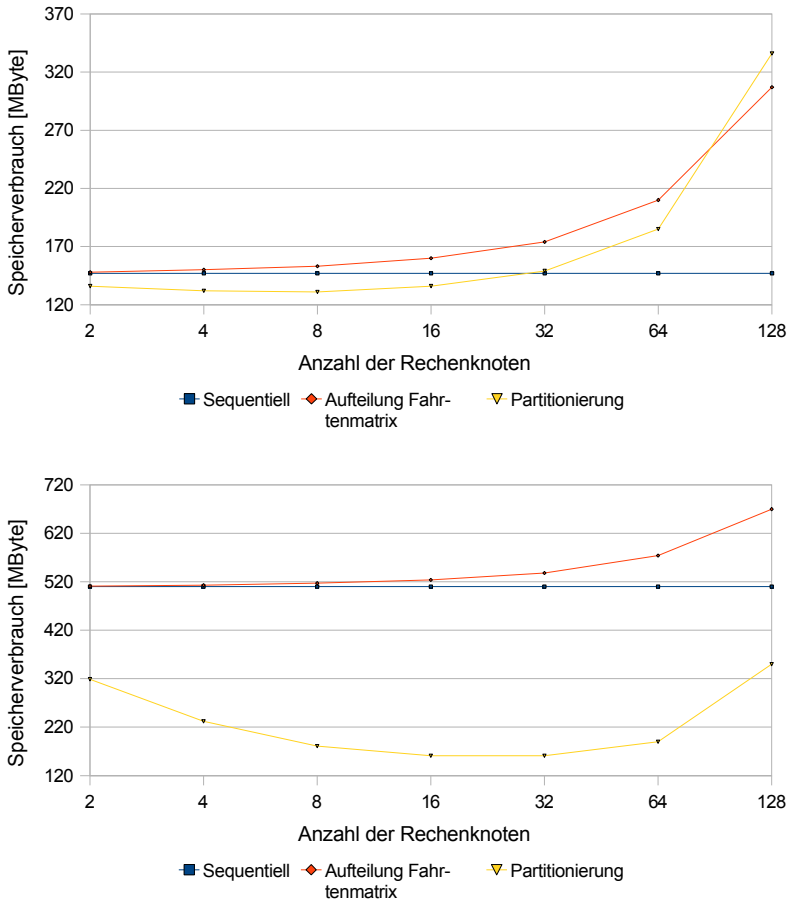


Abbildung 9.22: Spitzenspeicherverbrauch der makroskopischen Simulation des Individualverkehrs im Stadtnetz von Karlsruhe (oben) und im Verkehrsnetz der Region Stuttgart (unten) bei Anwendung der verschiedenen Parallelisierungsstrategien: Aufteilung der Fahrtenmatrix (orange), Partitionierung des Verkehrsnetzes (gelb). Als Referenzwert ist der Speicherverbrauch im sequentiellen Fall angegeben (blau).

9.5 Zusammenfassung

In diesem Kapitel wurden die im Rahmen dieser Arbeit entstandenen Komponenten unter verschiedenen Gesichtspunkten analysiert. Zunächst wurde gezeigt, dass die entwickelten Verfahren realistische Ergebnisse (hier: ein Fundamentaldiagramm, das mit gemessenen Kurven vergleichbar ist) erzeugen. Bei der Evaluierung der Kopplung zwischen makroskopischer und mikroskopischer Simulation konnte gezeigt werden, dass beide Ansätze – der OD-Ansatz und der integrierte Ansatz – gute Ergebnisse liefern, die vergleichbar mit der mikroskopischen Referenzsimulation sind. Es zeigte sich dabei, dass bei den betrachteten Szenarien der integrierte Ansatz dem OD-Ansatz stets leicht überlegen ist.

Bei der Untersuchung der Leistungsdaten des Frameworks wurde zwischen der mikroskopischen und makroskopischen Simulationswelt unterschieden. Im mikroskopischen Fall wurde exemplarisch an drei Routen demonstriert wie der im Rahmen dieser Arbeit entstanden Hierarchie-basierte Dijkstra den Suchraum deutlich reduzieren kann. Auch die Speedup-Werte und die Echtzeitverhältnisse für die verschiedenen Verkehrsnetze sind vielversprechend.

Für die makroskopische Simulation des Individualverkehrs wurden die verschiedenen Strategien (Aufteilung der Fahrtenmatrix und Partitionierung der Verkehrsnetze) untersucht. Im Falle der Verkehrsnetzpartitionierung ergab sich, dass die Anwendung einer Multilevelpartitionierung die Schnittkantenzahl im Vergleich zu einer rekursiven Koordinatenbisektion deutlich reduzieren kann. Hinsichtlich des Laufzeitverhaltens hat die Strategie der Aufteilung der Fahrtenmatrix Vorteile gegenüber der Partitionierungsstrategie. Demgegenüber kann bei letzterer der Speicherverbrauch deutlich gesenkt werden, was die Simulation größerer Szenarien ermöglicht.

10 Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde ein Framework zur Unterstützung paralleler mikroskopischer und makroskopischer Verkehrssimulationen entwickelt. Ziel war es, ausgehend von einer gleichen Datenbasis in Form von Verkehrsnetzen und Nachfragedaten – vorliegend als Quelle-Ziel-Matrizen – eine Umgebung zu entwickeln, welche die Durchführung unterschiedlichster Modelle im Sequentiellen und Parallelen ermöglicht. Dabei wurden exemplarisch drei verschiedene Modelle in das Framework integriert und auf ihre Parallelisierbarkeit hin untersucht.

Nachfolgend werden die im Rahmen dieser Arbeit erreichten Ergebnisse kurz zusammengefasst und bewertet. Anschließend wird ein Ausblick auf noch offene Fragestellungen und Erweiterungsmöglichkeiten gegeben.

10.1 Zusammenfassung und Bewertung

Die Zusammenfassung und Bewertung der erreichten Ergebnisse orientiert sich an der Gliederung der vorliegenden Arbeit und stützt sich dabei auf die drei inhaltlichen Säulen dieser Arbeit: Mikroskopische und makroskopische Simulationen sowie die Kopplung verschiedener Simulation und den daraus resultierenden hybriden Verkehrssimulationen.

10.1.1 Mikroskopische Verkehrssimulation

Kapitel 5 beschäftigte sich mit der mikroskopischen Verkehrssimulation. Es wurde eine flexible Modellerweiterung vorgestellt, die es ermöglicht, schnell und ohne größere Umstände, neue Komponenten (Fahrzeugtypen,...) zum Simulationsmodell hinzuzufügen. Hierzu wurde das ursprüngliche Nagel-Schreckenberg-Modell um Komponenten des heterogenen Verkehrs erweitert. Durch eine Verfeinerung des Zellgitters bei den zellulären Automaten, ist es möglich eine ganze Bandbreite von Fahrzeugtypen darzustellen und nach Belieben

weitere hinzuzufügen. Neuartig ist dabei auch der Ansatz, mit den gleichen Techniken, mit denen unterschiedliche Fahrzeugtypen modelliert und simuliert werden können, mit lediglich marginalen Anpassungen auch den motorisierten Öffentlichen Verkehr in Form von Linienbussen simulieren zu können.

Neben diesen Modellerweiterungen wurden Strategien zur Parallelisierung der mikroskopischen Simulation untersucht. Dazu wurden verschiedene Verfahren zur Partitionierung des Verkehrsnetzes und anschließender Simulation mit wahlweise dynamischer oder statischer Lastverteilung untersucht. Die Partitionen des Verkehrsnetzes können dabei je nach Auslastung um einzelne Sektoren wachsen oder schrumpfen. In bisherigen Arbeiten wurden hier häufig nur statische Strategien untersucht. Die vorgestellten Verfahren sind äußerst flexibel und erweiterbar. Die Modellerweiterungen um den heterogenen Verkehr besitzen allerdings noch weiteres Optimierungspotential.

10.1.2 Makroskopische Simulation des Öffentlichen Verkehrs

Kapitel 6 behandelte Umlegungsverfahren für den Öffentlichen Verkehr. Ziel war die effiziente Parallelisierung eines fahrplanfeinen Umlegungsverfahrens. Die Wahl fiel dabei auf ein solches Verfahren basierend auf einer Branch&Bound-Strategie, da dieses eines der derzeit besten, bekannten Verfahren für ein solches Vorhaben ist. Die vorgestellte Parallelisierungsstrategie ermöglicht, unter Replikation von Teilen des Verkehrsnetzes und der ermittelten Verbindungen, durch eine Aufteilung der Nachfragedaten eine sehr gute Beschleunigung der Rechenzeit. Durch die im Rahmen dieser Arbeit entstandene Strategie ist es erstmalig möglich, sehr detaillierte und sehr große Verkehrsnetze in vertretbarer Zeit mit diesem Verfahren zu berechnen. Die gewählte Strategie der dynamischen Lastverteilung mittels des Master-Slave-Konzepts erwies sich für die derzeit vorliegenden Verkehrsnetze und Nachfragedaten als ausreichend und besitzt noch weiteres Potential im Falle noch größerer Umlegungsszenarien.

10.1.3 Makroskopische Simulation des Individualverkehrs

Das anschließende Kapitel 7 hatte die Vorstellung von Parallelisierungsstrategien für Umlegungsverfahren des motorisierten Individualverkehrs zum Ziel. Nach einem kurzen Überblick über existierende Umlegungsverfahren fiel die Auswahl auf eine stochastische Umlegung für den Individualverkehr. Nach einer eingehenden Analyse des Laufzeitverhaltens des Verfahrens, konnten die rechenintensivsten Komponenten identifiziert werden. Im Rahmen dieser Arbeit wurden zwei Strategien entwickelt, die eine Parallelisierung dieses Umlegungsverfahrens zum Ziel hatten. In der ersten Strategie fiel die Wahl auf die Replikation des Verkehrsnetzes und anderer wichtiger Teilkomponenten. Die eigentliche Parallelisierung wurde durch eine Aufteilung der Nachfragedaten erreicht. Diese Teilnachfragen wurden zum Zwecke einer dynamischen Lastverteilung mittels eines Master-Slave-Konzepts verteilt und berechnet. Wie in Kapitel 9 aufgezeigt, skaliert dieses Verfahren gut. Allerdings ist es durch den zur Verfügung stehenden Speicher jedes einzelnen Rechenknotens

limitiert. Daher wurde in einer zweiten Strategie dieses Problem durch eine Partitionierung der Verkehrsnetze gelöst. Dabei wurden verschiedene Methoden der Netzpartitionierung, wie die rekursive Koordinatenbisektion und eine Multilevel-Partitionierung, analysiert und umgesetzt. Neu ist in beiden Fällen die Anwendung von parallelen Programmiertechniken zur Lösung einer stochastischen Umlegung. Mit Hilfe der im Rahmen dieser Arbeit entwickelten Konzepte ist es erstmalig möglich – wie im Fall des makroskopischen Öffentlichen Verkehrs – sehr große und detaillierte Verkehrsnetze in vertretbarer Zeit zu simulieren. Die entworfenen Strategien ermöglichen sowohl eine Berechnung auf dedizierten Hoch- und Höchstleistungsrechnern als auch im kleineren Maßstab den praktischen Einsatz in einem Verbund einiger Arbeitsplatzrechner.

10.1.4 Hybride Verkehrssimulationen

Kapitel 8 verfolgte einen etwas anderen Ansatz. Das im Rahmen dieser Arbeit entstandene Framework bietet, wie bereits besprochen, Möglichkeiten der Erweiterung und Zusammenführung verschiedener Simulationstypen. In einem ersten Schritt wurde exemplarisch eine Kopplung einer makroskopischen stochastischen Umlegung des Individualverkehrs mit der in Kapitel 5 vorgestellten mikroskopischen Simulation vorgestellt. Dabei wurden zwei neuartige Kopplungsstrategien entwickelt – der OD-Ansatz und der integrierte Ansatz. Bei ersterer handelt es sich um eine lose Kopplung, bei der die mikroskopische und die makroskopische Simulation nahezu unverändert eingesetzt werden können und die Kontrolle durch ein externe Koppelschnittstelle erfolgt. Beim integrierten Ansatz war eine Modifikation der mikroskopischen Simulation notwendig. Die Ergebnisse beider Strategien sind vielversprechend, besitzen aber gerade im Hinblick auf die Synchronisationsmechanismen noch Optimierungsbedarf.

Als zweites Beispiel diente die Kopplung einer einfachen, rudimentären Fußgängersimulation mit der vorgestellten mikroskopischen Verkehrssimulation. Dabei dienten Bushaltestellen, die zuvor in der Modellerweiterung in Kapitel 5 eingeführt wurden, als Interaktionspunkte zwischen Fußgängern und Straßenverkehr. Unter anderem wurde damit die Flexibilität der zur Verfügung gestellten Schnittstellen gezeigt, welche eine leichte Erweiterung der bestehenden Simulationen und eine Einbettung in das Framework ermöglichen. Einzelsimulationen können so – unter Berücksichtigung einer logischen Modellkompatibilität – relativ frei ausgetauscht werden. Diese Verfahren sind neuartig, besitzen aber noch Verbesserungsmöglichkeiten und dienen hier lediglich einer exemplarischen Veranschaulichung und eines Proof-of-Concepts.

10.2 Ausblick

Auch wenn im Rahmen dieser Arbeit zahlreiche Strategien und Modellerweiterungen untersucht, entwickelt und umgesetzt wurden, so existieren noch zahlreiche Möglichkeiten für

Erweiterungen und Verfeinerungen. Diese sollen in diesem Abschnitt skizziert werden und zum Teil durch aktuelle laufenden Arbeiten illustriert werden.

10.2.1 Modelle für die mikroskopische Simulation

Die in Kapitel 5 vorgestellten Modellerweiterungen und Anpassungen bieten noch viel Potential für weitere Ausgestaltungsmöglichkeiten. Einige sollen im Folgenden angesprochen werden.

Mehrstreifiger Verkehr

Das übernommene Mehrstreifigkeitsmodell, das im Rahmen dieser Arbeit an die Behandlung des heterogenen Verkehrs angepasst wurde, ist in seinem Kern ein Modell für die Simulation mehrstreifigen Verkehrs auf Autobahnen und ist für den Einsatz im Stadtverkehr nur bedingt sinnvoll. Das innerstädtische Überholverhalten unterscheidet sich zum Teil deutlich von dem ausserorts. Beispielsweise finden innerorts oft häufiger Fahrstreifenwechsel mit einer niedrigeren Hemmschwelle statt, als dies ausserorts der Fall ist. Ausgereifte Modelle sind allerdings noch immer Gegenstand intensiver wissenschaftlicher Diskussion. Das vorliegende Framework eröffnet die Möglichkeit, verschiedene Modelle für die Behandlung von Mehrstreifigkeit zu integrieren. Idealerweise wäre eine Erweiterung um ein innerstädtisches Modell sinnvoll. Bei Überholvorgängen oder Streifenwechseln eröffnete sich dann die Möglichkeit, je nach Ereignisort zwischen den beiden Modellen zu wählen und auf diese Weise ein realistischeres Verhalten zu simulieren.

Verfeinerung des Öffentlichen Verkehrs

Die vorliegende mikroskopische Simulation unterstützt ein einfaches Modell zur Simulation von motorisiertem Öffentlichem Verkehr in Form von Linienbussen. Die Haltestellen sind als Übergabepunkte zwischen einer Fußgängersimulation und der Verkehrssimulation konzipiert. Die ihnen zu Grunde liegenden Datenstrukturen sind einfache Warteschlangen. Dieses Modell bedarf noch weiterer Verfeinerungen beispielsweise in der Behandlung der Wartezeiten an den Haltestellen.

In der vorliegenden Implementierung existieren im Verkehrsnetz keine Haltebuchten für Linienbusse. Ein Linienbus blockiert unter Umständen eine komplette Fahrbahn (abhängig von der Anzahl der Fahrstreifen). Eine Erweiterung um Haltebuchten wäre für eine realistischere Behandlung erstrebenswert.

Schienenverkehr

Der real existierende Öffentliche Verkehr ist nur teilweise im Modell erfasst. Es fehlt bei der mikroskopischen Behandlung der schienengebundene Verkehr. Hierbei sind zwei Fälle zu unterscheiden. Erstens ist dies der Schienenverkehr auf Straßen (beispielsweise in Form von Tram- oder Straßenbahnen). Zweitens sind dies Züge des Regional- und Fernverkehrs, die nur äußerst selten mit dem Straßenverkehr interagieren.

Die Behandlung von Fall 1 ist zur Zeit Gegenstand einer Integration in den bestehenden Simulator. Hierzu werden entsprechende Streckentypen aus dem Verkehrsnetz ausgelesen, die dem Schienenverkehr zugeordnet sind. Die Integration in den Simulator erfolgt dabei nach dem gleichen Konzept des heterogenen Verkehrs, wie es bereits beschrieben wurde. Die Schienen werden als zellulärer Automat integriert (die Länge einer Zelle kann gegenüber der Zellenlänge im Straßenverkehr variieren). Überholvorgänge sind bei diesem Schienenverkehr verboten.

Eine Integration von Fall 2 wäre im Hinblick auf die Simulation von Park&Ride-Situationen hilfreich.

10.2.2 Verteilte Verkehrssimulationen

Im Rahmen einer Diplomarbeit wurden bereits ansatzweise die Möglichkeiten einer verteilten Verkehrssimulationsumgebung untersucht [158]. Die Grundidee hinter dieser verteilten Simulation ist die Verknüpfung verschiedener gleichartiger Verkehrssimulationen $S_0, \ldots, S_n$ zu einer großen Simulation S. Es ist dabei transparent, ob es sich bei den Einzelsimulationen um sequentielle oder parallele Simulationen handelt. Der Grundablauf einer solchen Simulation ergibt sich wie folgt. Zunächst äussert ein Rechner im Computernetz den Wunsch, eine detaillierte Simulation auf einem gewissen Gebiet durchzuführen. Es werden nun, gegebenenfalls ad-hoc, diejenigen Rechner im Computernetz bestimmt, die Teile des gewünschten Gebiets vorhalten und eine Simulation unterstützen. Diese werden dann spontan zu einer großen Simulation verknüpft. Anschließend beginnt die Simulation. Es existieren noch ein paar offene Fragen, die es zu klären gilt: Wie kann gewährleistet werden, dass sich die Teilverkehrsnetze überlappen, und wo liegen die Verknüpfungspunkte für die Netze? Wie kann eine effiziente Synchronisation der beteiligten Simulationen erfolgen? In der vorliegenden Diplomarbeit wurden zur Lösung dieser Probleme einige vereinfachende Annahmen getroffen. Es erfolgte exemplarisch die Verknüpfung von mikroskopischen Simulationen untereinander, ebenso wie von makroskopischen in Gestalt stochastischer Umlegungen.

10.2.3 Routenberechnungen

Sowohl bei der makroskopischen als auch der mikroskopischen Simulation spielt die Routenberechnung eine zentrale und rechenintensive Rolle. Im Rahmen dieser Arbeit wurden

zahlreiche Routensuchstrategien auf ihre Eignung für die gewählten Simulationstypen untersucht.

Makroskopische Simulation

Interessant wäre eine Untersuchung, inwieweit Heuristiken oder Vorwissen aus früheren Simulationen bei der stochastischen Umlegung genutzt werden könnten, um diejenigen aus den Abschnitten 7.2.4 und 7.2.5 weiter zu optimieren. Wünschenswert wäre eine Strategie, die es ermöglicht, die Quelle-Ziel-Paare der Nachfragedaten so geschickt auszuwählen und zu verteilen, dass möglichst viele Routen simultan berechnet werden können. Für eine optimale Lösung müssten die Routen und deren Abhängigkeiten schon vorab bekannt sein. Ein möglicher, einfacher Ansatz könnte sein, diejenigen Paare auszuwählen, deren Quell- und Zielknoten in weit entfernten Partitionen liegen. Auf diese Weise könnte eine Interferenz der Suchbäume beziehungsweise Partitionen reduziert werden.

Vielversprechend könnte es auch sein, eine Routensuche mit Vorberechnungsphase anzupassen. Im makroskopischen Fall hat sich diese Vorberechungsphase als Problem erwiesen, da sie in jeder äußeren Iteration von neuem durchgeführt werden muss. Es ergibt sich nun folgende Fragestellung: Ist es möglich – abhängig vom gewählten Algorithmus – die Suche so zu verändern, dass die Vorberechnungsphase nur ein einziges Mal zu Beginn berechnet werden muss und anschließend durch lokale Änderungen rascher aktualisiert werden kann?

Mikroskopische Simulation

Für die Erweiterungen der Routensuche im mikroskopischen Fall gibt es verschiedene Möglichkeiten. Im Rahmen zweier studentischer Arbeiten [128, 157] wurde untersucht, inwieweit sich eine dynamische Routenneuberechnung unter Hinzunahme von Schwarmintelligenz realisieren lässt. Bei einer solchen dynamischen Routenneuberechnung werden die einzelnen Fahrzeuge wie – der Natur nachempfunden – als Ameisen behandelt, die Pheromone ausstoßen, allerdings anders als in der Natur mit einer negativen Anziehung. Fahrzeuge hinterlassen während ihrer Fahrt durch das Straßennetz Pheromonspuren. Je höher die Pheromonkonzentration auf der Straße ist, desto dichter ist der Verkehr. Bei Überschreiten eines vom Benutzer definierten Schwellwerts, kann eine Neuberechnung der Route, das heißt die Bestimmung einer Alternativroute, angestoßen werden. Dieses Grundmodell ist in die mikroskopische Simulation integriert. Es bedarf allerdings einer Anpassung an den heterogenen Verkehr: Sind unterschiedliche Pheromontypen für die verschiedenen Fahrzeugtypen sinnvoll? Wie kann das aggressivere Überholverhalten von Zweiradfahrern mittels dieser Technik abgebildet werden?

Bei der Routensuche selbst könnte die Untersuchung einer *Sektorsuche* interessant sein. Ähnlich wie bei den in Abschnitt 2.1.6 vorgestellten optimierten Routensuchstrategien, ist es denkbar, eine Art Multilevel-Suche im partitionierten Verkehrsnetz durchzuführen. Wie gezeigt wurde, wird bei der Parallelisierung des mikroskopischen Verkehrsnetzes dieses

zunächst in Sektoren unterteilt. Basierend auf diesen Sektoren ist es denkbar, einen *Sektorgraph* zu erstellen, das heißt alle Knoten eines Sektors verschmelzen zu einem einzigen *Sektorknoten*. Die Sektorknoten werden anschließend auf geeignete Weise miteinander verknüpft. Für längere Routen könnten so recht schnell auf dem gröberen Graphen *Sektorrouten* ermittelt werden, die beschreiben, welche Sektoren von dem Fahrzeug besucht werden werden. Für die detaillierten endgültigen Routen würden dann lokale Suchen in den Partitionen genügen.

10.2.4 Ausnutzung von Shared Memory Systemen

In einigen Fällen der Parallelisierung, wie beim makroskopischen Öffentlichen Verkehr, liegt eine Replikation der Verkehrsnetze auf die einzelnen Rechenknoten vor. Handelt es sich bei den Rechenknoten zusätzlich um speichergekoppelte (shared memory) Systeme, so kann es interessant sein, eine Routensuche durchzuführen, die dies ausnutzt. Beispielsweise können auf diese Weise bei der fahrplanfeinen Umlegung mehrere Verbindungen beziehungsweise bei der stochastischen Umlegung mehrere Routen zeitgleich auf diesen Rechenknoten berechnet werden, da in beiden Fällen lange Zeit lediglich lesend auf die Verkehrsnetze zugegriffen wird.

10.2.5 Integration in ein Grid-Framework

Alle vorgestellten Simulationen eignen sich theoretisch für eine Integration in das Grid-Framework GridSFEA [121]. Dieses unterstützt Simulationen unter der Ausnutzung von *Checkpoints*, die auf einem Rechner durchgeführt wurden, zu stoppen und zu einem späteren Zeitpunkt ggf. auf einem anderen Rechner im Grid neu zu starten. In allen drei untersuchten Simulationsverfahren müsste diese Checkpoint-Funktionalität integriert werden. Hierzu wäre es notwendig, bei einem Anhalten den aktuellen Zwischenstand der Simulation abzuspeichern und diesen bei einem *Resume* wieder einzulesen und an der unterbrochenen Stelle die Simulation fortzusetzen.

Literaturverzeichnis

[1] *Verkehr in Deutschland 2006*. Statistisches Bundesamt, 2006.

[2] Mehdorn: Kunden profitieren von Bahn-Börsengang. *Verkehrsrundschau.de*, 2007.

[3] *Panorma of Transport*. Statistical Books. Eurostat, 2007.

[4] P. Adamson und E. Tick. Greedy partitioned algorithms for the shortest path problem. *International Journal of Parallel Computing*, 20:271–298, 1991.

[5] S. Akhter und J. Roberts. *Multi-Core Programmierung*. Intel Press, 2008.

[6] E. Ambt, J. de D. Ortzar und L. G. Willumsen. Metropolitan Origin-Destination Surveys: The State of the Art. In *Proceedings of the 8th World Conference on Transport Research*, 1998.

[7] G. M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, Band 30, S. 483–485, 1967.

[8] V. T. Arsan und R. Z. Koshy. Methodology for Modelling Highly Heterogeneous Traffic Flow. *Journal of Transportation Engineering*, 131(7):544–551, 2005.

[9] K. Axhausen. *Eine ereignisorientierte Simulation von Aktivitätsketten zur Parkplatzwahl*. PhD thesis, Universität Karlsruhe, 1989.

[10] D. A. Bader und J. J. Joseph. Practical Parallel Algorithms for Dynamic Data Redistribution, Median Finding, and Selection. Technical Report CS-TR-3494, College Park, 1995.

[11] M. Balmer, N. Cetin, K. Nagel und B. Raney. Towards Truly Agent-Based Traffic and Mobility Simulations. In *AAMAS '04: Proceedings of the Third International Joint*

Conference on Autonomous Agents and Multiagent Systems, S. 60–67, Washington, DC, USA, 2004. IEEE Computer Society.

[12] M. Balmer, K. Nagel und B. Raney. Large scale multi-agent simulations for transportation applications. *Intelligent Transportation Systems*, 8:1–17, 2004.

[13] M. Balmer und M. Rieser. Generating Daily Activity Chains from Origin-Destination Matrices. Arbeitsberichte Verkehrs- und Raumplanung 243, ETH Zürich, 2004.

[14] M. Bando, K. Hasebe, K. Nakanishi und A. Nakayama. Analysis of optimal velocity model with explicit delay. *Physical Review E*, 58:5429–5435, 1998.

[15] M. Bando, K. Hasebe, A. Nakayama, A. Shibata und Y. Sugiyama. Dynamical Model of Traffic Congestion and Numerical Simulation. *Physical Review E*, 51:1035–1042, 1995.

[16] J. Barcelo. The parallelization of AIMSUN2 Microscopic Traffic Simulator for ITS applications. In *Proceedings of the 3rd World Conference on Intelligent Transport Systems*, 1996.

[17] B. Barney. OpenMP. `https://computing.llnl.gov/tutorials/openMP/`, 2008.

[18] R. Bellman. On a Routing Problem. *Quarterly of Applied Mathematics*, 16:87–90, 1958.

[19] V. J. Blue und J. L. Adler. Cellular automata microsimulation of bi-directional pedestrian flows. *Journal of the Transportation Research Board*, 1678:135–141, 2000.

[20] M. Blum, R. Floyd, V. Pratt, R. Rivest und R. Tarjan. Time Bounds for Selection. *Journal of Computer and System Sciences*, 7:448–461, 1972.

[21] M. Brackstone und M. McDonald. Car-following: a historical review. *Transportation Research F: Traffic Psychology and Behaviour*, 2:181–196, 1999.

[22] D. Braess. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, 12:258–268, 1968.

[23] A. W. Brander und M. C. Sinclair. A comparative study of K-shortest path algorithms. In *In Proc. of 11th UK Performance Engineering Workshop*, 1995.

[24] W. Burghout. *Hybrid microscopic-mesoscopic traffic simulation*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2004.

[25] W. Burghout und H. N. Koutsopoulos. Hybrid Traffic Simulation Models: Vehicle Loading at Meso – Micro Boundaries. In *International Symposium of Transport Simulation, Lausanne*, 2006.

[26] C. Burstedde. Simulation von Fußgängerverhalten mittels zweidimensionaler zellulärer Automaten. Master's thesis, Institut für Theoretische Physik, Universität zu Köln, 2001.

[27] C. Burstedde, A. Kirchner, K. Klauck, A. Schadschneider und J. Zittartz. *Pedestrian and Evacuation Dynamics*, chapter Cellular Automaton Approach to Pedestrian Dynamics - Applications, S. 87. Springer, 2001.

[28] C. Burstedde, K. Klauck, A. Schadschneider und J. Zittartz. Simulation of pedestrian dynamics using a 2-dimensional cellular automaton. *Physica A*, 295:507–525, 2001.

[29] C. Cantarella und E. Cascetta. Dynamic process and equilibrium in transportation network: Towards a unifying theory. *Transportation Science A*, 25(4):305–329, 1995.

[30] J. Carrasco und J. de D. Ortzar. A review and assessment of the nested logit model. *Transport Reviews*, 21, 2001.

[31] E. Cascetta, A. Nuzzolo, F. Russo und A. Vitteta. A modified logit route choice model overcoming path overlapping problems. Specification and some calibration results for interurban networks. In *13th International symposium on transportation and traffic*, 1996.

[32] N. Cetin. *Large scale parallel graph-based simulations*. PhD thesis, Swiss Federal Institute of Technology ETH, 2005.

[33] N. Cetin, A. Burri und K. Nagel. A large-scale agent-based traffic microsimulation based on queue model. In *Swiss Transport Research Conference (STRC)*, 2003.

[34] R. E. Chandler, R. Herman und E. W. Montroll. Traffic dynamics: studies in car following. *Operations Research*, 6:165–184, 1958.

[35] D. Charypar, K. W. Axhausen und K. Nagel. An Event-Driven Parallel Queue-Based Microsimulation for Large Scale Traffic. Technical report, ETH Zürich, 2007.

[36] D. Chowdhury, L. Santen und A. Schadschneider. Statistical Physics of Vehicular Traffic and Some Related Systems. *Physics Reports*, 329:199, 2000.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest und C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd Auflage, 2001.

[38] M. Cremer. *Der Verkehrsfluss auf Schnellstraßen. Modell, Überwachung, Regelung*. Springer, 1979.

[39] M. Cremer und J. Ludwig. A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, 28(4):297–303, 1986.

[40] A. J. Daly. Improved methods for trip generation. In *Proceedings 25th European Transport Conference, London*, 1997.

[41] M. de Berg, M. van Kreveld, M. Overmars und O. Schwarzkopf. *Computational Geometry*. Springer, 2000.

[42] P. Deo und H. J. Ruskin. Simulation of Heterogeneous Motorised Traffic at a Signalised Intersection. In *ACRI 2006*, S. 522–531, 2006.

[43] R. Diekamp. *Entwicklung eines fahrzeugorientierten Verkehrsflusssimulationsprogramms*. PhD thesis, RWTH Aachen, 1995.

[44] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[45] A. Dupuis und B. Chopard. Parallel simulation of traffic in Geneva using cellular automata. *Virtual shared memory for distributed architectures*, S. 89–107, 2001.

[46] U. Elsner. Graph partitioning, a survey. Technical report, Technische Universität Chemnitz, 1997.

[47] H. Emmerich und E. Rank. An Improved Cellular Automaton Model for Traffic Flow Simulation. *Physica A*, 234:676–686, 1997.

[48] J. Esser und M. Schreckenberg. Microscopic Simulation of Urban Traffic Based on Cellular Automata. *International Journal of Modern Physics C*, 8:1025–1036, 1997.

[49] M. Fellendorf. VISSIM: Ein Instrument zur Beurteilung verkehrsabhängiger Steuerung. In *Tagungsband zum Kolloquium „Verkehrsabhängige Steuerung am Knotenpunkt“*. Forschungsgesellschaft für Straßen- und Verkehrswesen, 1994.

[50] M. Fellendorf. Static assignment versus dynamic simulation. In *Conference on Computational Engineering in System Applications*, 1998.

[51] M. Fellendorf, M. Friedrich und P. Vortisch. Kopplung makroskopischer und mikroskopischer Verkehrsmodelle - ein Verfahren für die Integration von großräumiger Planung und Detailplanung. In *Tagungsband der 18. Verkehrswissenschaftlichen Tage*, 2001.

[52] M. Fellendorf, T. Haupt, U. Heidl und W. Scherr. VISEM - An Activity Chain Based Traffic Demand Model. In *Transportation Applications Conference*, 1995.

[53] M. Fellendorf, T. Haupt, U. Heidl und W. Scherr. *Activity-Based Approaches to Travel Analysis*, chapter PTV Vision: Activity Based Demand Forecasting in Daily Practice. Elsevier Science, Oxford, 1997.

[54] R. Finkel und J. Bentley. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica*, 4(1):1–9, 1974.

[55] P.-O. Fjällström. Algorithms for graph partitioning, a survey. *Linköping Electronic Articles in Computer and Information Science*, 3(10), 1998.

[56] G. Flötteröd und K. Nagel. High Speed Combined Micro/Macro Simulation of Traffic Flow. In *Proceedings of Intelligent Transportation Systems Conference*, 2007.

[57] M. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comp.*, 21:948 pp., 1972.

[58] L. R. Ford und D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[59] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.

[60] M. L. Fredman und R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34:596–615, 1987.

[61] M. Friedrich. *Rechnergestütztes Entwurfsverfahren für den ÖPNV im ländlichen Raum*. PhD thesis, TU München, 1994.

[62] M. Friedrich, I. Hofsäß und S. Wekeck. Timetable-based transit assignment using branch & bound techniques. *Transportation Research Records*, 1752:100–107, 2001.

[63] M. Friedrich, G. Schleupen, M. Moltenbrey und H.-J. Bungartz. A parallel implementation of a schedule-based transit assignment algorithm for large networks. In *Proceedings of the 18th Symposium Simulationstechnique (ASIM 2005)*, Erlangen, Sept. 2005.

[64] M. Friedrich und S. Wekeck. A schedule-based transit assignment model addressing the passengershoice among competing connections. *Schedule-Based Dynamic Transit Modeling: Theory and Applications*, S. 159–173, 2004.

[65] M. Fukui und Y. Ishibashi. Traffic Flow in 1D Cellular Automaton Model Including Cars Moving with High Speed. *Journal of the Physical Society of Japan*, 65:1868–1870, 1996.

[66] C. Gawron. An Iterative Algorithm to Determine the Dynamic User Equilibrium in a Traffic Simulation Model. *International Journal of Modern Physics C*, 9:393–407, 1998.

[67] C. Gawron. *Simulation-Based Traffic Assignment – Computing User Equilibria in Large Street Networks*. PhD thesis, Universität zu Köln, 1999.

[68] D. Gazis, R. Hermann und R. Rothery. Nonlinear follow-the-leader-models of traffic flow. *Operations Research*, 9(4):545–567, 1961.

[69] D. C. Gazis. Car following theory of steady state traffic flow. *Operations Research*, 7:499–505, 1961.

[70] P. Gipps. A behavioural car following model for computer simulation. *Transportation Research B*, 15B:105–111, 1981.

[71] P. G. Gipps. A model for the structure of lane changing decisions. *Transportation Research B*, 20(5):403–414, 1986.

[72] A. Grama, A. Gupta, G. Karypis und V. Kumar. *Introduction to Parallel Computing*. Addison Wesley, 2. Auflage, 2003.

[73] W. Gropp und E. Lusk. The MPI communication library: its design and a portable implementation. In *Proceedings of the Scalable Parallel Libraries Conference, October 6–8, 1993, Mississippi State, Mississippi*, 160–165. IEEE Computer Society Press, 1994.

[74] W. Gropp, E. Lusk, N. Doss und A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–829, 1996.

[75] R. P. Guenthner und K. C. Sinha. Modeling bus delays due to passenger boardings and alightings. *Transportation Research Record*, 915:7–13, 1983.

[76] J. L. Gustafson. Reevaluating Amdahl's Law. *Communications of the ACM*, 31(5):532–533, 1988.

[77] R. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of Algorithm engineering and experimentation: 6th annual international workshop*, 2004.

[78] P. Hart, N. Nilsson und B. Raphel. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on System Science and Cybernetics*, 2(4):100–107, 1968.

[79] D. Helbing. Improved fluid-dynamic model for vehicular traffic. *Physical Review E*, 51:3164–3169, 1995.

[80] D. Helbing. Gas-kinetic derivation of navier-stokes-like traffic equations. *Physical Review E*, 53:2366, 1996.

[81] D. Helbing. *Verkehrsdynamik*. Springer, 1997.

[82] D. Helbing, A. Hennecke, V. Shevetsov und M. Treiber. Micro- and macro-simulation of freeway traffic. *Mathematical and computer modelling*, 35(5-6):517–547, 2002.

[83] D. Helbing und P. Molnar. Social Force Model for Pedestrian Dynamics. *Physical Review E*, 51:4282–4286, 1995.

[84] D. Helbing und M. Schreckenberg. Cellular automata simulating experimental properties of traffic flows. *Physical Review E*, 59:R2505–R2508, 1999.

[85] M. Hilliges. *Ein phänomenologisches Modell des dynamischen Verkehrsflusses in Schnellstraßennetzen*. PhD thesis, Universität Stuttgart, 1995.

[86] R. Hoar, J. Penner und C. Jacob. Evolutionary swarm traffic: If ant roads had traffic lights. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, Band 2, S. 1910–1915, 2002.

[87] S. Hoffmann und R. Lienhart. *OpenMP - Eine Einführung in die parallele Programmierung mit C/C++*. Springer, 2008.

[88] M. Hribar, V. Taylor und D. Boyce. Parallel shortest path algorithms: Identifying the factors that affect performance. Technical Report CPDC-TR-9803-015, Center for Parallel and Distributed Computing, Northwestern University, 1998.

[89] R. Jacob, M. V. Marathe und K. Nagel. A computational study of routing algorithms

for realistic transportation networks. *ACM Journal of Experimental Algorithms*, 4(6), 1999.

[90] H. F. Jordan und G. Alaghband. *Fundamentals of Parallel Processing*. Prentice Hall, 2002.

[91] A. Kappler. Entwurf Paralleler Algorithmen für die Simulation des Individualverkehrs in großen Verkehrsnetzen. Master's thesis, Technische Universität München, 2007.

[92] G. Karypis und V. Kumar. Analysis of multilevel graph partitioning. In *Supercomputing*, 1995.

[93] G. Karypis und V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. In *International Conference on Parallel Processing*, 1995.

[94] G. Karypis und V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. In *Supercomputing*, 1996.

[95] O. Kaumann, K. Froese, R. Chrobok, J. Wahle, L. Neubert und M. Schreckenberg. Online Simulation of the Freeway Network of NRW. In D. Helbing, H. Hermann, M. Schreckenberg und D. Wolf, Hrsg., *Traffic and Granular Flow '99*, 315–356, 2000.

[96] B. S. Kerner und P. Konhäuser. Cluster effect in initially homogeneous traffic. *Physical Review E*, 48(4):R2335–R2338, 1993.

[97] A. Keßel, H. Klüpfel und M. Schreckenberg. Microscopic simulation of pedestrian crowd motion. In M. Schreckenberg und S. Sharma, Hrsg., *Pedestrian and Evacuation Dynamics*, S. 193–202, Berlin, 2002. Springer.

[98] S. Khan und P. Maini. Modelling heterogeneous traffic flow. *Transportation Research Record*, 1678:234–241, 2000.

[99] H. Klüpfel. The simulation of crowd dynamics at very large events - calibration, empirical data, and validation. In P. Gattermann, N. Waldau und M. Schreckenberg, Hrsg., *Proceedings of the 3rd International Conference on Pedestrian and Evacuation Dynamics*, Berlin, 2006. Springer.

[100] H. Klüpfel. The simulation of crowds at very large events. In A. Schadschneider, R. Khne, T. Pschel, M. Schreckenberg und D. E. Wolf, Hrsg., *Traffic and Granular Flow '05*, Berlin, 2006. Springer.

[101] H. Klüpfel, T. Meyer-König, J. Wahle und M. Schreckenberg. Microscopic simulation of evacuation processes on passenger ships. In S. Bandini und T. Worsch, Hrsg., *ACRI 2000*, S. 63–71, London, 2000. Springer.

[102] S. Krauß. Towards a unified view of microscopic traffic flow theories. In M. Papageorgiou, Hrsg., *Proceedings of the 8th IFAC/IFIP/IFORS Symposium*, Band 2. Elsevier Science, 1997.

[103] S. Krauß. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. PhD thesis, Universität Köln, 1998.

[104] R. D. Kühne. Macroscopic Freeway Model for Dense Traffic – Stop-Start Waves and Incident Detection. In J. Volmueller und R. Hamerslag, Hrsg., *Proceedings of the 9th International Symposium on Transportation and Traffic Theory*, 1984.

[105] M. Lanthier, D. Nussbaum und J.-R. Sack. Parallel implementation of geometric shortest path algorithms. *Parallel Computing*, 29(10):1445–1479, 2003.

[106] K. Lee, P. Hui, D. Mao, B.-H. Wang und Q.-S. Wu. Fukui-ishibashi traffic flow models with anticipation of movement of the car ahead. *Journal of the Physical Society of Japan*, 71 (7):1651–1654, 2002.

[107] W. Leutzbach. *Introduction to the Theorie of Traffic Flow*. Springer, 1988.

[108] H. S. Levinson. Analyzing transit travel time performance. *Transportation Research Record*, 915:1–6, 1983.

[109] M. Lighthill und G. Whitham. On kinematic waves, II. A Theory of traffic flow on roads. *Proceedings of the Royal Society*, 229A:317–345, 1955.

[110] M. Luby und P. Radge. A bidirectional shortest-path algorithm with good average-case behavior. *Algorithmica*, 4(4):551–567, 1989.

[111] J. Ludmann. Konzept eines Verkehrssimulationsprogramms. Master's thesis, RWTH Aachen, 1989.

[112] S. Maerivoet und B. D. Moor. Cellular automata models of road traffic. *Physics Report*, 419:1–64, 2005.

[113] S. Marinosson, R.Chrobok, A. Pottmeier, J. Wahle und M. Schreckenberg. Simulation des Autobahnverkehrs in NRW. In D. Tavangarian und R. Grützner, Hrsg., *SCS/ASIM - 16. Symposium in Rostock, Simulationstechnik*, S. 517–523, 2002.

[114] T. V. Mathew, P. Gundaliya und S. L. Dhingra. Heterogeneous Traffic Flow Modeling and Simulation Using Cellular Automata. In *9th ASCE International conference on Applications of advanced technology in transportation (AATT06)*, 2006.

[115] Message Passing Interface Forum. MPI: A Message Passing Interface Standard, 1994.

[116] R. H. Möhring, H. Schilling, B. Schütz, D. Wagner und T. Willhalm. Partitioning Graphs to Speed Up Dijsktra's Algorithm. *ACM Journal of Experimental Algorithmics*, 11(2.8), 2006.

[117] M. Moltenbrey und H.-J. Bungartz. Design and implementation of a fine grain microscopic traffic simulator with integrated timetable-based public transportation. In *Proceedings of the 19th Symposium Simulationstechnique (ASIM 2006), Hannover*, S. 153–158, Erlangen, Sept. 2006. SCS Publishing House e.V.

[118] L. Montero, E. Codina, J. Barcelo und P. Barcelo. Combining Macroscopic and Microscopic for Transportation Planning and Design of Road Networks. In *Proceedings of the 19th ARRB Transport Research Conference*, 1998.

[119] D. J. Morgan. A microscopic simulation laboratory for advanced public transportation system evaluation. Master's thesis, Massachusetts Institute of Technology, 2002.

[120] R.-P. Mundani, H.-J. Bungartz und S. Giesecke. Integrating Evacuation Planning into an Octree-Based CSCW Framework for Structural Engineering. In *Proceedings of the 22nd Conf. on Information Technology in Construction*, S. 435–440, 2005.

[121] I. L. Muntean. *Efficient Distributed Numerical Simulation on the Grid*. PhD thesis, TU München, 2008.

[122] K. Nagel, J. Esser und M. Rickert. Large-scale traffic simulations for transportation planning. *Annual Review of Computational Physics VII*, 2000.

[123] K. Nagel und M. Rickert. Parallel implementation of the transims microsimulation. *Parallel Computing*, 27(12):1611–1639, 2001.

[124] K. Nagel und A. Schleicher. Microscopic traffic modeling on parallel high performance computers. *Parallel Computing*, 20:125–146, 1994.

[125] K. Nagel und M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Phys. I*, 2:2221–2229, December 1992.

[126] K. Nagel, D. Wolf, P. Wagner und P. Simon. Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1611–1639, 1998.

[127] S. Narasimhan. *Simulation and Optimized Scheduling of Pedestrian Traffic*. PhD thesis, Universität Stuttgart, 2007.

[128] T. Nitsche. Schwarm-basierter Verkehrsfluss. Systementwicklungsprojekt, TU München, 2007.

[129] P. S. Pacheco. A User's Guide to MPI. Department of Mathematics, University of San Francisco, 1998.

[130] M. Papageorgiou. *Applications of Automatic Control Concepts to Traffic Flow Modeling and Control*. Springer, 1983.

[131] M. Papageorgiou. Some remarks on macroscopic traffic flow modelling. *Transportation Research A*, 32, 1998.

[132] M. M. B. Pascoal. Implementations and empirical comparison of k shortest loopless path algorithms. In *9th DIMACS Implementation Challenge - Shortest Paths*, 2006.

[133] H. J. Payne. Models of freeway traffic and control. *Mathematical Model of Public Systems*, 1:51–61, 1971.

[134] L. A. Pipes. An Operational Analysis of Traffic Dynamics. *Journal of Applied Physics*, 24:274–281, 1953.

[135] PTV AG. *VISUM-Handbuch*. PTV Planung Transport Verkehr AG, 2006.

[136] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill, 2003.

[137] A. Reuschel und Z. Öster. Fahrzeugbewegungen in der Kolonne. *Österreichisches Ingenieur-Archiv*, 4:193–215, 1950.

[138] P. I. Richards. Shock waves on the highway. *Operations Research*, 4:42–51, 1956.

[139] M. Rickert. *Traffic Simulation on distributed memory computers*. PhD thesis, University of Cologne, 1998.

[140] M. Rickert und K. Nagel. Dynamic traffic assignment on parallel computers in transims. *Future Generation computer systems*, 17(5):637–648, 2001.

[141] H. Sagan. *Space-Filling Curves*. Springer, 1994.

[142] P. Sanders und D. Schultes. Highway hierachies hasten exact shortest path queries. *ESA*, S. 568–579, 2005.

[143] P. Sanders und D. Schultes. Engineering highway hierachies. *ESA*, S. 804–816, 2006.

[144] J. L. Santos. K shortest path algorithms. In *9th DIMACS Implementation Challenge - Shortest Paths*, 2006.

[145] S. Saunders. A Comparison of Data Structures for Dijkstra Single Source Shortest Path Algorithm. http://www.cosc.canterbury.ac.nz/research/reports/ HonsReps, 1999.

[146] A. Schadschneider. *Microscopic Simulation of Pedestrian Crowd Motion*, chapter Cellular Automaton Approach to Pedestrian Dynamics – Theory. Springer, 2001.

[147] W. Schnabel und D. Lohse. *Grundlagen der Straßenverkehrstechnik und der Verkehrsplanung, Band 2*. Verlag für Bauwesen, 1997.

[148] V. Schneider und R. Könnecke. Simulating Evacuation Processes with ASERI. In *Proceedings of International Conference on Pedestrian Evacuation Dynamics (PED)*, 2001.

[149] T. Schnekenburger und G. Stellner. *Dynamic Load Distribution for Parallel Applications*. Teubner, 1997.

[150] F. Schulz, D. Wagner und C. D. Zaroliagis. Using multi-level graphs for timetable information in railway systems. In *ALENEX'02: Revised Papers from the 4th International Workshop on Algorithm Engineering and Experiments*, 2002.

[151] H. Schütt. Entwicklung eines sehr schnellen, bitorientierten verkehrssimulationssystems für straßennetze. *Schriftenreihe der Arbeitsgruppe Automatisierungstechnik der Technischen Universität Hamburg-Harburg*, Heft 6, 1990.

[152] H. P. Simao und W. B. Powell. Numerical methods for simulating transient, stochastic queueing networks. *Transportation Science*, 26:296, 1992.

[153] H. D. Simon und S.-H. Teng. How good is recursive bisection. *SIAM J. Sci. Comput*, 18:1436–1445, 1997.

[154] P. Simon und K. Nagel. A simplified cellular automaton model for city traffic. *Physical Review E*, 58(2):1286–1295, 1998.

[155] P. M. Simon und K. Nagel. Simple queuing model applied to the city of Portland. *International Journal of Modern Physics*, 10:941–960, 1999.

[156] U. Sparmann. Spurwechselvorgänge auf zweispurigen bab-richtungsfahrbahnen. *Forschung Straenbau und Straenverekehrstechnik*, 263, 1978.

[157] M. Steff. Anwendungen von Swarm Intelligence in Simulation. Systementwicklungsprojekt, TU München, 2007.

[158] M. Steff. Entwurf und Implementierung einer verteilten Verkehrssimulationsumgebung. Master's thesis, TU München, 2008.

[159] J. Stokes. Into the Core: Intel's next-generation microarchitecture. *ars technica*, 2006.

[160] Y. Sugiyama und H. Yamada. Simple and exactly solvable model for queue dynamics. *Physical Review E*, 55:7749–7752, 1997.

[161] T. Ungerer. *Parallelrechner und parallele Programmierung*. Spektrum Akademischer Verlag, 1997.

[162] U. Vandebona und A. Richardson. The effects of fare-collection strategies on transit level of service. *Transportation Research Record*, 1036:79–87, 1985.

[163] Verkehrsforum.de. Verflüssigung des Verkehrs, November 2001.

[164] D. Wagner und T. Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Proceedings of 11th European Symposium on Algorithms*, 2003.

[165] B.-H. Wang, L. Wang und P. M. Hui. One-dimensional fukui-ishibashi traffic flow model. *Journal of the Physical Society of Japan*, 66(11):3683–3684, 1997.

[166] J. Wardrop. Some theoretical aspects of road traffic research. In *Proceedings of Institute of Civil Engineers, London, PART II, Vol. 1*, S. 325–378, 1952.

[167] J. Werth. Galilei-invariante Fahrzeugwechselwirkungen im Straßenverkehr. Master's thesis, Universität Duisburg, 1998.

[168] R. Wiedemann. Simulation des Straßenverkehrsflusses. *Schriftenreihe am Institut für Verkehrswesen, Universität Karlsruhe (TH)*, Heft 8, 1974.

[169] R. Wiedemann und T. Schwerdtfeger. Makroskopisches Simulationsmodell fr Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO). *Forschung Straßenbau und Straßenverekehrstechnik*, Heft 500, 1987.

[170] A. Wimmer. Entwurf und Implementierung eines parallelen mikroskopischen Verkehrssimulators für den Einsatz auf Hochleistungsrechnern. Master's thesis, Technische Universität München, 2007.

[171] Y.Sheffi. *Urban Transportation Networks*. Prentice Hall, New Jersey, 1985.

[172] F. B. Zhan und C. E. Noon. Shortest Path Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 32(1):65–73, 1998.

[173] F. B. Zhan und C. E. Noon. A comparison between label-setting and label-correcting algorithms for computing one-to-one shortest paths. *Journal of Geographic Information and Decision Analysis*, 4, 2000.