Traffic Simulation
on
Distributed Memory Computers

Inaugural-Dissertation
zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von

Marcus Rickert

aus Bergisch Gladbach

8. Dezember 1997

DRAFT

Berichterstatter:	Prof. Dr. Achim Bachem
	Prof. Dr. Rainer Schrader

Tag der letzten mündlichen Prüfung: 2. Februar 1998

Contents

Acknowledgments — Danksagung

During the work for my thesis I have obtained help from many people. At this point I would like to express my gratitude to all of them. In particular, I thank ...

Prof. Dr. Achim Bachem for supervising my thesis. He started the traffic research group at the Center for Parallel Computing in Cologne with the help of Kai Nagel. Later he co-founded the research initiative "Forschungsverbund Verkehrsimulation und Umweltwirkungen NRW" (FVU-NRW).

Prof. Dr. Rainer Schrader who is the current chairman of the Center of Parallel Computing.

Kai Nagel for the abundance of hints and advice he gave me. His initial effort lay the foundation for the approach to traffic simulation that my thesis is based on. He also reminded me repeatedly of any pending deadlines. Without him this text would not have been finished in time.

Chris Barrett for supporting my thesis as part of the TRANSIMS project at the Los Alamos National Lab.

Ganz besonders meinen Eltern, die mich während meiner ganzen Studienzeit unterstützt haben.

Meinen Kollegen Christian Gawron, Andreas Latour, Kai Nagel, Michael Schreckenberg und Peter Wagner, mit denen ich gemeinsam veröffentlich habe.

Der Deutschen Forschungsgemeinschaft (DFG). Meine Arbeiten zur Dissertation wurden finanziell vom Graduiertenkolleg "Scientific Computing" an der Universiät zu Köln im Zeitraum von Januar 1995 bis Dezember 1997 unterstützt.

Den Vorsitzenden des Graduiertenkollegs Prof. Dietrich Stauffer und später Prof. Ewald Speckenmeyer.

Den Mitgliedern der Verkehrsgruppe am Zentrum für Paralleles Rechnen (ZPR) für die gute Zusammenarbeit: Rolf Böning, Christian Gawron, Stephan Hasselberg, Stefan Janz, Stefan Krauss, Peter Oertel, Andrea Vildosola und Peter Wagner. Peter Wagner hat noch in letzter Minute ein paar Aussagen im Text "geradegerückt" und hilfreiche Tips gegeben.

Jörg Esser für seine Arbeiten im Bereich des Online-Routings, die Auslöser für Kapitel 5 waren.

Allen anderen Mitarbeitern des ZPR, die ich während meiner Studienzeit und meiner Promotionszeit kennengelernt habe.

All people who proof-read my thesis: Terence Kelly, Kai Nagel, Martin Pieck, Michael Kramer, Frank Meisgen, Jack Morrison, Harald Regnery, Thomas Renard, Paula Stretz, Peter Wagner, and last but not least Katja Wolf. As a matter of fact, she found most typos.

Michael Olesen, who provided the toolbox *graphx* that I used as graphical interface to my micro-simulator PAMINA, and Frank Meisgen for his efforts to adapt the PLB algorithm to the needs of PAMINA.

Terence Kelly, who helped me save weeks of coding by introducing me to the utility `[g]awk`. Without his help I would still be writing data evaluation programs.

All people who are (or used to be) associated with TRANSIMS: Dick Beckman, Kathy Berkbigler, Brian Bush, Stephen Eubanks, Roger Frye, Sandra Hull, Riko Jacob, Lyn Kendrick, Deborah Kubicek, Tino Lopez, Madhav Marathe, Darryl Morgeson, Martin Pieck, Steen Rasmussen, Doug Roberts, Myron Stein, Paula Stretz, and many more. Riko Jakob wrote the first version of the route-planner RP2.

Dem ZPR, dem Regionalen Rechenzentrum Köln und dem Rechenzentrum PC^2 in Paderborn für die zur Verfügung gestellten Rechner.

All members of group TSA-DO/SA for the computing time on their workstations.

Those who contributed directly: Andreas Latour for Figure 2.12, Kai Nagel for Figure 3.7, Thomas Pfenning for Figure 6.2, and Martin Pieck for Figures 4.24 and 4.25.

The North Central Texas Council of Government (NCTCOG) for the road network and origin-destination data on which most of my simulations were based.

All others whom I forgot.

Deutsche Kurzbeschreibung

In dieser Arbeit werden verschiedene Aspekte moderner Verkehrssimulation beschrieben, beginnend mit einem Modell für den Verkehr auf einem einzelnen Straßenabschnitt bis hin zum kompletten Straßennetzwerk. Der Simulator PAMINA wird benutzt um selbstkonsistente Routensätze zu berechnen, die schließlich als Basis für die Experimente mit Online-Routing dienen.

Die folgende deutsche Zusammenfassung ist im wesentlichen eine Übersetzung des Kapitels 7. Eine WWW-Version dieser Arbeit is unter der URL

http://www.zpr.uni-koeln.de/~mr/dissertation/

abrufbar. Die Seite enthält außerdem Informationen darüber, wo der Quellcode des Simulators PAMINA und eine begleitende Anleitung erhältlich sind.

Kapitel 2 — Verkehr auf Straßenabschnitten

Der Verkehr auf zwei Spuren zeigt gegenüber dem Verkehr auf einer Spur neue Aspekte. Wie in der Realität ist das Überholen ein wichtiger Bestandteil der Fahrdynamik. Kapitel 2 enthält eine Erweiterung des Regelsatzes des einspurigen zellularen Automaten (CA) mit drei Parametern, die den grundlegenden Spurwechselprozeß beschreiben. Die symmetrische Version behandelt Links-Rechts- und Rechts-Links-Wechsel gleich. Die asymmetrische Version bevorteilt die Links-Rechts-Übergänge, womit die Richtlinien auf autobahnähnlichen Straßen simuliert werden können. Die erhaltenen Fundamentaldiagramme sind den in der Realität ermittelten ähnlich. Wir können auch zeigen, daß das Phänomen der Ping-Pong-Spurwechsel erheblich durch Einführung einer zusätzlichen Spurwechselwahrscheinlichkeit reduziert werden kann.

Der Look-Back-Parameter, der bestimmt, wie weit ein Fahrer vor dem Wechsel "über seine Schulter" zurückschaut, wird als grundlegend für die Gleichmäßigkeit des Verkehrsflusses aufgezeigt. Durch die Reduzierung des Parameters von v_{max} Gitterplätzen auf Null wird die Laminarität des Flusses, besonders im asymmetrischen Fall, zerstört. Im tatsächlichen Straßenverkehr erhöhen egoistische Spurwechselmanöver die Unfallwahrscheinlichkeit beträchtlich. Deswegen kann — obwohl der CA-Regelsatz keine Unfälle erlaubt — das Ausmaß an Flußunregelmäßigkeiten als Hinweis dafür angesehen werden, wie sicher der Verkehr fließt.

Kapitel 3 — Ausführung von vorberechneten Routen

In Kapitel 3 wird das zweispurige Modell als Grundlage für den Simulator PAMINA verwendet. Durch Kombination von einfachen Netzwerkbausteinen zu Verbundelementen sind wir in der Lage, sowohl Fernstraßennetze (bzw. Autobahnnetze) als auch innerstädtische Straßennetze abzubilden. Das deutsche Autobahnnetz mit einer Gesamtlänge von 75000 [km] dient dabei als erstes Beispielnetz. Außerdem ist der Simulator in der Lage individuelle Routen auszuführen, eine Eigenschaft, die sich im folgenden als wichtig herausstellen wird. Allerdings werden bei diesen ersten Simulationen die Routen noch zufällig ausgewählt.

Einen ersten Test für PAMINA führen wir im Rahmen der TRANSIMS-Case-Study[1] durch. Sowohl das Straßennetz als auch der zugrundeliegende Routensatz sind zu diesem Zeitpunkt noch vorläufig: das Straßennetz umfaßt keine kleinen innerörtlichen Straßen und der Routensatz ist in sofern noch nicht ausgewogen, als daß Fahrer versuchen, Staus zu umgehen.

Wir untersuchen, wie die "Wiedergabetreue" der Simulation durch Aktivieren oder Deaktivieren der Geschwindigkeitsbeschränkungen und Ampeln beieinflußt wird, indem die tatsächlichen Reisezeiten mit den vom Routenplaner vorhergesagten verglichen werden. Es ergibt sich, daß die Simulation mit aktivierten

[1] *Case-Study* bezieht sich hier auf eine spezielle Fallstudie in Dallas, Texas, die aus dem Simulationsgebiet der *Study-Area* und bestimmten Szenarien besteht.

Geschwindigkeitsbeschränkungen bei gleichzeitig deaktivierten Ampeln die beste Übereinstimmung bietet. Es soll an dieser Stelle betont werden, daß es im Prinzip nicht das Ziel der Mikrosimulation ist, die Schätzungen des Routenplaners exakt zu reproduzieren. Nur die Mikrosimulation selbst ist in der Lage, den Aufbau von Staus über Straßenabschnittsgrenzen hinweg korrekt wiederzugeben. Deswegen wird es immer Diskrepanzen zwischen Routenplaner und Mikrosimulation geben, sobald abschnittsweise die Verkehrsnachfrage die Kapazität überschreitet.

Während der Simulation der Study-Area fällt besonders das Phänomen der Grid-Locks auf. Diese Zustände, bei denen regional die Fahrzeuge so dicht stehen, daß sie sich gegenseitig (permanent!) am Weiterfahren hindern, treten bei hohen Fahrzeugdichten auf. Sie werden dadurch verursacht, daß es den Fahrern — zumindest in der in diesem Kapitel verwendeten Version — nicht möglich ist, ihre Route während der Fahrt zu ändern. Die Anzahl der Grid-Locks hängt von der verwendeten Wiedergabetreue ab: ohne Geschwindigkeitsbeschränkungen und ohne Ampeln treten nie Grid-Locks auf. Dasselbe gilt für aktivierte Geschwindigkeitsbeschränkungen. Die Hinzuschaltung der Ampeln hat noch größere Auswirkungen auf den Verkehrsfluß: das System entwickelt unabhängig vom Zustand der Geschwindigkeitsbeschränkungen immer Grid-Locks. Das unterstreicht, daß in einem innerstädtischen Straßennetz der Gesamtdurchsatz hauptsächlich durch den Durchsatz an Kreuzungen bestimmt ist. Die Einführung von Geschwindigkeitsbeschränkungen hat dagegen wenig Einfluß, da bei den im Stadtverkehr vorkommenden Dichten die mittlere Geschwindigkeit der Fahrzeuge sowieso bereits mit denen der Geschwindigkeitsbeschränkungen vergleichbar ist.

Wir führen einen weiteren Parameter q_r ein, um die Länge der Rotphasen an Ampeln zu justieren. Durch Variation des Parameters zwischen 0 und 1 können wir kontinuierlich von einem Verkehrsmodell ohne Ampeln zu einem mit Ampeln übergehen. Im Bereich $q_r = 0.6 \ldots 0.65$ registrieren wir einen Übergang: unterhalb von $q_r = 0.6$ entwickelt keiner der Läufe einen Grid-Lock, oberhalb von $q_r = 0.65$ haben sämtliche Läufe Grid-Locks. Für Läufe innerhalb des Intervalls stellen wir ein stochastisches Verhalten fest.

Kapitel 4 — Iterative Routenadaption

Im letzten Kapitel entsprachen die in PAMINA verwendeten Routensätze denen, die für die TRANSIMS-Case-Study durch einen iterativen Prozeß mit dem TRANSIMS-Mikrosimulator erstellt wurden. In Kapitel 4 wählen wir einen konsistenteren Ansatz: Der Routensatz für PAMINA wird iterativ über Rückkopplung durch Reisezeiten aus PAMINA-Simulationsläufen gewonnen. Dank der hohen Ausführungsgeschwindigkeit von PAMINA und Verbesserungen im Datenaustausch zwischen Planer und Routenkonverter ist es möglich, die Laufzeit drastisch zu reduzieren. Verglichen mit TRANSIMS, das 1996 ca. 8.5 Stunden für einen Lauf benötigte, können wir mit PAMINA die Zeitspanne für eine Iteration, bestehend aus *Planer — Mikrosimulation — Routenkonverter*, auf 35-45 Minuten reduzieren. Dies gestattet uns die Durchführung von umfangreichen Experimenten zur iterativen Routenadaption. Wir definieren drei einfache Parameter, von denen vermutet wurde, daß sie den adaptiven Prozeß beeinflussen könnten: der Ausgangsroutensatz, die Auswahlkriterien für neu zu planende Routen und die Neuplanungsrate. Glücklicherweise stellt sich heraus, daß im Rahmen der hier durchgeführten Experimente keiner der Parameter das *Resultat* der Iteration entscheidend beeinflußt. Besonders die Unabhängigkeit vom Ausgangsroutenplan ist vorteilhaft, da sonst unter Umständen sehr viel Aufwand für die Erstellung eines "guten" Ausgangsplans investiert werden müßte. Es zeigt sich jedoch, daß der Zeitaufwand bis zur Relaxation durch die Wahl eines linear altersabhängigen Auswahlkriteriums[2] erheblich gegenüber einer rein zufälligen Auswahl reduziert werden kann. Die Gesamtneuplanungsrate[3] f_{acc} sollte bei mindestens 2 liegen, um ausreichend relaxierte Routensätze zu erhalten, was anhand von Messungen der Gesamtreisezeit aller Fahrzeuge ermittelt wird. Es sollte zu einem späteren Zeitpunkt zumindest ein Lauf mit $f_{acc} \gg 3$ durchgeführt werden, um zu

[2] Die Wahrscheinlichkeit für die Neuplanung einer Route hängt hierbei linear vom Alter der Route ab, d.h. von der Anzahl der Iterationen, die seit lder letzten Neuplanung vergangen sind.

[3] Die Gesamtneuplanungsrate f_{acc} ergibt sich aus der Summe der bis zur jeweiligen Iteration angefallenen einzelnen Neuplanungsraten.

überprüfen, ob die Relaxation tatsächlich zu ihrem Ende gekommen ist und auf welchen Wert sich die endgültige Gesamtreisezeit beläuft.

Zwei Artefakte treten im adaptiven Iterationsprozeß auf: (a) Die Anzahl der Routen, die in das Simulationsgebiet eingespeist werden, nimmt um ca. 10% ab, weil der Routenplaner immer mehr Routen aufgrund der stark angestiegenen Reisezeiten innerhalb der Study-Area auf außerhalb der Grenzen des Gebiets verlegte. Durch Einführung eines Ebene-0-Korrekturfaktors c für alle Reisezeiten außerhalb der Study-Area ist es möglich, diesen Effekt bis zu einem gewissen Grad umzukehren. In einem Lauf mit linearer Korrektur nimmt die Anzahl der Pläne nur um ca. 5% ab, in einem weiteren Lauf mit Korrektur $\sqrt{c}$ um ca. 7.5% Prozent. Diese Abhängigkeit läßt die Vermutung zu, daß ein Faktor der Gestalt c^q mit $q \in [0.5, 2]$ dazu benutzt werden könnte, um die Anzahl der durch die Study-Area gerouteten Fahrzeuge gezielt zu beeinflussen. Es stellt sich ebenfalls heraus, daß die Ebene-0-Korrektur nur als nachträgliche Korrektur eingesetzt werden sollte, *nachdem* der Routensatz ausreichend relaxiert ist. Dadurch kann der extreme Verlust an Fahrzeugen durch verspätete Einfahrt in die Study-Area vermieden werden, der während früher Phasen der Iteration auftritt.

Ein weiteres Artefakt besteht in der großen Anzahl von Fahrzeugen, die am Ende der Simulationszeit an den Grenzen der Study-Area in Warteschlangen aufgestaut sind. Dieser Effekt wird durch die unzureichende Rückkopplung dieser Wartezeit in die Planung verursacht. Wir verwenden eine Korrektur, bei der die durchschnittliche Wartezeit für den Eintritt in einen Straßenabschnitt zur Reisezeit auf dem Abschnitt selbst addiert wird. Die Resultate zeigen, daß nach einer Gesamtneuplanungsrate von $f_{acc} > 1.5$ keiner der untersuchten Iterationsläufe aufgestaute Fahrzeuge am Ende der Simulation aufweist.

Obwohl die Routensätze nach ausreichender Gesamtneuplanungsrate ($f_{acc} > 2.0$) als gut relaxiert angesehen werden können, zeigen die Gesamtreisezeit und die Anzahl der Fahrzeuge in der Study-Area dennoch Fluktuationen zwischen aufeinanderfolgenden Iterationen. In zukünftigen Betrachtungen wird untersucht werden, wie die Amplitude dieser Fluktuationen mit denen des vom Planer bereitgestellten Nachfrageprofils zusammenhängt, das Aufschluß darüber gibt, wie viele Fahrzeuge auf einem Straßenabschnitt anfallen würden, falls es keine Rückstaus und stochastisches Fahrerverhalten gäbe. Als Vorbereitung werden für die meisten Iterationsläufe bereits die benötigten Daten gesammelt.

Anschließend führen wir einen Vergleich von drei Simulatoren auf dem gleichen Simulationsgebiet durch. Der TRANSIMS Mikrosimulator, PAMINA und der Simulator VSS werden jeweils mit ihren eigenen selbstkonsistenten Routensätzen verwendet. Die Ergebnisse für die Gesamtreisezeit und die zeitabhängige Anzahl der Fahrzeuge in der Study-Area zeigen zwar qualitative Übereinstimmung, es gibt jedoch quantitative Unterschiede. VSS liefert Resultate unterhalb derer von PAMINA, was leicht durch die Tatsache erklärt werden kann, daß VSS das gesamte Gebiet von Dallas - Fort Worth simuliert: Staus, die außerhalb der Study-Area auftreten, führen zu einer kleineren Anzahl von Fahrzeugen innerhalb der Study-Area. Die Verläufe der TRANSIMS-Kurven sind äquivalent für kleine Gesamtneuplanungsraten ($f_{acc} \approx 1.0$). In späteren Iterationen ($f_{acc} \approx 3.0$) fallen die PAMINA Verläufe schließlich unter die von TRANSIMS ab. Daher kann angenommen werden, daß die TRANSIMS-Routensätze nicht ausreichend relaxiert sind. In zukünftigen Simulationen sollten daher immer nur entsprechend relaxierte Routensätze als Vergleichsgrundlage dienen.

Die Vergleiche beschränken sich bis zu diesem Zeitpunkt auf entweder räumlich oder räumlich und zeitlich aggregierte Werte. Um eine bessere Vorstellung von den Unterschieden der Simulatoren zu bekommen, benutzen wir im folgenden schwächer aggregierte Daten. Zuerst betrachten wir die zeitabhängigen Reisegeschwindigkeiten der Fahrten in die Study-Area für TRANSIMS und PAMINA. Wie im vorhergehenden Vergleich zeigen die Kurven einen qualitativ ähnlichen Verlauf: beginnend mit hohen Reisegeschwindigkeiten während der frühen Morgenstunden erreichen die Geschwindigkeiten ein Minimum um ca. 8:00 Uhr. Danach erhöhen sich die Geschwindigkeiten wieder. Ihre Amplitude ist jedoch für PAMINA größer als für TRANSIMS. Das ist insofern auf den ersten Blick verwunderlich, als daß TRANSIMS eine geringere Fahrzeugdichte zur Rush-Hour in der Study-Area aufweist und die Höchstgeschwindigkeiten auf die gleiche Weise berechnet werden. Deswegen sollte dieser Vergleich noch einmal wiederholt werden, und zwar mit der gleichen Verzögerungswahrscheinlichkeit $p_d = 0.2$, die bei TRANSIMS benutzt wird. PAMINA verwendete bisher $p_d = 0.3$, was die Beschleunigung etwas verkleinert. Besonders bei mittleren und hohen Dichten haben geringere p_d einen direkten Einfluß auf die Durchschnittsgeschwindigkeit.

Dieser Effekt wirft auch die grundsätzliche Frage nach der Bestimmung der Höchstgeschwindigkeit innerhalb eines CA-Modells auf. Wegen der Granularität kann die Geschwindigkeit nur in Stufen von ca. 24 [km/h] verändert werden. Leider führt die Verwendung der Verzögerungswahrscheinlichkeit p_d zur Feineinstellung der Höchstgeschwindigkeit auch zu Veränderungen der Fahrdynamik bei kleinen Geschwindigkeiten.

In einem zweiten Experiment vergleichen wir die Abbiegeraten für eine ausgesuchte Menge von Kreuzungen innerhalb der Study-Area. Die Übereinstimmung zwischen TRANSIMS und PAMINA ist zufriedenstellend. Im allgemeinen sind die Raten für TRANSIMS niedriger als für PAMINA, was auf die geringere Fahrzeugdichte in TRANSIMS zurückzuführen ist. Beide Simulatoren zeigen jedoch zum Teil erhebliche Abweichungen gegenüber gemessenen Zählraten, die von den amerikanischen Behörden (NCTCOG) zur Verfügung gestellt worden waren. Es kommen hauptsächlich zwei Ursachen in Frage: (a) Die Start-Ziel Beziehungen aus dem Jahre 1990, die als Datengrundlage für die Routensätze diente, stimmen nicht mit dem verwendeten Straßennetz aus dem Jahre 1996 überein. Eine Veränderung der Infrastruktur führt auch entsprechende Änderungen in der O-D Matrix mit sich. (b) Beide Simulationen bestrafen das Abbiegen an Kreuzungen nicht ausreichend. Eine realistischere Repräsentation der Abbiegevorgänge sollte die zu große Anzahl reduzieren können. Dies wird Gegenstand zukünftiger Untersuchungen sein. Zusammenfassend bleibt aber zu betonen, daß die in dieser Dissertation angewandten Methoden einige der ersten Resultate liefern, die realistisch genug sind, um sinnvoll mit gemessenen Daten verglichen werden zu können.

Kapitel 5 — Online-Routing

In Kapitel 5 verwendeten wir PAMINA um eine umfassende Untersuchung von Online-Routing durchzuführen. Wir nehmen an, daß die Fahrer in zwei Gruppen unterteilt werden können: *Subskribenten*, die Zugriff auf ein Fahrzeugleitsystem haben, und *Nichtsubskribenten* ohne Zugriff. Den Anteil der Subskribenten mit Zugriff bezeichnen wir im folgenden mit *Marktanteil*.

Ein Simulationslauf wird wie folgt durchgeführt: In regelmäßigen Zeitintervallen versuchen alle Subskribenten ihre verbleibende Reisezeit neu hochzurechnen. Sie dürfen jeweils die *augenblicklichen* Reisezeiten der Straßen auf ihrer Restroute verwenden, die sich innerhalb des maximalen Planungshorizonts befinden. Wenn die Hochrechnung um einen gewissen Anteil schlechter ist als die ursprüngliche Schätzung des Routenplaners, wird eine neue Route mit einem auf den Planungshorizont beschränkten Standard-Dijkstra-Algorithmus berechnet. Wenn die neue Route wiederum um einen gewissen Anteil besser als die Hochrechnung ist, wird sie vom Fahrer übernommen. Dies gilt als ein *Umroutungsvorgang*.

Das erste Experiment wird über einen Routensatz mit einer Gesamtneuplanungsrate von $f_{acc} = 1.0$ durchgeführt. Die Straßenbedingungen werden gegenüber denjenigen, auf denen der Routensatz beruht, nicht verändert. Die Resultate können wie folgt zusammengefaßt werden:

(i) Während der Rush-Hour aktualisieren ungefähr die Hälfte aller Subskribenten regelmäßig ihre Hochrechnung. Wiederum die Hälfte davon erhält eine neue Route.

(ii) Bei Ankunft am Ziel haben Subskribenten eine im Durchschnitt bis zu 28% kürzere Reisezeit als wenn sie keine neuen Routen erhalten hätten. Der Vorteil verringert sich mit zunehmenden Marktanteil: bei einem Marktanteil von 50% beträgt die Verkürzung nur noch 6%. Dieser Effekt stellt ein Dilemma für die Anbieter von Fahrzeugleitsystemen dar. Zum einen ist ein großer Marktanteil erstrebenswert, um durch Kauf von Hardware entstandene Investitionen zu finanzieren, anderseits ist es naheliegend, daß der *Anreiz*, ein Fahrzeugleitsystem zu kaufen und den Service zu subskribieren mit der zur erwartenden Reisezeitverkürzung wächst.

(iii) Die Qualität der Umroutung verbessert sich, wenn ein Fahrzeug mehr als einmal eine neue Route erhält. Sie wird gemessen, indem die Reisezeit mit Fahrzeugleitsystem verglichen wird mit der

Reisezeit der gleichen Fahrt ohne Fahrzeugleitsystem. Dies gibt einen Hinweis darauf, wie Leitsysteme implementiert werden könnten: Nach Erhalt einer neuen Route folgt der Fahrer zunächst dem neuen Vorschlag, versucht aber gleichzeitig immer wieder, die neu eingeschlagene Route durch Neuberechnung zu verbessern.

(iv) Bis zu einem Marktanteil von 40% hat das Online-Routing einen positiven Effekt auf *alle* Fahrzeuge im System. Dies kann anhand der erfolgreich ausgeführten Routen und der durchschnittlichen Reisezeit festgestellt werden. Für 40% Marktanteil beträgt die Verkürzung der Reisezeit im Durchschnitt 6%. Für höhere Marktanteile hat das Online-Routing einen insgesamt negativen Effekt: Zwar gibt es vereinzelt noch Läufe mit verkürzten Zeiten, aber insgesamt nimmt die Varianz der durchschnittlichen Reisezeit so stark zu, daß sich die Verbesserung nicht mehr auswirkt. Online-Routing bei hohen Marktanteilen macht das System demnach leicht instabil.

(v) Die Erhöhung des Zeitintervals von 120 [sec] auf 240 [sec] hat einen leicht negativen Einfluß. Intuitiv ist dieser Effekt gut zu verstehen, denn der Umroutungsprozeß basiert immer auf den *aktuellen* Reisezeiten aller Straßen und nicht auf einer Hochrechnung für den *zukünftigen* Zustand. Die Erhöhung des Intervals vergrößert demnach diese Diskrepanz.

Wir führen exakt die gleichen Läufe nochmals mit einem Routensatz durch, der eine Gesamtneuplanungsrate von $f_{acc} = 1.5$ aufweist. Wie sich schon bei der iterativen Routenadaption herausgestellte, nutzen Routensätze späterer Iterationen das Verkehrsnetzwerk besser aus. Es war daher zu vermuten, daß das Online-Routing für einen besser relaxierten Routensatz mit einer geringeren Wahrscheinlichkeit gute Alternativrouten liefert. Die Ergebnisse unterstützen diese Vermutung: (a) Der Vorteil von Subskribenten verringert sich um Faktor fünf, und (b) es gibt zwar einen positiven Effekt für alle Fahrer bei kleinen Marktanteilen, aber der negative Effekt durch die hohe Varianz der Reisezeiten bei größeren Marktanteilen bleibt bestehen.

Die grundlegende Frage ist nun: *Wie gut relaxiert ist ein realistisches Verkehrssystem?* Wenn es eher dem Fall $f_{acc} = 1.5$ ähnelt, wird sich die Implementation des hier beschriebenen Online-Routing-Verfahrens als sehr schwierig erweisen. Eine Verbesserung könnte sich durch die Extrapolation zukünftiger Reisezeiten ergeben, die zusätzlich über Standard-Verkehrssituationen abgeglichen werden, die von der Mikrosimulation hochgerechnet werden. Diese können zum Beispiel vom Wochentag oder von den Wetterbedingungen abhängen. Allerdings wirft dieser Ansatz das Problem auf, daß die Situation, die Ausgangspunkt der Hochrechung ist, sich nicht einfach als Evolution eines bestenden Routensatzes reproduzieren läßt.

In den beiden letzten Experimenten untersuchen wir realistischere Verkehrssimulationen, indem wir *nicht reproduzierbare* Stauereignisse betrachten. Wir stören den Fluß auf einem Straßenabschnitt zwischen 8:00 Uhr und 9:00 Uhr durch Herabsetzen der Geschwindigkeitsbeschränkung von 4 auf 1 [site/sec] in CA-Einheiten. In der Realität könnte dies zum Beispiel durch einen Unfall oder durch eine Baustelle verursacht werden. Der entscheidende Punkt ist, daß *keiner der Fahrer im voraus von der Störung weiß.* Deswegen gibt es auch keine Möglichkeit für den externen Routenplaner entsprechend zu reagieren. Die Simulationsergebnisse zeigen, daß bereits der verminderte Durchfluß auf einem Straßenabschnitt zu einer merklichen Verminderung des Gesamtdurchsatzes führt. Online-Routing mit kleinen Marktanteilen behebt dieses Problem, allerdings führen größere Marktanteile wieder erneut zu erhöhten Reisezeiten.

Sodann erhöhen wir die Anzahl der Störungen von eins auf zehn. Dabei zeigt das System einen beträchtlich niedrigeren Durchsatz noch bis hin zu 20% Marktanteil. Ab zwanzig Prozent verbessert sich die Situation des Systems auffallend. Und auch für größere Anteile bleibt der positive Trend bestehen: Wir erhalten zunehmend kürzere Reisezeiten bis zu Marktanteilen von 90%. In dieser Hinsicht unterscheidet sich das stark gestörte System von allen anderen bis jetzt untersuchten Szenarien. Es bleibt zu überlegen, ob das *gleichzeitige* Auftreten von zehn Störungen in einem solch kleinen Netzwerk nicht übertrieben wirkt. Aber selbst wenn die Realität zwischen einer und zehn Störungen liegt, zeigen die Ergebnisse, daß die Menge an nicht-reproduzierbaren Störungen ein wichtiger Aspekt für den Erfolg eines Online-Routing-Verfahrens im Bezug auf die Verbesserung der Gesamtsituation ist. In den hier vorgestellten Simulationen werden

Fahrer immer als "erfahren" angesehen. Etwaige Verbesserungen für Fahrer, die nicht mit dem Gebiet und dem Stauaufkommen vertraut sind, bleiben noch zu untersuchen.

Kapitel 6 — Paralleles Rechnen

Im letzten Kapitel liegt das Hauptgewicht auf den rechnerischen Aspekten der Verkehrssimulation. Durch Betrachtung von einfachen Parametern des Straßennetzwerkes, z.B. Anzahl der Kreuzungen und Straßenabschnitte, und der Hardware, z.B. Kommunikationsbandbreite und -latenz, können wir eine obere Schranke für die parallele Effizienz der Simulation herleiten. Es zeigt sich, daß bei Bus-Kommunikationsnetzen, die in vielen Shared-Memory-Systemen eingesetzt werden, hohe Effizienzen nur für kleine und mittlere Rechnergrößen bis 64 CPUs erreichbar sind. Eine weitere Verbesserung kann nur auf Rechnern mit einem zweidimensionalen Kommunikationsnetzwerk erreicht werden.

Der Anhang A enthält eine Beschreibung der *Parallel Toolbox*, die zur Parallelisierung verwendet wird. Mit ihrer Hilfe ist ein dynamisches Load-Balancing nach dem Prinzip der Domain-Decomposition-Methode mit PVM als Message-Passing Bibliothek möglich. Es ist das erste Mal, daß ein Verkehrsnetz der Größe der Bundesrepublik in Echtzeit[4] simuliert werden kann. Dieser Ansatz erlaubt sowohl die Verwendung eines Shared-Memory-Rechners, wie z.B. der SGI Power Challenger mit 16 CPU's als auch eines Workstation-Clusters von 12 SUN Sparc 5, das ungefähr die gleiche Rechenleistung liefert. Darüber hinaus erlaubt die Toolbox das Einfügen und Entfernen von Rechnern während der Laufzeit der Simulation.

Es bleibt zu erwähnen, daß trotz der zunehmenden Rechengeschwindigkeit moderner Computer Verkehrssimulationen weiterhin eine große Menge an Rechenzeit verbrauchen. Für die in den Kapiteln 4 und 5 vorgestellten Ergebnisse waren allein ca. 630 Stunden paralleler Rechenzeit auf 6 CPUs und ca. 130 Stunden sequentieller Rechenzeit auf einer CPU notwendig. Wenn der Mikrosimulator nicht parallelisiert worden wäre, hätte eine einzelne CPU mit 250 [MHz] 3900 Stunden ununterbrochen gerechnet. Das entspricht einem Zeitraum von fünfeinhalb Monaten.

[4] Echtzeit bedeutet, daß eine Sekunde der Simulationszeit in einer Sekunde simuliert werden kann.

Chapter 1

Introduction

Over the last decades, the world economy has constantly grown resulting in a continuously increasing demand for transportation. This is true for both goods and people. Mobility — as a sign of independence — is commonly regarded as one of the most important factors of the quality of life. In 1995, there were 40.4 million vehicles registered in Germany which corresponds to one vehicle for every other citizen [39]. It is not surprising that traffic has become an important factor in our daily lives.

Lately, however, we have been experiencing more and more of the downside of vehicular traffic. As street networks have not been extended at such a pace as the demand for transportation, the occurrence of traffic jams has become a major problem in densely populated areas. The resulting delay has direct and indirect consequences:

- Time spent "on the road" can generally not be used for productive work. People who have to travel professionally lose part of their income through traffic jams. Also, there is an indirect economical effect: being confined to a car in a traffic jam is a stressful situation which reduces the ability to work efficiently once the work place has been reached.
- Vehicles stalled in traffic jams use additional fuel. This is not only another cost factor to drivers, but also causes damage to the environment through pollution.
- Recurrent congestion largely restricts the independence of drivers. Instead of adjusting trips to activities, drivers have started to plan their activities carefully in order to avoid congestion.

Since the construction of new streets is either financially not affordable or politically unsupportable[1] it has become more and more important to use the existing network *more efficiently*. There are three different time-scales at which one can influence the efficiency:

a) Long term considerations (over years) refer to the construction of new infrastructure. Here, it is important to decide *where* to add the infrastructure to obtain the best overall benefit.

b) On smaller time-scales like days ore weeks, one can try to analyze the impact of temporary measures such as road constructions or road blocks. Moreover, sudden, non-recurrent peaks of the demand structure can be investigated: additional traffic caused by simultaneous cultural events (i.e. concerts or sports events).

c) The lower end of the time-scale is dominated by decisions made *just before* or *during* a trip. So-called Advanced Traffic Management/Information Systems (ATMS/ATIS) analyze the current traffic situation and give advice to drivers on how to avoid traffic jams.

[1]This is especially true for European countries.

In all cases, traffic simulation has proven to be an important tool to model and analyze the traffic state. In this thesis, we will concentrate on different aspects of traffic simulation on modern computer architectures.

In the following chapter we will describe models handling traffic on a single multi-lane street segment. The Cellular Automaton (CA) model for traffic flow on a link developed by Nagel and Schreckenberg will serve as a starting point. The CA rule-set will be extended to allow for multi-lane traffic to capture lane-changes. We will present results obtained from simulations of traffic on loops with periodic boundary conditions.

In Chapter 3 the CA multi-lane traffic segment will be used as one of the building blocks of the micro-simulator PAMINA for traffic in street networks. We will also present the structure of other components, such as highway junctions and city intersections. As a first test we will use a route-set consisting of individual routes generated for the TRANSIMS[2] case-study to drive the simulation. Traffic lights and speed-limits will serve as switches to influence the "fidelity" of the simulator. We will investigate the phenomenon of grid-locks and how they are influenced by the choice of fidelity.

PAMINA will be used in Chapter 4 to iteratively adapt route-sets from a given initial route-set based upon an origin-destination matrix. The link travel-times generated by the micro-simulation will serve as feedback to improve the dynamic performance estimate for each individual link. We will see that the speed of the relaxation process can well be monitored by inspecting the overall travel time of all vehicles in the simulation area (called study-area). We will identify parameters that help accelerating the relaxation without having an impact on the final state towards which the iteration relaxes. Two artifacts which occurred during some iterations and their remedies will be pointed out. The chapter will be concluded by a comparison of three micro-simulators processing route-sets for the same study-area.

In Chapter 5 the previously generated route-sets will be used to conduct an experiment of online routing. During a simulation run a certain fraction of all drivers will be allowed to access online information. If a driver forecasts an increased trip time due to an impending congestion, a shortest path algorithm computes a sufficiently good alternative route. The main issues of this chapter will be the questions, a) how much the quality of the re-routing process will depend on the market saturation of the online service, and (b) how much impact (both positive and negative) the system will have on drivers who do *not* have access to the re-routing service.

The computational aspects of traffic simulation will be the main issue of Chapter 6. Based upon simple parameters which can be obtained from the street network and the underlying hardware, an upper bound for the parallel efficiency will be derived. Also, this chapter we will present benchmark results of the PAMINA micro-simulator both for dynamic load-balancing and static load-balancing with external load feedback.

In Chapter 7 we will summarize the work presented in this thesis and discuss the results obtained from the simulation runs. Whenever applicable, we will give suggestions as to how the micro-simulation can be improved in future experiments.

There are two appendices. The first appendix deals with the technical implementation of the PAMINA micro-simulator. The second one contains details related to traffic simulation and the iterative process used in Chapter 4.

An online version of this thesis is available at the URL:

http://www.zpr.uni-koeln.de/~mr/dissertation/

The page also contains instructions on how to retrieve the source code of the micro-simulator PAMINA and its User's Guide.

[2]For a description of the TRANSIMS project, see Section 3.1.

Chapter 2

Traffic simulation on a link

As a preparation for the simulation experiments of subsequent chapters we would like to give a short instruction to traffic flow on a *single link*. We deliberately exclude all phenomena which are caused by complex interactions of street networks. For most considerations, one can assume that this single has periodic boundary conditions: vehicles basically move in circles without ever reaching a network element such as an intersection or a junction.

We start out by describing fundamental microscopic models with individual vehicles and macroscopic models that only consider vehicle densities. Later, we introduce the Nagel/Schreckenberg cellular automaton model of traffic flow and define a two-lane extension to include lane-changing and passing. We conclude this chapter by presenting simulation results for the two-lane model.

2.1 Existing models

2.1.1 Continuous models

One way to look at traffic is related to flow of liquids. Instead of considering individual vehicles (particles), the dynamics are defined by the vehicle density $\varrho(x,t)$ and the vehicle flow $j(x,t)$. To allow for consistent results — as in fluidynamics — both the continuity equation

$$\partial_x \varrho(x,t) + \partial_x j(x,t) = D\partial_x^2 \varrho \tag{2.1}$$

and the momentum conservation equation

$$\partial_t v + v\partial_x v = F/m + \nu\partial_x^2 v \tag{2.2}$$

have to hold. We already included diffusion terms for both. With a given relation for $j(x,t) = j(\varrho(x,t))$, Lighthill and Whitham [58] found a solution for equation 2.1 (for $D = 0$) exhibiting both laminar flow and discontinuities called shock waves. For $D > 0$ the discontinuities disappeared resulting in "smeared" shock waves.

Real-world car-following behavior can be modeled by assuming a velocity–dependent[1] gap $\Delta x(t)$

$$v(t) = C\Delta x(t). \tag{2.3}$$

Provided that drivers need a certain time τ to adopt their velocity one can expand equation 2.3 with respect time and obtain

$$a(t) = \tau^{-1}[V(\Delta x(t), ...) - v(t)] \tag{2.4}$$

[1]This results in the rule of thumb to have at least a gap of $v/2$ meters to the car ahead, if v is given as the current speed in [km/h].

as a relation between the current velocity and the acceleration $a(t)$. It can be seen as an exponentially decreasing approach to some gap-dependent function $V(\Delta x(t))$ with halftime τ. Kühne and others (for a review see [77, 78]) combined equations 2.2 and 2.4 to obtain

$$\partial_t v + v\partial_x v = \frac{1}{\tau}[V(\varrho) - v] - \frac{c_0^2}{\varrho}\partial_x \varrho + \nu \partial_x^2 v. \tag{2.5}$$

Link performance functions

Traffic flow models models are usually calibrated with real-world measurements by trying to match the so-called fundamental diagrams, defined by either $j = j(\varrho)$ or $j = j(v)$. The shape of these functions is an *emergent property* of the underlying model.

If the fundamental diagrams cannot be sufficiently adjusted by parameters, it is questionable if basic properties of traffic flow are conserved. Therefore, a straight-forward alternative approach is to *prescribe* the relationships. To do so, one has to depart from the notion of local densities and velocities, though. Links are either treated as one unit or at least as a coarse grid of minimum length. In the first case, for example, a link performance function [111] returns the travel-time t with respect to the current flow j on link, its capacity c and a free-flow travel-time t_0:

$$t(j) = t_0 \left[1 + J\frac{j}{c-a}\right]. \tag{2.6}$$

The parameter J can be used for calibration. It is obvious, that in equation 2.6 any information about the distribution — let alone the interaction — of vehicles is lost. It does, however, reflect the basic relationship between travel-time and flow: the expected time increases sharply as the flow approaches capacity. Link performance functions have been used in network simulation models because of their high computational speed.

The *cell transmission model* [22] splits a link into cells with a fixed length l. For a given update interval t and a maximum velocity v_{max}, the cell length is usually selected as $l = v_{max}/t$, so that the fastest vehicle can only transverse a maximum of one cell per update. Additional link parameters are the current number of vehicles on the link $N(t)$ and the maximum inflow of vehicles into the link $Q(t)$. As with the link performance function, the cell transmission model completely neglects vehicle interactions. It does, however, give a coarse vehicle distribution along the link.

Queueing models

It is possible to neglect the interactions of vehicles on a link while maintaining individual information of vehicles. This is achieved by handling vehicles in *queues*. Upon entry into a link, link performance functions are used to determine the estimated travel-time through a link according to the vehicle's individual properties. The resulting scheduled arrival time at the end of the link is used to insert the vehicle into a time-sorted exit queue. In every simulation time-step, the queue is checked for vehicles that are scheduled to exit from the link. Provided there is enough room in the destination segments, the vehicles will be removed from the queue. See [112] for a review and [40] for a practical implementation of this approach.

2.1.2 Discrete models

In recent times, simulations of traffic flow based on cellular automata (abbreviated as "CA", see [119] for a review) have gained considerable importance. Originating from the fundamental work by Wolfram [129], CA's had already been used in many fields of physics as a means for fast computation. As a rule, a problem given for continuous time and space is transferred into a time-discrete counterpart defined on a spatial grid. A grid site is either occupied by a particle or it is empty. Any interactions between particles are modeled by next-neighbor relationships between grid sites.

Applying this approach to traffic flow, Nagel and Schreckenberg [84] extended the range of rules from nearest neighbors to a range of 5 grid sites and introduced six discrete velocities $0 \ldots 5$. The simulation results exhibited a striking resemblance to realistic traffic behavior. For $v_{max} = 1$ Schadschneider and Schreckenberg found an analytic solution [105]. For higher v_{max}, these analytic approaches lead to good approximations for the average behavior [106]. Further analytic results can be found in [15]. Nagel [77, 78] pointed out the strong connections between particle hopping models and fluid-dynamical approaches for traffic flow.

Within the TRANSIMS project, Barrett et al. have considered using a finer grid than 7.5 [m] for their simulations [6]. Reducing the grid-size to 3.75 [m] allows for a higher maximum vehicle occupation on city streets and the implementation of more realistic vehicle lengths.

Krauss [53] started out with the original CA single-lane rules in mind and transferred them into a space-continuous[2] version. The approach allows higher maximum flows and independent handling of the stochasticity in free flow and congested flow regimes, both of which simplifies the validation of the model parameters to real-world measurements. In another modification [54] the rate at which vehicles could brake was reduced to more realistic[3] values. Simulations yielded a hysteresis for the flow-density dependency that had previously been observed in real-world traffic: during the recovery from a jam, the flow on a link never reaches the previous maximum again *unless* the density drops considerably below the critical density at least once. This effect is also know as *capacity gap*.

Single lane model

For the convenience of the reader we would like to outline the single lane model[4]. The system consists of a one dimensional grid of L sites with periodic boundary conditions. A site can either be empty, or occupied by a vehicle of velocity zero to v_{max}. The velocity is equivalent to the number of sites that a vehicle advances in one update — provided that there are no obstacles ahead. Vehicles move only in one direction. The index i denotes the number of a vehicle, $x(i)$ its position, $v(i)$ its current velocity, $v_d(i)$ its maximum speed, $pred(i)$ the number of the preceding[5] vehicle, $gap(i) := x(pred(i)) - x(i) - 1$ the width of the gap to the predecessor. Note that in the original model all vehicles had the same maximum velocity v_{max}. We now allow for different desired velocities $v_d(i)$ to include an inhomogeneous fleet. At the beginning of each time-step the rules are applied to all vehicles simultaneously (parallel update, in contrast to sequential updates which yield considerably different results). Then the vehicles are advanced according to their new velocities.

- **IF** $v(i) < v_d(i)$ **THEN** $v(i) := v(i) + 1$ **(S1)**
- **IF** $v(i) > gap(i)$ **THEN** $v(i) := gap(i)$ **(S2)**
- **IF** $v(i) > 0$ **AND** $rand < p_d(i)$ **THEN** $v(i) := v(i) - 1$ **(S3)**

S1 represents a constant acceleration until the vehicle has reached its maximum velocity v_d. **S2** ensures that vehicles having predecessors in their way slow down in order not to run into them. In **S3** a random generator is used to decelerate a vehicle with a certain probability modelling erratic driver behavior. The free–flow average velocity is $v_{max} - p_d$ (for $p_d \neq 1$).

[2] The location and the velocity of a vehicle are continuous, but the update interval is still discrete

[3] In the Nagel/Schreckenberg model vehicles traveling at 120 [km/h] can come to a full stop within one second!

[4] Wimmershoff provides an online demo for the single-lane CA model [128]

[5] *A precedes B* in this context means that A is followed by B

Multi-lane models

Traditional car-following theory (see [42]) by and large never dealt with multi-lane traffic. Modern microscopic traffic simulation models (e.g. [10, 16, 37, 64]) obviously handle multi-lane traffic by necessity. Cremer and coworkers [20, 107] even treat multi-lane traffic in the context of cellular automata models. Yet, all these papers approach the problem by using heuristic rules of human behavior, without checking which of these rules exactly cause which kind of behavior. In validations then (e.g. [64]), it often enough turns out that certain features of the model are not realistic; and because of the heuristic approach it is difficult to decide which rules have to be changed or added in order to correct the problem.

For that reason, a more systematic approach is justified. Our approach here is to search for a *minimal* set of rules which reproduces certain macroscopic facts. The advantage is that relations between rules and macroscopic behavior can be more easily identified; and as a welcome side-effect one also obtains higher computational speed.

We choose CA models as starting point for this investigation because their highly discrete nature reduces the number of free parameters even further. It is clear that a similar analysis could be applied to continuous microscopic models, hopefully benefiting from the results obtained here.

Nagatani examined a two-lane system with completely deterministic rules and $v_{max} = 1$ [70, 73], where cars either move forward or change lanes. A very unrealistic feature of this model are states in which blocks of several cars oscillate between lanes without moving forward at all. This was corrected by introducing randomness into the lane changing [71]. Latour has developed the two-lane model which served as the basis for the one discussed here [55]. In [99] a more elaborate rule set was used for two-lane traffic which reproduced the phenomenon of increased flow with an imposed speed limit. Most queueing models are not truly multi-lane, but emulate multiple lanes by switching the order of vehicles on one lane whenever a passing would have occurred in reality [3]. Gawron [40] uses the number of lanes to scale the capacity of link queues. The model, however, does not include any explicit interactions between lanes.

2.2 A generic two-lane model

The single lane model is not capable of modelling realistic traffic mainly for one reason: a realistic fleet is usually composed of vehicles types having different desired velocities. Introducing such different vehicle types in the single lane model only results in *platooning* with slow vehicles being followed by faster ones and the average velocity reduced to the free–flow velocity of the slowest vehicle [9, 72].

We introduce a two-lane model [99, 102] consisting of two parallel single lane models with periodic boundary conditions and four additional rules defining the exchange of vehicles between the lanes. The update step is split into two sub-steps:

1. Check the exchange of vehicles between the two lanes according to the new rule set. Vehicles are only moved *sideways*. They do not *advance*. Note that in reality this sub-step regarded by itself is infeasible since vehicles are usually incapable of purely transversal motion. Only together with the second sub-step our update rules make physically sense.

   This first sub-step is implemented as strict parallel update with each vehicle making its decision based upon the configuration at the beginning of the time-step.

2. Perform independent single lane updates on both lanes according to the single lane update rules. In this second sub-step the resulting configuration of the first sub-step is used.

A somewhat generic starting point for modeling passing rules is the following: (T1) The driver looks ahead if somebody is in his way. (T2) The driver looks on the other lane if the situation is better there. (T3) The driver looks back on the other lane if somebody would be obstructed by the lane change.

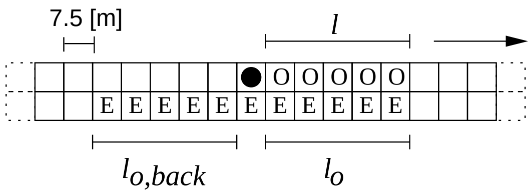

Figure 2.1: *Two-lane CA model: geometry describing lane-changing rules* — The vehicle (denoted by a filled circle) will change to the right (lower) lane if any of the sites marked with "O" is occupied and the sites marked with "E" are empty. The vision ranges are l for look-ahead in the current lane, l_o look-ahead in the other lane, and $l_{o,back}$ for look-back in the other lane.

Technically, we keep using $gap(i)$ for the number of empty sites ahead in the same lane, and we add the definitions of $gap_o(i)$ for the forward gap on the other lane, and $gap_{o,back}$ for the backward gap on the other lane. Note that if there is a vehicle on a neighboring site both return -1. The generic multi-lane model then reads as follows. /epsfysize=4cm A vehicle i changes to the other lane if all of the following conditions are fulfilled (see Figure 2.1):

- $gap(i) < l$ (**T1**),
- $gap_o(i) > l_o$ (**T2**),
- $gap_{o,back}(i) > l_{o,back}$ (**T3**), and
- $rand() < p_{change}$ (**T4**).

l, l_o, and $l_{o,back}$ are the parameters which decide how far a driver looks ahead in his own lane, ahead in the other lane, or back in the other lane, respectively.

The most important parameters of the two-lane model are *symmetry*, *stochasticity*, and *direction of causality*. In Table 2.1 we associate the parameters of our rule set with the previously mentioned characteristics.

characteristic	yes	no
symmetry	**T1** for L$\rightarrow$R	no **T1** for L$\rightarrow$R
stochasticity	$prob_c < 1$	$prob_c = 1$
backward causality	$l_{o,back} > 0$	$l_{o,back} = 0$

Table 2.1: *Characteristics of the two-lane CA rules*

Symmetry: The rule set defining the lane changing of vehicles can be both symmetric and asymmetric. The symmetric model is interesting for theoretical considerations whereas the the asymmetric model is more realistic.

Stochasticity: The single lane model proved that a strictly deterministic model is not realistic: the model did not show the desired spontaneous formation of jams. In the case of the two-lane model the lack of stochasticity in combination with the parallel update results in strange behavior of slow platoons occupying either lane: since none of the vehicles has reached its maximum velocity and all evaluate the other lane to be better there is collective change sidewise which is usually reversed over and over again until the platoon dissolves or the platoon is passed by other vehicles.

We introduce stochasticity into the two-lane rule set to reduce the effective number of lane changes and thus dissolve those platoons. The simulation revealed that this effect is equally important in the asymmetric free–flow case (see 2.2.2).

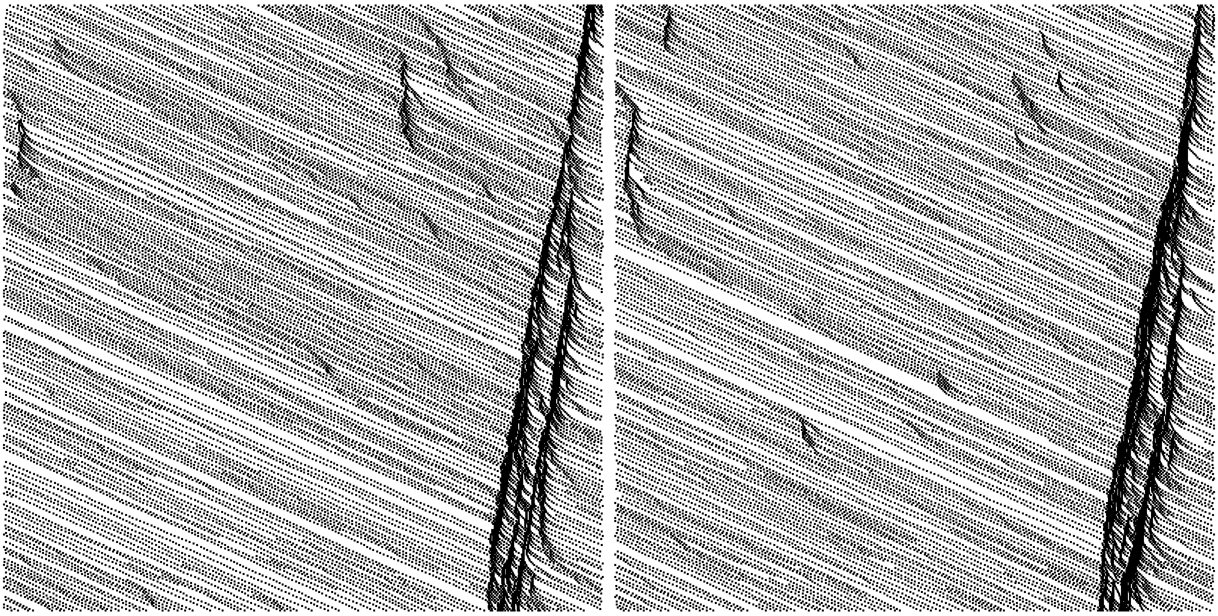

Figure 2.2: *Space-time plot for two-lane CA-model: symmetric,* $l_{o,back} = 5$ — Space is plotted left to right. Time is plotted top to bottom. The symmetric case exhibits laminar flow interspersed with traffic jams. The flow on both lanes (left and right halves) looks equal.

Direction of Causality: In the single lane model a vehicle only looks ahead (= downstream = in the direction of vehicle flow) so that causality can only travel upstream (= in the direction opposite of vehicle flow). A reasonable lane changing rule must include a check of sites *upstream* in order not to disturb the traffic of the destination lane. This would result in causality traveling downstream.

2.2.1 A lane changing model

As an example, we start with $l = v + 1$, $l_o = l$, $l_{o,back} = v_{max} = 5$, and $p_{change} = 1$. That means that both l and l_o are roughly proportional to the velocity, whereas looking back is not: It depends mostly on the expected velocity of other cars, not on one's own.

In the symmetric version of this model, cars remain on their lane as long as they do not "see" anybody else. If they see somebody ahead in their own lane (i.e. $gap < v+1$), they check the other lane if they can switch lanes and do so if possible. Afterwards, if they are satisfied, they remain on this lane until they become dissatisfied again.

In the asymmetric version, cars always try to return to the right lane, independent of their situation on the left lane.

In the next section we will present some results obtained from the described rule set.

2.2.2 Some simulation results

Space-time-plots both of the symmetric and the asymmetric version are shown in Figures 2.2 and 2.3. For these plots, we simulated a system with a length of 12,000 sites of which we plot 400 sites in 400 consecutive time-steps. The density is 0.09 which is slightly above the density of maximum flow (see Figure 2.4). Vehicles go from left to right (spatial axis) and from top to bottom (time axis). Traffic jams appear as solid areas with steep positive inclination whereas free flow areas are light and have a more

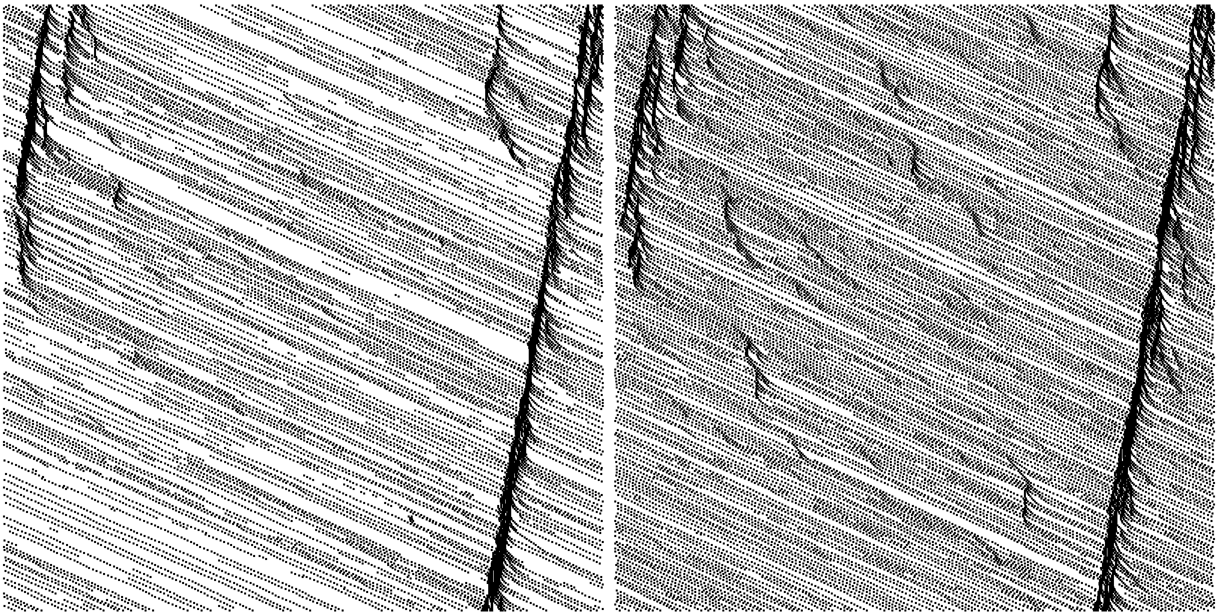

Figure 2.3: *Space-time plot for two-lane CA-model: asymmetric, $l_{o,back} = 5$* — In the asymmetric case the left and right lanes show different densities. Vehicles are more likely to be found on the right lane. Short trajectories on the left correspond to passing events.

shallow negative inclination. Each plot is split into two parts: the left part containing the left lane and the right part containing the right lane, respectively.

Note that Plot 2.3 (left lane) gives a good impression of the great number of lane changes through the high frequency of short vehicle life lines appearing and disappearing: These are vehicles that temporarily leave the right lane to avoid an obstacle. They go back to their old lane as soon as the obstacle has been passed. It will be confirmed quantitatively that indeed the rate of lane changes is much higher for the asymmetric model than for the symmetric model.

Before going on, we would like to describe our standard simulation set-up for the following observations. Note that quantitative simulation results were obtained with a much larger system than the qualitative space-time plots. We simulated a system of length

$$L \;=\; 133,333 \; sites \;\approx\; 1000 \; km$$

with closed boundary conditions: Traffic was running in a loop. We started with random initial conditions, i.e. N cars were randomly distributed on both lanes around the complete loop with initial velocity $v_0 = 0$. Since the system is closed, the average density per lane is now fixed at

$$\langle \rho \rangle_L = \frac{N}{2\,L} \;,$$

where the "2" stands for the number of lanes.

The simulation was then started, 1000 time-steps were executed to let the transients die out, and then the data collection was started. The flow which is found in the fundamental diagrams is both space and time[6] averaged as follows

$$\langle j \rangle_{L,T} = \frac{1}{T}\,\frac{1}{L}\,\sum_{i}^{L}\,\sum_{t}^{T/5}\, v(i, 5t) \;.$$

[6] We gather statistics every fifth time-step only, since subsequent time-steps are correlated.

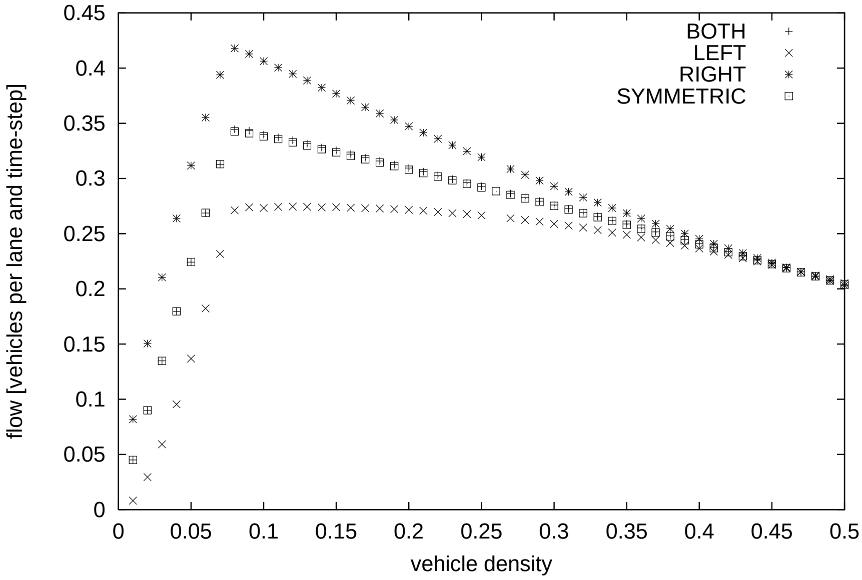

Figure 2.4: *Two-lane model flow for* $l_{o,back} = 5$, $p_{change} = 1.0$ — While in the asymmetric model the flow on the right lane is considerably higher than in the left lane, the *average* flow is exactly as high as in the symmetric case.

Values for lane change frequency and ping-pong lane change frequency are obtained by the same averaging procedure except that statistics are gathered every time-step, since *by definition* ping-pong lane changes occur in subsequent time-steps.

We usually used $T = 5000$, and the same procedure was repeated for each density found in the plots. With a resolution of $\Delta\varrho = 0.01$ an average plot took about 22 hours of computation time on a Sparc 10 workstation.

Flow behavior

By comparing these models with each other and with earlier results, we make the following observations (Figure 2.4 unless otherwise noted):

(i) Both for the symmetric and the asymmetric version, maximum flow is higher than twice the maximum flow of the single lane model (Figure 2.5). This is to say that, in spite of the additional disturbances which the lane changing behavior introduces into the traffic flow, the general effects are beneficial, probably by diminishing large deviations from "good" flow patterns.

(ii) Both for the symmetric and the asymmetric version, the combined two-lane flow reaches a maximum at $\rho_{jmax} \approx 0.08$, which is at or near a sharp bend of the flow curve.

(iii) For the asymmetric model, flow on the left lane keeps increasing slightly for $\rho > \rho_{jmax}$, but this is over-compensated by the decreasing flow on the right lane.

(ii) and (iii) together lead one to the speculation that maximum flow in the asymmetric case here actually is connected to a "critical" flow on the right lane and a "sub-critical" flow on the left lane. Any addition of density beyond here leads to occasional break-downs on the right lane and thus to a much lower flow there. Obviously, such interpretations would have to be clarified by further investigations, and the word "critical" would have to be used with more care, such as is pointed out in [80] for the single lane case.

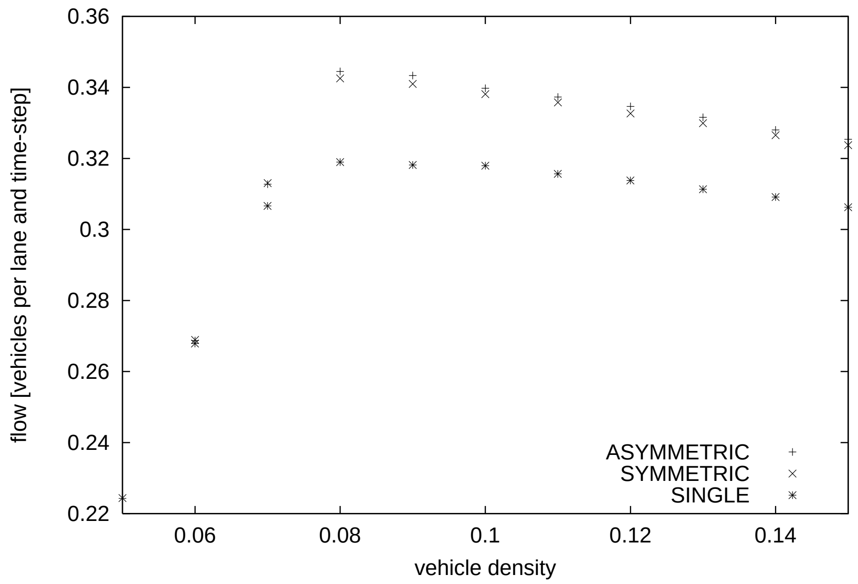

Figure 2.5: *Comparison of flows between single-lane and two-lane models* — The two-lane model has a higher maximum flow than the single-lane model both in the symmetric and the asymmetric cases. The other lane can be used to locally resolve minor congestion through lane changing.

(iv) For both lanes combined, the curves for symmetric and asymmetric traffic actually look basically similar. If the above interpretation is right, this means that the overall density of maximum flow is a fairly robust quantity, but one can stabilize one lane at a much higher density if this density is taken from the other lane.

(v) At very low densities in the asymmetric case, flow on the left lane, j_{left}, only slowly builds up. This is to be expected, since at least two cars have to be close to each other to force one of them on the left lane, leading to a mean field solution of $j_{left}(\rho) \approx \rho^2$ for $\rho \approx 0$.

(vi) For $\rho > 0.4$, flows on both lanes in the asymmetric models are very similar and similar to the lane flows in the symmetric models.

Lane changing behavior

To obtain some further insight into the lane changing dynamics, Figure 2.6 shows the frequency of lane changing both for the asymmetric and the symmetric model.

(i) Note that in the asymmetric case there is a sharp bend in the curve, which is not found for the symmetric case. This bend is also near ρ_{jmax}, giving further indication that the dynamics above and below ρ_{jmax} are different.

(ii) For the symmetric case, lane changing occurs with less than half the frequency compared to the asymmetric case.

(iii) In the symmetric case, the lane changing frequency per site for small densities increases approximately quadratically up to rather high densities, whereas the same quantity for the asymmetric model grows approximately linearly already for fairly low densities. This suggests that for the symmetric case a mean field description of interaction, $P(change) \propto \rho^2$, would be valid up to comparably high densities. For the asymmetric case, it is fairly obvious that this does not work. Since

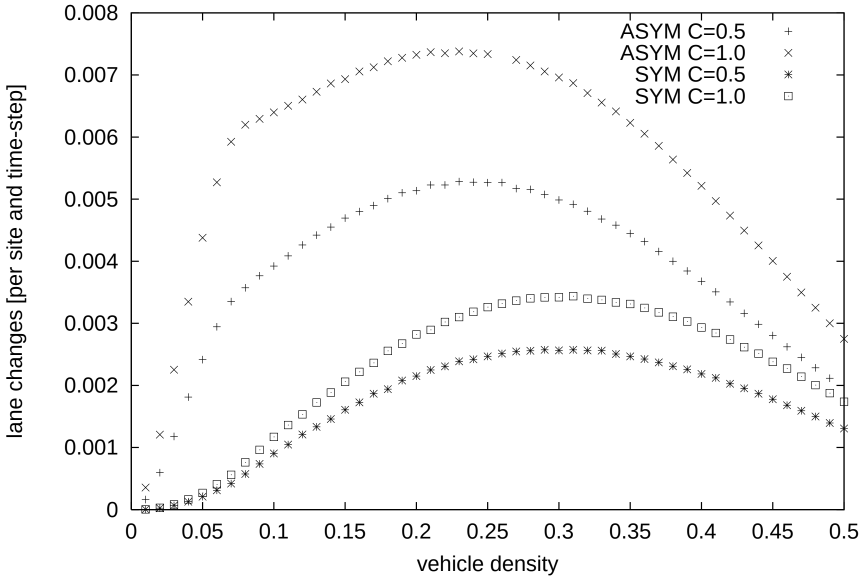

Figure 2.6: *Simple lane changes for* $l_{o,back} = 5$ — Introducing an additional probability of $C = 0.5$ for lane-changing does not reduce the frequency of lane-changes proportionally (with respect to the original case with $C = 1.0$).

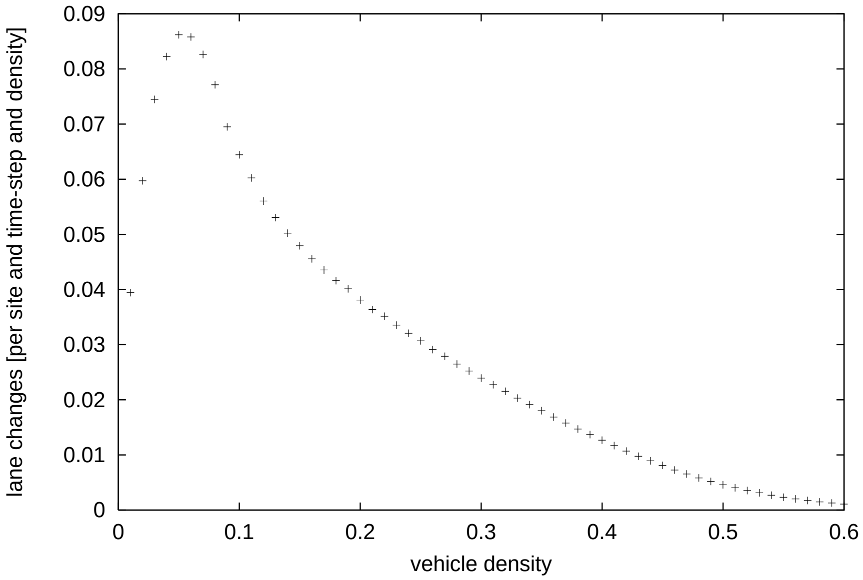

Figure 2.7: *Lane changes per normalized by density for* $l_{o,back} = 5$, $p_{change} = 1.0$ — The frequency of lane changes per vehicle density shows a significant peak below the critical density of the two-lane model.

the vehicles have a strong tendency to be on the right lane, already a density of 0.04 per lane would be a density of 0.08 if everybody were on the right lane. Yet, $\rho = 0.08$ is known to be already a density of high interaction in single lane traffic. Since this high interaction tends to spread out vehicles [105, 106], each additional vehicle simply adds its own share of lane changes, making the relation roughly linear.

(iv) The maximum number of lane changes occurs at densities much higher than ρ_{jmax}. The lane

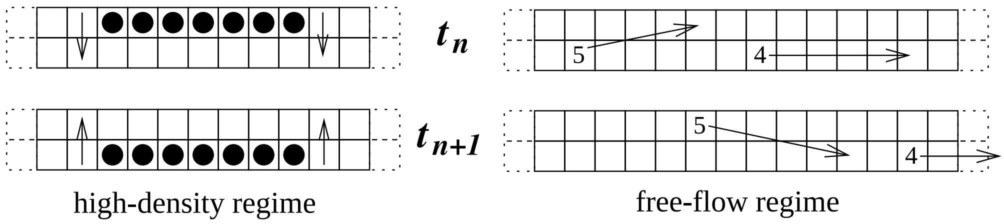

Figure 2.8: *Ping-Pong behavior in two-lane CA-model* — LEFT: In high density regimes vehicle collectively change lanes without advancing (except for the first vehicle of the platoon). RIGHT: In low density regimes the asymmetric model exhibits a high frequency of unsuccessful passing attempts. This is due to the stochastic change of velocity between v_{max} and $v_{max} - 1$.

changing probability *per vehicle*, however, reaches a maximum below the critical density (Figure 2.7).

Ping-pong lane changes

An artifact of the previously described algorithm is easily recognizable when one starts with all cars on the same lane, say the right one. Assuming a fairly high density, all cars see somebody in front of them, but nobody on the left lane. In consequence, everybody decides to change to the left lane, so that *all* cars end up on the left lane (see left-hand side of Figure 2.8). Here, they now all decide to change to the right lane again, etc., such that these coordinated lane changes go on for a long time ("cooperative ping-pong effect"). This effect has already been observed by Nagatani for the much simpler two-lane model [69, 70].

One way around this is to randomize the lane changing decision [71]. The decision rules remain the same as above, but even if rules T1 to T3 lead to a yes, it is only accepted with probability p_{change}. With this fourth lane changing rule, patterns like the one above are quickly destroyed.

In order to quantify the effects of a different p_{change}, simulations with $p_{change} = 0.5$ were run. The observations can be summarized as follows:

(i) The flow-density curves are only marginally changed (Figure 2.9).

(ii) The frequency of lane changes is decreased in general, but, except for $\rho < \rho_{jmax}$ in the asymmetric case, by much less than the factor of two which one would naively expect (Figure 2.6). This implies, that there is usually a *dynamic* reason for the lane change: if it does not occur in one time-step due to $p_{change} < 1$, then it is re-tried in the following time-step, etc.

(iii) For a better quantification in how far a $p_{change} < 1$ actually changes the pattern of vehicles changing lanes back and forth in consecutive time-steps, we also determined the frequency of "ping pong lane changes", where a car makes two-lane changes in two consecutive iterations. Obviously, there are left-right-left (lrl) and right-left-right (rlr) ping-pong lane changes.

Figure 2.10 shows that reducing the probability to change lanes, p_{change}, from 1 to 1/2 has indeed a beneficial effect: The number of ping-pong lane changes decreases by about a factor of five. The TRANSIMS micro-simulator [86] takes advantage of this result by rejecting lane-changes with a small probability.

Yet, for the symmetric case, the frequency of ping-pong lane changes is more than an order of magnitude lower in both cases anyway. This indicates that in simulations starting from random initial conditions, the cooperative effect as described further above as cooperative ping-pong effect does not really play a role for the statistical frequency, because this effect should be the same for the symmetric and the asymmetric model. Instead, the cause of the ping-pong lane changes in the asymmetric model is as follows (refer to the right-hand side of Figure 2.8): Assume just two cars on the road, with a gap of 5 between them. With respect to velocity, both cars are in the free driving regime, and their velocities will fluctuate between 4

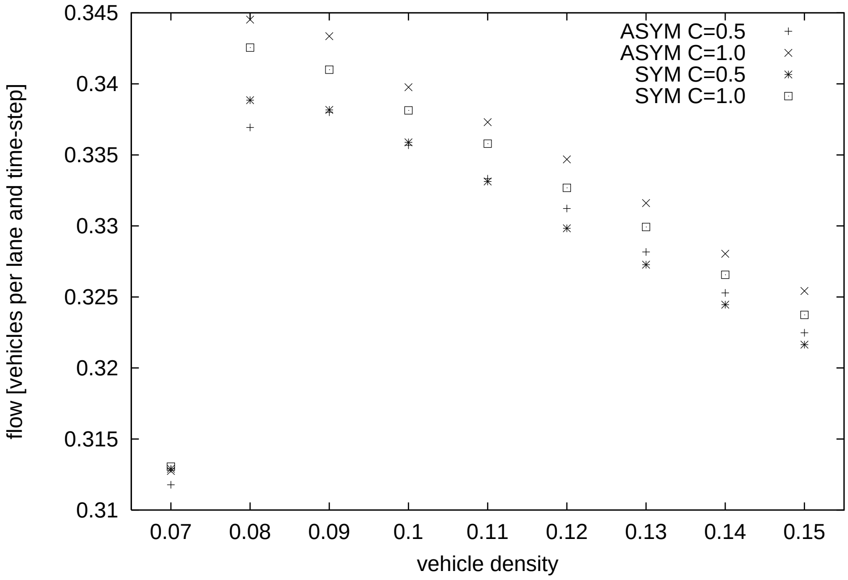

Figure 2.9: *Flow in two-lane model for* $l_{o,back} = 5$ *and different* p_{change} — Reducing the lane changing probability p_{change} also reduces the maximum flow slightly for both the symmetric and asymmetric cases. The change, however, is rather small. Note the magnified scale of the y-axis.

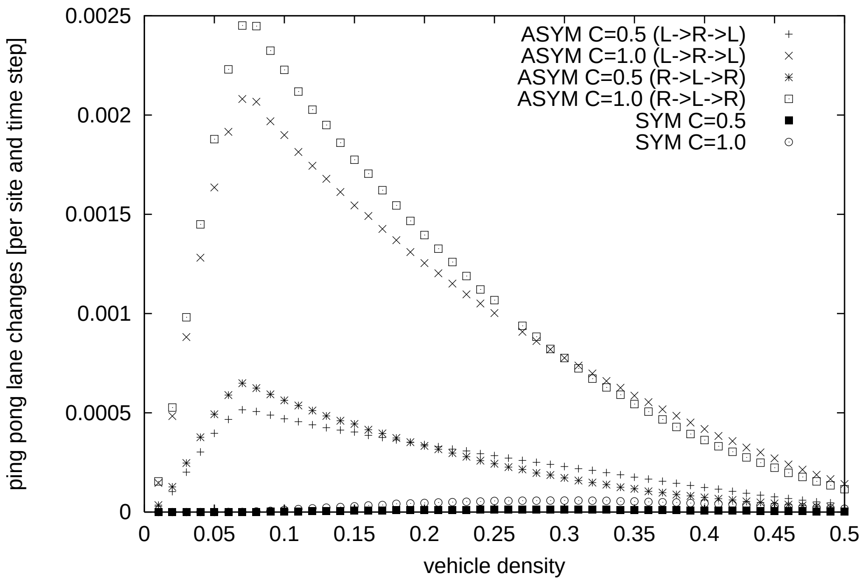

Figure 2.10: *Ping-pong lane changes for* $l_{o,back} = 5$ *and different* p_{change} — The impact of $p_{change} < 1.0$ on ping-pong lane changes is much more drastic than on the overall frequency.

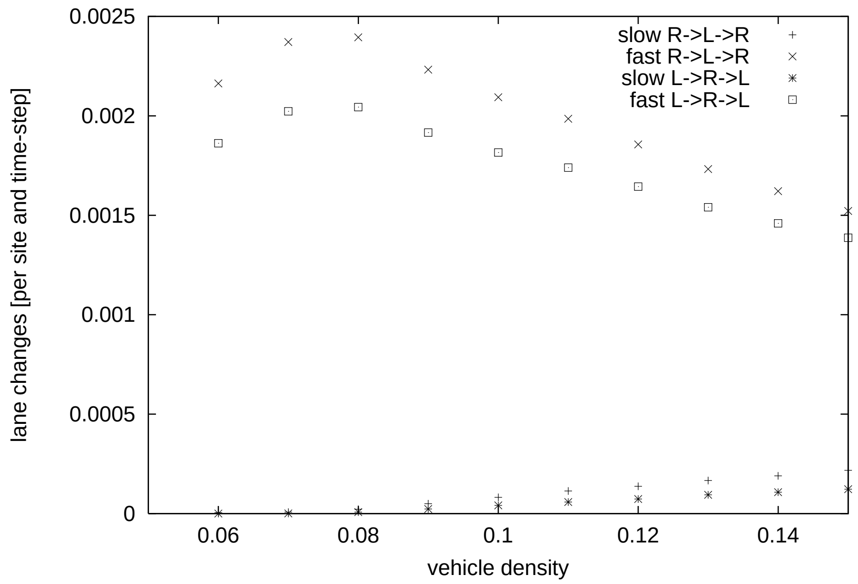

Figure 2.11: *Slow and fast ping-pong lane changes* — Most ping-pong lane changes of the asymmetric model are caused by the "tailgating effect" of fast vehicles with $v \geq 4$.

and 5. Now assume that the following car has velocity 5 from the last movement. That means that it looks 6 sites ahead, sees the other car, and changes to the left lane. Then, assume that in the velocity update, the leading car obtains velocity 5 and the following car obtains velocity 4. Then, after the movement step, there is now a gap of 6 between both cars, and in the lane changing step, the follower changes back to the right lane. And this can happen over and over again in the asymmetric model, but will not happen in the symmetric model: Once the following car in the above situation has changed to the left lane, it will remain there until it runs into another car on the left lane.

To investigate this second kind of ping-pong lane changes we ran simulations recording whether a ping-pong lane change was made at low velocities $0 \leq v \leq 3$ or high velocities $4 \leq v \leq 5$. Figure 2.11 shows a very distinct peak for *fast* ping–pong–changes at low densities whereas *slow* ping–pong–changes have a lower peak at higher densities similar to that of the symmetric case.

This gives a strong indication that most lane changes are actually caused by the "tailgating effect" as described above, which is an artifact of the rules. It is, though, to be expected that this behavior does not have a strong influence on the overall dynamics: It mostly happens in the free driving regime; as soon as, for example, another car is nearby in the left lane, it is suppressed by looking back and forward on the other lane.

2.2.3 Other parameter combinations

We would like to mention two other parameter combinations. They are presented because they generate artifacts which contradict the common sense one would apply to the phenomena of traffic flow.

(i) Latour [55] reduced the lookahead to $l = v$ instead of $l = v + 1$. While this change is negligible for vehicles at higher velocities it becomes crucial to vehicles stopped in a jam: assuming the current velocity to be zero the vehicle looks *zero* sites ahead and decides to remain in the current lane due to the non–fulfilled rule $T1$. This state will persist until the predecessor moves even if the other lane is *completely* free! Figure 2.12 shows the impact of the reduced look ahead on overall flow: for density $\rho > 0.75$ there is no perceptible flow in the right lane which corresponds to a traffic jam that occupies more or less the whole right lane.

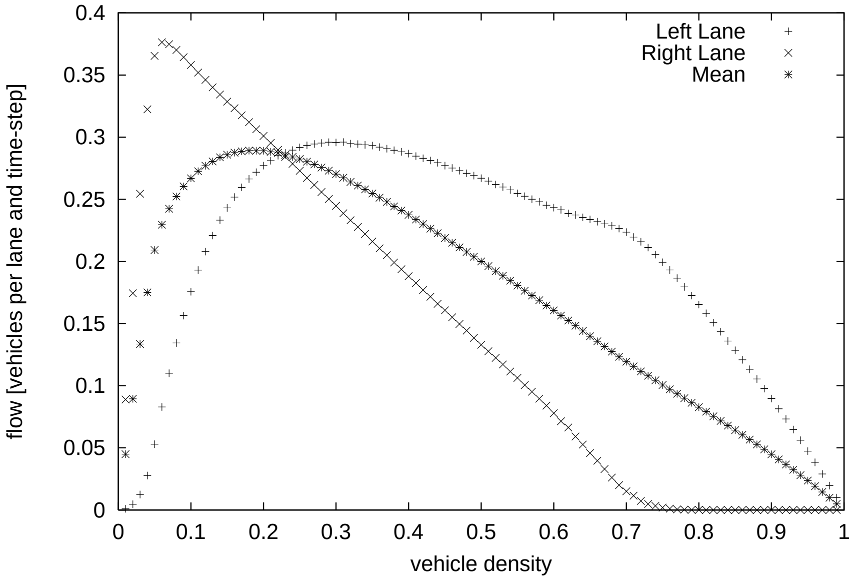

Figure 2.12: *Flow in Latour's model 1 with $l = v$* — Reducing the look-ahead to v causes artifacts for stalled vehicles. They will not change to the left lane because of rule $T1$, even if the left lane is vacant. Above density 0.75 there is no measurable flow on the right lane. This figure was taken from [55].

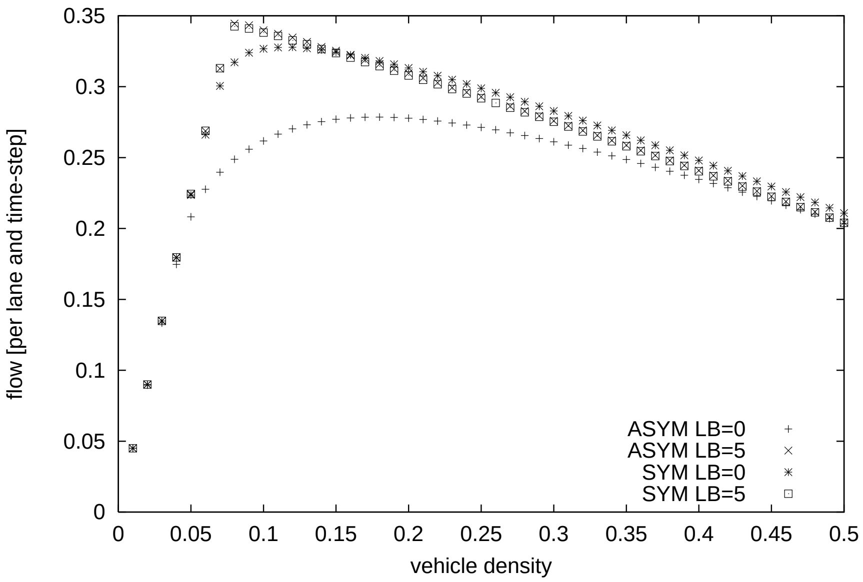

Figure 2.13: *Flow in two-lane model for $p_{change} = 1.0$ and different $l_{o,back}$* — Reducing the look-back range from 5 to 0 considerably reduces the maximum flow, especially in the asymmetric case.

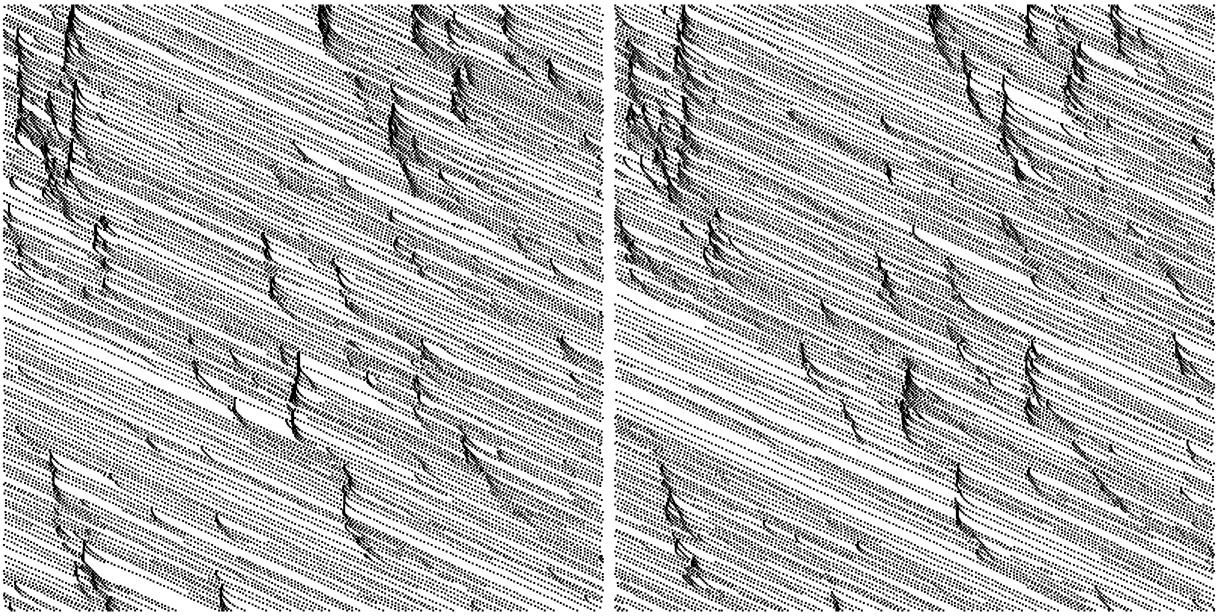

Figure 2.14: *Space-time plot for two-lane CA-model: symmetric,* $l_{o,back} = 0$ — Reducing the look-back increases the probability for small traffic jams. Free-flow regimes are rather short.

(ii) In the second case we reduced the look-back to $l_{o,back} = 0$. Vehicles no longer check whether their lane changing could have a disadvantageous effect on the other lane which corresponds to a very egoistic driver behavior. In real-world traffic, insufficient look-back is also an important cause for accidents. Figure 2.13 shows flow density relationships for look-back $l_{o,back} = 5$ and $l_{o,back} = 0$. It is obvious that the decrease in look-back also decreases the maximum flow at critical densities. Moreover $l_{o,back} = 0$ splits the curves of the symmetric and asymmetric cases which used to be almost identical for $l_{o,back} = 5$: the lack of look-back is much more disadvantageous for asymmetric than for symmetric rules.

In Figures 2.14 and 2.15 we used $l_{o,back} = 0$ for the symmetric and asymmetric rule sets with one plot per lane. It is clearly visible (compare to Figures 2.2 and 2.3) how $l_{o,back} = 0$ completely disrupts the laminar flow regime. Vehicles change lanes without looking back; and due to the formulation of the model this behavior does not cause accidents, but it causes the obstructed vehicles to make sudden stops. Since these stops are caused more or less randomly, the regime becomes much more randomly disturbed than before, somewhat reminiscent of the Asymmetric Stochastic Exclusion Process (see [77, 106]).

As seen before the effect is even more drastic for the asymmetric rule set since the number of lane changes is higher than in the symmetric case. In Figure 2.15 with $l_{o,back} = 0$ dynamics are dominated by small traffic jams caused by lane changes, while in Figure 2.2 there are still some fairly laminar areas.

2.2.4 Comparison to reality

Compared to reality (e.g. [64]), the lane change frequency in the asymmetric models presented here is by about a factor of 10 too high. Using $p_{change} = 0.1$ would correct this number, but it is not a good fix dynamically: It would mean that a driver follows a slower car in the average for 10 seconds before he decides to change lanes. Besides, it was shown that about 90% of the lane changes in the asymmetric models here are produced by an artificial "tailgating dance", where a follower changes lanes back and forth when following another car. It remains an open question in how far artifacts like this can be corrected by the current modeling approach or if it will be necessary to, e.g., introduce memory: If one remembers

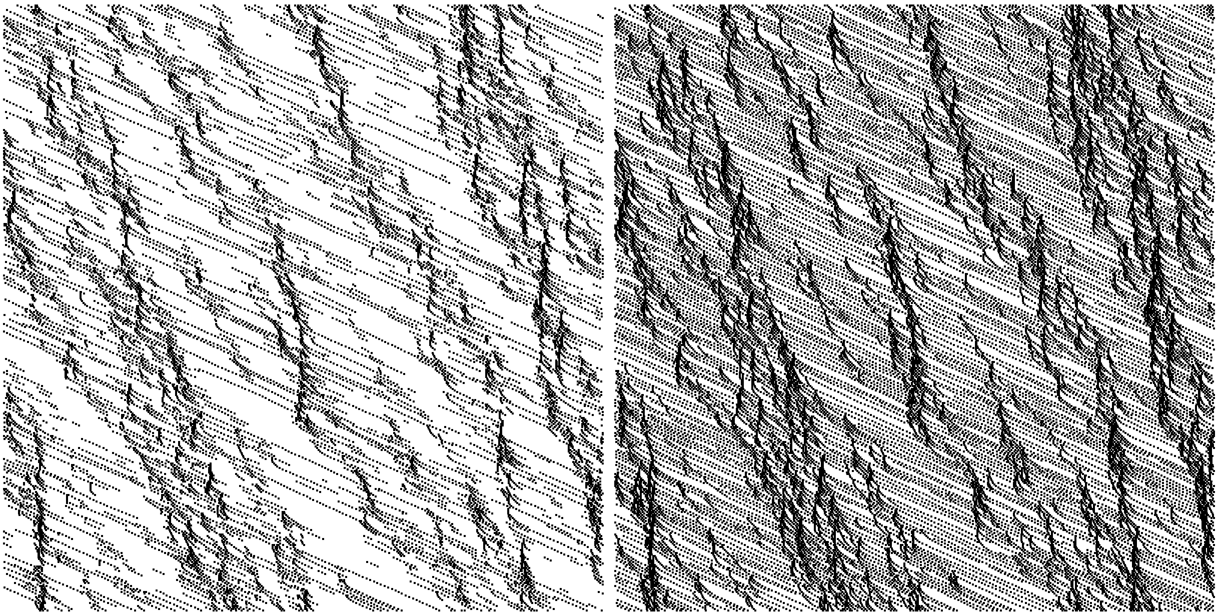

Figure 2.15: *Space-time plot for two-lane CA-model: asymmetric, $l_{o,back} = 0$, left + right lanes —* In the asymmetric case, the laminar flow is completely destroyed. The system is dominated by disrupting lane-changes.

to just have changed lane from right to left, one will probably stay on that lane for some time before changing back.

Another defect of the models presented in this paper is that the maximum flow regime is most probably represented incorrectly. Both measurements (e.g. [64] or Figure 3.6 in [62]) or everyday observation show that real traffic shows a "density inversion" long before maximum flow, that is, more cars drive on the left lanes than on the right lanes. This effect is more pronounced for countries with higher speed limits in combination with a passing prohibition resulting in a separation of faster passenger vehicles and slower trucks. Let us denote by ρ_{jmax}^{1lane} the density of maximum flow of the *single lane* case. It follows for the real world two-lane case that at a certain point the left lane will have a density higher than this density ρ_{jmax}^{1lane} whereas the right lane has a density lower than ρ_{jmax}^{1lane}. When further increasing the overall density, then the flow on the left lane will decrease whereas it still increases on the right lane. It is unclear if the net flow here increases or decreases; but it should become clear that instabilities here are caused by the left lane first. This is in contrast to the models presented here, for which the right lane reaches the critical density first.

2.3 New developments

Wagner, Nagel, and Wolff [126] modified the two-lane models of the CA model slightly to generate the density inversion described above. Simulations yield a cross-over between left and right lanes at density of approximately $\varrho_{cross} \approx 0.06$ which is well below the density $\varrho_c \approx 0.15$ at which the maximum overall flow occurs. These results are similar to empirical data provided by the German Ministry of Traffic [117]. The model even reproduces the related density-dependent lane usage on three lanes [57].

Chapter 3

Executing Pre-computed Routes

Over the last years, there has been increased interest in microscopic traffic simulators worldwide. In contrast to earlier implementations which could only handle small street networks in reasonable time, current state-of-the-art implementations exploit the architecture of modern computer systems to increase their performance considerably. Also, there has been a shift away from macroscopic underlying traffic models to simple microscopic ones such as the cellular automaton approach.

In this chapter we start out by giving a short overview of existing micro-simulators that are able to execute route-sets within regional street networks. The remainder of the chapter will be dedicated to the description of the micro-simulator PAMINA (Parallel Microscopic Network Algorithm), which will serve as the core tool for all computations presented in this work.

3.1 Simulation models

Currently, there are several commercial and non-commercial traffic simulation packages available. Some [92, 93] are based on macroscopic traffic models, which neglect individual characteristics of vehicles including route-plans. Although their computational performance is usually higher than microscopic models, they lack the ability to run activity-based simulations based upon route-plans. In the following, we will outline those micro-simulators capable of executing individual route-plans. For a more comprehensive survey of micro-simulators see [123].

NETSIM [98] was originally developed for the Federal Highway Administration. Later, it was integrated with TRAF simulations system resulting in the new common name TRAF-NETSIM. It puts special emphasis on handling stochasticity of driver decisions correctly, since random driver behavior at the microscopic level is known to have a considerable impact on aggregated measurements. In one test-bed [60] the whole core street network of Austin (TX) was simulated on a CRAY vector computer.

INTEGRATION [1, 32] is a microscopic traffic-simulator developed at the Queen's University in Kingston, Canada. Its dynamics are based upon a car-following model with a macroscopic calibration for the desired free speed, the speed at capacity, and the jam density of each link. The network capabilities of the simulator include among other things lane-changing, incidents, freeway intersections, turning movement restrictions, traffic signals, loop detectors, and vehicle probes.

DYNASMART (=Dynamic Network Assignment Simulation M for Advanced Road Telematics , see [18, 45]) is a micro-simulator driven by individual route-plans. It provides the capability to explicitly model trip maker en-route decisions in response to online information.

DYNEMO [108] represents a special case within this list of traffic-simulators. In principle, it is a *macroscopic* model, since it uses segmented links with link performance functions. The necessary input for these

feature	PAMINA I	PAMINA II	PAMINA III
individual speed limit	no	no	yes
transfer rates	yes	no	no
route-plans	no	random	yes
traffic signals	no	no	yes
message passing library	PVM	PVM	PVM/MPI
parallelization	direct	Toolbox 1.0	Toolbox 2.0
off-line load balancing	yes	yes	yes
off-line feedback	no	no	yes
online load balancing	no	yes	prepared

Table 3.1: *Overview over PAMINA versions*

functions such as density and mean velocity, however, is retrieved by aggregating over *individual* vehicles. Therefore, DYNEMO is capable of processing route-plans as every other model presented here.

TRANSIMS [86, 121] is a traffic research project funded by the American Federal Highway Administration [88]. It comprises modules to (a) generate a synthetic population from census data, (b) generate activities from the synthetic population, and (c) generate a route-set from the activities. The simulator itself, which is also based upon the Nagel-Schreckenberg model, was parallelized using workstation clusters with a distributed memory programming model. A more detailed description can be found in Section 4.4.

The traffic research effort FVU-NRW [87] funded by the German federal state Nordrhein-Westfalen uses a modular traffic simulator called PLANSIM-T [41]. Its structure corresponds to that of TRANSIMS. Since privacy laws in Germany largely restrict access to census data for research purposes, route-sets are more likely to be obtained from origin-destinations matrices. The matrix is computed from homogeneous groups of population distributed over the simulation area. Actual traffic counts are used to calibrate the OD-flows, which currently still poses a problem [125]. The micro-simulator features three built-in underlying traffic models: (a) the original Nagel-Schreckenberg model [94], (b) a refined continuous model described in [53], and (c) a low resolution queuing model, which replaces incoming lanes by queues. PLANSIM-T uses the thread programming paradigm on shared memory multi-processor computers.

The city traffic simulator CASim [17, 31] developed at the university of Duisburg, Germany (also within the framework of the FVU-NRW) uses the original Nagel-Schreckenberg CA. The simulation can be driven by both turn counts at intersections and individual route-plans. Furthermore, it can be influenced by online traffic count data which is used to perform dynamic re-routing.

The PARAMICS [90, 16] simulator developed at the Edinburgh Parallel Computing Center uses (in its original version) a Connection Machine CM-200. New releases have been ported to message-passing systems like the CRAY T3D/E.

3.2 Overview of PAMINA

Using the CA model developed by Nagel and Schreckenberg as a starting point, we extended the model to include traffic in street networks. The goal was to maintain the simplicity of the CA model as far as possible by using a few building blocks based upon the original model. We have implemented three versions (see Table 3.1) of the micro-simulator PAMINA which we would like to describe in this chapter.

The first version PAMINA I [99] already used the multi-lane extension of the Nagel-Schreckenberg CA to simulate traffic on links. Vehicles are generated using sources with time-dependent insertion rates. At each of the traffic nodes (e.g. junction or ramp) vehicles are transferred to their new respective destination links according to time-dependent transfer rates. In this respect it is similar to CASim. There is, however, no calibration of the transfer rates as in the traffic simulation of the city traffic of Duisburg. Vehicles are removed from the network at sinks with time-dependent absorption rates. The implementation used a distributed memory approach with PVM as the message passing library.

The second version PAMINA II focused on the computational aspects of load-balancing. In contrast to PAMINA I, which used a static load-balancing scheme, PAMINA II used online measurements of execution time to balance the computational load on all CPUs of the parallel computer system. It was also used as a feasibility proof for the simulation of the whole Autobahn network of Germany in real-time [103]. PAMINA II will be discussed in detail in Chapter 6.

The third and current version PAMINA III shifted the focus to the actual application of the traffic simulator. The network model was extended to include simple signalized intersections. Also, vehicles now follow individual routes on their trips through the network instead of obeying transfer-rates or random routes. PAMINA III (or simply PAMINA in the following) will be described in detail in this chapter. Its applications will be discussed in Chapters 4 and 5.

The main objective of PAMINA is to execute a route-set (a list of route-plans) in a street network. Each route-plan is defined by a source, a destination, a list of intermediate net points, and a departure time which are used as follows: After the vehicle has been instantiated[1] at the given departure time, it will be inserted into the simulation network at the origin. It will then execute the route-plan until it reaches its destination. Finally, it will be removed from the system after statistics about its actual travel time have been collected.

A simulation run is initiated by supplying a map defining the geometry of the street network and the route-set. Vehicles will be instantiated according to their departure times until all routes have been processed. The simulation will continue until a given percentage of all instantiated vehicles have reached their destination. For the simulation runs of Chapter 4 the simulation time is set to a fixed interval (here 7 hours).

3.3 Network elements

One of the first applications of the network simulator PAMINA was to simulate the whole Autobahn network of Germany in *real-time*, which is to say that one simulation second takes as long as one wall-clock second. The network was given as a graph with nodes and links which had to be represented in the simulator. In the following sections we will describe this network representation.

The network representation used in PAMINA is a graph in which each intersection is represented by a node[2] and each street segment between intersections corresponds to two edges[3]. Moreover, there are nodes defined by the *natural* boundaries of a road network with node degree one, called *terminators*, and additional nodes with degree two where vehicles can enter or exit the network, called *ramps*[4]. This network is usually supplied in two sets of objects: (a) the *set of nodes*, each of which has a unique number (id) and the geometric location of the object, given in rectangular coordinates relative to an arbitrary, but fixed point, and (b) the *set of edges*, each of which has two references (by id) to nodes and optional information such as name, number of lanes, or speed limit.

3.3.1 Street segments

The directed connection (edge or link) between two nodes is represented as a grid equivalent to the model by Nagel/Schreckenberg and its two-lane extension. The characteristics *length*[5], *speed limit*, and *number*

[1] In object-oriented programming languages the term *instantiate* is used for the dynamic creation of a memory object (e.g. vehicle). These objects usually have a limited life-time before they are *deleted* or *disposed*.

[2] Also known as *vertex*.

[3] A segment can correspond to one edge or two edges depending on whether the two directions are equivalent, or not. PAMINA uses the latter, even if all characteristics of both directions are identical, since this symmetry is broken during simulation, anyway.

[4] The segments feeding the ramps are not part the network. Therefore they do not increase the degree of a ramp. If the map were extended to include lower hierarchies, ramps would also have degrees larger than two.

[5] The length of a street segment is either explicitly given or derived from the Euclidean distance of the two nodes.

of lanes are used to adapt the CA model. The size of the grid is computed by using the grid-site length of 7.5 [meter] as a unit.

It is important to note that the characteristics mentioned so far are *constant* for the whole segment. Typical details like additional turning lanes in front of intersections may be modeled by inserting additional nodes to split a given segment and assigning different parameters to the various parts.

Lane changing

The lane changing rules were taken from Section 2.2. If a link has more than two lanes, we enforce a left–to–right lane-changing priority. This prevents two vehicles from moving to same common site in case both have to change lanes according to their lane changing rule set.

For PAMINA II, special rules apply in the vicinity of ramps: exiting vehicles are required to remain on their exit lane. They are no longer permitted to change lanes.

Speed Limit

In contrast to the original CA model which assumed a maximum speed limit of approximately 120 km/h (freeway traffic), the speed limit within a city is usually lower. In order to match individual speed limits of the simulation area, for each segment we introduced a CA speed limit

$$v_{sl} = \lfloor v_{sl}^{real} / l_{site} + 0.5 + p_d \rfloor$$

where v_{sl} is the CA speed limit given in sites per time-step, v_{sl}^{real} the real speed limit given in meters per second, l_{site} the CA site length given in meters, and p_d the deceleration probability of the CA rule set. Moreover, v_{sl} is forced to be in $[1 \ldots v_{max}]$. Note that p_d is used to compensate for its reduction of the average free-flow velocity to $v_{max} - p_d$. Using the speed-limit as a reference for the free speed is based upon the assumption that most drivers will go as fast as permitted. Depending on regional driving habits, the free speeds may have to be adjusted to match actual measurements.

3.3.2 Basic network elements for freeway junctions

We designed eight basic network elements which serve as building blocks for junction templates. Similar to the design of the traffic CA, the basic elements are discrete in time and space. Their rule-sets are of comparable complexity.

Source (SRC) A source is associated with a network node and the outgoing lanes of a street segment. It carries a queue of vehicles that are to be inserted into the street segment. Every time-step it scans the first site on each lane for vacancy. Each vacant site is then filled by those vehicles in the queue which have been waiting the longest. Compared to the "look back" rule of the two-lane rule-set for lane changing, this is a rather simple behavior. A more elaborate rule-set can be found in [75].

Sink (SNK) A sink is associated with a network node and the incoming lanes of a street segment. Every time-step it scans the last (closest to the node) v_{max} sites of each lane to detect vehicles that carry a route-plan whose destination node is the same as its own network node. Whenever such a vehicle is found, it is removed from the site and deleted after some information about the route-plan execution has been collected.

Transfer Segment (TS) Transfer segments are multi-lane CA grids of length $l_{transfer}$ used to let vehicles travel from absorption ranges to insertion ranges. As for lane-changing they function as though they were normal grids. The beginning and the end of a transfer segment is either defined by a connector or a block preventing vehicles from advancing any further.

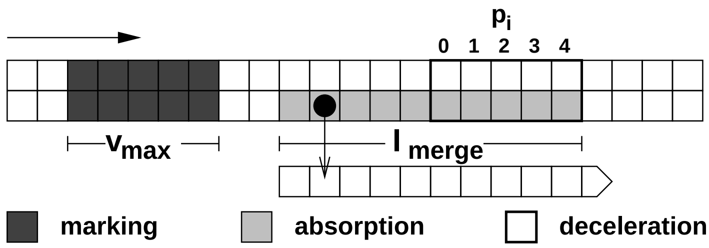

Figure 3.1: *Marking, Deceleration, and Absorption*

Marking Range (MR) The marking range is used to mark vehicles for exit. It is associated with a source street segment and a destination street segment. Every time-step it scans a range of sites of length v_{max} on the source segment for vehicles required to exit onto its destination segment. Whenever such a vehicle is found, it will be marked with a flag which influences CA lane-changing behavior.

Absorption Range (AR) An absorption range is associated with a source street segment, a destination street segment and a transfer segment. At every time-step it scans each lane (see Figure 3.1) over a range of length l_{merge} on the source segment for vehicles that are marked for exit to its destination segment. Whenever such a vehicle is found and the corresponding site on the transfer segment is vacant, it will be moved from the source segment to the transfer lane preserving its current velocity. At the same time the flag which caused the transfer will be reset.

Deceleration Range (DR) A deceleration range is associated with a street segment. At every time-step it scans a certain set of lanes (not necessarily all lanes) over a range of length v_{max} (see Figure 3.1) for vehicles marked for exit and currently located in a lane leading them to a wrong destination. Whenever such a vehicle is found, its maximum velocity is decreased as follows: Let v_i denote its current velocity, $v_{i,max}$ its maximum velocity, and p_i the position of the vehicle, where $p_i = 0$ ($p_i = v_{max} - 1$) represents the first (last) site of the range of the range, respectively. Then

$$\begin{aligned} v_{i,new} &= \min(v_i, v_{i,max}, v_{max} - p_i - 1) \\ v_{i,max,new} &= \min(v_{i,max}, v_{max} - p_i - 1) \end{aligned}$$

represent the new current velocity and new maximum velocity, respectively. The vehicle will never be able to leave the range in forward direction and will eventually stall at the end of the range. It can only proceed by changing lanes and thus restoring its maximum velocity to its old value.

Insertion Range (ER) The insertion range is the counterpart to the absorption range. It is associated with a transfer segment and a destination street segment. At every time-step it scans each lane at the end of the transfer segment over a range of length l_{merge} for vehicles. Whenever a vehicle is found and the corresponding site on the rightmost lane of the destination segment is vacant the vehicle is transferred from the transfer segment to the destination segment.

Connector (CN) The connector is associated with a source street segment and a destination street segment. At every time-step it copies boundary information from the outgoing lanes of the destination segment to the incoming lanes of the source segment and vice versa. The associated CA grids can thus be regarded as continuous so that no special rules are needed to guarantee consistent update between segments.

3.3.3 Freeway junctions

The representation of Autobahn junctions or freeway junctions as one of composite structures is an over-simplification. Figure 3.2 depicts an intersection of degree four with its given *substructure* and the corresponding simplified graph structure.

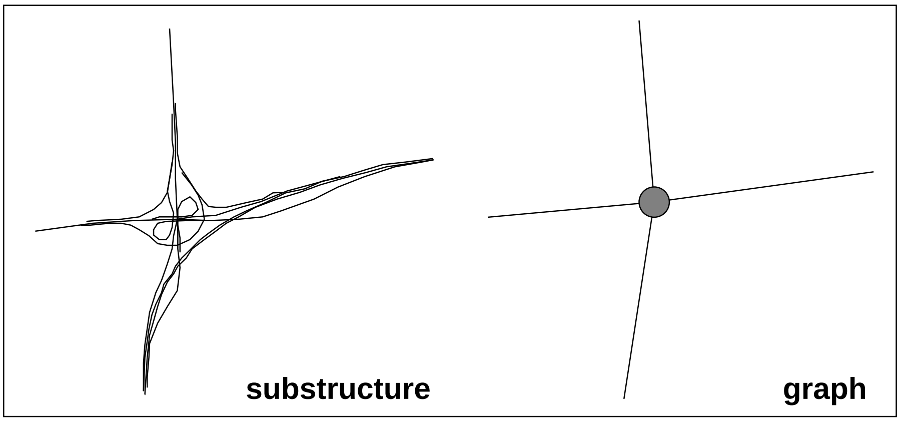

Figure 3.2: *Substructure of an junction of degree 4* — The left-hand side depicts the substructure of an intersection of degree four. All deceleration, transfer, and acceleration lanes are explicitly coded. The intersection in question is *AK Jüchen* of Autobahn A44 and A46. The right-hand side shows the corresponding graph representation which PAMINA uses as input format.

The reason why the early versions of PAMINA were still based upon such a simple input data format is simply due to its general availability. More elaborate data formats — especially for the German street network — have only started to emerge. It will be years before a really comprehensive data base will be available. Meanwhile, the concept was to provide a set of substructures which can be used as templates to model the real-world intersections.

Composite network elements (substructures)

The main characteristics of a junction like throughput and capacity will be captured by defining adequate template substructures and choosing available parameters accordingly. These are: the number of lanes $n_{transfer}$ of the transfer segment, the maximum velocity and related CA parameters of the transfer segments, the lengths and positions of the merge, absorption, and insertion ranges, as well as the relative position of the ranges with respect to each other. Throughput measurements of real intersections can be used to validate the templates and compare them to other models that have a higher fidelity of their intersection representation [41, 63, 86].

Each composite network element is composed from several basic network elements described above. In addition to their individual functionality they share the ability to forward vehicles with respect to a canonical *through-direction*. This is done by using connectors to pass boundary information of an incoming segment to the corresponding outgoing segment and vice versa. Which lanes are to be regarded as through-lanes will be described below. Table 3.2 shows an overview about how many basic elements form one composite element.

Composite Element	TS	SRC	SNK	MR	DR	AR	ER	CN
terminator	0	1	1	0	0	0	0	1
ramp	4	2	2	2	2	2	2	2
intersection degree 3	4	0	0	4	4	2	2	2
intersection degree 4	8	0	0	8	8	8	8	4

Table 3.2: *Composite Network Elements*

Terminator A terminator consists of one source and one sink. They are necessary to define boundary conditions for the traffic volume generated by routes.

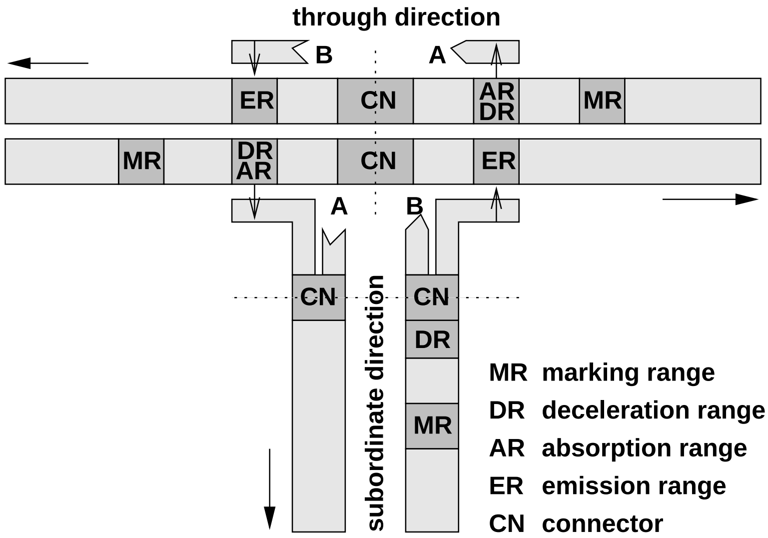

Figure 3.3: *Junction of degree 3*

Ramp The ramp serves as an origin and destination for route-plan execution. Since vehicles can leave into two distinct directions there are two sources (and, of course, two sinks) per ramp. The through-direction is trivially defined.

Junction of degree 3 The substructure of an junction of degree 3 (see Figure 3.3) is designed to be asymmetric: two of the three incident street segments are regarded as through-direction, one is subordinate. Since information about the actual structure is usually not available, the geometric graph information is used to determine the through-direction as follows: for each pair of incident segments the enclosing angle is calculated. Of those the pair with the angle having the smallest deviation from π is regarded as through-direction, the remaining segment as subordinate.

Due to the discrete nature of the CA there is another characteristic: vehicles coming in on the subordinate segment have to change lanes to reach either the left $\lfloor n_{lanes}/2 \rfloor$ lanes for a left turns or the right $\lceil n_{lanes}/2 \rceil$ lanes for a right turn. Suppose two vehicles A and B end up on the very last sites of the incoming segment with A bound to make a right turn, but located on one of the left lanes, and B bound to make a left turn, but located on one the right lanes. In such a case the vehicles will block each other from changing lanes, resulting in a dead lock which is not automatically resolved through the CA rule set. Therefore at every time these locations are scanned for pairs of vehicles which fulfill the condition above. In case such a pair is found the vehicles will simply be switched.

Junction of degree 4 The substructure of an intersection is regarded to be completely symmetric resembling a *clover*. Opposing incident street segments are canonical through-directions. Figure 3.4 shows the structure of this type of intersection. Note that due the symmetry only 2 out of 8 transfer segments need to be displayed.

3.3.4 Simple city intersections

In PAMINA III we added another simple intersection type. For city traffic there is usually no need to provide for transfer lanes since the extent of city intersections can be assumed to be zero[6]. Therefore, the structure can be considerably simplified. Figure 3.5 depicts the geometry of an city intersection. All incoming lanes of each incoming segment are equivalent. At the very end of each incoming lane v_{max} sites are scanned for vehicles *before* the usual rules of motion are applied. During each time-step, at most

[6]In TRANSIMS it takes a vehicle at about one time-step to transverse an intersection.

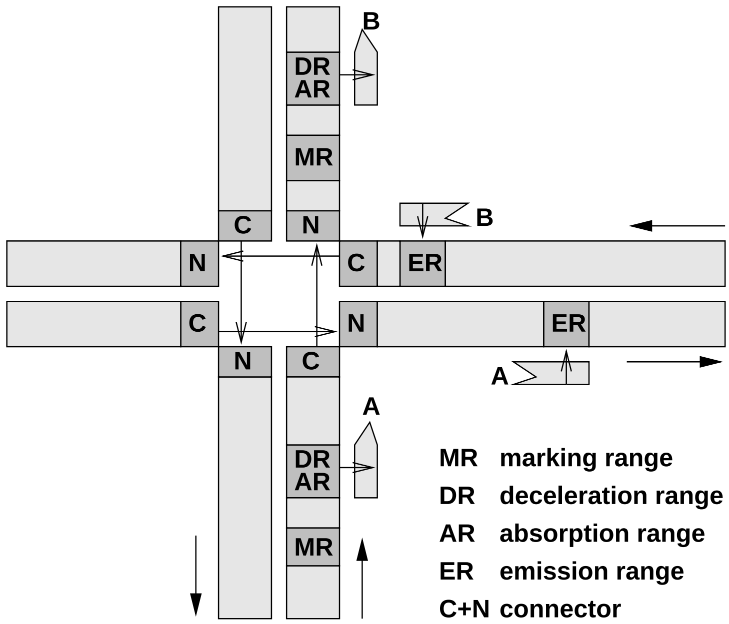

Figure 3.4: *Junction of degree 4*

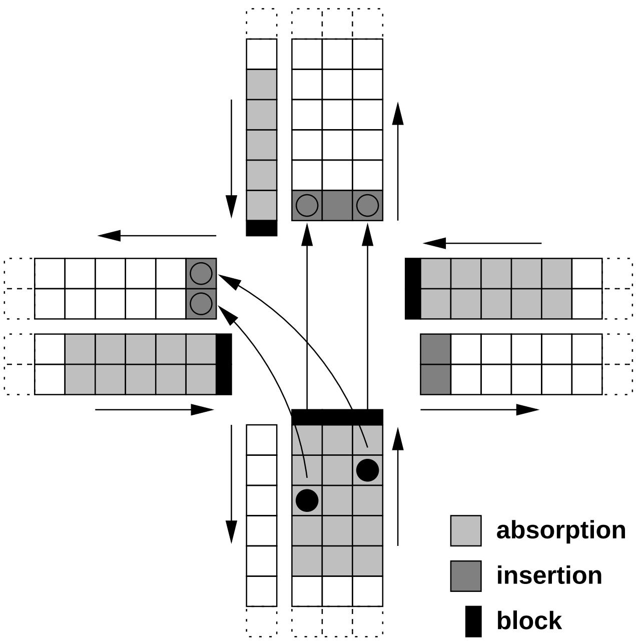

Figure 3.5: *Geometry of a city intersection* — In case of the left turn depicted above, the vehicle on the left lane has no corresponding lane on its destination link. It will be inserted into the leftmost lane.

one vehicle per incoming lane of the source segment can be moved to one of the insertion sites of the destination segment. If possible, the same lane is used on the destination segment. If that is not feasible and if the destination segment has fewer outgoing lanes than the incoming segment has incoming lanes, the vehicle is inserted into the leftmost lane. If that site is occupied, the next neighboring site off to the right is checked until a vacant site is found or the right-most lane is reached. Note that the scanning of the incoming lanes is always done beginning at the site nearest to the intersection. Thus, the order of the vehicles with respect to each other is not changed. In order to ensure unbiased processing of all incoming lanes, the scanning is done in a round-robin fashion with respect to consecutive time-steps.

For a vehicle approaching the intersection there are two alternatives: either it is absorbed from the scanning area and inserted into the destination segment or it proceeds according to its CA rules of motion. Due to block at the end of the lane (or earlier due to other preceding travelers) it will eventually stop. This behavior is important to model the spill-backs in real world traffic. The queues are resolved as one would expect: As soon as the situation on the destination lane(s) improves, vehicles are removed one by one starting at the site nearest to the block. Note that a vehicle may be blocked by other vehicles (having a different destination) although its destination segment is vacant. This effect may cause grid-locks which will be discussed in Section 3.4.2.

Approach and turning behavior

In contrast to other traffic simulations with resolution at city-street level, we do not model a special behavior for vehicles while approaching or transversing intersections. In our implementation all incoming lanes are equivalent. This was done for two reasons. First, modeling detailed approach and turning behavior requires extensive geometric information which is often not available or not consistent. Second, the current rules of motion show a quickly decreasing lane-changing probability as soon as the density exceeds a certain threshold. This is mainly due to a strict 'look-back' rule (see 2.2.2) which checks for following traffic on the neighboring lane. In contrast to freeway conditions, where this rule maintains the desired traffic jam waves, here, it would prevent proper lane-changing. This again would result in vehicles queuing up, since they could not change to their respective turning lanes [86].

Traffic lights

Traffic lights are modeled by activating the scanning mechanism for the duration of the green phase T_g and deactivating it for the length of the non-green-phase T_r (which includes both the red phase and transition phases). Since there is only one phase per incoming segment, any direction-specific phasing information is averaged over all directions weighted by the number of active lanes into the respective direction. Let i be the incoming segment, j the outgoing segment, $T_g(i,j)$ the length of the green phase from i to j, and $l(i,j)$ the number of lanes going from i to j. The overall green phase will be computed as

$$T_g(i) = \frac{\sum_j l(i,j) T_g(i,j)}{\sum_j l_g(i,j)}.$$

The overall red phase $T_r(i)$ is computed accordingly using all $T_r(i,j)$. Note that due to this averaging, the complete phase cycles $T_{rg}(i) = T_g(i) + T_r(i)$ (green phase + red phase) of the incoming segments may differ from each other, resulting in a continuous phase shift. This is different from real world traffic light installations where the starting time is usually defined by taking a multiple of $T_{rg}(i)$ and adding a relative offset.

Interferences

Two types of interference can occur at an intersection: (a) vehicles that have to obey right of way must wait for gaps in the crossing or oncoming traffic stream, and (b) vehicles that have right of way are obstructed by others blocking the intersection. In the simplest version of our intersection neither of these interferences are handled. However, it is simple to force a reduced throughput through the intersection by examining the overall occupancy of the v_{max} last sites of all outgoing segments and introducing an additional transfer probability. This probability would be one if all sites in the examined area are vacant, smaller than one if all occupied with a functional (possibly linear) transition between the two extremes.

Sources and sinks

In contrast to a freeway junction, a city intersection may be a vehicle source and/or a vehicle sink. This is required for route-plans which start or terminate a node. In this case the absorption range at the end of the incoming segments (see Figure 3.5) will be used to absorb vehicles that have reached their final destination. The insertion sites will be used to insert new vehicles.

3.3.5 Parking accessories

Parking accessories are equivalent to sources and sinks, only that they are located right *on* the link. For the simulation of city traffic they serve as insertion (deletion) points for all trips starting (ending) in driveways or parking lots on that link.

Vehicles can be inserted at an accessory if more than 2 consecutive sites are vacant. This is done to guarantee that during heavy congestion not all gaps in the link are filled by vehicles from the accessory source queue.

3.3.6 Route-sets

The first attempt of including route-sets into a medium scale traffic simulation was done in one of the early versions of TRANSIMS [5]: the interstate traffic of the city of Albuquerque was simulated on a single workstation to show the general feasibility of this approach. Nagel [75, 82, 85] used a parallel computer with two CPU's to run a parallel net simulation based upon the single-lane CA with individual route-plans. He examined iterative route-selection behavior of a group of drivers traveling through the network. The NRW-FVU, TRANSIMS, and PARAMICS groups are currently designing or already using large-scale traffic simulations that include route execution. INTEGRATION [60] and DYNASMART [45] have also been used with individual route-plans, albeit at a smaller scale.

For PAMINA, route-sets represent the third major input for the simulation beside nodes and edges. Each route-plan contains information about the node id of the origin, the scheduled departure time-step from the origin, the estimated travel time in simulation time-steps, a list of node id including the destination as its last entry, and optional vehicle data.

PAMINA expects the routes to be sorted according to their departure time-step. Thus the evaluation of the route-plan is reduced to the following scheme: at every time-step routes are sequentially read until a departure time-step is found that is larger or equal the current time step. For each route a vehicle is created and loaded with that route. The vehicle is appended to the insertion queue of the source associated with the given origin id. If there is more than one source per id (e.g. ramps or accessories) also the second node id of the route-plan is scanned to determine the outgoing segment and thus the specific source. Note that the scheduled time of insertion may differ from the actual time of insertion, since the source can only add vehicles to the grid if there are vacant sites. During intervals of high insertion rates there will be a certain number of vehicles waiting (or 'pending') in the source queues.

In addition to the traffic volume generated by routes PAMINA II provides a mechanism to generate background traffic (see A.2) of a given density. In this case vehicles will be homogeneously instantiated across the network before the actual simulation is started resulting in the selected density. In contrast to their routed counterparts they do not carry route-plans at all. Instead, they change directions at intersections according to a given turning probability. In case an non-routed vehicle is about to leave the network, it will be *reflected* with its current lane and velocity leading into the opposite direction.

3.4 Simulating city traffic

As a first test [101] for PAMINA with realistic route-plans, we used a preliminary route-set generated for TRANSIMS case-study. We used two maps (see Figure 3.6): the complete Dallas/Fort Worth Area and a small excerpt of the latter called *study-area*. The study-area map comprises all streets except small ones in residential areas and similar areas. The large map further contains all minor and major arterials for Dallas and all major arterials for Forth Worth.

The plan-sets which were available at that time contained only trip departure times between 7 am and 10 am of which we selected those between 7 am and 8 am as period of interest. We thus started the simulation at 7 am and let it run at least until 8 am. After that, the simulation either terminated when (a) 99% of all route-plans had been executed, or (b) a grid-lock was detected. In this context we assume the system to be grid-locked if the number of vehicles in the system is constant for more than 600 time-steps. For the CA model we used the deceleration probability of $p_d = 0.3$ in all simulations.

For each simulation run all plans can be regarded as static. For the time being, we do not perform any online re-routing. A plan-set is generated from an activity set consisting of a source plus a departure time on the one hand and destination on the other hand. The plan-set used for the study presented here (also referred to as *plan-set 11*) is a very preliminary plan-set which was generated in the course of the Dallas/Fort Worth case-study of TRANSIMS. Work on plan-sets in the context of the same case-study can be found in [61, 79] and in Chapter 4.

The plan-sets in TRANSIMS are sequences of links plus estimates when these links will be reached. From this, one can calculate when every driver *expects* to be on a certain link. In consequence, one can calculate the complete time dependent *demand* structure, which would correspond to the traffic which would happen if everybody could act according to her expectations.

This demand structure is best displayed in terms of the ratio of flow demand to capacity, also called V/C ratio.[7] If the V/C ratio is exactly one, the number of vehicles which are planning to use a certain link is equal to the maximum number of vehicles the link can let through. If the V/C ratio is larger than one, then the number of vehicles in excess of the capacity will not be able to get through and will be queued up at the entrance of the link. This means that these vehicles will not be at other links downstream when they expected, so that the demand structure downstream of a congested link (i.e. V/C > 1) becomes wrong. From this argument it becomes immediately clear that one has to run the micro-simulation to obtain dynamically correct traffic patterns.

Figure 3.7 shows a graphical display of the demand structure of a plan-set which is similar to the one used for the simulation runs.

All plan-sets are computed for the whole Dallas/Forth-Worth area which means that all routes have to be restricted to the study-area if only that portion of the map is simulated. The truncation of the plans is done in a straight-forward way: any route that contains at least one segment within the study-area will be part of the restricted plan set. Its departure will be delayed by the amount of time that the vehicle would spent outside the study-area before it reaches the first segment within the simulation area. For all edges transversed up to entry, we use the cruising velocities assumed by the planner (also see 4.3.2).

After the start of the simulation, route-plans are executed as follows: (a) At the time-step given by the departure-time, a vehicle is created at the departure node (source) of the route. (b) The vehicle is inserted into a queue associated with the source. (c) Each time-step the queue is scanned for pending vehicles. If possible, the vehicle is removed from the queue and inserted into the first segment, from where it starts

[7] V stands for volume, C for capacity. One has to keep in mind that V refers to volume *demand*.

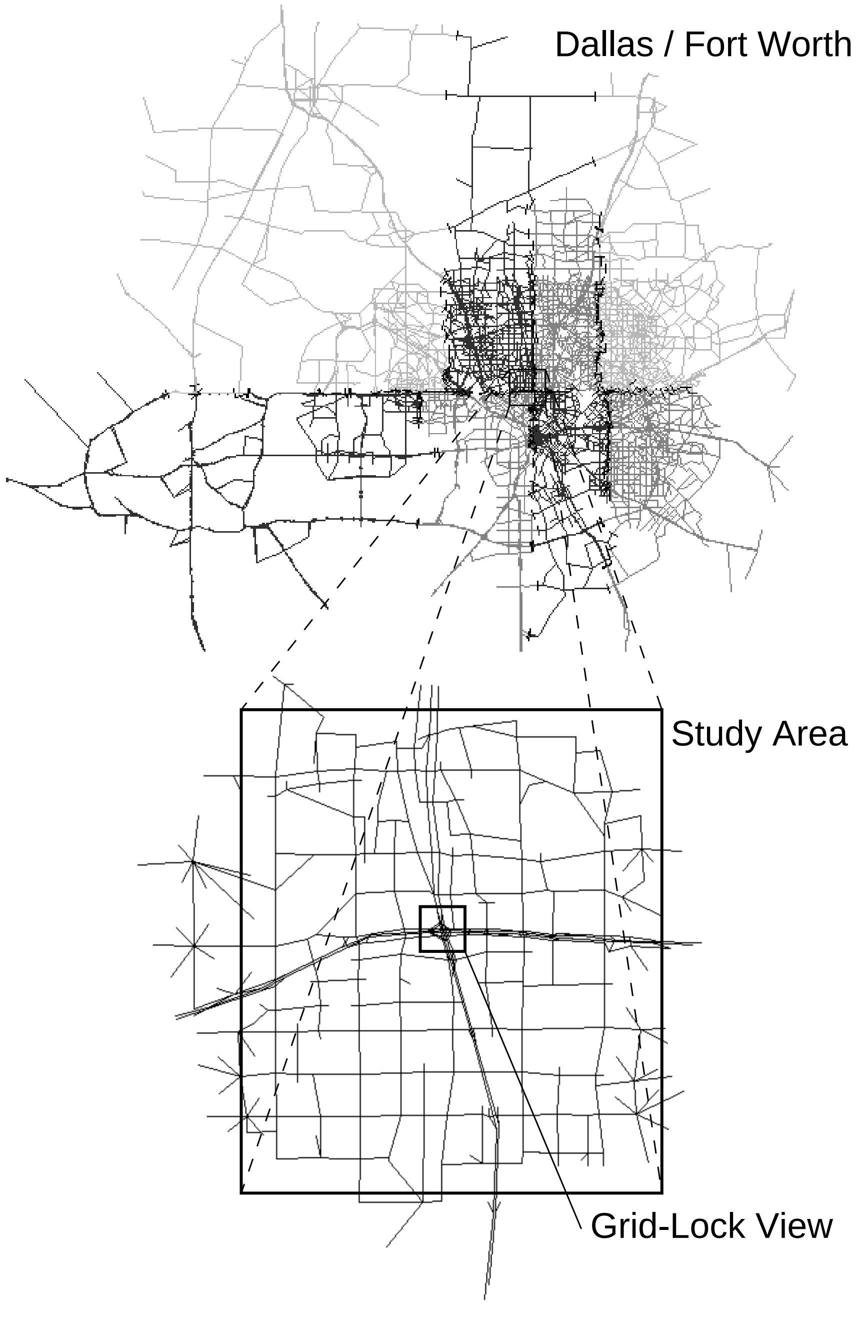

Figure 3.6: *Map of Dallas / Fort Worth* — The different shades of gray in the Dallas / Fort Worth map correspond to the mapping to different processors of the parallel computer topology. Note that the resolution decreases with growing distance from the study-area. For the simulations in this chapter the study-area itself contains only major and minor arterials. The study-area with streets (including local streets) can be seen in Figure 4.21. The small rectangle marked as *Grid-Lock View* can be seen in Figure 3.13.

Figure 3.7: *V/C-ratios* — White links mean that demand is lower than capacity ($V/C < 1$); for gray links demand is between capacity and two times capacity ($1 \leq V/C < 2$); for black links demand is larger than two times capacity ($2 \leq V/C$). The time labels (17:00-18:00) refer to an equivalent afternoon rush hour. This figure was generated using the FlashPlan visualization utility of the TRANSIMS application suite.

following its route-plan. (d) As soon as it reaches the destination, the vehicle is removed from the segment. The travel time is recorded for statistical evaluation.

Note that all vehicles try to execute their route-plans independent of the actual traffic conditions that they encounter along their way. In heavily congested areas, vehicles often spill back across intersections because they cannot enter their next respective destination segments. This can result in complete grid-locks of the simulation area, which cannot be resolved anymore within the current rule-set. This artifact will be discussed next.

During the simulation we keep track of: the number of vehicles inserted so far, the number of vehicles currently in the network, and the number of vehicles that have reached their destination. Upon arrival of each vehicle we store the estimated travel time (computed by the router beforehand) and the actual travel time. These can be compared to check the prediction quality of the router.

Except for the curves depicting the number of vehicles in the study-area (Figure 3.11), we considered only vehicles that arrived before time-step 1800. Also, all curves have been aggregated over 10 simulation-runs (using different random seeds). They have been normalized by the respective number of vehicles that have reached their destinations before time-step 1800. Moreover, the area underneath each curve has been normalized to one.

3.4.1 Levels of Fidelity

It must be the principal goal of a traffic simulation within a cellular automaton context to keep the interaction rule-set as small as possible. This has two advantages: (a) The number of parameters will be small, which reduces the probability of artifacts and makes the model more credible, if it can be successfully validated using those parameters. (b) A simple rule-set usually results in efficient coding, which is essential, considering the before-mentioned necessity of statistical averaging.

We examine different levels of fidelity by simulating the same plan-set after activating different combinations of characteristics of intersections (traffic-lights) and street segments (speed limit) which were described above. We ran simulations for the combinations listed in table 3.3. For the *low-fidelity* (lf) model both traffic lights and speed-limits were deactivated. In the *speed-limit* (sl) model only traffic lights were activated, in the *traffic-lights* (tl) model only traffic-lights. The *high-fidelity* (hf) model contained both features.

short name	lf	sl	tl	hf	rl
long name	low fidelity	speed limits	traffic lights	high fidelity	reduced lights
speed limit	no	yes	no	yes	yes
traffic lights	no	no	yes	yes	q_r

Table 3.3: *Parameters of simple city simulation* — Parameters (bottom rows) defining the fidelity (right columns) of the simulation: In case of an active speed limit the maximum velocity v_{max} is reduced from its original value of 5 to a segment-specific value. In case of active traffic lights, the transfer at intersections is decreased by introducing periodic red phases during which some of the incoming segments are blocked. Reduced non-green phases (q_r) are discussed in Section 3.4.3.

Delay of arrival

Since upon arrival of each vehicle its actual travel time t_{actual} is recorded, we could compute the distribution of relative delay d

$$d = \frac{t^{trav}_{act} - t^{trav}_{sched}}{t^{trav}_{sched}}$$

with respect to the scheduled trip time t^{trav}_{sched} forecast by the router. Note that negative values denote early arrivals. Figure 3.8 shows the results for plan-set 11 in all fidelities. It is obvious that in mode

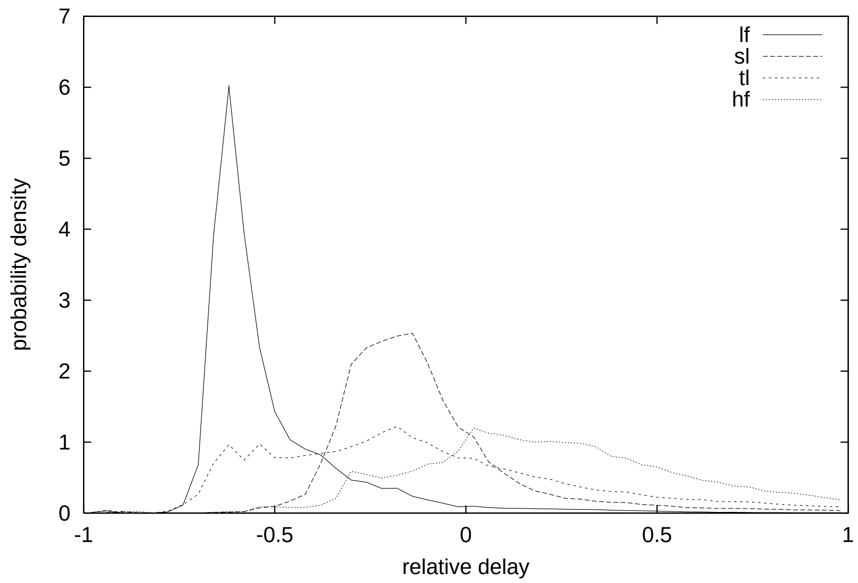

Figure 3.8: *Distribution of relative delays for plan-set 11* — The low fidelity model *lf* shows a characteristics peak at negative relative delays since vehicles are not delayed by either speed-limit or traffic-lights. Higher fidelities have their peaks shifted to positive delays.

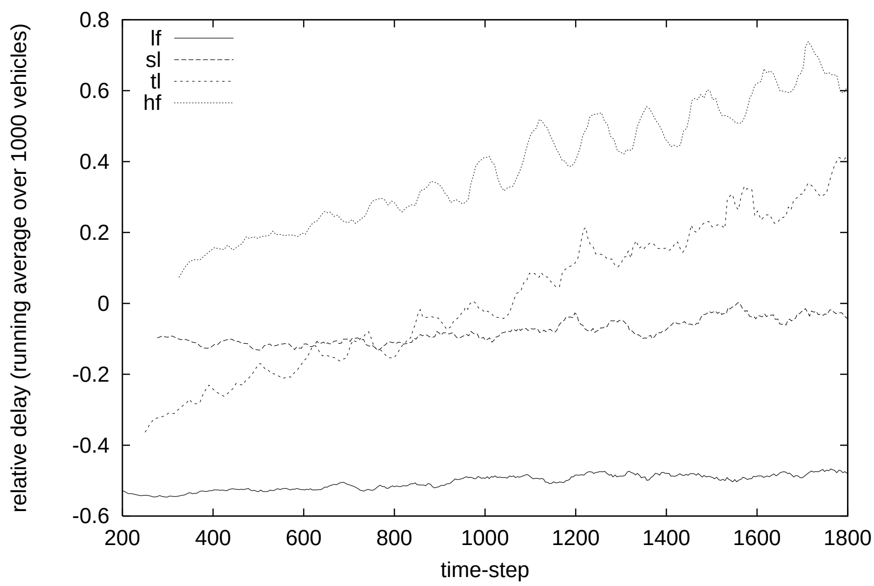

Figure 3.9: *Running average of relative delays for plan-set 11* — The curve for fidelity *sl* is closest to zero relative delay throughout the simulation. Fidelities with activated traffic lights have a rising tendency. This is caused by growing congestion inside the study-area: The number of vehicles never reaches a plateau. The low fidelity simulation executes routes to quickly resulting in large negative delays.

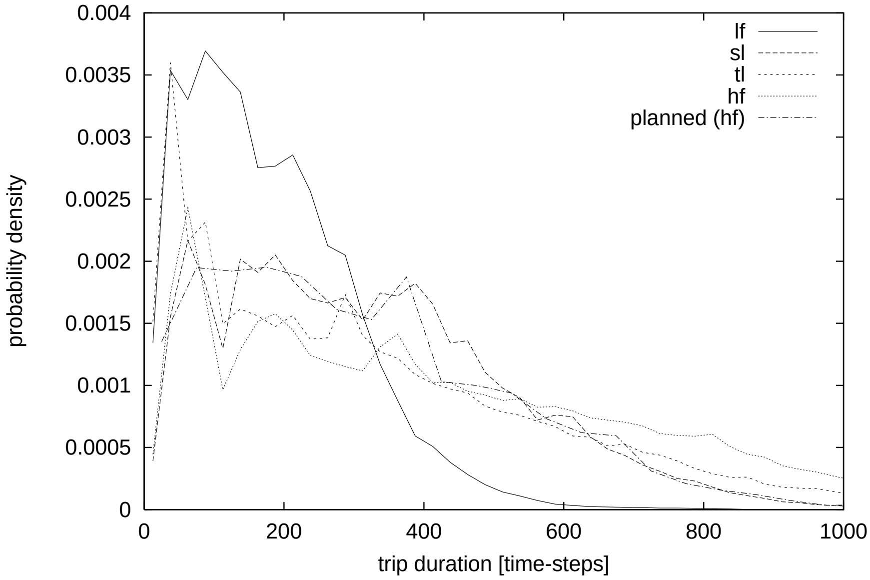

Figure 3.10: *Distribution of trip duration* — The distribution of planned trip duration as projected by the planner is best matched by fidelity *sl*.

lf (due to the missing speed limit) route-plans are executed much too fast. There are hardly any late-comers at all. In modes *sl* and *tl* the peak is already shifted towards zero delays but still biased. Mode *hf* generates a distribution which peaks almost exactly at zero delay. The average, however, is shifted towards positive delays. This can be verified in Figure 3.9, which displays the running average (over the latest 1000 vehicles) of the relative delay. As can be seen, in the uncongested regime the travel times of the micro-simulation are in general shorter than what the planner had expected. Yet, in highly congested situations, travel times in the micro-simulation become longer than what the planner had expected.

We have to point out that it cannot be the goal of the dynamic micro-simulation to reproduce static results forecast by the planner. These comparisons only serve as a consistency-check. It is to be expected that results of the micro-simulation will considerably differ, e.g. in places where the implicit aggregation of the planner smoothes over sudden peaks in traffic load.

Trip Duration

Figure 3.10 depicts the distribution of trip durations for all fidelities. As expected mode *lf* has its peak shifted far left towards small trip durations. Mode *tl* has a significant peak at approximately 50 [sec], where it reaches the same value as *lf*. This is due to the fact that the probability of encountering a traffic light is a very small for short trips, rendering modes *lf* and *tl* equivalent in that regime. Modes *tl* and *hf* have a very slow descent towards large trip durations due to grid-locks. The curve of trip duration as projected by the planner is best matched by mode *sl*.

3.4.2 Reproducibility and grid-locks

Since the CA model contains a stochastic element, we receive a unique evolution of the simulation for each seed of the random number generator. In a sub-critical system the network is able to transport all vehicles (albeit with delay) so that all runs will look similar on a macroscopic level. In a system with a network throughput incapable of handling the loading, the system will most likely grid-lock (see below). Between the two extremes we find a regime in which the specific configuration may either block

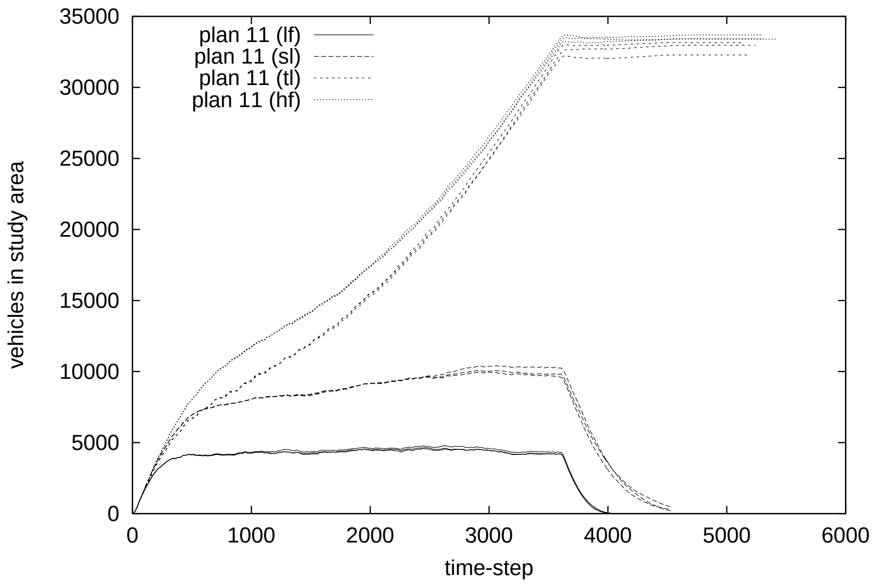

Figure 3.11: *Vehicles in study-area for plan-set 11* — While the vehicle count reaches an equilibrium for fidelities *lf* and *sl*, the other two fidelities grid-lock.

or not block. Figure 3.11 depicts the number of vehicles which are in the study-area at a given time-step. Fidelities *lf* and *sl* belong to the sub-critical regime. Both curves reach a plateau (at 4500 vehicles after time-step 500 for *lf* and at 10,000 vehicles after time 2500 for *sl*) after an initial loading phase representing an equilibrium between the insertion and deletion rates of vehicles. After the loading phase all remaining vehicles are discharged within 400 time-steps for *lf* and within 900 time-steps for *sl*.

Modes *tl* and *hf* belong to the super-critical regime. They never reach an equilibrium between insertion and deletion during the loading phase; and the plateau after the loading of the network is due to grid-lock.

See [7] for a similar (albeit much smaller scale) investigation on the relation between network loading and network throughput.

Grid-locks

In this simulation, a grid-lock situation can be determined by a horizontal line after time-step 3600 (e.g. the end of vehicle insertion). This is caused by closed loops in the traffic network in which all sites are occupied. Similar grid lock situations were reported in [28, 79]. Figure 3.12 depicts a simplified intersection. In the left half traffic is already dense, though not grid-locked. Due to high demand and the red-phases at the intersection, the segments of the loop are no longer cleared. In the right half the whole loop is blocked: the first vehicle in each lane is forced to make a right turn into another lane which is also blocked. This phenomenon (in its strict form) cannot be seen in real-world traffic, because drivers move out of the lanes and pass on the on-coming lane or they abandon their current route and choose a detour. Figure 3.13 shows a screen-shot of a simulation run with plan-set 11 in mode *hf*. Vehicles are represented by dark dots, lane boundaries by grey lines. Right at the center, there is a small grid-locked loop blocking traffic from all incoming directions.

3.4.3 Reduced non-green phase length

Above we have seen cases of sub-critical and super-critical loading of the network. Although the respective models are defined by different types of rule-sets, they can be regarded as two specific cases of a more

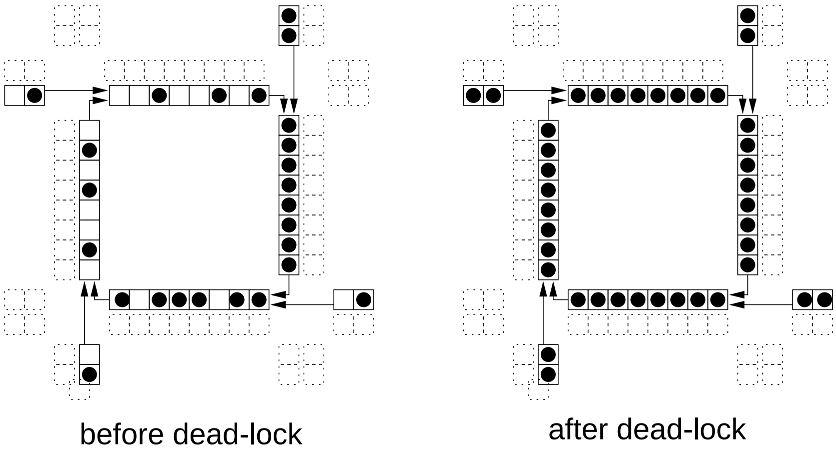

Figure 3.12: *Geometry of a grid-lock* — Left: Just before a grid-lock situation. Right: The grid-lock was caused by vehicles which are required to make right turns. The first vehicle of each grid is waiting for the other vehicle to leave the side. Since this dependency is circular, it cannot be resolved anymore.

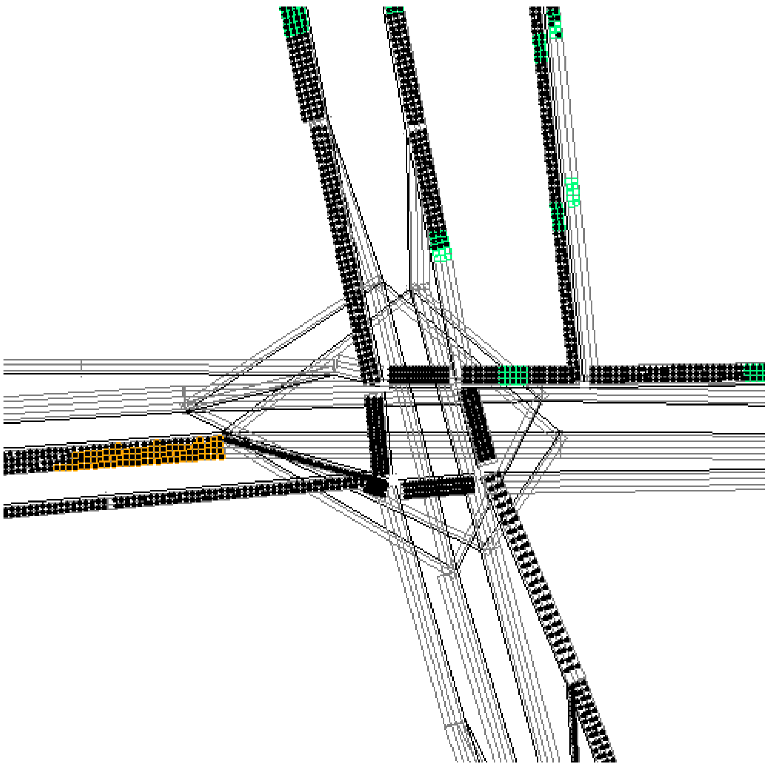

Figure 3.13: *Grid-lock in study-area (plan 11, fidelity* hf*)* — The screen shot shows a grid-lock configuration found while executing plan-set 11 at high fidelity. The location of this excerpt within the study-area can be seen in Figure 3.6.

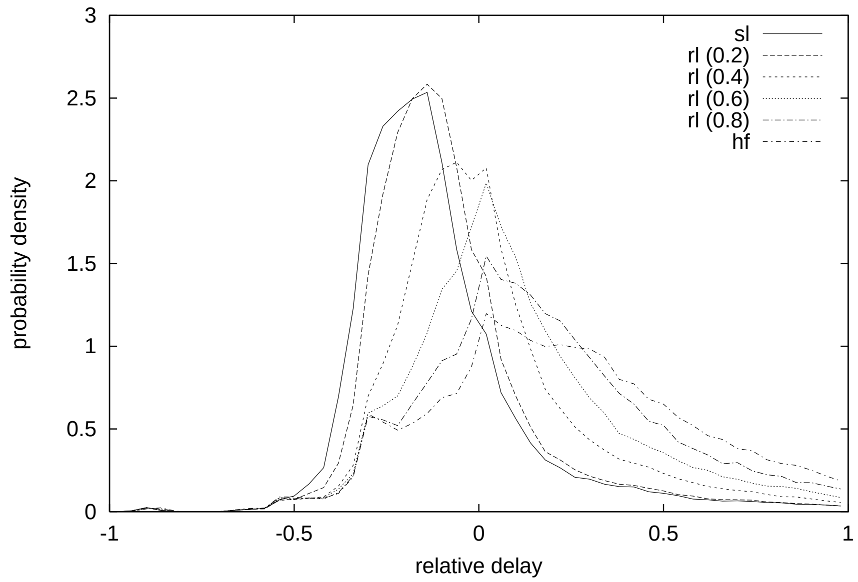

Figure 3.14: *Distribution of relative delay of plan-set 11 (different q_r)* — The distribution of relative delays exhibits a smooth transition between fidelity *sl* (corresponding to $q_r = 0$) and fidelity *hf* (corresponding to $q_r = 1.0$).

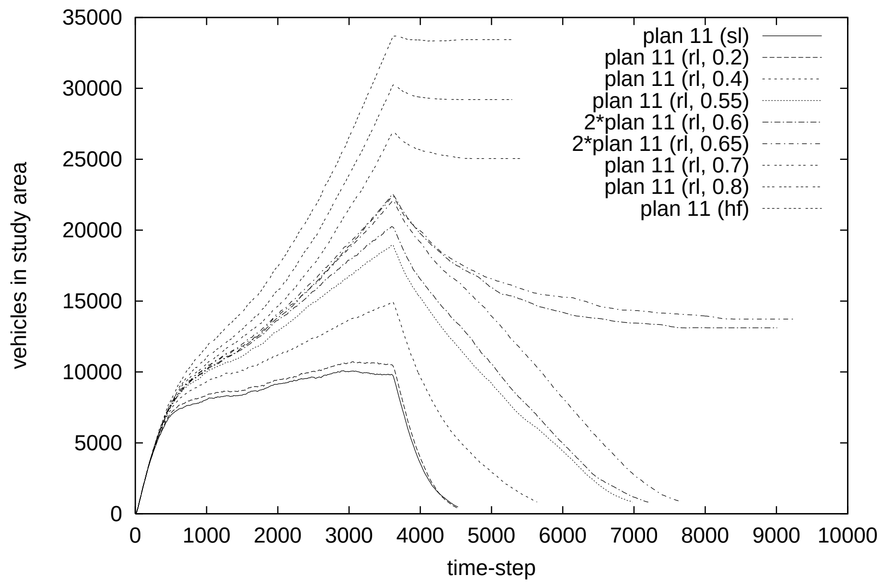

Figure 3.15: *Vehicles in study-area (different q_r)* — The curves for the number of vehicles in the study-area show a transition at $q_r \simeq 0.65$. For larger values the simulation grid-locks, for smaller values it does not.

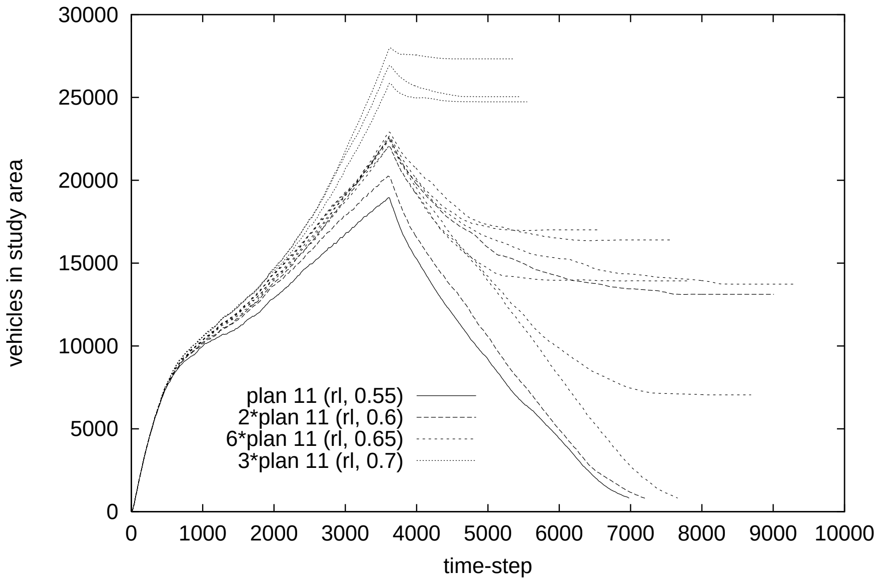

Figure 3.16: *Vehicles in study-area (q_r = 0.55, 0.6, 0.65, 0.7)* — For $q_r = 0.6$ and $q_r = 0.65$ the simulation generates both grid-locking and non-grid-locking runs. Note that the only difference between the runs is the seed of the random generator.

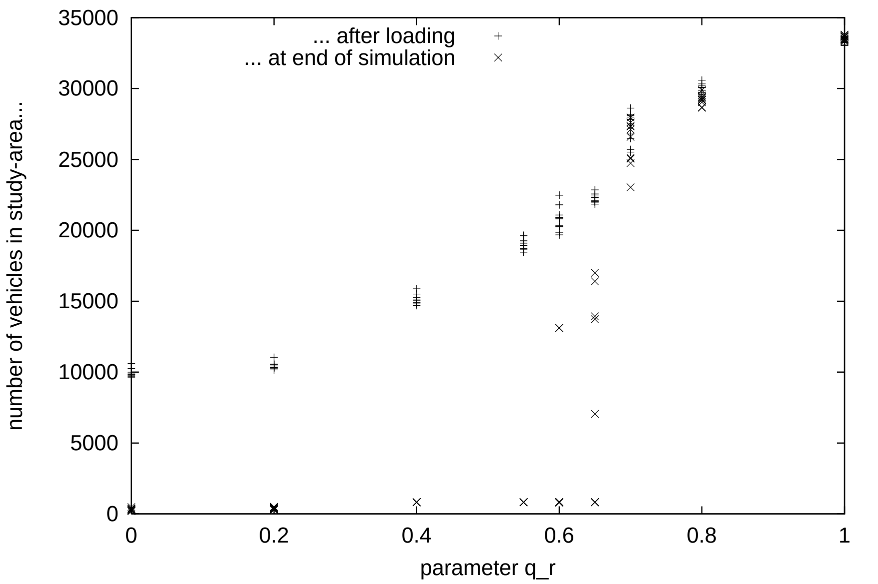

Figure 3.17: *Vehicles in study-area as a function of q_r* — The number of vehicles in the study-area after the loading phase increases almost linear with q_r. The number of vehicles at the end of the simulation phase, however, shows a transition around $q_r = 0.6 \ldots 0.65$.

general rule-set (called *rl*) in which in the effective red-phase $T_{r,eff}$ is computed by multiplying the original phase-length T_r by a certain factor $q_r \in [0 \ldots 1]$. Consequently, $q_r = 0$ represents mode *sl* while $q_r = 1$ represents mode *hf*.

We have made several runs for different values of q_r between 0 and 1. Both Figures 3.14 and 3.15 show a smooth transition between modes *sl* and *hf* (for 0.6 and 0.65 blocking and non-blocking representatives were chosen) as far as delay and trip duration are concerned. Values below 0.6 show a secure sub-critical (no grid-locks), and values above 0.65 a secure super-critical behavior. For 0.6 and 0.65 the system has a certain chance of reaching a grid-lock (1 out of 10 runs for $q_r = 0.6$ and 5 out of 10 for $q_r = 0.65$), which can better be seen in Figure 3.16. Figure 3.17 shows the number of vehicles at the end of the loading phase and at the end of the simulation plotted versus q_r.

A similar effect was reported for simple 2-dimensional grid models [21, 38, 69], except that in these studies the overall density was changed instead of the efficiency of the network components. Intuitively, the grid-lock effect seems to be the same. Yet, further investigations will be necessary to understand in how far simple models on a 2-dimensional grid can indeed offer insight for real-world city traffic, which is admittedly happening in 2-dimensional space but is composed of traffic on 1-dimensional links.

Chapter 4

Iterative Route Adaption

As we have seen in the previous chapter, traffic simulation has benefited a lot from the development of new simulation techniques and the increasing computational power of today's computers. Handling the functionality of the micro-simulator is not an easy task, but it can be solved by thorough research, validation and implementation. The most imminent problem of traffic simulation, however, has not been solved yet: providing the input data to drive the simulation. So the question is rather "*Where* and *When* do people go?" than how to simulate microscopically. In reality vehicular traffic is generated by millions of individuals who use their past experience and up-to-date information to chose one of many possible paths through the traffic network. If we knew the origin, the departure time and the destination for each and every individual, we could use this information to create a so-called *origin-destination matrix* (abbreviated as *O-D matrix*). Unfortunately, as a rule this data is not available!

In this chapter, we will concentrate on the task of generating reasonable input data for the micro-simulation. First we outline, how route-sets have been produced traditionally. In Section 4.2 we describe an iterative adaption method to produce a route-set using simulation feedback. We continue with Section 4.3, in which we use PAMINA to conduct comprehensive iterations runs. As a conclusion, in Section 4.4 we compare PAMINA to two other simulators. Each of the simulators uses route-sets that were obtained from the same type of iterative adaption process.

4.1 Introduction

As mentioned above, the central prerequisite of traffic simulation is to know *how many drivers* depart in a given time-interval from *which source* to *which destination*. One way to retrieve partial information about the origin-destination matrix is to conduct a sampled survey of drivers. The disadvantages are obvious: (a) the fraction of drivers will be small, (b) the data may be heterogeneously distributed over the simulation area, (c) many of the drivers will not be able to give route descriptions, (d) some of the drivers will give wrong data for whatever reason. It is obvious that extrapolating any route or origin destination information based upon survey results is — to say the least — difficult.

4.1.1 Drawback of traffic count data

Another possibility to drive the micro-simulation is to discard the individuality of vehicles and regard only streams of vehicles (refer to Figure 4.1). This simplifies the problem of input data to one of collecting vehicle count information which could be obtained from counting devices distributed across the simulation area. Provided that sufficient information (also see the discussion about retrieval of count information in Section 5.1) is available there are two possibilities:

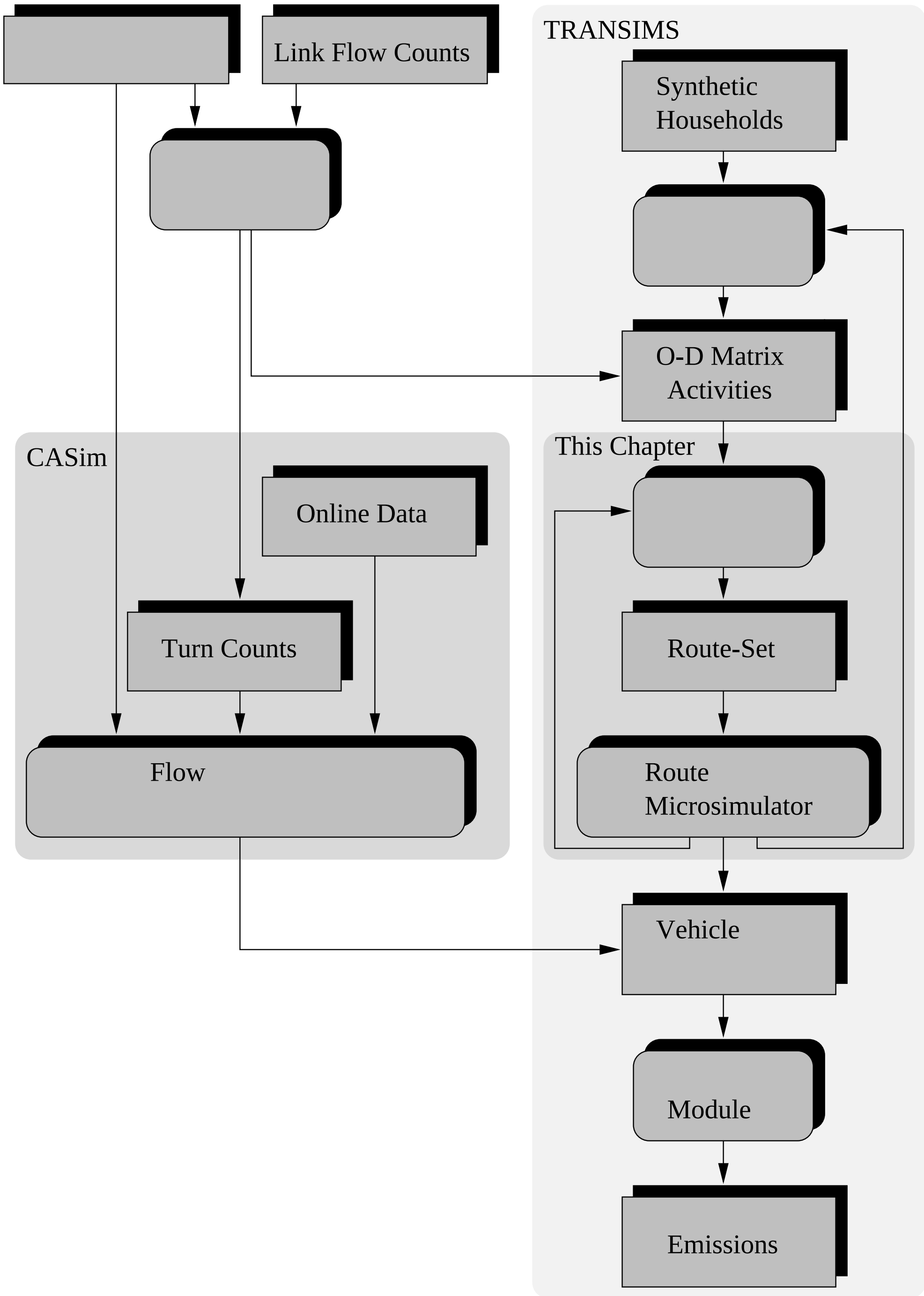

Figure 4.1: *Data flow in micro-simulation application suites* — In this diagram rectangles with round corners denote applications, the other rectangles denote data. Arrows denote the flow of data between the modules. On the right hand side the TRANSIMS micro-simulation suite is sketched. Note that there are at least two levels of feedback: between the micro-simulator and the route-planner on the one hand (inner feedback) and the micro-simulator and the activity generator on the other hand. This chapter will focus on the inner feedback loop which is shown in a more detailed form in Figure 4.2.

- The counting data is used to generate an origin-destination matrix by solving a non-linear program. The problem is that for a given intersection with n incoming links there are $n(n-1)$ turning dependencies, but only $4n$ link counts. This relationship continues for larger networks leading to about $O(N^2)$ origin destination relations for N intersections with only $O(N)$ counts. The result is a highly under-determined system of equations which is difficult to solve. Using time-dependent correlations between inflows and outflows yields some additional information but also requires to use a fine time-grid which worsens statistics for each time-bin. Assumptions about the maximum entropy do not yield conclusive results either [51].

  Note that by creating an origin-destination matrix from the anonymous count information can be "personalized" again. This is currently being done at the FVU-NRW project (see [87]).

- The counting data is only used to generate turn count information for each individual intersection. Absolute counts are converted into relative turning probabilities for each incoming direction with respect to each outgoing direction. Additionally, information about all inflows and outflows at vehicle sources and sinks is included. It is possible to drive a simulation using only this type of data as Esser [31] has shown for the city traffic of Duisburg. Nagel [75] notes that turning probabilities are especially bad downstream from a stochastically occurring congested area: vehicles that have been delayed in a jam will not arrive during their correct turning probability anymore, but rather during that of a later time-bin.

  Another problem arises if new streets are added to the underlying traffic network. Since there are no traffic counts available for the new street, there will not be any flow on the corresponding link of the micro-simulation either. This clearly shows the uselessness of traffic counts for the comparison of scenarios with varying traffic infrastructure.

  One advantage of the anonymous approach is that vehicles can be added and removed dynamically to match traffic counts provided by some online source. In [29] traffic states are dynamically reproduced by either using the density or the average velocity as a reference.

4.1.2 Using origin-destination matrices

Instead of using deriving origin-destination matrices from unreliable and incomplete traffic flow counts, in TRANSIMS [116] another approach was chosen: based upon census data[1], artificial households are created matching the statistical properties of the original census [8]. Each household has a family with members requiring trips (such as home-to-work or work-to-shopping) to different places in the network. A combination of trip origin, trip destination, and trip departure time is called *activity*. The set of all activities for a given time period and area is called *activity list*. It can be regarded as a detailed origin-destination matrix that has not been aggregated into time-bins. Note that, for the time being, all route-sets in TRANSIMS are still based upon origin-destination data, and not on activity lists.

First results with the TRANSIMS case-study [79] reveal that not only the route-sets but also the activity lists will be influenced by the results of the micro-simulation. This feedback, however, takes place on a different time-scale: instead of changing route choice on a day-to-day basis, changes of the activity lists rather reflect changes of behavior as to *when* and in *which order* trips are planned. Drivers facing the same congestion every day over and over again will eventually chose a different departure time or maybe combine several trips into one and switch to another day of the week.

4.1.3 Equilibrium assignment

In the following we would like to outline how an activity list (or origin-destination matrix respectively) can be converted into a route-set serving as input for the micro-simulation. We will call this process *assignment* or *routing* and the executing agent *router*. There are two main prerequisites:

[1] This approach may cause difficulties in some European countries where census data is not available for research purposes, not even in the strongly simplified form used for TRANSIMS.

- There has to be some relationship between the travel-time on a link and the currently assigned traffic volume with respect to its capacity which is the maximum number of vehicles that can go through the link in a given time unit. Commonly used functions exhibit a steep increase in travel-time as volume approaches capacity.
- We require a cost function judging the quality of a chosen route. In most cases this will be the trip-time. Another option is to convert the trip-time into some monetary unit and then add other expenses, such as vehicle maintenance fees on a per-mile basis and tolls. The latter is especially important if the impact of road-pricing is to be investigated [81]. When we speak of a *shortest* path, we actually mean the one with the lowest associated costs.

A simple algorithm (called *all-or-nothing*) works like this: (a) choose one origin-destination volume between S and D, (b) compute the shortest path between S and D and add the volume to all links that are part of the shortest path, (d) update the costs of all links involved, (e) repeat until all O-D volumes have been assigned. Obviously, this approach has major drawbacks:

- The symmetry between O-D pairs is broken because they are *sequentially* processed by the algorithm. Whoever is first to be handled has a greater probability of being assigned a shorter path.
- O-D relationships of large volume cannot be split up resulting in unnecessarily high costs for volumes which could otherwise be distributed over several alternative routes.
- After the algorithm has terminated, there may be links that have a volume assigned to them which exceeds capacity.

A simple improvement (often called *capacity restraint*) is to only assign a fraction $1/n$ of a given O-D volume and run the loop n times instead. By increasing n, the level of asymmetry can be suppressed to any given threshold. Also, alternative routes will be used. The third problem, however, may not go away. Transferred to the real traffic network a volume above capacity for a given link results in a traffic volume *at* capacity for the link and increasing spill-back into the feeding links. The spill-back will increase the travel-time for the feeding links rendering the initial assumption about their assigned volumes obsolete.

4.1.4 Quasi-dynamic assignment

The previous section has shown that the static equilibrium assignment has clear disadvantages. Note that traditionally its use was still justified. On the one hand, origin-destination matrices were averaged over large periods of time (e.g. a whole day or a whole week). Therefore, the traffic volumes seldom exceeded capacities since rush hours were smeared out over the whole day. Also, most dynamic approaches were known to be computationally costly and generally not feasible for realistic problem sizes until several years ago.

A major improvement of traffic assignment can be achieved by using a time-dependent origin-destination matrix and by providing a better feedback by links above capacity. The term *quasi-dynamic* for this type of assignment was chosen because most models still lack the travel-time feedback from a simulation.

The model SATURN [124] reduces those O-D volumes going through crowded links by restricting the volumes to capacity. The remaining "suppressed" volume is saved to a file to be interactively inspected by a user. The simulation DNA (= <u>D</u>ynamic <u>N</u>etwork <u>A</u>ssignment, see [118]) allows to feed back the suppressed volume automatically and begin a new planning round with partial O-D streams starting at the intersections at which the capacities were exceeded.

The model DYNEMO (= <u>Dy</u>namic <u>Ne</u>t <u>Mo</u>del, see [108]) was one of the first to use the results of a simulation as feedback for link travel-times. In fixed time intervals the routes of all vehicles in the system are updated according to the current traffic state. DYNEMO is not truly dynamic, however, since the O-D matrix itself has to be time-independent. Only the route-selection is time-dependent.

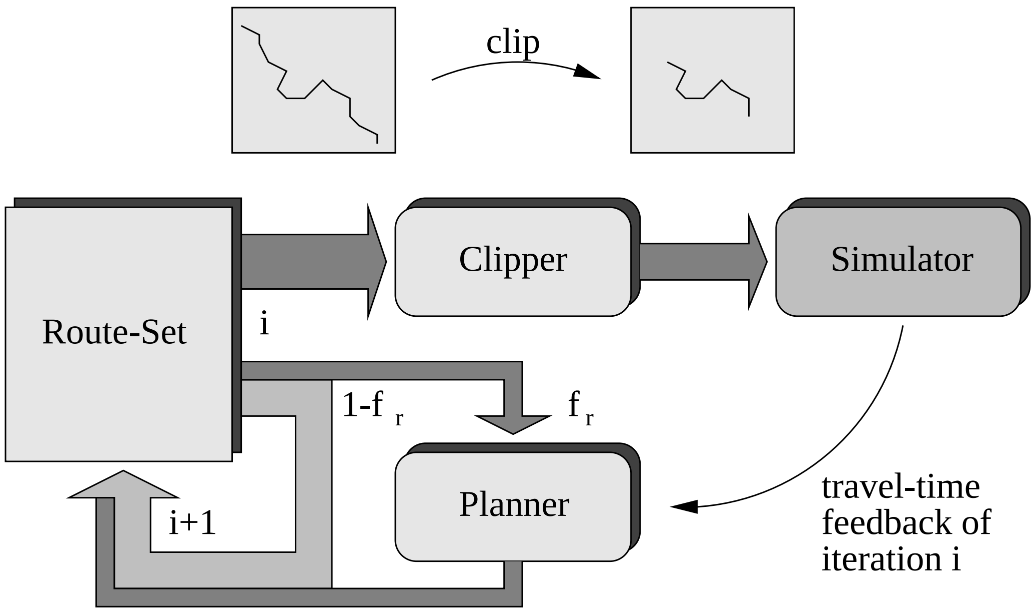

Figure 4.2: *Iterative assignment with simulation feedback* — For iteration $i+1$ the fraction f_r of the previous route-set is re-planned by the router using link travel-times of iteration i. The remaining fraction $1 - f_r$ is simply reused. Before the route-set is fed into the simulation it is clipped to the boundaries of the study-area.

The model DRUM (= Dynamische Routensuche und Umlegung, see [110]) uses an iterative approach. Starting with a given set of time-dependent link travel-times it does a time-dependent assignment of routes. A stochastic variation of the link travel-times permits the generation of alternative routes for origin destination pairs. The time-dependent demand for each link is converted into a new link travel-time for the next iteration of the algorithm. DRUM does not consider spill backs caused by V/C-ratios larger than one. Therefore, it is questionable if the iteration process will yield satisfactory results.

4.2 Iterative assignment with simulation feedback

Nagel and Barrett use iterative re-planning for a case-study within the context of the TRANSIMS project [79]. In the remaining portion of this chapter we would like to concentrate on this iterative approach in which the micro-simulation and the planner modules are executed in turns to generate a self-consistent route-set. The planner module uses the link travel-times of the previous simulation run to re-plan a certain fraction of the previous route-set (refer to Figure 4.2). This repeated planning process is to mimic the decision process observed for a fleet of human drivers: most of the drivers will rely on the same routes they have used lately. The remaining fraction will try to find a new route according to the latest information about the performance of the street network. Since both the demand coded in the origin-destination matrix *and* the travel-time feedback are dynamic, the iterative adaption with simulation feedback can be regarded as a *truly dynamic assignment*.

4.2.1 Restricted route choice

A slightly different iterative process was tried for the transit traffic of the Autobahn network of Nordrhein Westfalen [76]. For each origin-destination pair, the ten shortest paths were were computed *beforehand* using the Euclidean distance as cost function. Each driver retrieved a copy of the alternatives corresponding to his origin and destination. The iteration was executed as follows:

preparation phase Each driver chooses one of his ten fastest paths. During the first ten iterations he tries all ten alternatives sequentially. Afterwards he usually chooses the path that has proven to be the fastest. With a small probability $p_{other} = 0.05$ he tests one of the other routes.

execution phase The micro-simulation is started with all drivers executing their current routes. The travel-time of the current route is stored for later comparison.

After 15 iterations the variation of consecutive iterations is small. Surprisingly, the network throughput actually decreases through re-planning. In contrast to the results obtained in the Dallas - Fort Worth test-bed and in the next section. One reason may be that the choice of 10 options is too restrictive, especially since the options were computed beforehand and have therefore little relation to the actual traffic pattern generated by the simulation. It seems very reasonable, therefore, to allow all drivers to completely reconsider their route choice once in a while without any restriction whatsoever.

A similar approach is pursued by Gawron [40] for the iterative adaption of route-plans in a simple artificial network. Drivers choose from a limited number of predefined paths with a certain probability distribution. After each iteration both the subjective costs of routes and the probability distribution are updated according to carefully selected functions. The iterative process reaches an equilibrium after approx. 100 iterations.

4.2.2 Corridor simulations

Another way to restrict the adaptive planning to a simple case is to consider all routes between several origin and *one* destination. This geometry is called a *corridor simulation*. Emmerink et al [27] investigated the influence of pre-trip and en-route information in an iterative day-to-day dynamic framework. Their test-bed was a simple artificial network consisting of 18 nodes and 28 links with a single origin and a single destination. Hu and Mahmassani [45] used an artificial network consisting of 50 nodes and 144 links with 32 origins and 10 destinations. The geometry of the network — one fast highway with parallel slower streets — also resembled a corridor.

4.2.3 Selecting the departure time

So far, the only variable in route planning process was the route choice itself. Although eventually TRANSIMS will include the choice of departure time, currently, this additional option is not being considered. One major obstacle lies in defining a reasonable departure interval and providing a utility function rating (punishing) the time difference between the actual departure time and the time that was given by the activity triggering the departure.

In [18] and [47, 48] simple commuting networks were used to investigate how iterative re-planning influences the choice of departure time. The routes of all drivers were fixed leading from a residential area through a heavily congested corridor to the office area. Each driver tried to arrive at his office at a fixed time and leave his home *just in time*.

4.2.4 Finding the shortest path

The main component of the router module is the shortest path algorithm. For a given source, departure time, and destination it tries to find a route which imposes the least "costs" on the driver. The usual approach is to define time-dependent link costs $c_j^{trav}(i)$ (also called *weights*) for each link j and each time-bin i. As mentioned before, costs can be a combination of travel-time, distance, financial costs (such as tolls). They may also include preferences for certain street types (such as scenic roads) or prohibitions (trucks through residential areas). Therefore, a better expression for the shortest route would be the *minimum cost route*.

In principle, each driver has an individual *subjective view* of the costs of a link depending on his financial and social background. It is possible that two drivers choose different routes because their link costs vary considerably, and neither of the routes has to be the shortest route (measured as the sum of the Euclidean lengths of the links) or fastest route (measured as the sum of travel times on the links). For the work presented here, however, we use a simplified approach by regarding the whole fleet of drivers as homogeneous with the *link travel time* as the only cost factor. This may cause certain instabilities because minor changes in link travel-times may result in many drivers modifying their routes collectively (see 4.2.6). Nagel [79] reports a positive effect of a 30%-noise that was added to the original cost function to create an artificial individual view of the network.

Once the link costs are defined there is a wide variety of algorithms available to compute an exact solution of the minimum cost route. Among those Dijkstra's algorithm (see i.e. [2]) is probably the most commonly used one. Variations of the original algorithm try to improve the run-time performance depending on specific characteristics of the cost functions and the underlying graph (see [19]). In the case of a street network which is basically planar and the number of edges E is approximately as large as the number of nodes N (except for a constant factor), the most Dijkstra's algorithm has a complexity of $O(N \log N)$ for the computation of all minimum cost paths from one source to all other destinations in the network. This version will be used in this chapter within the router module and in Chapter 5 for online re-planning.

Another way to route vehicles is to use heuristic approaches mimicking a driver's decision process. Within the TRANSIMS project there have been attempts to implement an algorithm called *likely-path* using the geometric location of the destination to select the next link in a route (see [4]). Starting at the source, the likelihood of the next link to be chosen depends on the angle between the link and the destination: the smaller the angle, the larger the likelihood. Unfortunately, this algorithms did not assign enough traffic to highways, since a highway connection usually does not follow a shortest Euclidean path: In many situations drivers have to move *away* from the destination to access a highway ramp. For this and other reasons the *likely path* approach was abandoned later on.

4.2.5 Equilibrium

In principle, it is possible to formulate the route-set for a given O-D matrix and link performance function as the solution of an optimization problem: all routes are to be selected in such a way that the overall costs are minimized. This minimum has to exist since the sum of all independent minimum cost paths is a lower bound. The solution is called the *system optimum* since the traffic is in a system-wide cost minimum.

Unfortunately, for any realistic network size the linear program resulting from the optimization problem is far too complex to solve on today's computers. Moreover, in a real-world system, this type of optimum is very hard to achieve anyway, because all drivers would have to have knowledge about the decisions of all others and some of them may have to chose routes which are longer than those they would have chosen independently. Therefore, a more realistic definition is the so-called *Nash-equilibrium* in which no driver can reduce the cost of his route unilaterally by selecting a new route.

Note that for both equilibria the sum of all costs is well-defined but that there may be more than one route configuration yielding these costs. For very simple networks, both equilibria may be equal, so that after a while the system optimum is reached although all drivers try to optimize their routes independently. Braess [14] reports a case in which the construction of a new street destroys the symmetry between both equilibria resulting in an overall worse network performance. Therefore, this phenomenon is often called the *Braess-paradox*, although in fact it is only the clash between two different optimization criteria.

4.2.6 Stability

During the iterative process routes are re-planned based upon the link travel-times of the previous iteration which are subject to stochastic fluctuations. Re-running the simulation with the same route-set but using

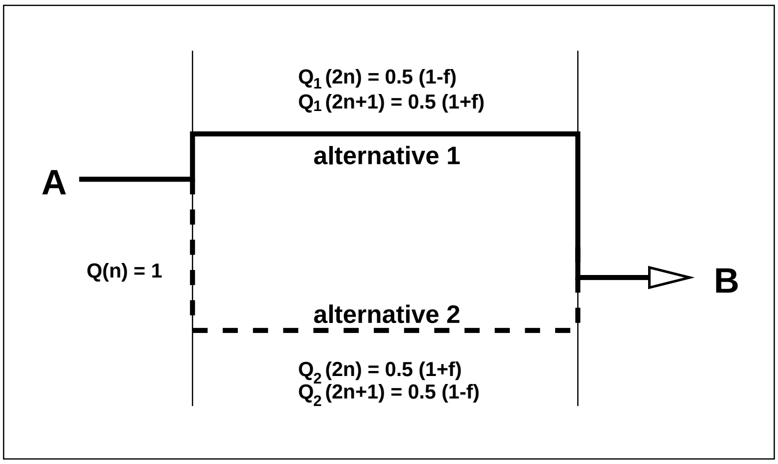

Figure 4.3: *Oscillations between two alternative routes* — There are two alternative routes between source A and B. In iteration n, if the link travel-times are slightly different, the router will move the fraction f_r of the flow $Q_1(n)$ going from A to B to the alternative flow $Q_2(n+1)$. In the next iteration this process is reversed resulting in periodic fluctuations.

different seeds for the random generators also yields different traffic patterns: links which are operated very close to capacity may cause spill-backs if there are some additional vehicles that arrive earlier than in other runs. In a domino effect spill-backs may have an impact on large areas of the network. Consequently, for the router the view of the traffic differs slightly in each iteration resulting in a possible shift of routes, even though the iteration may already have relaxed. Consider the following situation (see Figure 4.3): for a given section of a route between locations A and B there are two similar routings, using either the solid or the dotted paths. Even if after a given iteration n the flow $Q(n) = 1$ between A and B is equally distributed between alternatives 1 and 2, the stochasticity of the micro-simulation will disturb the feedback travel-times for the links. As a consequence, one of the alternatives will be shorter than the other so that a fraction f_r of routes is moved from the longer to the shorter alternative. In the next iteration this effect is reversed because of the usually linearly increasing link travel-time with respect to link-density. Therefore, the routes assigned to the alternatives are subject to an oscillation of relative amplitude of at least f_r.

4.3 Using PAMINA for iterative assignment

At this point we will use the micro-simulator PAMINA to iteratively adapt a route-set consisting of all routes going through an 8x8 [km] area inside Dallas. The map includes all street types such as highways, arterials, and local streets in residential areas (see Table B.1). For the experiments, the initial route-set was derived from trip-table provided by the North Central Texas Council of Governments (NCTCOG). Refer to [79] for a more specific description of this data.

4.3.1 Router and feedback data

In this section we will outline the router module used in the iteration runs and the nature of the data used as feedback from the micro-simulation to the planner.

During the micro-simulation each link of the street network is constantly monitored. Every 10 time-steps the current velocities of all vehicles on each link are accumulated. Every $t_b = 900$ time-steps (15 minutes) the values are converted to link travel-times and written to a file. During grid-lock on a link the sum of velocities is zero which would result in an infinite travel-time. Therefore, the travel-time is limited to the time a vehicle would spend on the link if it were going at 1/100th of the free speed of the link. If the link is vacant, the travel-time is set to be exactly free speed.

The TRANSIMS iteration uses two files for its feedback process. The first one is equivalent to the one described above containing average link travel-times. The data, however, is retrieved differently: vehicles leaving links trigger events which are collected in a statistics file. After the simulation a post processor aggregates all travel-times into time-bins. There are two reasons for a vehicle count of zero: (a) no vehicle left the link, because the there was no traffic on the link at all, or (b) no vehicle *could* leave the link because a link up-stream was completely grid-locked. A second statistics file containing the link occupancies is used to decide which case applies.

The planner reads the link travel-times $t(i)$ and assigns them to 15-minute time-bins indexed by i. For each route of the input route-set (e.g. the route-set of the previous iteration) the planner first determines whether it is re-planned or simply written to the output route-set without any modification.

In case the route is to be re-planned the planner runs a time-dependent Dijkstra shortest-path using the previously collected link travel-times. Two slightly different versions of the planner are used. Let T be the distance of a node which is to be labelled next in Dijkstra's algorithm. It corresponds to the wall-clock time at which a vehicle following the shortest path would arrive at the given node.

RP1 is the route-planner which was also used in the iterative reference run of the TRANSIMS project. It uses the time-bin $i = \lfloor (T + t_b)/t_b \rfloor$ which corresponds to a "look-ahead" time-shift of t_b. This improves the relaxation of the TRANSIMS iteration by anticipating congestion on links [79]. For time-bins after 12:00 pm the planner assumes free speeds for all links again. Turning prohibitions at intersections are handled by preventing the shortest path algorithm from proceeding into any prohibited direction. Note that this approach may exclude complicated paths that visit a node more than once to circumvent prohibitions. Since the number of affected plans is very small, this error can be neglected.

RP2 is the research version of the planner. Since its source code is more easily accessible it is used for most of the runs. In contrast to RP1, the time-bin is computed as $i = \lfloor T/t_b \rfloor$. Also, in contrast to RP1 the research version completely ignores left turn prohibitions. For time-bins after 12:00 pm, the travel-times of time-bin 11:45 am are used.

In both cases using $t(i)$ can only be regarded as an estimate, since it is averaged over a period of 15 minutes. RP1 tends to take time-bins which may be too far in the future, whereas RP2 does the opposite: it only uses data which is slightly outdated. A more exact approach would be to use T to compute the first bin i as RP1 does, but then to proceed as sketched in Section B.3. The effect of using $t_{eff}(i)$ instead of $t(i)$ is that we now average the travel-time over all bins that the vehicle "visited" while it was on the link. The first and the last bins are only weighted with the respective fraction that they were used.

4.3.2 Route-set conversion

Before the route-set can be used as input for the simulation, the routes extending over the whole planning area of Dallas - Fort Worth have to be clipped to the simulation area as follows: For each route of the original route plan, we determine the first link inside the study-area. Check if the estimate of the entrance time into that link lies within the simulation-time window 5:00 am to noon. If so, the beginning of the route is clipped up to the first link inside the area. The entrance time into the study-area is used as the new departure time. If the route leaves the study-area again, the remaining portion of the route will also be clipped. The new scheduled travel-time corresponds to the time spent inside the study-area. After conversion, all routes are sorted according to their new departure times.

For the travel-time estimates outside the study-area the free speeds of the links are used. This results in differences to another micro-simulation (see VSS in 4.4.1) used for the same test-bed which actually simulates traffic outside the study-area.

Also note that in principal there may be routes entering and leaving the simulation area more than once. The current version of the micro-simulator does not handle these routes correctly, but treats them as though they had only one entrance point into the study-area. The fraction of affected routes can be assumed to be small due to the convex shape of the study-area.

4.3.3 Iteration parameters

The re-planning process as it is used within the current context is defined by a small set of parameters: the choice of the *initial route-set*, the *re-planning fraction*, and the *route selection*. They will be described next.

Initial route-set

At the start of the iteration there is a choice between three different initial route-sets that were generated from the O-D matrix based on

- free speeds (called *FS*) as time-independent link-weights,
- the logical link-lengths (called *SP* for shortest path) as time-independent link-weights,
- an empty route-set (called *VD* for void) in which all routes were deactivated. In this case "re-planning" a route means: Plan the route for the first time and activate it. Once a route is activated it remains activated throughout the iteration.

Re-planning fraction

Choosing the re-planning fraction f_r is a trade-off between computational speed and stability of the iterative process. Large fractions allow fast re-planning of all routes in the initial route-set. As we will see later on, one prerequisite of relaxation is that most of the routes have been re-planned at least once. As the iteration proceeds, however, large fractions have the clear disadvantage of moving routes back and forth between similar alternatives (see [79]).

The aim of the iteration process should be either to keep the re-planning fraction small throughout the process or at least reduce the fraction more and more as the iteration approaches relaxes.

Route selection and route aging

The above-mentioned re-planning fraction does not yet define *which subset* of the previous route-set is to be re-routed. One can easily imagine a selection according to the following criteria:

- Criteria associated with the traffic network, such as routes going through a certain part area of simulation area. This is related to traffic state information that may only be available on certain types of streets (e.g. major arterials).
- Criteria associated with the individual driver such as demographic information on the driver or the origin of the trip, i.e. wealthy drivers may have better access to information.
- Criteria associated with data retrieved from the simulation such as the planned travel-time or the actual travel-time.
- Criteria associated with data retrieved from the planner such as the iteration of the last re-planning of a route.

All groups of criteria except for the last must be handled with care since they may bias the selection towards a subset of plans within the route-set. For example: selecting those routes first whose actual travel-times exhibit the largest difference to the scheduled travel time is dangerous if there are regions in the network that are only connected to the remaining part through a small number of bottlenecks. Re-planning those routes will not improve the performance since the network does not offer any real alternatives to the bottlenecks. We have, therefore, decided to use the age (denoted as "a") of a route, that is the number of elapsed iterations since it was re-planned last, as the only parameter. In particular, we chose the following versions of route selection:

random The routes are selected completely at random with the probability f_r. The planner does not distinguish between routes in any way. All routes are picked with the same probability $P_{rnd} = f_r$.

scheduled random The routes are also randomly selected, but the re-planning fraction was explicitly chosen for every iteration. We used this scheme only for the first two test-runs, since it required user interaction. The iteration of the TRANSIMS-14 run was also based upon a schedule (see Table 4.1).

linear age The likelihood P to select a route of age a is given by $P_{lin}(a) = qa$ were q is a factor that only depends on f_r and is thus a constant of the iteration.

forced reduction with random smoothing The N routes are split into two groups: those routes that have never been re-planned yet, and those that have been re-planned at least once. For n iterations we re-plan N/n plans of the first group and an additional fraction of f_r of the second group. For iterations after n only the second portion of re-planning remains. The probability in iteration i is computed as $P_{red} = f_r$ if $i > n$ or the route has already been re-planned, and $P_{red} = 1/(n + 1 - i)$ otherwise.

All of the above versions eventually lead to a stationary age distribution $f(a)$ which can be analytically predicted (see B.1). The first and the last versions result in a decreasing exponential with

$$f_{rnd}(a) = f_r(1 - f_r)^a,$$

the second one results in a normal distribution

$$f_{lin}(a) = f_r e^{ba^2}.$$

Since the linear age version tries to re-plan older routes sooner than younger ones, the age distribution is biased towards younger plans compared to the random version. This can be seen in Figure B.1 for $f_r = 0.05$.

4.3.4 Relaxation

So far we have only described how we execute the iterative process and what parameters we use to influence it. A very important aspect is how to measure the actual improvement achieved by re-planning. As we have seen in Section 3.4 the micro-simulation exhibits grid-locks whenever loops in the network links are completely filled with vehicles. One obvious measure of improvement would be a reduction of the number of grid-locks. The left-hand side of Figure 4.4 shows a first result from the iteration using random route-selection and a re-planning-fraction of $f_r = 0.01$. We see the number of vehicles in the study-area plotted against the simulation time (one curve for every tenth iteration). It is easy to detect two transitions in the development: a) Between iterations 30 and 40 the simulation stops grid-locking with the vehicle count remaining constant after a certain point in time, b) between iterations 60 and 70 the throughput of the system has improved so far that practically all vehicles are able to leave the study-area during the simulated time until noon. The remaining approx. 1000 vehicles are not grid-locked (see 4.3.5). As the iteration continues and all grid-locks have been dissolved, these curves get less and less informative since

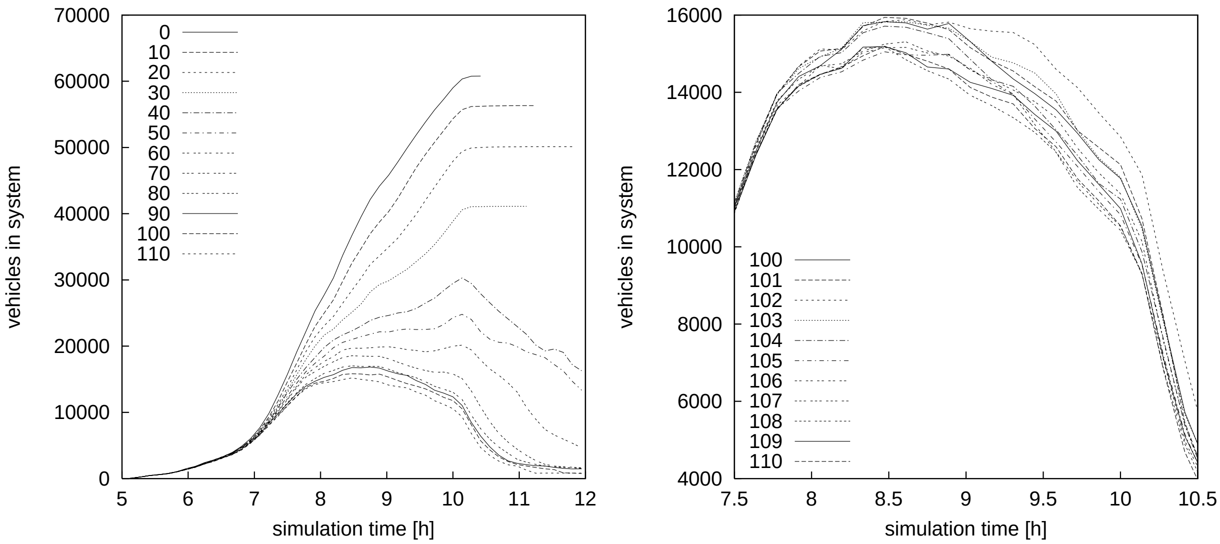

Figure 4.4: *Run 4: Number of vehicles in the study-area* — LEFT: For early iterations the number of vehicles in the study-area exhibits grid-locks (horizontal lines). As the process proceeds the grid-locks disappear until eventually most of the vehicles are able to leave the study-area. RIGHT: The last ten iterations show very little variation. Note the different scale! The number of vehicles in the study-area proves to be an impractical measure for the progress of the iteration process.

the absolute difference between simulations decreases. The right-hand side of Figure 4.4 shows the last 11 of the 110 iterations. Although there is still some improvement visible between the set of 100 through 104 and the set of 105 through 110, the change is not as drastic as in earlier iterations.

Changing the display from the number of vehicles in the study-area to the aggregated travel-time gives more insight. The left-hand side of Figure 4.5 shows a continuous decrease with some underlying noise. The improvement is mainly caused by three effects:

a) The traffic volume is increasingly distributed among all street types. The original route-set is biased towards fast routes (FS) using the network hierarchies in a heterogeneous fashion.

b) Since the capacities of the access links to the study-area are limited, there is a growing spill-back of vehicles at the boundaries as soon as demand exceeds capacity. The feedback for link travel-times does not punish those queues in the original version. In Section 4.3.5 we will introduce a correction to the feedback which takes care of queue delays.

c) The number of routes that are routed through the study-area decreases. The right-hand side of Figure 4.5 depicts the number of routes planned, inserted, and executed. After 110 iterations only 263281 of the initial 294883 routes remain reducing the traffic load by roughly 10%. This artefact will also be discussed in Section 4.3.5.

iteration	1 - 3	4 - 6	7	8 - 9	10	11	12	13 - 14	accumulated
run 1	0.20	0.10	0.05	0.05	0.02	0.02	-	-	1.09
run 2	0.20	0.10	0.05	0.05	0.02	-	-	-	1.07
TRANSIMS 14	0.10	0.10	0.10	0.05	0.05	0.05	0.05	0.02	0.99

Table 4.1: *Iteration parameters for runs with scheduled re-planning fraction* — The entries show the re-planning fraction f_r for each iteration. All of the runs listed here used planner RP1 with random re-planning selection.

Figure 4.6 shows sum travel-times for selected runs (see Table 4.2). Apparently, the iteration number does not relate to the improvement of the iteration unless the re-planning fractions were the same for all

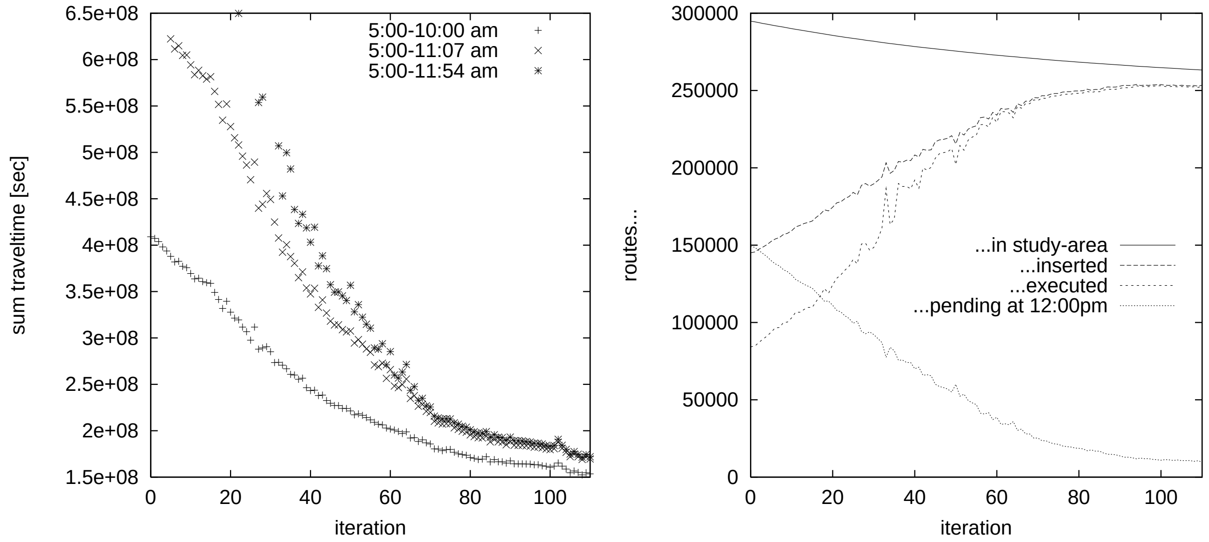

Figure 4.5: *Run 4: Sum of travel-times and executed routes* — LEFT: Accumulated travel-time tt_{acc} of all vehicles in the study-area as a function of the iteration number. It is obvious that despite the high number of iterations the value of tt_{acc} has not sufficiently relaxed. RIGHT: Number of routes routed through the study-area, those inserted into the study-area, and finally those successfully executed by noon.

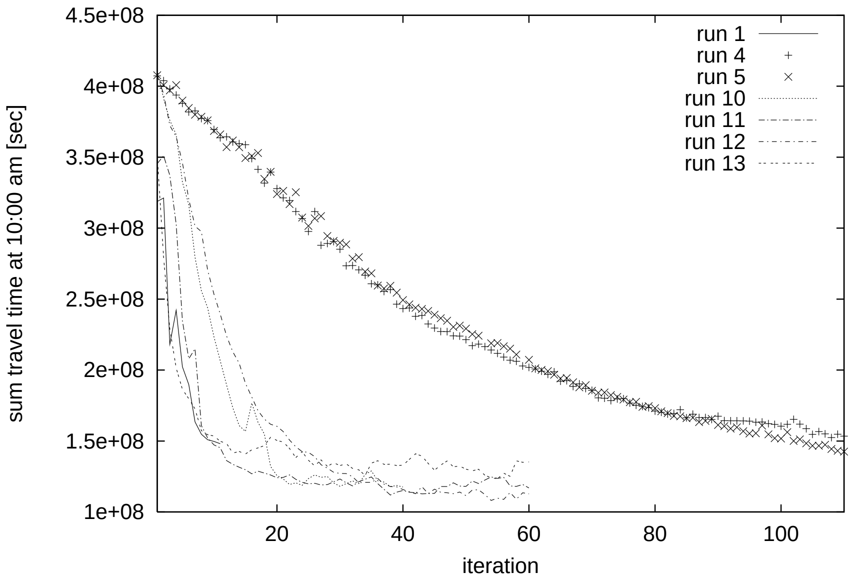

Figure 4.6: *Sum travel-time for selected runs* — The plot depicts the sum travel-time with respect to the iteration number. Run 1,3,4, and 5 are not sufficiently relaxed. Runs 10, 11, and 12 level off at around $1.2 * 10^8[sec]$, run 13 slightly higher.

run	4	5	7	8	10	11	12	13/16	14	15	17
planner	RP1	RP2	RP2	RP2	RP2	RP2	RP2	RP2	RP2	RP2	RP2
init. route-set	FS	FS	VD	SP	FS	FS	FS	FS	FS	FS	FS
iterations	110	110	60	60	20	60	60	60	80	80	60
reductions	-	-	-	-	20	-	-	-	-	-	-
fraction f_r	0.01	0.01	0.05	0.05	0.01	0.05	0.05	0.05	0.05	0.05	0.05
selection	rnd	rnd	age	age	red.	age	rnd	age	age	age	rnd
level-0-corr.	-	-	-	-	-	-	-	-	lin.	sqrt	-
queue feedb.	-	-	-	-	-	-	-	yes	-	-	yes

Table 4.2: *Parameter combinations of iteration runs* — For the initial route-set *FS* denotes *free speed*, *VD* denotes *void* and *SP* denotes *shortest path*.

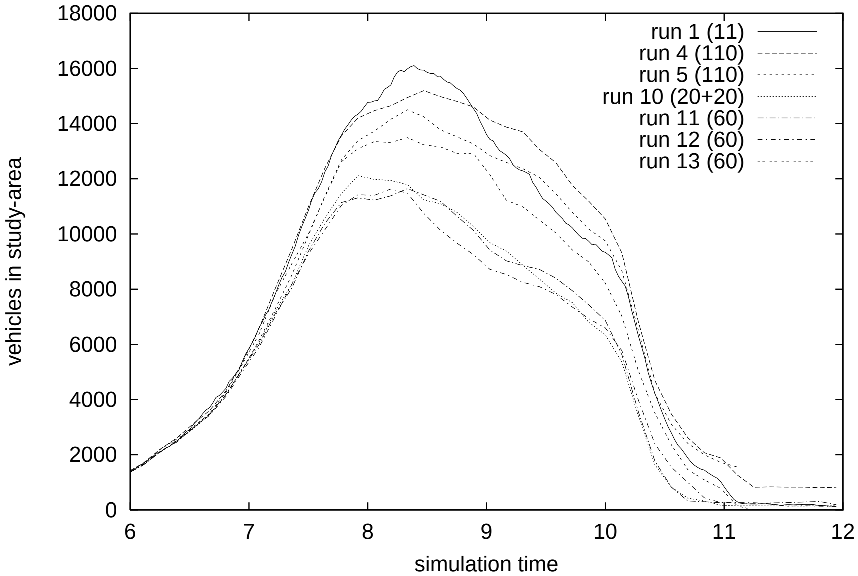

Figure 4.7: *Vehicles in study-area for selected runs* — Runs with a small accumulated re-planning fraction (runs 1, 4, and 5) still show higher peaks at rush hour than those with a high fraction (runs 10, 11, and 12). Run 13 has a higher peak for reasons discussed in 4.3.5.

runs. It does, however, give a notion for the speed of the relaxation since the running time of the router is almost independent of the re-planning fraction for small fractions[2]. Due to the slow relaxation of runs 4 and 5, they do not reach the assumed equilibrium at approx. $1.2 * 10^8$ [sec] aggregated travel time, despite the high number of iterations (e.g. 110). Figure 4.7 shows the number of vehicles in the study-area with respect to simulation-time. Although all curves exhibit a similar shape, there are distinct differences in the maximum number of vehicles. Possible explanations are that the runs have actually relaxed into different states or that the iterations 1, 4, and 5 are not sufficiently relaxed. Next, we will see that the latter is true.

Accumulated re-planning fraction

A more useful comparison can be obtained by using the *accumulated re-planning fraction* f_{acc} as the ordinate of the plot. This fraction is defined as the sum of all individual re-planning fractions up to a

[2]This is due to the large constant portion imposed by I/O. The micro-simulation itself, of course, is completely independent of the re-planning fraction.

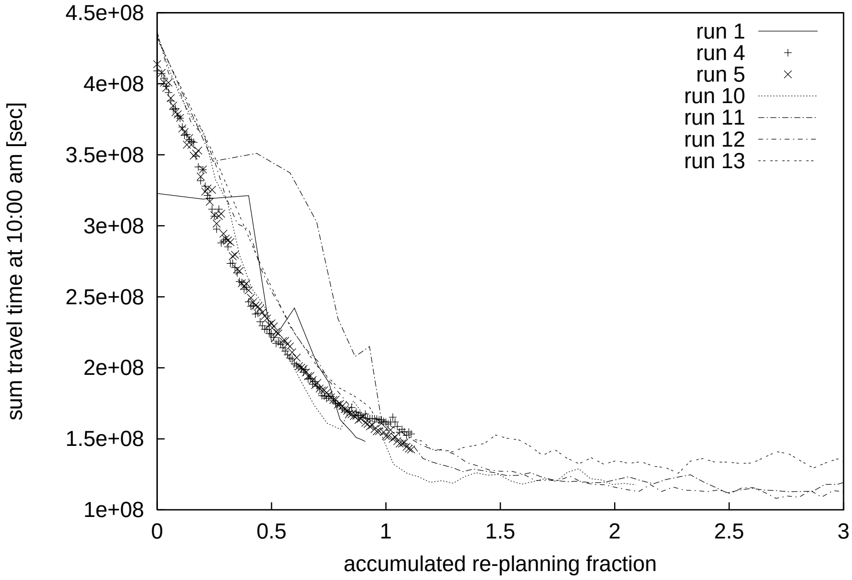

Figure 4.8: *Relaxation by accumulated re-planning fraction (all iterations)* — With respect to the accumulated re-planning fraction all curves (except for run 13) collapse into one. The differences of runs 1 and 11 are caused by exceptionally few or many grid-locks, respectively. Also see Figure 4.9.

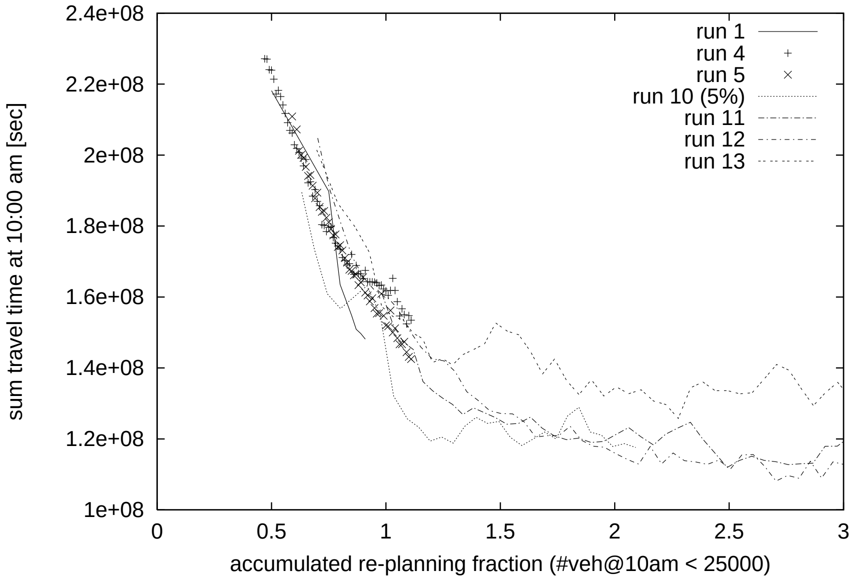

Figure 4.9: *Relaxation by accumulated re-planning fraction (non-grid-locking iterations)* — The curves were restricted to those iterations with less than 25,000 vehicles in the study-area at 10:00 am.

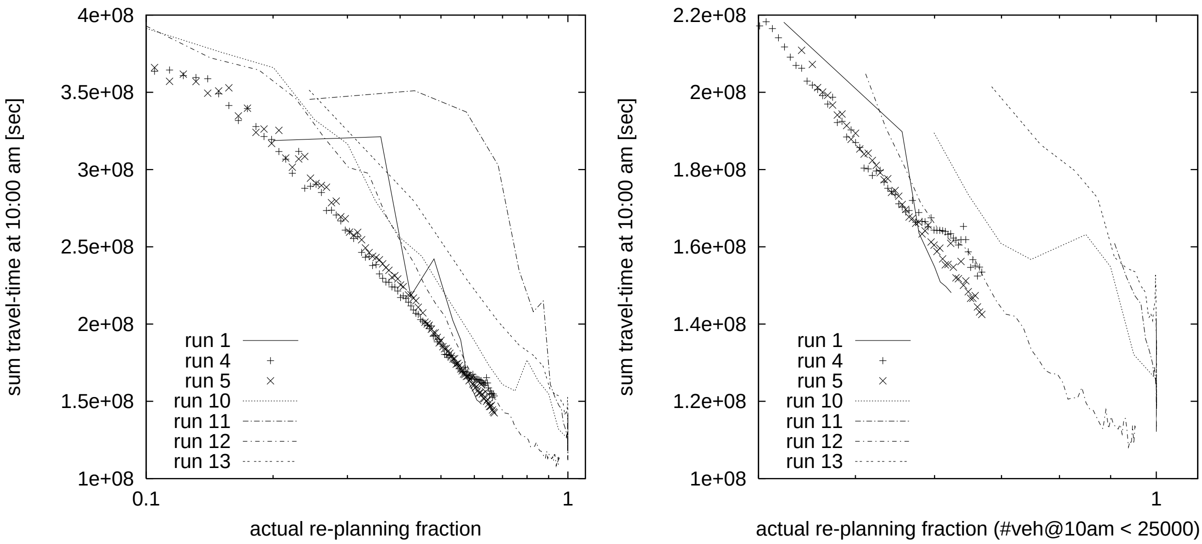

Figure 4.10: *Relaxation by actual re-planning fraction* — LEFT: Sum travel-time plotted against the actual re-planning fraction, which corresponds to the fraction of routes that have been re-planned at least once. RIGHT: Only those iterations with less than 25,000 vehicles in study-area at 10:00 am are plotted. There is a clear difference visible between runs 1, 4, 5, and 12 (all random selection) on the one hand, and runs 11 and 13 (both linear age selection) on the other hand. Run 10 seems to lie in between.

given iteration. Figure 4.8 shows the same curves as in Figure 4.6 plotted against f_{acc}. The curves for all runs coincide very well. Runs 1 and 11 exhibit some points outside the general slope. This is due to extreme grid-locking of run 11 during early iterations and very few grid-locks during the first iteration of run 1. Note that run 13 levels off at a slightly higher sum travel-time than the other runs. This will be discussed in 4.3.5.

In Figure 4.9 the similarity becomes even more obvious after removal of all iterations with vehicle counts below 25,000 at 10:00 am. The peaks of runs 1 and 11 have disappeared. Run 10 is the first one to reach a sum travel-time of approx. $1.2 * 10^8$ shortly after an accumulated re-planning fraction of one. At this point, it is the only run in which *all* routes have been re-routed at least once.

Actual re-planning fraction

In addition to the accumulated re-planning fraction we define the *actual re-planning fraction* as the fraction of plans that have been re-planned *at least once*. The left-hand side of Figure 4.10 shows the sum travel-times plotted with respect to their actual re-planning fraction. As before, the curves on the right-hand side are restricted to those iterations with vehicle counts below 25,000 at 10:00 am. The plot clearly separates the runs with random selection (runs 1, 4, 5, and 12) from those with linear age selection (runs 11 and 13). Run 10 with forced reduction selection is at first closer to the random selection, but later on approaches the linear age selection curves. The figure also shows that using the actual re-planning fraction as ordinate makes little sense as soon as the fraction approaches one: For runs 10 through 13 the points are simply stacked above one since the fraction is limited to 1 by definition.

4.3.5 Artifacts

The planner and the micro-simulation do not operate on the same scope. Since the micro-simulation only executes those portions of the routes *within* the study-area one expects phenomena which are due

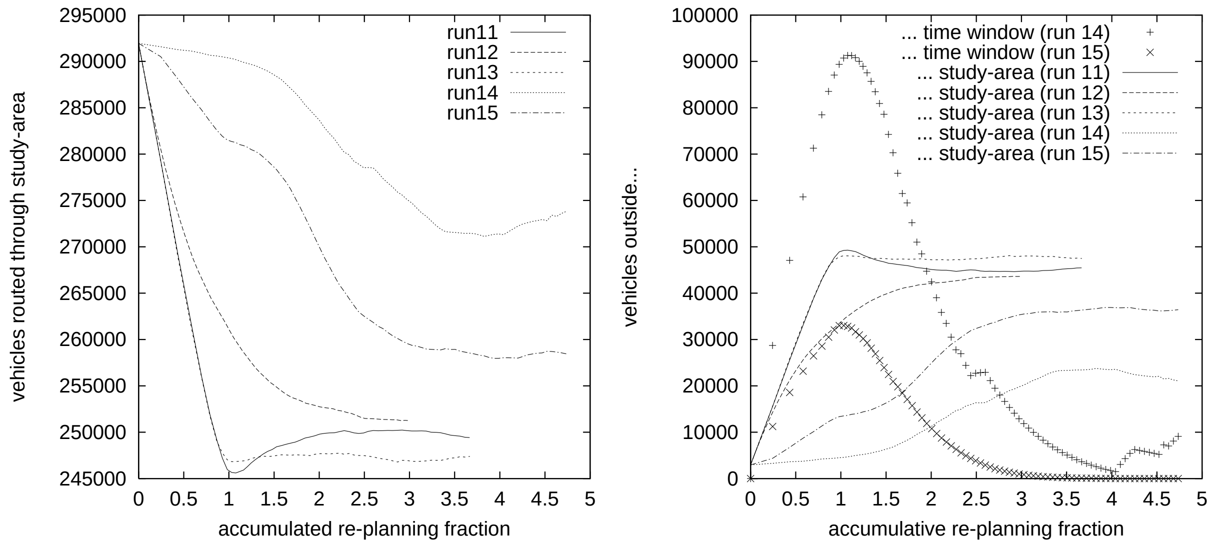

Figure 4.11: *Route loss through re-planning* — The left-hand side shows the number of routes routed through the study-area for different runs. The right-hand side shows the number of routes lost because they lie outside the time-window or outside the study-area. Note that run 14 seems to become instable again after a re-planning fraction of 4. This is due to the high number of vehicles routed through the study-area.

to boundary effects. We would like to describe two observations we made during the simulation runs and potential remedies.

Loss of Vehicles

The traffic which is generated for the study-area produces higher link-travel times than those derived by the given free speeds[3]. Therefore, the link travel-time feedback will generally discourage the planner from using links in the study-area compared to the links outside which keep their high free speeds. We differentiate between two effects:

- Routes with either source or destination inside the study-area may have a peculiar shape: instead of computing an overall fastest path, the planner will rather minimize the portion inside the study-area.
- Routes with neither source nor destination (through-traffic) inside the study-area may be re-planned such that they avoid the study-area altogether. The same effect is reported in [79] where the number of plans was reduced from 285,393 to 258,501 after an accumulated re-planning fraction of 0.95.

The left-hand side of Figure 4.11 shows the loss of routes during the iteration process. The number decreases from the original value of 294,800 to approx. 250,000. Run 11 shows a peculiar behavior: a minimum is reached slightly above an accumulated re-planning fraction of one. After that, the number of plans increases again to level off at around 254,000. This increase is due to queuing at the boundaries of the study-area which will be described at the end of this section.

We extended the router to include a so-called *level-0 correction.* Before a re-routing step for each link j (with length L_j) and each time-bin i, we converted the link travel-times $T_j^{trav}(i)$ into link-velocities

[3]Except for those time periods with very little traffic during which a measured average speed (almost speed-limit) may actually be higher than the corresponding free speed.

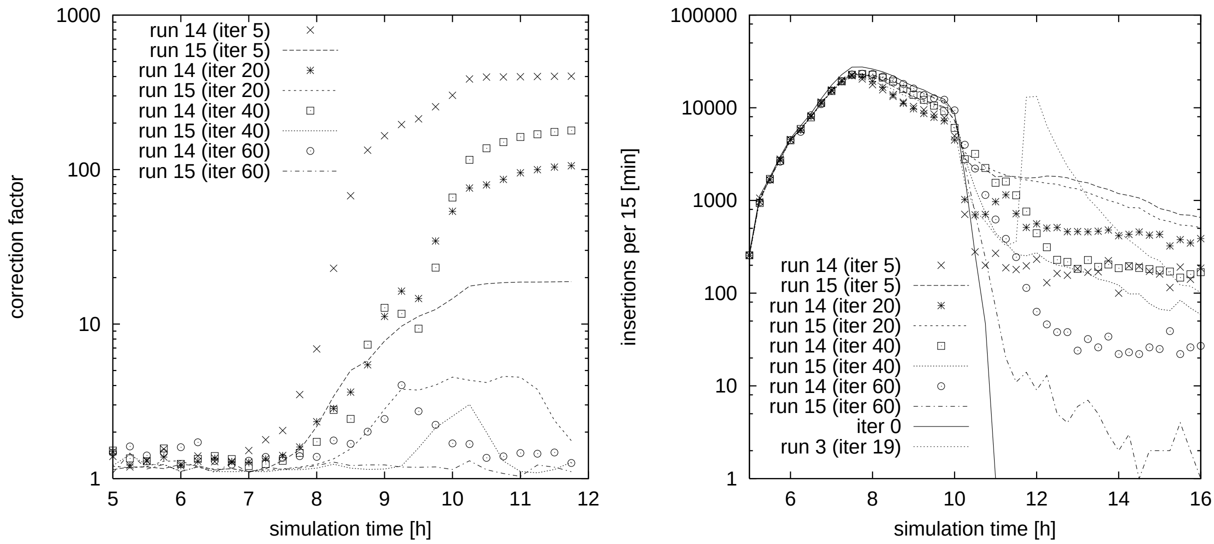

Figure 4.12: *Level-0 feedback: correction factor and insertion rates* — LEFT: The plot shows the correction factor for several iterations of runs 14 and 15. Note the logarithmic scale on the y-axis. RIGHT: Number of insertions per 15 [min] interval. The route-set of iteration 0 serves as a reference: there are no insertions after 11:00 am.

$v_j(i) = L_j/T_j^{trav}(i)$. Then, we averaged the ratio $v_j^{free}/v_j(i)$ for all links resulting in a single factor

$$c(i) = \frac{\sum_j L_j v_j^{free}/v_j(i)}{\sum_j L_j}$$

for each time-bin. This factor itself (in run 14) or its square-root (in run 15) was multiplied to all link travel-times *outside* the study-area. Now, looking at the whole planning area from far away, the study-area should no longer look worse than the remaining portion. The left-hand side of Figure 4.12 depicts the correction factor $c(i)$ for different iterations. For early iterations, the factor increases considerably due to the large number of grid-locks during rush hour. As grid-locks vanish, the correction factor diminishes, too.

Run 14 with activated level-0 correction (see Figure 4.11) loses less than half of the cars compared the other runs. Instead of losing routes with respect to space, we start losing routes with respect to time. The right-hand side of Figure 4.12 depicts the number of vehicles entering the study-area for each 15 minute bin. Compared to the original curve of iteration zero, early iterations show a large number of vehicles scheduled to be inserted after 12:00 pm. As the iterative process continues the shift to late insertions times decreases, but still after 60 iterations there are measurable counts outside the simulation period. This figure also shows an artifact of the router RP1: an earlier run with level-0 feedback exhibits a steep peak of insertions at noon. This is due to the fact that RP1 uses the free flow link travel-times for all time-bins after 12:00. All routes that have been started before noon but have not reached the study-area due to strongly increased travel-times before noon are suddenly accelerated and reach the study-area within a very short time-interval. For that reason all other level-0 feedback runs were executed with router RP2.

For run 15 the loss with respect to time is not as strong as in run 14. On the other hand, there are fewer plans remaining inside the study-area, too.

Figure 4.13 gives more insight into the stability of the route-sets obtained from the level-0 feedback. Run 15 still exhibits strong fluctuations in the number of routes that were successfully executed until noon. This, of course, is not a result of the level-0 feedback as such, but simply due to the high number of vehicles that are still routed through the study-area. The load is about 25,000 vehicles higher than in run 13 approaching the maximum capacity of the study-area.

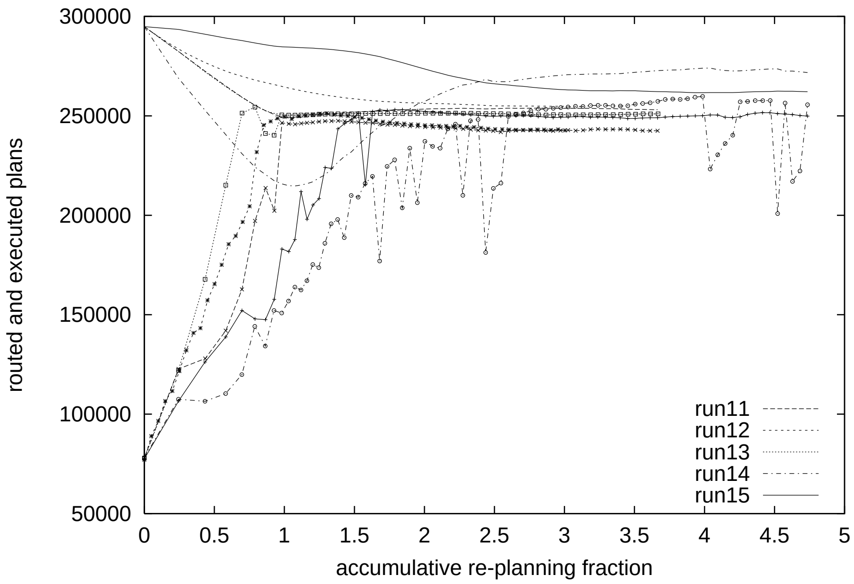

Figure 4.13: *Level-0 feedback: routed and executed vehicles* — Curves without points denoted the number of routes through the study-area. Curves with points shows the corresponding number of executed routes. Run 14 exhibits a very instable route execution compared to all other runs. Even after an accumulated re-planning fraction of 3.0 there are still iterations with partial grid-locks.

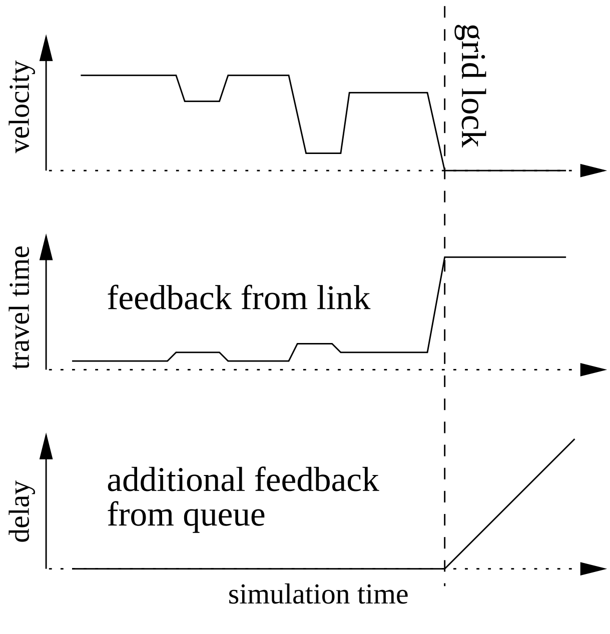

Figure 4.14: *Function of queue feedback* — After a grid-lock (top) occurs on one of the links, the normal travel-time feedback yields a time-independent penalty (center). The queue-feedback adds an additional penalty which increases with the average waiting time of all vehicles pending at the entrance of the link.

Queue feedback

The vehicle insertion of the original route-set (see iteration 0 on the right-hand side of Figure 4.12) is mainly restricted to the time-period between 5:00 am and 10:00 am. One would expect that due to the

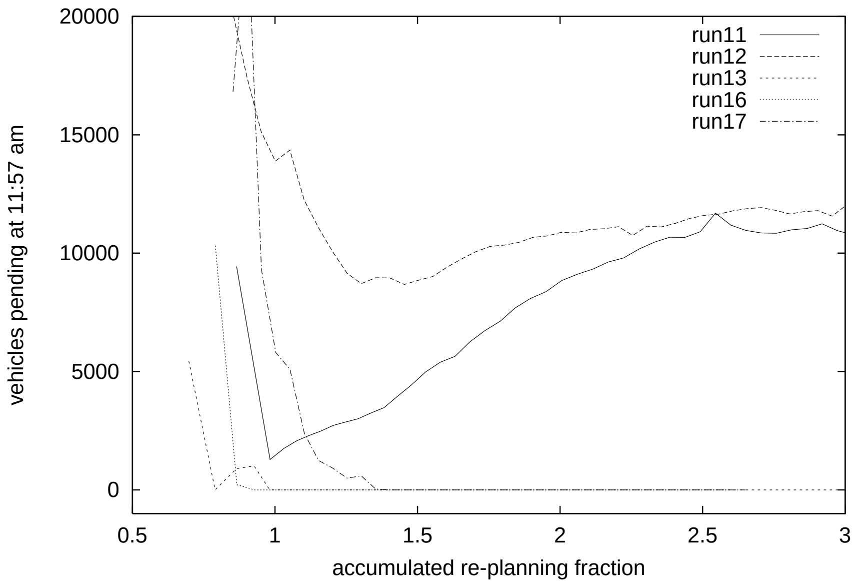

Figure 4.15: *Vehicles pending at 11:57 am* — In contrast to runs 11 and 12, the other runs 13, 16, and 17 with queue feedback do not exhibit any pending vehicles anymore.

rather small spatial extent of the study-area all vehicles should be able to exit by around 11:00 am once the grid-locks have vanished. Surprisingly, this is not the case. Figure 4.15 shows the number of vehicles pending at 11:57 am. As seen before for run 11 in Figure 4.11, here runs 11 and 12 have a minimum shortly after accumulated re-planning fraction 1.0. At the end of the iteration both have more than 10,000 vehicles waiting in queues by noon. The queuing of vehicles at the boundaries to the study-area is mainly caused by insufficient travel-time feedback from the micro-simulation (see Figure 4.14). As long as no grid-lock occurs on an feeding link, the travel-time can be well estimated by the average-velocity. As soon as all vehicles on a link are stalled, the travel-time — at least theoretically — should be infinite, but is actually computationally bounded by a very large (= unattractive) value. The value is constant and independent from the duration of the grid-lock. The overall travel-time, however, increases since the waiting time (=delay) in the queue has to be included.

For run 13 we introduced delay counters for all sources of the study-area. At each time-step we added the number of vehicles currently queued as delay to the counter. For each vehicle that was inserted and removed from queue we subtracted the number of time-steps it had been waiting in the queue. Each 900 seconds (the usual interval at which the travel-times were collected) we added the average waiting time (= delay counter divided by the number of vehicles queued) to the travel-time of the link associated with a queue. The result can be seen in Figure 4.15. Run 13 does not show any queued vehicles after an accumulated re-planning fraction of 1.0.

4.3.6 Reproducibility of iteration results

The parameter combination of run 11 was reused to produce the results of run 16. The only differences between the two runs are the random seeds used in the simulator and in the planner. Figure 4.16 shows the curves for sum travel-time and routes/executed routes. The similarity of the plots is very good.

Moreover we compared runs 7, 8, and 11 which only differ by their initial route-set and their random seeds. Figure 4.16 shows the similarity of the results. As expected, the choice of the initial route-set only makes a difference for early stages of the iteration. After 40 iterations most of the routes have been re-planned at least once anyway so that the "knowledge" about the initial route-set is lost. However, the

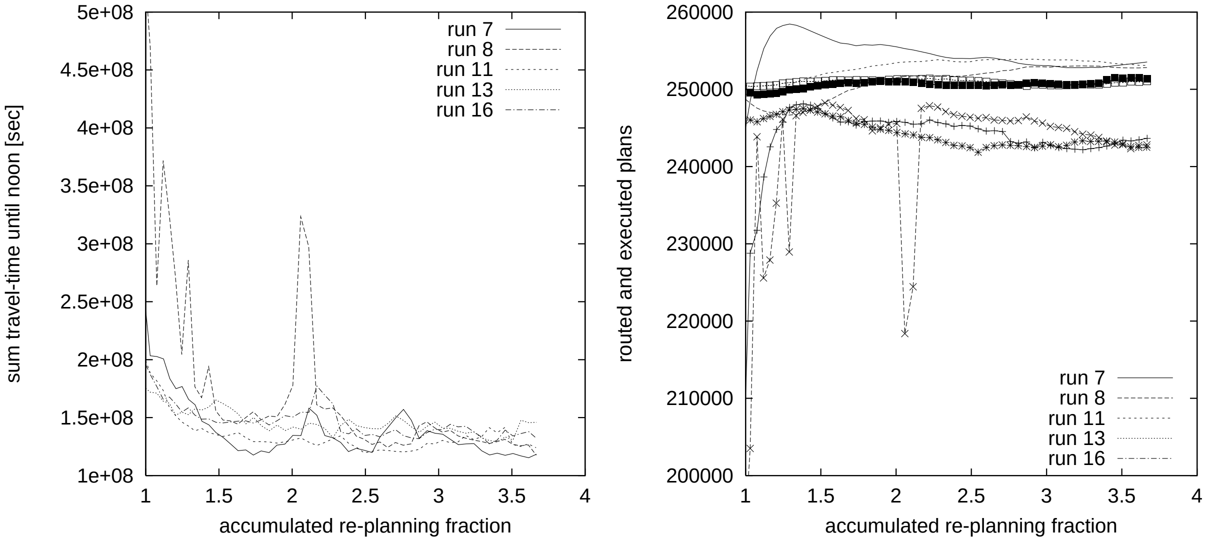

Figure 4.16: *Reproducibility of results.* — Runs 13 and 16 share the same iteration parameters. They only differ by the seed of the random generator. Runs 7, 8, and 11 only differ by the initial route-sets and the random seeds. Especially the right-hand side shows the good reproducibility.

feature	TRANSIMS	PAMINA	VSS
simulation area	study-area	study-area	Dallas/Ft.Worth
multi-lane CA	yes	yes	no
lane-change	symmetric	asymmetric	n/a
lane-change conflict	even-odd TS	L to R priority	n/a
speed limit	yes	yes	no
v_{max}	3	4	5
p_{decel}	0.2	0.3	0.25
approach behavior	yes	no	no
traffic lights	exact	average	random
turning pockets	yes	no	no
intersection queues	yes	no	no
lane interference	yes	no	no
stop/yield signs	yes	no	no
vehicle loss	yes	no	no
pre-loading	yes	no	n/a
pre-processing $[min]$	120	6-8	12
execution $[min]$	330	30-35	15-20
post-processing $[min]$	60	-	-

Table 4.3: *Overview of simulator features active in case-study comparison* — This table refers to PAMINA III. For an overview of the versions of PAMINA see Table 3.1.

initial route-set *SP* of run 8 seems to be less stable than the other two: after an accumulative re-planning fraction of two it still produces two iterations with heavy grid-locks.

4.4 Example: TRANSIMS case-study

In Chapter 3 we have seen that PAMINA is able to reproduce reasonable and intuitive results for a simple simulation set-up. All the runs were done without any prior validation or tuning of simulation parameters. The question is now whether such a simple approach can yield any useful results especially in comparison to other micro-simulations. Within the TRANSIMS project we had the opportunity to compare three simulators operating at different levels of fidelity. The first one is the TRANSIMS simulator itself representing the highest level of fidelity. Next in line is PAMINA which simplifies the approach behavior to intersections and the handling of intersections themselves. PAMINA is about 15 times faster than the TRANSIMS simulator so that more intensive investigations were possible. The third simulator VSS (= Very Simple Simulator, [114]) was implemented by P. Simon. It manages to use single lane CA links for the whole network by scaling the number of routes appropriately. This simulator is another order of magnitude faster than PAMINA. VSS was used to execute routes in the planning network instead of clipping them to the study-area[4]. We would like to describe the two other simulators in more detail.

TRANSIMS Simulator

TRANSIMS is like PAMINA a microscopic traffic simulator using the symmetric multi-lane extension of the Nagel/Schreckenberg CA model for its traffic links. The approach behavior close to intersections, however, is different: A given destination link can usually be reached by a subset of incoming lanes only. At about 70 grid cells prior to the intersection, vehicles start to move to the correct lane that will take them to their respective destination link. The closer they get to the intersection, the more "urgent" the incentive to change lanes becomes. Vehicles that are not able to reach a lane corresponding to their destination will go straight through the intersection. These "lost vehicles" [79] will be removed from the network at the next upcoming intersection. As a result the number of fully executed trips may actually be smaller than the number of routes in a plan.

TRANSIMS differentiates between unsignalized (with stop-signs or yield-signs) and signalized intersections with a detailed direction-dependent phasing scheme. In all cases vehicles are required to check for interfering traffic on oncoming or destination lanes. This is done by requiring a certain number of sites close to the destination site to be free from other vehicles. For a detailed description of the flow characteristics of the TRANSIMS simulator see [86].

The TRANSIMS pre-processor that converts the original route-plan into a format which is understood by the simulator allows to *pre-load* the traffic network for a given simulation start-time. All vehicles that have a departure time before the simulation start-time are inserted at their *scheduled* position using the free speeds of all links. That way the loading phase during which there is a lack of vehicles in the system can be shortened. Within the context of the comparisons presented here, this additional feature does not play a role, since the simulation was started at 5:00 am. At this time the number of vehicles is so small that any loading effects can be neglected.

Very Simple Simulator (VSS)

Another way to reduce the complexity of the micro-simulator is to replace the computationally expensive multi-lane links by single-lane links. In VSS the ratio between the flow 7800 [veh/hour] of the maximum throughput link (4 lanes) and the flow 1800 [veh/hour] on a standard single-lane link is used as a scaling factor. This factor defines what fraction of the original route-plan is actually fed into the simulation.

At intersections the inflow into destination links is defined by a transition probability p_T between 0 and 1 which is determined from the individual throughput of the respective destination link. $p_T = 1$ corresponds to maximum throughput for the area under consideration. Since the link with the highest capacity is a

[4] In principle, all three simulators are capable of simulating the whole Dallas - Forth Worth network. One reason not to use the medium and high fidelity models was the insufficient resolution of the street network outside the study-area.

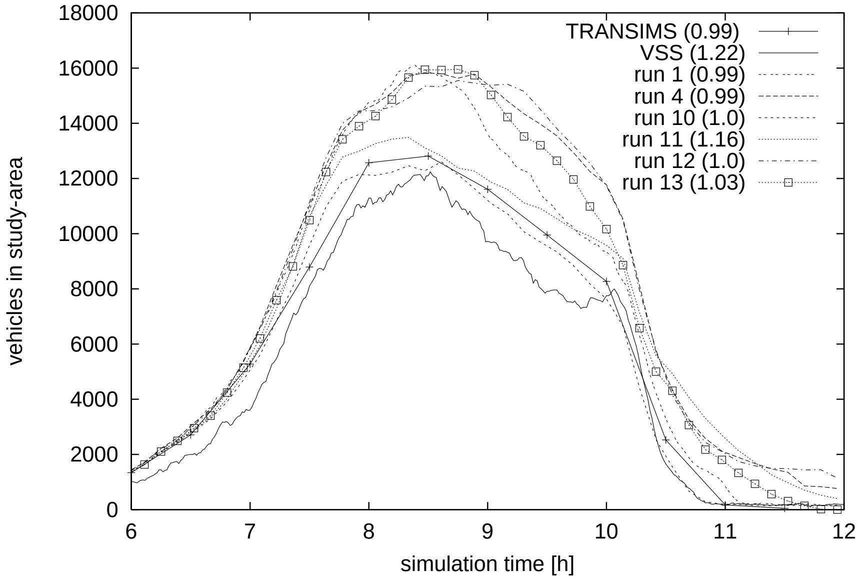

Figure 4.17: *Case-study test-bed: vehicles in study-area with* $f_{acc} \approx 1.0$ — The iteration numbers of the PAMINA runs were chosen to be as close as possible to the accumulated re-planning fraction of the TRANSIMS run ($f_{acc} = 0.99$). For run 11 we had to take $f_{acc} = 1.16$ since all earlier runs were still slightly grid-locked.

four-lane highway, a probability of approximately $p_T \approx 0.25$ corresponds to a single-lane freeway. In normal links with traffic controls at their end this value may be reduced even more. Note that the relationship between p_T and the flow is not linear (see [114]).

In contrast to the setup of the previous two micro-simulators, VSS uses the routes as they are generated by the planner and *not* in their representation clipped to the study-area. This generates slightly different results for the loading phase of the network which will be discussed later.

4.4.1 General Comparison

For a first comparison of the simulators we plotted the number of vehicles in the study-area as a function of the simulation time. The problem was that for the TRANSIMS simulation only data for one run (called *TRANSIMS-14*) was available. It is based upon a route-plan generated in fall 1996. The iterative re-planning scheme is characterized by the re-planning fractions given in Table 4.1. The number of lost vehicles was about 15,000 to 20,000. Figure 4.17 depicts the results of several runs obtained from PAMINA (see Chapter 4) and one of the results from VSS. We chose the curves of those iterations which were closest to the accumulated re-planning fraction ($f_{acc} = 0.99$) of the TRANSIMS run. Run 10 matches the TRANSIMS run best throughout the simulation period. Most other PAMINA runs exhibit a considerably higher maximum number of vehicles at 8:30 am. The VSS run is generally somewhat lower than all other curves. This effect is especially obvious for early simulation hours until 7:30 am. It is caused by the fact that VSS simulates the routes for the whole Dallas - Fort Worth area, so that traffic jams occurring *outside* the study-area reduce the number of vehicles entering the study-area [113].

In Figure 4.18 we have replaced the PAMINA curves by later iterations. For iteration 60, route-sets can be assumed to be well relaxed (see 4.3.4). The curves for runs 11 and 12 now lie below the TRANSIMS curve after 7:45 am. The curve for run 13, however, matches the TRANSIMS curve very closely, especially after 9 am. It even reproduces a little ledge at 10:30 am correctly, which is not visible at all in run 11.

Unfortunately, it remains unclear if the underlying route-plan used for the TRANSIMS run is as well relaxed as runs 10 through 13 for PAMINA. If one compares the sum travel-times, it seems very unlikely:

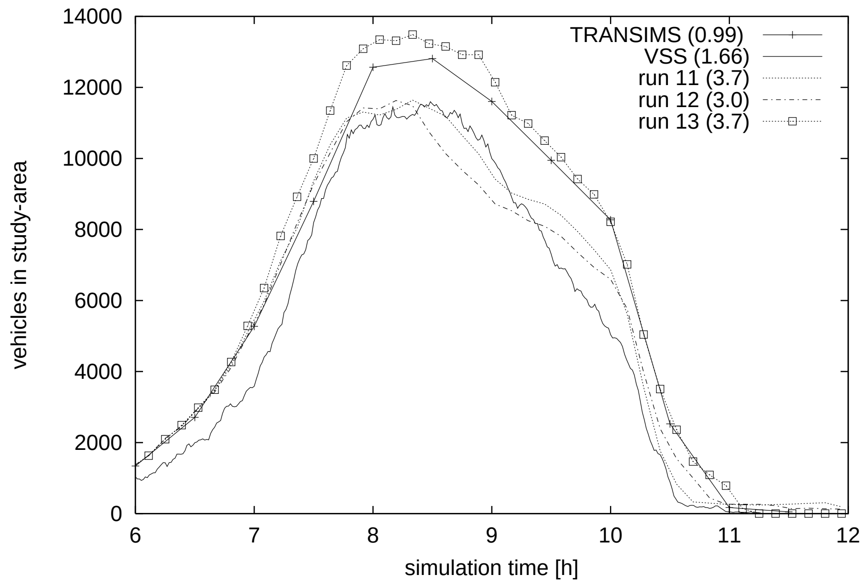

Figure 4.18: *Case-study test-bed: vehicles in study-area in a well relaxed system* — For large iteration numbers the PAMINA runs 11,12, and 13 can be assumed to be well relaxed (also see Figure 4.8). The curves depict the results of iteration 60.

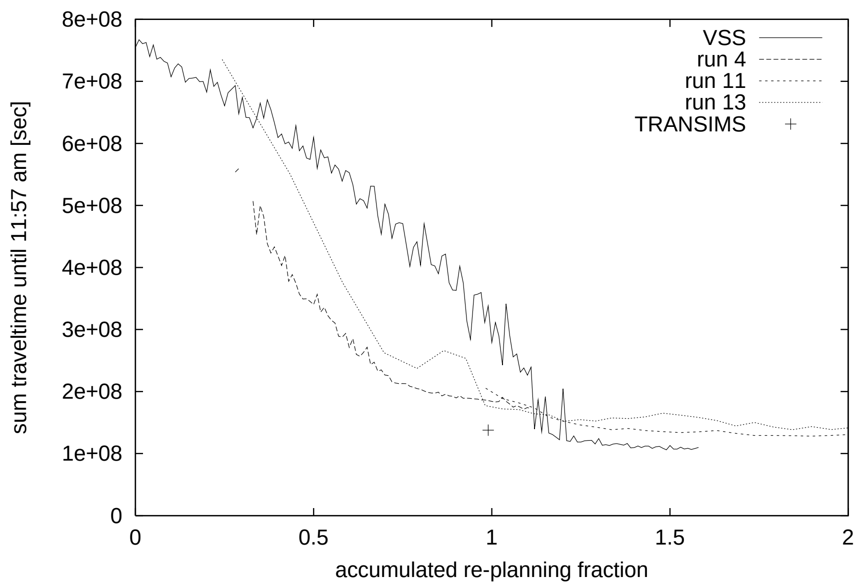

Figure 4.19: *Sum travel-times for different simulators* — The curves show the sum travel-time of all vehicles for several runs. Unfortunately, for TRANSIMS only one value can be displayed with $f_r = 0.99$. Note that the slopes for VSS and PAMINA are qualitatively different: while for PAMINA the improvement of travel-time slows down, for VSS it even seems to accelerate before the end of the relaxation process.

The sum of travel-times of run 11 is about 20% lower than the in the TRANSIMS-14 run.

Figure 4.19 shows the sum travel-time for three runs of PAMINA, one run of VSS and the TRANSIMS-14 iteration. We make the following observations:

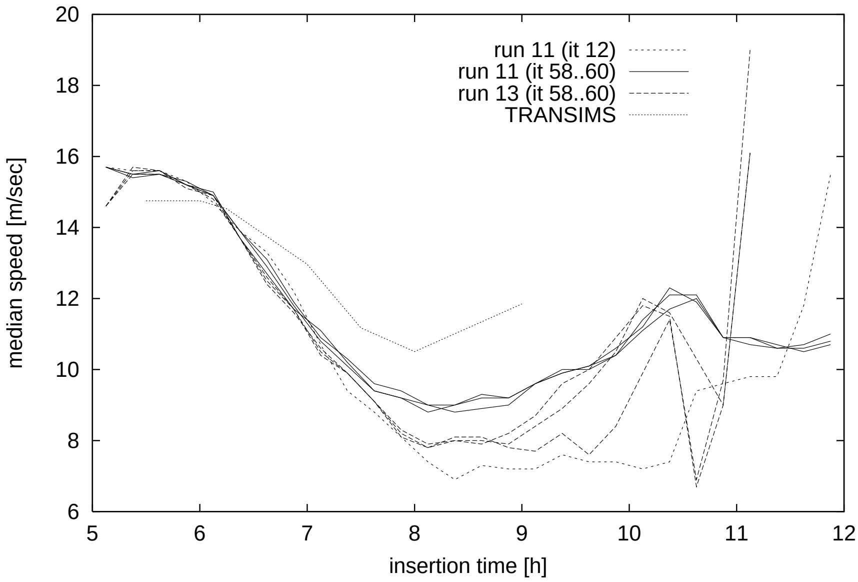

Figure 4.20: *Comparison of median speeds* — Both PAMINA and TRANSIMS exhibit the same qualitative shape with a pronounced minimum during rush hour.

(i) The final sum of all runs are in the range $1.0 \ldots 1.4 * 10^8$ [sec]. As seen before, the queue-correction run 13 is slightly higher. VSS has the lowest sum travel-time, which could be expected from Figure 4.18, since the number of vehicles is generally lower throughout the simulation period.

(ii) The qualitative shape of the curves is different for VSS and PAMINA. While the PAMINA curves show a steadily decreasing slope towards later iterations, the VSS curve seems to have a linear slope first and a slightly increasing slope just before the end of the relaxation process.

(iii) The VSS curve exhibits much stronger fluctuations than the PAMINA curves, especially with respect to run 4 which has the same re-planning fraction. A factor of $\approx \sqrt{5}$ can be explained by the smaller number of vehicles that are simulated by VSS. As mentioned above, the original route-set was scaled by a factor of $1/5$.

Also, this effect may again be due to VSS's simulation area: grid-locks not only occur inside the the study-area but also outside. For all grid-locks on feeding links outside the study-area the travel-time (or rather waiting-time) does not contribute to the curve.

So far all comparisons referred to highly aggregated data. In the following we will reduce the degree of aggregation.

4.4.2 Specific comparisons

For the purpose of comparison to one of TRANSIMS's many simulation statistics we restricted the collection of data to those vehicles that had their origins outside the study-area and their destinations within. Specifically, during the simulation we collected the following: for each vehicle i entering the study-area we stored the time of entry T_i, the location of entry S_i, the location of the destination D_i, and the travel-time t_i between S_i and D_i. From these we computed the Euclidean distance L_i between S_i and D_i and the average speed $v_i = L_i/t_i$. Finally, we chose the median v_{med} for all v_i in each 15-minute time-bin of T_i. Figure 4.20 shows the results for TRANSIMS and three iterations of run 10. The curves match qualitatively: both start at high median speeds in the early morning and reach a local minimum at around 8:00 am. Quantitatively, the TRANSIMS curve shows less significant extrema.

We also included a curve for iteration 12 of run 11 into the diagram. It is surprising that despite the similarity of run 11 and TRANSIMS in Figure 4.17, there is a clear difference for later hours: the PAMINA curve is well below the TRANSIMS curve and has a wide-spread minimum between 8:30 am and 10:30 am.

For runs 11 and 13 we plotted the average speeds for three consecutive iterations (58, 59, and 60). While run 11 shows very little variance throughout the simulation period, run 13 does so, especially after 9:00 am. This is partly due to the increased number of vehicles in the study-area at that time, which results in a higher probability of grid-lock. Also, run 11 may use the vehicles in queues as particle buffers: if for any reason a grid-lock is dissolved, the system can be filled again with vehicles waiting in queues. Apparently, this seems to have a stabilizing effect for the traffic inside the study-area, like ramp metering has a positive effect on highway traffic. The overall benefit, however, may be questionable, since some drivers are queued for hours *outside* the study-area.

4.4.3 Travel speed by origin

Using the angle between S_i and the center of the study-area (see Figure 4.21) we divided the incoming vehicles into four groups corresponding to the directions East (sector 0), North (sector 1), West (sector 2), and South (sector 3). For run 11 Figure 4.22 depicts the mean travel-speed by sector and the overall mean. It is obvious that the quality of traveling into the study-area strongly depends on the origin drivers come from. Drivers from the South have the highest travel-times before the rush-hour (before 7:30 am) and after the rush-hour (after 9:00 am). During the rush-hour drivers from the North have fastest access to the study-area. Their travel-speeds ($12 \ldots 16$ [m/sec]) show the weakest overall time-dependence of all sectors. Vehicles from the North and the East maintain their low trip-speeds until the end of the simulation period. Since there are no grid-locks at 11:00 am in run 11, this average speed is mainly dominated by vehicles being ejected from queues at the boundaries of the study-area. Since the insertion logic allows flows very close to link capacity, what we see here are basically a few links operating at capacity.

4.4.4 Turn counts

As a preliminary result we would like to present a comparison that was made in corporation with M. Pieck and K. Nagel. The NCTCOG provided turn count data for selected intersections inside the study-area. Therefore, both in the PAMINA and the TRANSIMS micro-simulators turn-count statistics were included. Figure 4.24 shows a comparison between the PAMINA run 11 and one of the TRANSIMS runs for the same time-period between 8:00 am and 9:00 am. Figure 4.25 depicts a comparison between PAMINA and actual counts. We make the following observations:

(i) The counts are very similar considering the short time interval of 15 [min] and the high level of disaggregation.

(ii) Generally, the turn counts in TRANSIMS are higher that those in PAMINA, which is due to the larger number of vehicles in the study-area at this time of day: approx. 12,500 for TRANSIMS and approx. 11,000 for PAMINA.

(iii) Compared to the actual data, the PAMINA counts are slightly biased towards left and right turns. This effect is caused by the lack of penalty for turns: in contrast to real-world drivers who generally dislike making to many turns in a route, the route planner chooses the shortest path independent of geometric considerations.

(iv) PAMINA counts are closer to TRANSIMS than to the actual counts. This fact insinuates that both micro-simulations have the same problem, which may have been caused by the common input data. Both route-sets are based upon the same O-D matrix, which is known to be from a different year (1990) than the street network (1996). The O-D matrix did not yet include a Northern extension a major freeway. As a result the simulations show insufficient counts in North-South direction for the two rightmost arterials.

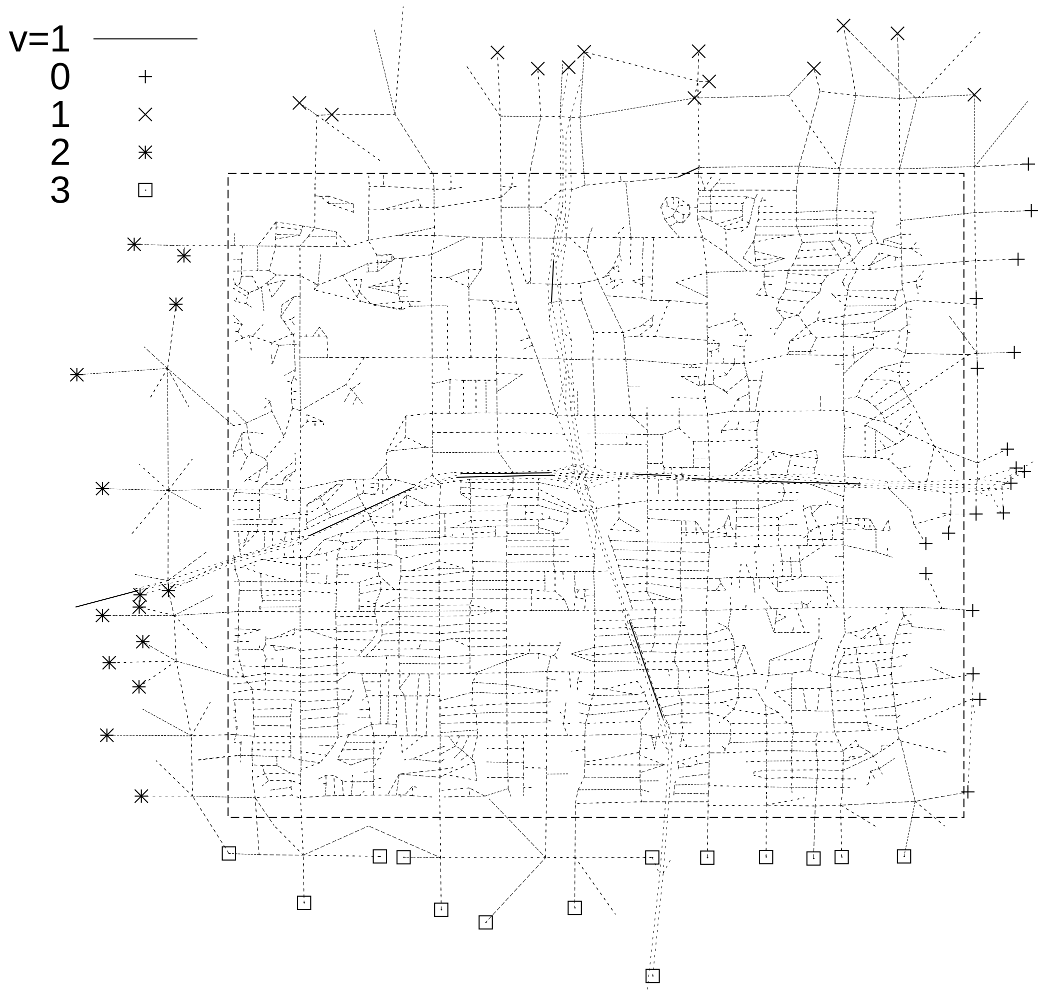

Figure 4.21: *Study-area: route origins by sector* — Map of the study-area including all local streets. The sources of the four sectors are marked with small symbols. All destinations lie within the rectangle plotted with dotted lines. This map also shows the links with reduced speed-limit that were used in the online simulation.

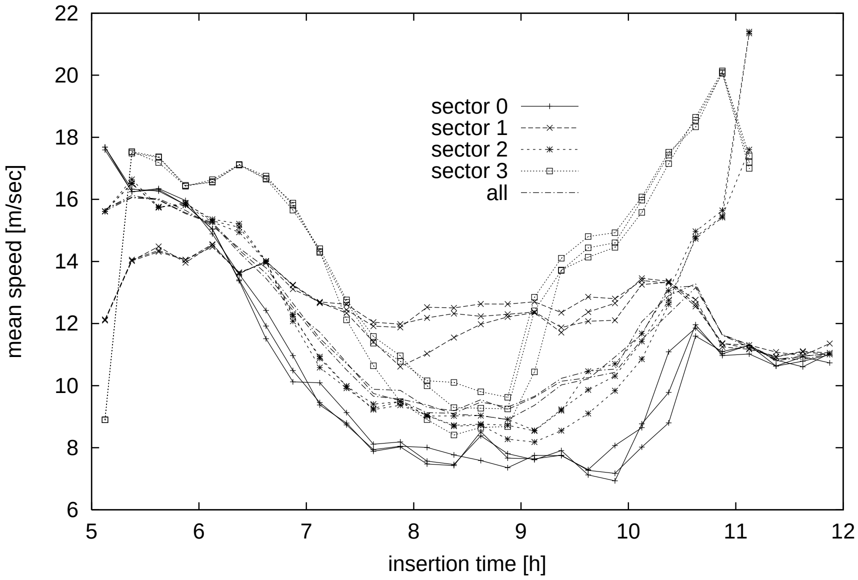

Figure 4.22: *Run 11: average velocities by sectors* — The curves display the mean speed of vehicles coming in from the four sectors. Vehicles originating in the East and in the North have low travel-speeds until the end of the simulation period. Vehicles from the West and South undergo the usual rush hour between 7:00 am and 10:30 am. Afterwards their average speed is back to free flow.

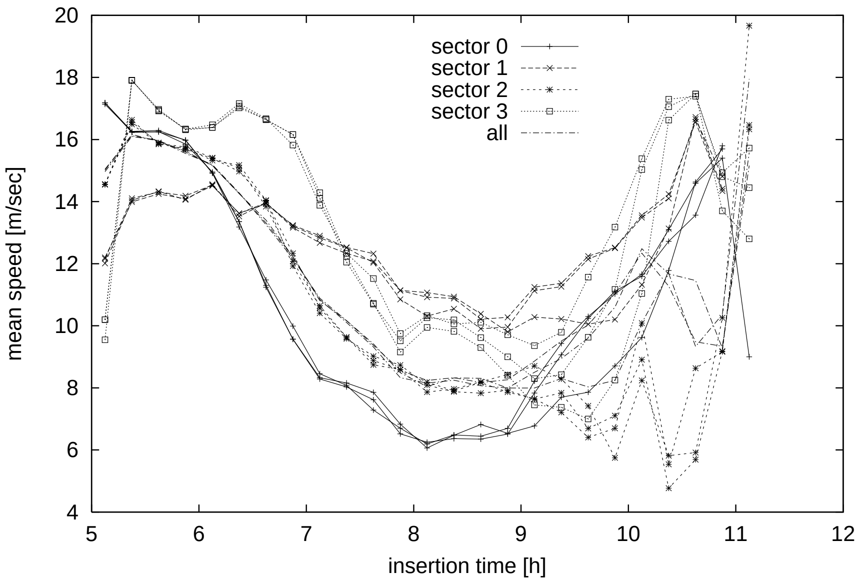

Figure 4.23: *Run 13: average velocities by sectors* — Run 13 exhibits average speeds which are generally somewhat lower than those of run 11. At 11:00 am, however, all directions have almost reached free flow speeds again. Note that due to the low number of vehicles after 10:30 am the curves tend to be rather erratic after that time.

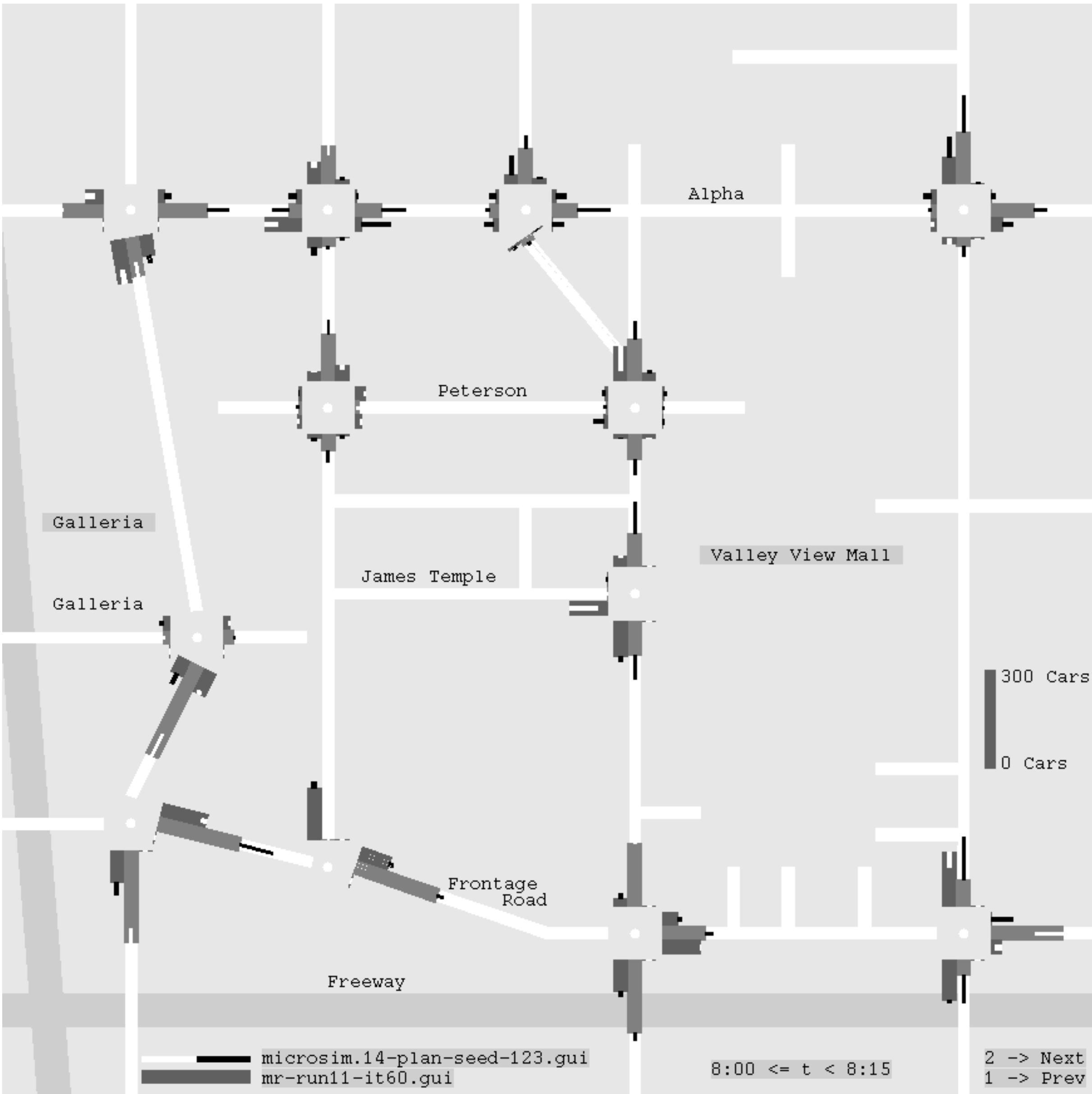

Figure 4.24: *Comparison with TRANSIMS: turn counts for selected intersections* — Grey bars at intersections denote PAMINA approach counts. Short black bars denote higher counts for TRANSIMS, white inverted bars denote lower counts for TRANSIMS.

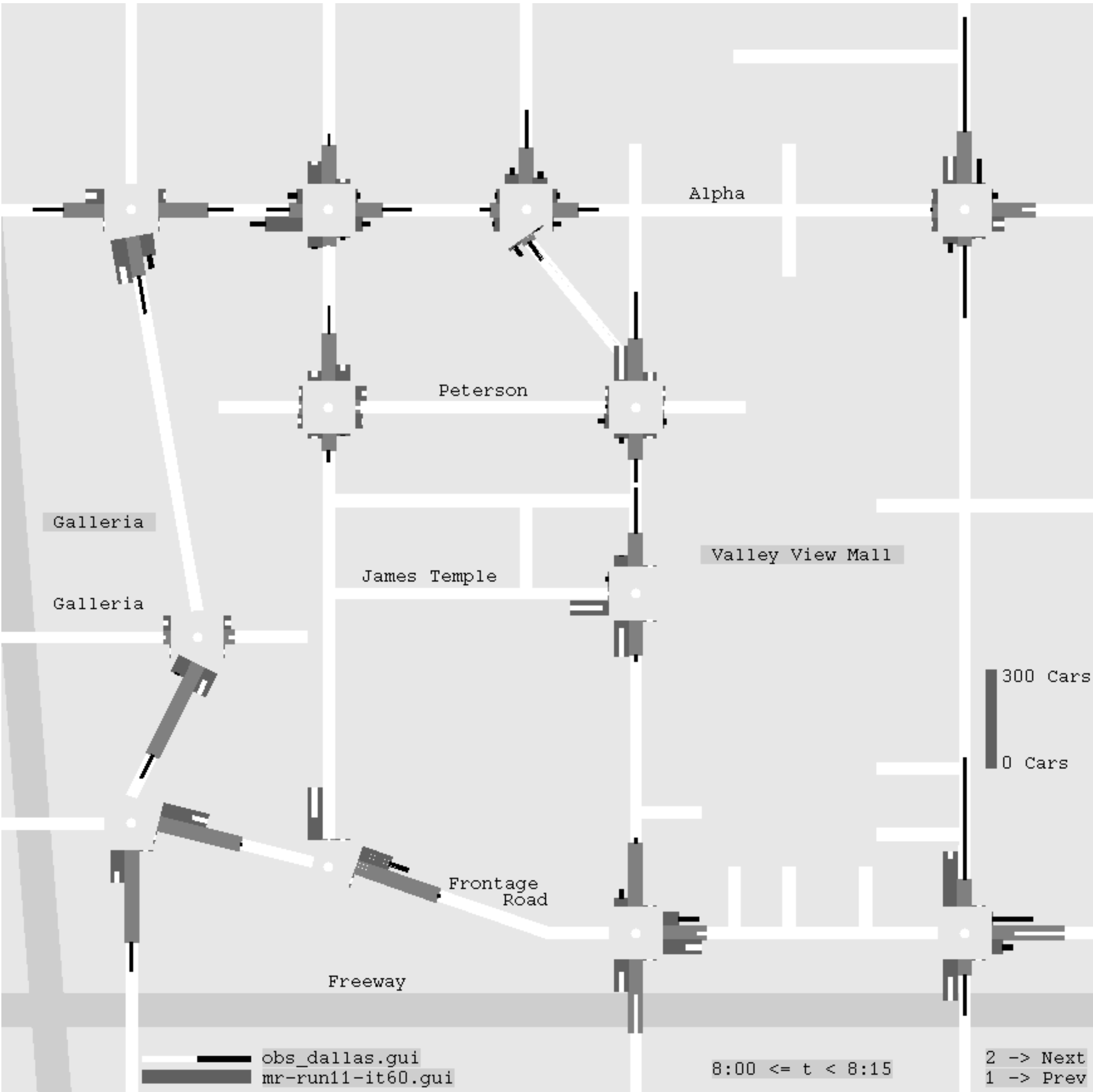

Figure 4.25: *Comparison with reality: turn counts for selected intersections* — Grey bars at intersections denote PAMINA turn counts. Short black bars denote higher counts in reality, white inverted bars denote lower counts in reality.

TRANSIMS will implement node penalties (represented by additional waiting time at nodes) in the near future [74]. Also, for the next test-bed for Portland (Oregon) the O-D matrix, the network, and any actual counts will be from the same year. All improvements combined are expected to match turn counts to the actual values.

Chapter 5

Online Routing

In chapter 4 we used micro-simulation to provide feedback for the dynamic traffic assignment. This process yielded a route-set which is expected to resemble reality within the margins defined by the fidelity of the micro-simulation. Results from simulations of this type can be used to forecast the impact of infrastructure changes on the environment or sub-populations. They may serve as decision criteria for city and traffic planners to determine which alternative is the most appropriate. The time-scale on which these decisions are made lie in the range of several years. Consequentially, for these computations the traffic network can be regarded as static with a few sets for representative scenarios (e.g. week day vs. weekend or holiday).

In this chapter we would like to consider a completely different time-scale. Instead of changing routes through an external router which is activated *after* the simulation has been executed, we will start modifying routes *while* a vehicle is traveling through the network. Therefore, the time interval at which the traffic network is monitored, will be reduced to the order of minutes. As before the Dallas - Fort Worth study-area will serve as the test-bed.

5.1 Introduction

With increasing traffic volume and likelihood of congestion the design of intelligent traffic information and guidance systems has become more and more popular. In the following we outline the main obstacles. We will use the term *subscribers* for all drivers with access to a route guidance system and the term *non-subscribers* all others. The fraction of subscribers will be called *market saturation*.

Emmerink et al investigated the influence of online information in non-recurrent congestion [26] and recurrent congestion [27] for a small artificial corridor network. They find that in both cases online information has a positive overall benefit for all market saturations. The maximum benefit is usually reached at medium saturations around 50%.

Esser [30] uses the core[1] network of the city of Duisburg (Germany) for routing experiments with en-route data. He finds a similar result: the average speed of all vehicles in the network is a monotonously increasing function of the market saturations. For some of the scenarios, however, the maximum benefit is already reached at 60%.

In practice, most intelligent route guidance systems have not passed their experimental test phase. This is due to a set of technical and algorithmic problems which have only been partially solved. Among others these are:

Collecting state information The first step of a functioning guidance system must be to retrieve an up-to-date representation of the current state for the whole traffic network in question. The clas-

[1] The network consisted of 107 intersections and 280 links.

sical approach is to install local counting devices that provide information about vehicle count and optionally about vehicle velocity. The availability of these devices is very heterogeneous and their installation is cost-intensive. Of course, in order to ensure that the system works efficiently, a certain spatial saturation has to be reached which would require an enormous investment by either public institutions or commercial investors. Counting devices have a built-in inaccuracy which may influence prediction quality (see [29]). Also most devices only provide good information if the flow has not broken down yet: It is impossible to differentiate between an empty street and one with stalled vehicles in it. In this case, cameras may be used to decide which case applies.

Once installed, the collected data has to be transferred to control centers. The fact that often there is no permanent data connection so that all data is collected and *stored* locally infers the purpose of many of counting loops to provide long-term surveys of traffic flow. In this case the expensive hardware is useless for online applications. Only if state information can easily be collected in short intervals (no more than 15 minutes), it can be integrated into an online system.

Lately, it has been suggested to use data gathered by vehicles actively participating in the traffic flow. These *floating probe* vehicles (see [29]) have storage devices on-board to collect information about location and average-velocity which are later transmitted to some control center. Again as with the permanently installed counting devices, a certain density of probes is necessary to provide a consistent view of the network. In addition, it is questionable whether on-board devices will be as readily accepted, because it will be difficult for providers to guarantee that information about location and time of travel will be handled anonymously. Of course, this is less a technical question than one of consumer acceptance. The state information need not necessarily contain any information relating to the vehicle identity. The question is whether consumers will *trust* their data to be safe. Critics claim that it is possible to reconstruct complete trip itineraries from partial information and associate these with individual drivers.

Note that the two approaches are not mutually exclusive. Both can contribute to an overall view of the traffic state. It will be interesting to compare and validate results in areas where information from both sources are available.

Providing alternative routes At this point we assume that we actually have a consistent view of the traffic network, no matter how this information was retrieved. We distinguish between two major types of traffic guidance: (a) the collected information is made available to drivers who have bought or leased online traffic information systems. It can be used to graphically enhance maps of the traffic network by adding color-coded information on congestion or travel-times on links. The important aspect is that in this case, *the driver himself* still makes the decision whether to deviate or stick to the current route. No hints are given about a potential detour. That is why this approach is called *passive*: The information system only provides information about the current state but nothing else. (b) The second approach is to use the state information to *actively* suggest alternative routes to the driver. This can be done locally, in which case an on-board computer determines an alternative route, or by the provider of the online information.

With increasing market saturation the impact of vehicles using alternative routes will result in a non-negligible feedback on the traffic network. More specifically, re-routed vehicles may create new congestion in areas which would have been free otherwise. This latter case raises the issue of fairness (see below), which is concerned with the question in how far the orchestrated re-routing for subscribers of the system can have negative effects on all non-subscribers.

One approach to reduce the effect of an “online” feedback — that is guiding a large fraction of vehicles through a detour — is to use shortest-path algorithms that deliver reasonable alternative routes in addition to the actual shortest path. Finding these routes proves to be more difficult than one would expect because conventional k-shortest-path algorithms (i.e. [35]) deliver results which were obtained by replacing an increasing number of links by slightly longer alternatives. Transferred onto the real traffic network this corresponds to artifacts such as changing from the highway to a parallel frontage road at one exit and reversing the process at the next ramp. Thus, one possible criterion for the term “reasonable” [109, 91] in this context could be that the alternative route overlaps very little with the shortest route, but is almost cost-equivalent (see [49]). Using shortest-path algorithms that start looking for a shortest path at both the origin and the destination may be a first step although they usually do not work in traffic networks with time-dependent link weights.

Communication Another related technical problem is how to transfer state-information (or route-alternatives) from and to vehicles. One solution is to install a network of beacons which communicate with passing vehicles over a short range (e.g. several meters) and relay their data to control centers. Although this approach is technically feasible and already in use in one test-bed (Berlin) it requires a completely new (and additional) network of hardware-transmitters.

Another approach is to use the cellular telephone network as communication network. In the long run, this will definitely be more cost-effective than new hardware, since the cellular telephone market is expanding at high rates. The amount of information which needs to be transmitted is small as compared to the conventional voice information. Current systems work at data rates of about 4 [KByte/sec] which means that complete alternative routes could be transferred in a fraction of a second. Even the establishment of a connection is possible within a second because most cellular phone networks use digital control signals. Therefore, a complete transaction of communicating the current location, the current route, and requesting a new route could be done in less than 2 seconds. Depending on the response time of the server providing the route alternatives the connection can be held (quick response time) or terminated (long response time). In the latter case the on-board device would be called back by the server as soon as an alternative route has been computed.

Acceptance Alternative routes that are suggested by the provider are based upon the *current* traffic state or estimates for the future state respectively. Both are subject to errors which may reduce the quality of an alternative considerably, or in the worst case make it worse than the original route. Depending on the likelihood of these negative experiences with a route guidance system, the acceptance will decrease. This will be especially disadvantageous to the provider if the service is payed on a pay-per-use basis. One way to regain the trust of subscribers would be to offer a service which is based on a pay-per-success basis. In this case the provider has to investigate with which likelihood he can provide sufficiently good suggestions so that he breaks even. The Dallas - Fort Worth test-bed allows for a first measure of the commercial success of a simple online routing system.

The route suggestion itself is also subject to acceptance by the drivers. A quick detour may be shorter but very hard to follow if it leads through areas that the driver is not familiar with. Other drivers may object to left-turns, going through residential areas, using free-way ramps too often, and so on. It should be clear that for providing a good route the driver's preferences must be taken into consideration.

Fairness Providing additional information to a subset of drivers may improve their average travel-time in recurrent and/or non-recurrent congestion. The commercial success of the service provider will be determined by the predictability and quality of the route-alternatives. If the service only had an impact on the drivers subscribing to it, the situation of the remaining drivers would completely decouple. Unfortunately, there is a feedback (be it positive or negative) due to re-routing taking place which will affect the whole fleet of drivers. In case the impact is mainly negative — for example if the average travel-time of the non-subscribers increases measurably — legislation may demand that a certain penalty be paid to compensate for the increased social costs imposed on non-subscribers. If the average travel-time even increases for all drivers, the social costs would be accompanied by additional environmental costs.

On the other hand, a centralized and orchestrated route guidance system may be the only way to move the traffic situation closer to a system optimum instead of the current user equilibrium. It is well known, however, that system optimal solutions usually require a certain fraction of the drivers to take routes which are *longer* than those they have used previously. Within the context

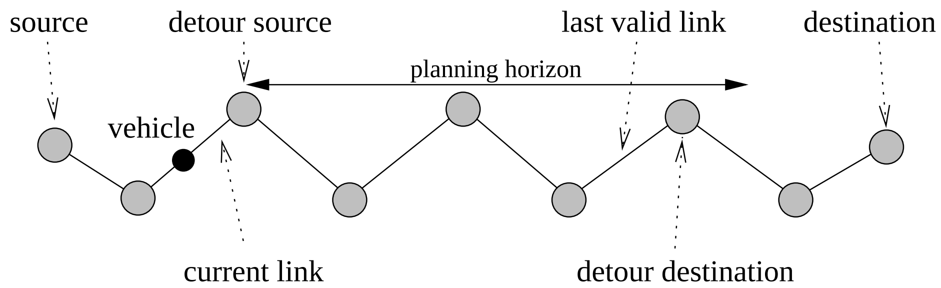

Figure 5.1: *Geometry of an online-detour* — For a vehicle the head of its current link serves as the source node for the detour search. The last node that is still within the planning horizon serves as detour destination.

of commercial service providers, it seems unrealistic to expect subscribers to pay for service which worsens their situation. But even if the average travel-time increases, the predictability — which is often regarded as a desirable feature a route [127] — will improve. Unfortunately, pushing traffic closer to optimality unveils its chaotic nature [81].

One of the issues of the remaining part of this chapter will be to estimate the impact on subscribers and non-subscribers as well as on both groups combined.

5.2 Re-routing algorithm

So far we used static route-sets for all simulations with PAMINA. Each vehicle followed its route-plan independent of the *current* road conditions until it reached its destination. This approach in conjunction with the discreteness of the traffic model resulted in grid-locks which could not be resolved anymore (see 3.4.2). Using the iterative re-planning improved the situation to such an extent that in later route-sets grid-locks no longer occurred.

The question is now, how the static approach can be enhanced by additionally using an online re-planning scheme. "online" in this context does not mean that we feed real-world online state information (e.g. data provided by online monitoring devices such as counting loops or video cameras) into the re-planner but instead use the micro-simulation as provider of state information. The starting point for each experiment is a route-set obtained from the iterative re-planning process. This serves as a specific "test day" for the traffic in the simulation area. Without online re-planning we can test the quality of this route-set by executing the route-set several times using different random seeds. For each route through the study-area we will obtain average trip-times and their respective variances.

5.2.1 Criteria triggering re-routing

Using the static case as the base-case, we now allow a certain subset of drivers to access online information during the trip through the study-area. The fraction of drivers equipped with these intelligent routing devices is called the *market saturation* m_{o-l}. Every t_{o-l}^{update} time-steps (in the range of $120 \ldots 240$ seconds) all equipped vehicles are monitored. For each vehicle the following steps are executed (see Figure 5.1):

- Starting with the *current link* (ending in the potential *detour source node*), the vehicle adds up the current travel-times t_{o-l}^{trav} for all future links in its route-plan until it reaches the first link whose head is further[2] away from the detour source node than the given planning horizon h_{o-l}. For all remaining links the travel-times contained in the original route are used. The sum of all individual travel-times yields a new arrival time T_{o-l}^{arr}.

[2] Manhattan-distance

Note that here the "historic" information of the old route is used to allow some reasonable comparison. The situation that we investigate can be compared to the daily commute to work for which each driver memorizes his previous trip time. Of course, there are other ways to include older information. In a day-to-day simulation of the street network of Nordrhein-Westfalen (NRW) [76], however, it was shown that there is no decisive difference between using only the previous trip or additional older trips (in this case with an exponentially decreasing weight).

- The driver compares his scheduled arrival time T^{arr}_{sched} contained in its original route-plan to the new estimate T^{arr}_{o-l} provided by the online router. If the relative estimated delay

$$d_{o-l} = \frac{T^{arr}_{o-l} - T^{arr}_{sched}}{t^{trav}_{sched}}$$

  is greater than a given threshold d^{min}_{o-l} the driver will *request* an alternative route from the online router. Otherwise (and in the case that the vehicle is only two links away from the destination) no action is taken.

- The online router starts a shortest-path search using the *detour source node* and the to-node of the last valid link as *detour destination node*, which is the last node of the current route within in the planning horizon. The router computes a new estimated travel-time for the re-routed portion which is combined with the portion from the detour destination node to the ultimate route destination node yielding a re-routed arrival time T^{arr}_{r-r}. The relative re-routing delay (negative values denote an improvement) is given as

$$d_{r-r} = \frac{T^{arr}_{r-r} - T^{arr}_{o-l}}{t^{trav}_{sched}}.$$

- If d_{r-r} is smaller than a given threshold d^{min}_{r-r} the vehicle uses the new alternative route. This is counted as an re-routing event. The router will increment a counter associated with the vehicle to keep track of how may times the vehicle has been re-routed. The estimates for all links up to the detour destination node are replaced by those of the new route. All estimates after that (including the arrival time) are corrected by adding the time difference $T^{arr}_{r-r} - T^{arr}_{sched}$ to the old values.

5.2.2 shortest-path algorithm and edge weights

During the simulation each link j (with length L_j, number of sites S_j, and free speed v^{free}_j) is monitored every t_{sample} time-steps for vehicles. The number N_j of vehicles found on link j and the sum of their velocities V^{sum}_j are accumulated over the update interval of t^{update}_{o-l} time-steps. At the end of each interval the travel-time t^{trav}_i is computed as follows

$$t^{trav}_j = \begin{cases} N_j L_j / V^{sum}_j & \text{if} \quad V^{sum}_j > 0 \\ L_j / v_{min} & \text{if} \quad V_j = 0 \quad \text{and} \quad N_j / S_j > \varrho_{thresh} \\ L_j / v^{free}_j & \text{otherwise} \end{cases}$$

For a given source S, destination D, and departure time T^{depart} the router executes the following steps: If S is the same as the previous source S_{prev} and $T^{depart} = T^{depart}_{prev}$ we re-use the shortest-path tree of the previous computation. Otherwise we compute a new shortest-path tree using a simple label-setting Dijkstra [19] algorithm. The travel-times t^{trav}_i are used as link weights. Only those nodes are considered for labeling that are in a Manhattan distance of h_{o-l} to S. The algorithm stops only after all reachable nodes have been labeled even if D has already been labeled. This allows the router to re-use the shortest path tree several times, since the requests are processed link by link: All vehicles on the same link share at least the first (= source) node of their potential detour. The label at D is used as the travel-time from S to D.

Note that this algorithm works with time-independent edge-weights. Within the context of the test-bed this can well be justified, since the average travel-time of all routes is about 10...12 minutes. For re-routing in larger areas, one would have to include travel-times obtained from other sources than current measurements.

5.3 Scenarios

5.3.1 Re-routing parameters

In Section 5.2.1 we have described four parameters that will be used to influence the behavior of the online re-planning. These are:

Market Saturation The impact of the market saturation m will be important for the marketability of online re-routing devices. Potential customers will only accept devices that provide a certain guarantee of success. If fees for route alternatives are only due in case they were successful (e.g. resulting in a travel-time shorter than some predefined average) providers will be interested in estimating what minimum success-rate would be necessary to break even.

Planning Horizon The planning horizon h_{o-l} (maximum spatial look-ahead) of the detour search has an important impact on the run-time behavior of the shortest-path algorithm. In a traffic map intersections are homogeneously distributed over the map area. The number of nodes to be labeled within a given distance h_{o-l} from the source node scales with $O(h_{o-l}^2)$. From this point of view the value should be kept as small as possible. On the other hand, increasing the value may enable the algorithm to find more useful alternatives.

Note that a small h_{o-l} not only reduces the number of labeled nodes considerably, but also decouples re-planning regions in a distributed route guidance system.

Update Interval The update interval for the link travel-times T_{o-l}^{update} will represent a trade-off between a quick response to increasing demand on links (short intervals) on the one hand and good statistics of link travel-times on the other hand. As far as computational efficiency is concerned, shorter intervals also increase the amount of communication between re-planning regions. This is directly transferable to the hardware requirements of a real-world route guidance system: the faster the update, the higher the required bandwidth.

Accumulated Re-planning Fraction The result of the iterative re-planning process is a self-consistent route-set which exhibits little fluctuation with respect to additional re-planning. The question is if reality is actually that close to a possible equilibrium. In case it is not, route-sets obtained from earlier iterations may be more similar to a real-world configuration of route-choices. Therefore, we will use route-sets from different iteration runs as base-cases for our re-planning experiments.

5.3.2 Simulation setup

For each experiment we run the simulation five times with no online re-planning ($m = 0$) to obtain data for the base-case. Each run uses a different set of random seeds. Afterwards we run a set of five runs for each parameter combination that we investigate.

In contrast to the iteration runs presented in Chapter 4 we changed the behavior of the planner RP2 slightly. In order to provide current travel-time estimates for all drivers, all routes — even those that are not re-planned — are updated to reflect the travel-times of the previous simulation run. Otherwise, the incentive to request a new route-plan would be considerably reduced for early iterations, since most of the drivers still have estimates dating back to iterations when grid-locks were very common. Figure 5.2 shows

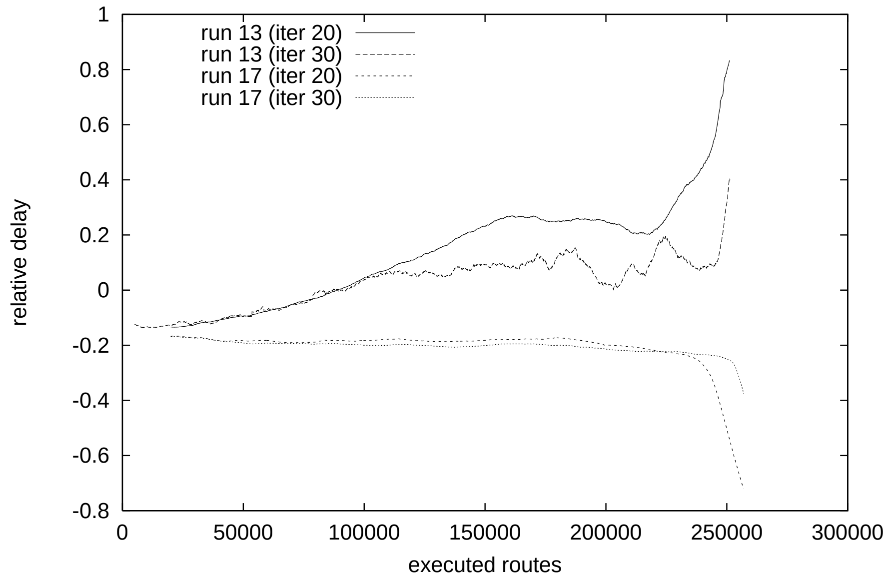

Figure 5.2: *Update of travel-times* — The curves show the running average of the relative delay for different runs. While run 13 uses scheduled arrival times based upon old link travel-time feedback files, run 17 re-computes all estimates using the latest run.

relative delays for runs 13 and 17 which use the same parameter combination but differ in the travel-time estimates. Run 17 shows a time-independent behavior whereas run 13 starts out by underestimating travel-times but considerably overestimates travel-times later in the simulation. We chose the parameter combination

- planning horizon $h = 5$ $[km]$ (corresponding to half the diameter of the study-area),
- update interval $T_{o-l}^{update} = 120$ $[sec]$, and $t_{sample} = 10$ $[sec]$,
- accumulated re-planning fraction $f_{acc} = 1.0$, and
- $v_j^{min} = 0.01 * v_j^{free}$,
- $d_{o-l}^{min} = d_{r-r}^{min} = 0.1$, and $\varrho_{thresh} = 0.05$

to be the reference for all our comparisons. These and most other combinations were sampled for market saturations between 0 and 90% in intervals of 10%.

5.3.3 Recurrent congestion

The first case that we investigate is that of recurrent congestion. The route-set that is used for the simulation contains a configuration of drivers that have chosen their routes over a re-planning period of 20 iterations. Each driver has chosen his route based upon a snap-shot of the previous iteration. Since the micro-simulation is non-deterministic, the traffic conditions in areas with large variance may be completely different for the current run. Therefore, there is a certain chance that choosing a different route will improve the trip-time of a driver although *on the average* the current route is already a good choice. This is where the online routing comes into play which is able to use the current snap-shot instead of an outdated one.

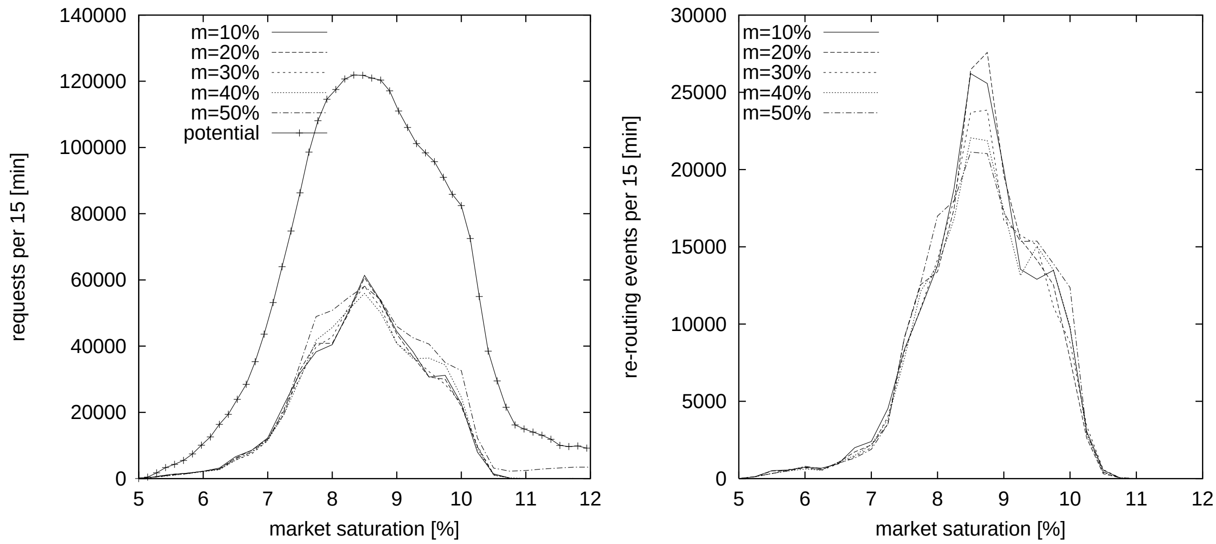

Figure 5.3: *Requests and re-routing events (iteration 20)* — LEFT: the number of potential requests derived from the number of vehicles in the system, and the actual number of requests scaled to a market saturation of 100%. RIGHT: the number of successful re-routing events with at least d_{r-r}^{min} relative improvement.

Frequency of service

In following we will present simulation results describing the different criteria of the decision making process outlined in Section 5.2.1. First we take a look at the frequency with which the online service is queried by subscribers. The left-hand side of Figure 5.3 shows the number of requests that the router has to handle for different market saturations $m_{o-l} = 0.1, 0.2, 0.3, 0.4, 0.5$. The counts have been accumulated in time-bins of 15 minutes and normalized to a market saturation of 1.0. The curves exhibit little difference with respect to the market saturation. As a reference we have included the number of *potential requests* which can be obtained from the number of vehicles in the study-area. Potential requests are equivalent to requesting a travel-time estimate for the remaining portion of a trip. One can see that only half of all drivers actually issue requests, since the other half has obtained an estimate that differs relatively less than $d_{o_l}^{min}$ from its "memorized" travel-time of the previous run.

The right-hand side of the same figure shows the number of requests that are answered by returning a new route. Note, that this requires the detour to be relatively better by at least d_{r-r}, so that about half of the vehicles do *not* receive a new route. The scaled number of re-routing events is also almost independent of the market saturation except for the peak period at 8:30 am, during which the higher saturations yield a smaller chance of being re-routed (about 20% decrease of $m_{o-l} = 0.5$ over $m_{0-l} = 0.1$). Remember that — at this point — the fact of being re-routed does *not* necessarily result in a shorter trip-time, but only corresponds to a shorter *projected* trip-time. Figure 5.4 shows the rate with which requests are answered by a returning a new route-plan. The curve exhibits a peak at about 8:30 am. This is the time when the average-velocity of routes into the study-area reaches its minimum (see Figures 4.22 or 4.23).

There is another peak at 5:30 am which may be an artifact due to wrong travel-time information: Travel-time data is only available for a link, if vehicles currently use it. Otherwise the travel-time is computed (as in the planner) to be length divided by free speed (see 3.3.1) resulting in *longer* travel-times for sparsely populated links. Since the time-bin for the external router is 15 minutes instead of 2-4 minutes for the online re-planning, the probability of a link to be empty is larger in the online case. In these cases the online re-router may find detours on links with underestimated trip-times. Later — as the study-area fills up — most links have travel-times from actual vehicles, so that the artifact disappears.

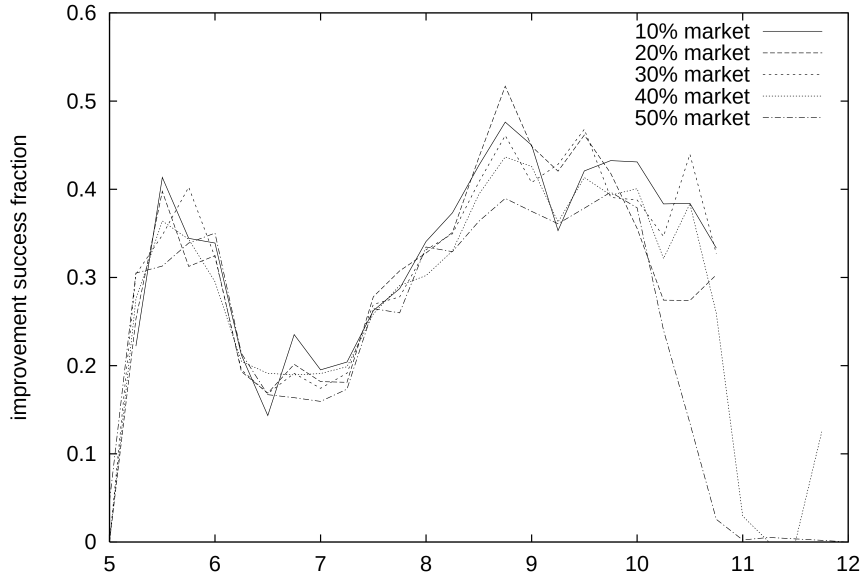

Figure 5.4: *Success fraction (iteration 20)* — Fraction of re-routing events that were successfully handled by returning a new route. The peak at 5:30 am is due to insufficient link travel-time statistics at that time of day.

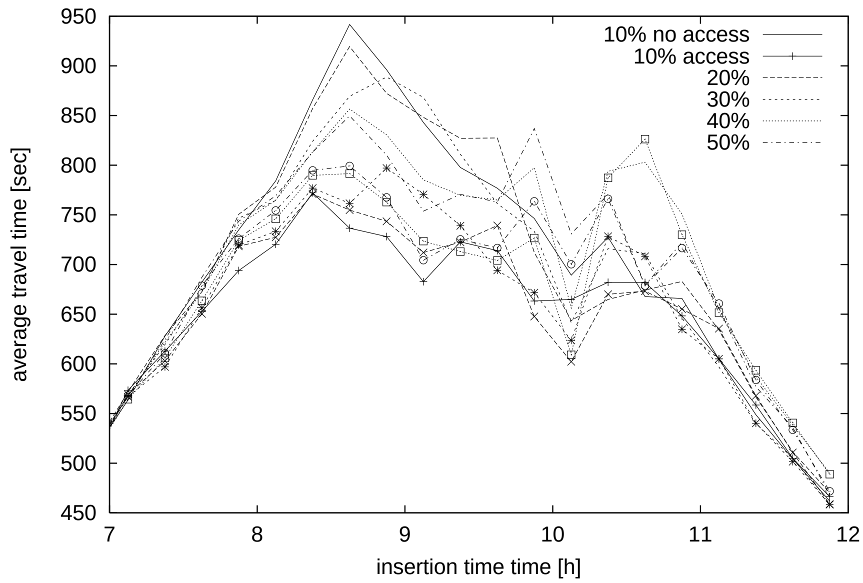

Figure 5.5: *Quality of service (iteration 20)* — Average travel-times of non-subscribers (lines without symbols) and subscribers (corresponding lines with symbols).

Quality of service

When subscribers are offered a route alternative they expect an improvement in travel-time with respect to the original route they would have taken. Unfortunately, due to the uncertainty of state information and prediction the alternative may not live up to its promise. Therefore, it is important to have some quantitative measure about how well the re-routing process works.

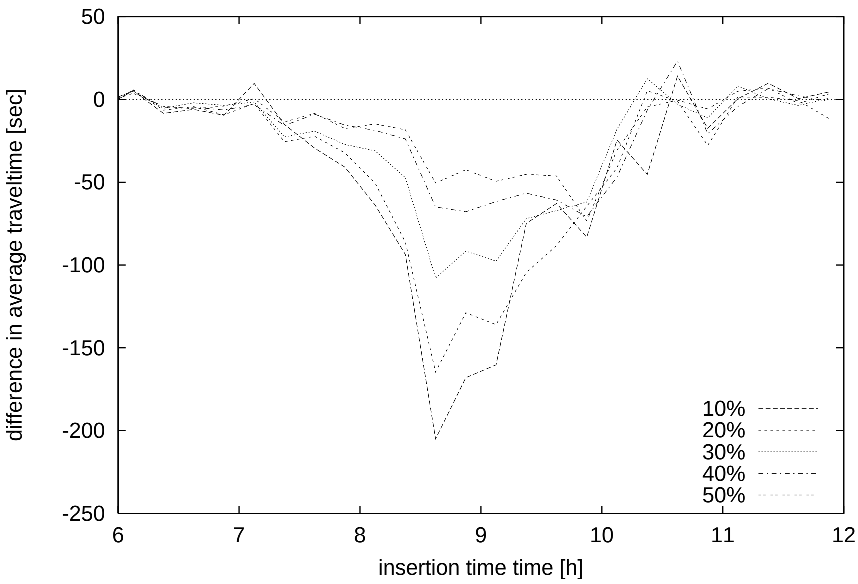

Figure 5.6: *Difference of average travel-time between subscribers and non-subscribers (iteration 20)* — Negative values denote shorter travel-times for subscribers. The advantage of subscribers over non-subscribers decreases as the market saturation grows.

First we will look at the overall benefits for subscribers compared to non subscribers. We compute the average travel-time of all subscribers and non-subscribers and average them over time-bins (with respect to their insertion time) of 15 minutes. In Figure 5.5 the average travel-time is plotted for different market saturations. Up to 10:30 am the curves for non-subscribers lie higher than the respective curve for non-subscribers. The difference of travel-time decreases as the market saturation increases.

Figure 5.6 shows this result more clearly. During the peak of the rush hour, a ten percent market saturation yields an average improvement of 200 seconds which corresponds to 28%! Of course, this does not prove the overall benefit of such a service since the improvement could be strongly biased towards a small subset of subscribers enjoying extreme reductions in travel-time while others have no benefit or may even be delayed. Still, Figure 5.6 may be important for marketing purposes since it clearly shows that the system works especially well if restricted to a small clientele. One surprising result is that the advantage of re-routing vanishes after 10:30 am. Figure 5.7 shows the number of insertions per 15 minutes and the number of vehicles in the study-area for the case of zero market saturation. It becomes obvious that — although there are still thousands of vehicles in the study-area — the number of new insertions drastically drops at 10:00 am and is already vanishingly small at 10:30 am. The effect we see, thus, is due to the fact the route-set does not extend far enough beyond the peak of the rush hour. The reduced number of vehicles has a negative effect on the statistics. Hence, all results after 10:00 am have to be interpreted with caution.

It is interesting to take a look at the distribution of relative delays upon arrival at the destination. From the five runs that were executed for market saturation $m = 0.2$ we gathered the subscribers into three groups (see Figure 5.8): those who were never re-routed ($rr = 0$), those who were re-routed one to five times ($rr = 1$), and those who were re-routed six to ten times ($rr = 2$). Trips with more than ten re-routing events were not considered because of their insufficiently small counts. The curve for $rr = 0$ is mainly a normal distribution centered around zero delay since this subset does not differ from non-subscribers at all except for the fact that they "know" they cannot receive better alternative routes. The curve for $rr = 1$ is also peaked at zero delay, but is biased towards negative delays, which is to say that there are more subscribers arriving earlier than non-subscribers. The effect becomes even more obvious for curve $rr = 2$, where the peak has actually shifted towards negative delays. This means that within

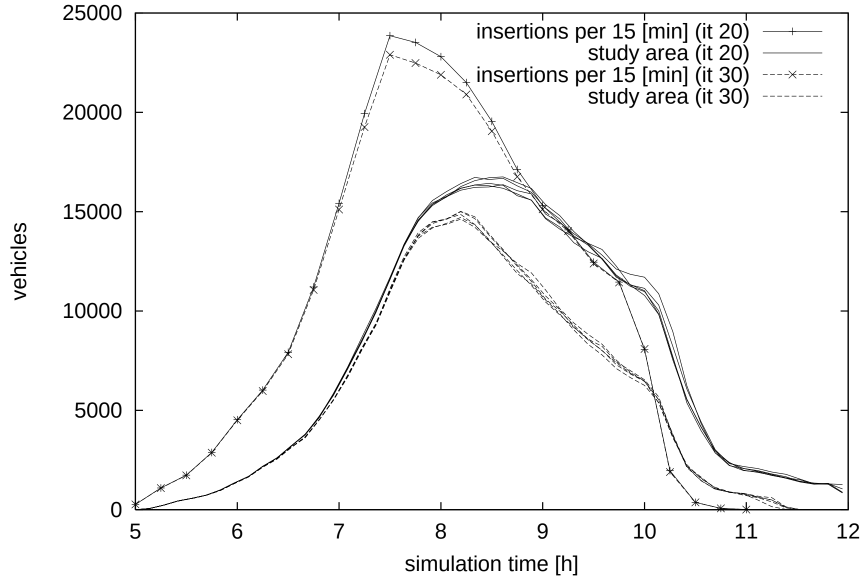

Figure 5.7: *Iteration 20 and 30: insertions and vehicles in study-area* — The curves depict the insertions into the study-area per 15 minute interval and the number of vehicles in the study-area. Note that there are hardly any insertions after 10:00 am.

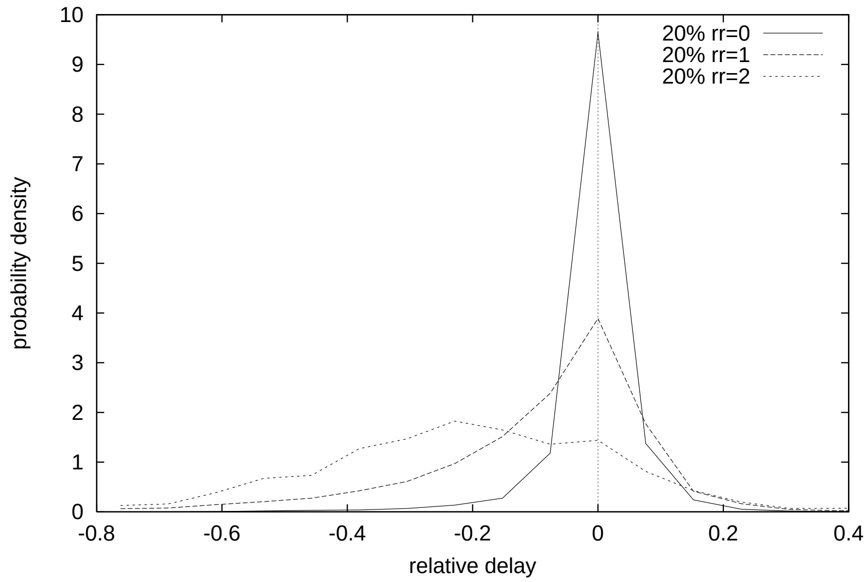

Figure 5.8: *Quality of service for $m_{o-l} = 0.2$ (iteration 20)* — The curves show the relative delay of re-routed vehicles for different re-routing counts. The more often a vehicle is re-routed the more the distribution is biased towards negative delays: it pays to be re-routed several times.

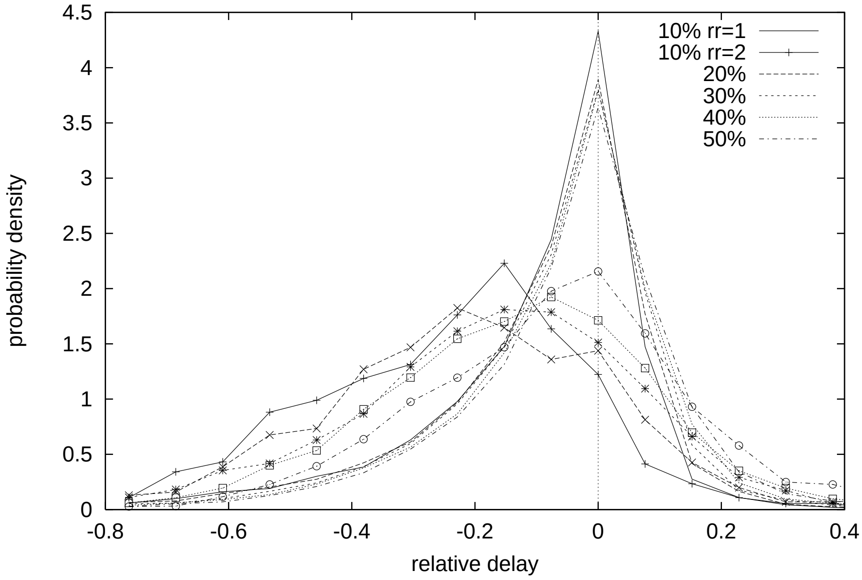

Figure 5.9: *Quality of service (iteration 20)* — In this figure the routes with few re-routing events ($rr = 0$) were omitted. While the curves for $rr = 1$ are basically independent of the market saturation, for $rr = 2$ the benefit decreases with growing market saturation.

our re-routing approach *it is worth while to be re-routed more than once.* Figure 5.9 gives an overview about several market saturations m. Here we have omitted the $rr = 0$ curves. It is interesting to see that the curves for $rr = 1$ are practically independent of m (although the fraction of positive delays slightly increases with m). The real difference occurs for $rr = 2$ where there is a continuous shift to negative delays for decreasing market saturation.

Fairness

Traditionally, traffic control systems have never been restricted to a subset of traffic participants. For example, radio transmitted information about congestion is available to practically the whole population — although it is very rarely used (see [115]). Traffic control measures, such as actuated traffic lights or variable traffic signs, affect the whole stream of vehicles passing by without discrimination.

Individual route guidance is a service that would only be available — at least during its market introduction — to a subset of all drivers. Traffic infrastructure, however, is still largely funded by public taxes. It is therefore logical that the benefits of a traffic network should be equally be spread among all participants. Note that there may be a regional competition of interests to accept only desired traffic (such as shoppers or delivery trucks) but to reject other traffic that only passes through.

In this context, a traffic control system that does not affect non-subscribers negatively may be much easier to sell or enforce than one that is known to be biased. Therefore we have investigated in how far the re-routing affects the overall traffic condition, compared to the base-case with a market saturation of zero. Figure 5.10 depicts the difference in average travel-times of all drivers with respect to the base case. Small saturations $m = 0.1 \ldots 0.3$ have hardly any effect for early hours and late hours with a short disadvantageous phase between 9:00 am and 9:30 am. Higher saturations $m_{o-l} = 0.4 \ldots 0.5$ have a stronger effect both positively until 9:45 am, and negatively afterwards. The curves show an a rather large dip after 10:00 am. As mentioned before, this portion has little significance because of the less balanced vehicle fleet of later hours.

If the travel-times of individuals decrease, it is to be expected that the overall throughput of the traffic

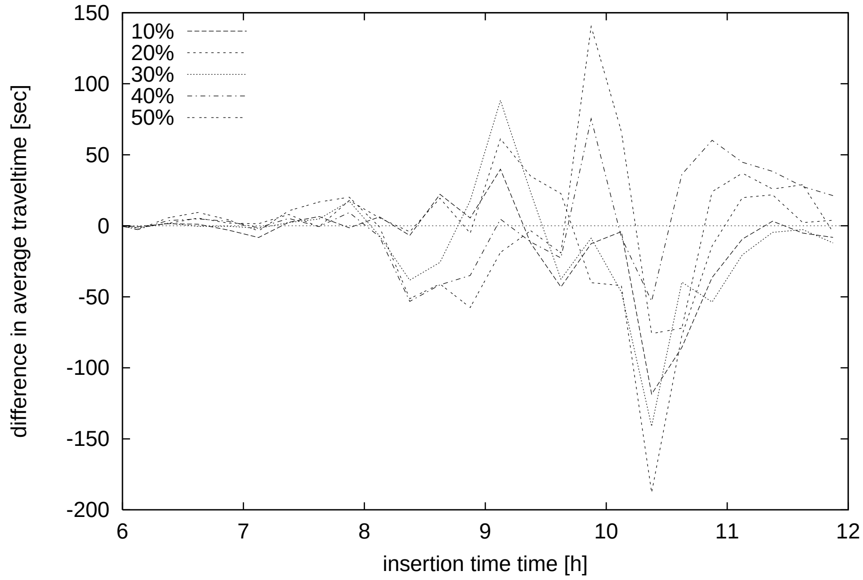

Figure 5.10: *Difference of travel-time of all to base-case (iteration 20)* — The curves show the impact of the guidance system on all drivers. Negative values denote an overall improvement.

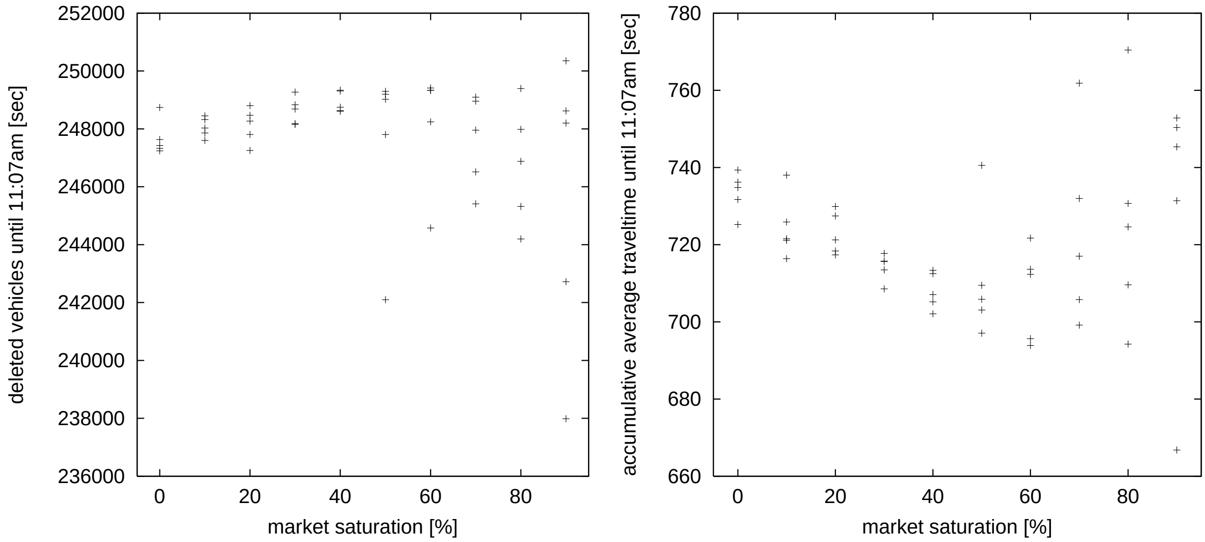

Figure 5.11: *Deletions and accumulative average travel-time (iteration 20)* — Both the number of executed routes (left) and the average travel-time (right) exhibit an improvement for market saturations up to 40%. Above that the improvement is negligible, but the variance increases considerably.

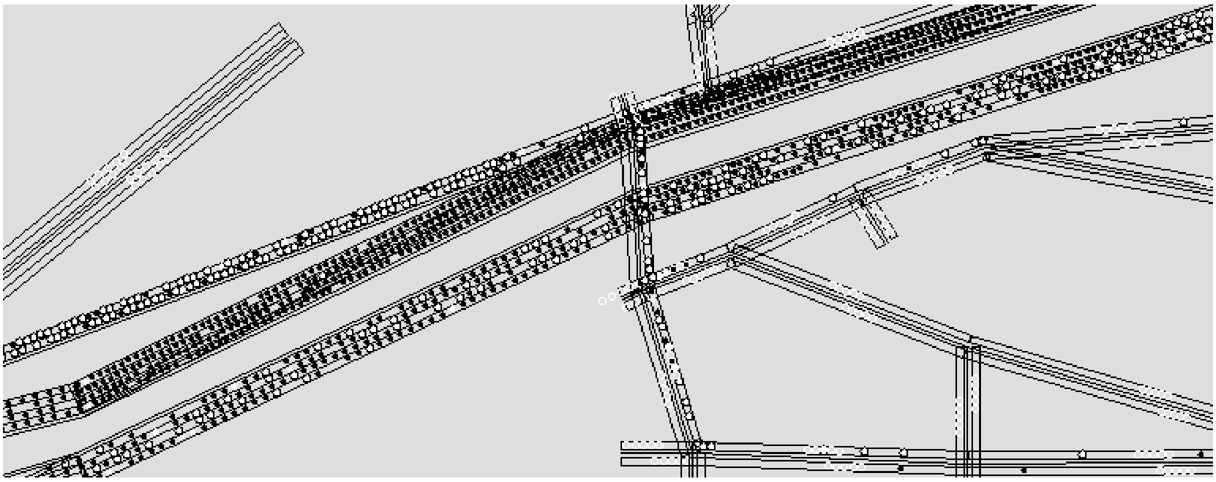

Figure 5.12: *Screen-shot of re-routed vehicles for* $m_{o-l} = 0.3$ — The screen-shot shows subscribers (open circles) that have been re-routed to avoid the jam in East-West direction. The jam mostly consists of non-subscribers (solid circles). Most subscribers end up on a street North of the jam. The network as a whole is used more efficiently.

network will improve. Figure 5.11 shows the number of vehicles that have successfully reached their destinations between 5:00 am and 11:57 am. For market saturations up to $m = 0.4$ we see a slight increase of vehicles. The variance between the five runs of each saturation is relatively small. Figure 5.12 shows the advantageous effect of re-routing: in case of traffic jams, subscribers start using links that have previously been empty.

For higher saturations the variance becomes considerably larger and the average number decreases again. This is due to additional traffic jams caused by the feedback of the re-routing.

The right-hand side of Figure 5.11 shows the accumulative travel-time until noon divided by the the number of vehicles that have reached their destination. There is improvement up to a saturation of $m = 0.6$ although for $m = 0.5$ and $m = 0.6$ the variance has already increased. For saturations over $m = 0.6$ the average travel-time rises again with a large increase of variance: approximately 12% for $m = 0.8$. Compared to the base-case the situation is worse by about 10%.

In contrast to the results reported by Emmerink and Esser, we find that there is no overall positive effect of online routing for all market saturations in our simulation. There may be several causes:

i) The network sizes used in [27, 30] are much smaller than the ones used in PAMINA. Small networks may exhibit effects which qualitatively differ from larger networks because of additional symmetries. Also, any occurring effects are "magnified" because of the small number of available routes.

ii) Both groups used artificial loading curves for the network. In the Dallas case-study an actual rush-hour is modelled with route-plans retrieved from a realistic origin-destination matrix. It seems questionable if artificial demand curves can be used as a base for real-world scenarios.

iii) At least in the case of [27], the process which is used for the iterative dynamic assignment yields a *constant* average travel-time after 100 iterations. This implies a traffic system which is much less influenced by stochastic variations than the simulation with PAMINA.

Other parameter combinations

In the following we will change two major parameters of the re-routing process. In order to quantify the effects caused by these changes it is important to look at the intrinsic variance of the system for runs with identical parameter combinations but different random seeds. We used the original set of base-case runs

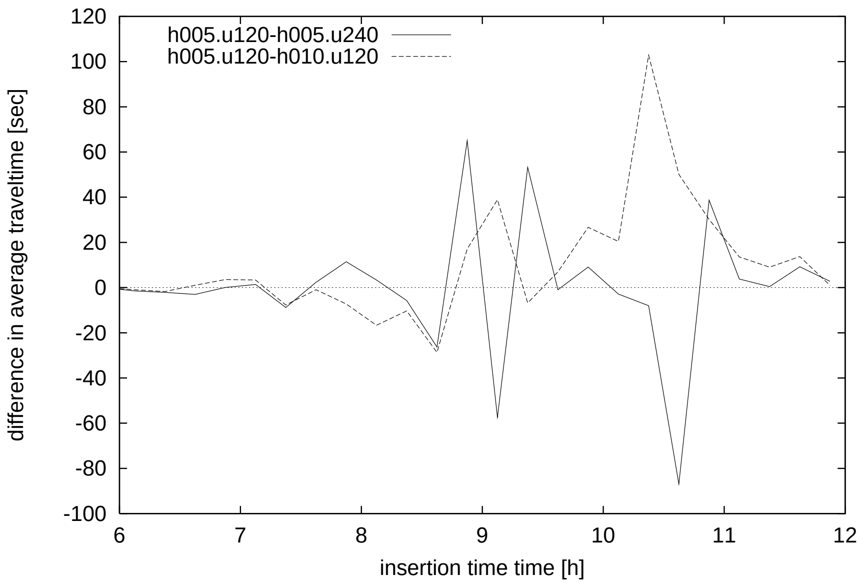

Figure 5.13: *Intrinsic variance of average travel-time (iteration 20)* — Difference between simulations for identical parameter combinations but different random seeds.

as reference and compute the respective difference of average travel-times to the sets of bases cases of the two parameter combinations. The base-cases are equivalent because for market saturation $m_{o-l} = 0$ the online re-planning parameters do not play a role. The results are depicted in Figure 5.13. Before 8:30 am the intrinsic variance is about 20 $[sec]$, between 8:30 am and 9:30 am about 50 $[sec]$, and about 100 $[sec]$ afterwards.

Planning Horizon

The planning horizon h_{o-l}, which is the maximum look-ahead of the shortest-path algorithm, is one of the parameters influencing the quality of the re-routing service. Figure 5.14 depicts the difference of average travel-time between $h_{o-l} = 10$ $[km]$ and $h_{o-l} = 5$ $[km]$. For small market saturations $m = 0.1 \ldots 0.3$ increasing the horizon has a beneficial effect. For higher market saturations the effect is reversed. Note that for $m = 0.5$ subscribers have actually *higher* travel-times at around 9:30 am.

The fairness for $h_{o-l} = 10$ $[km]$ can be seen in Figure 5.15. In contrast to $h_{o-l} = 5$ $[km]$ all market saturations yield positive (at 8:30 am) or indifferent results until 10:30 am. In this case the decreased efficiency for subscribers at high saturations turns out to be beneficial for all drivers.

Update Interval

The next modification was to increase the update-time t_{o-l}^{update} from 120 $[sec]$ to 240 $[sec]$. Figure 5.16 shows that the situation of subscribers becomes *worse* (between 7:45 am and 9:00 am) for all market saturations or does not change at all. This is consistent with the findings of Esser who reports a maximal positive effect of online-line routing for $t_{o-l}^{update} = 100$ $[sec]$. The fairness for non-subscribers is difficult to tell for early simulation hours before 10:00 am (see 5.17) since the difference in average travel-times oscillates strongly. Still, small saturation rates ($m = 0.1$) have a slightly positive influence.

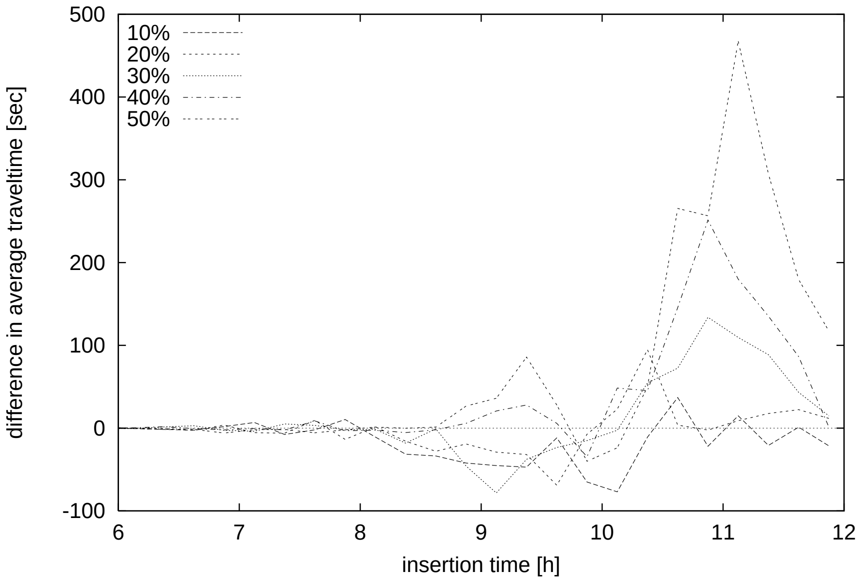

Figure 5.14: *Difference of average travel-time for different h_{o-l}* — The curves show the difference of average travel-time for subscribers using $h_{o-l} = 10[km]$ and $h_{o-l} = 5[km]$. Negative values denote a benefit for $h_{o-l} = 10[km]$.

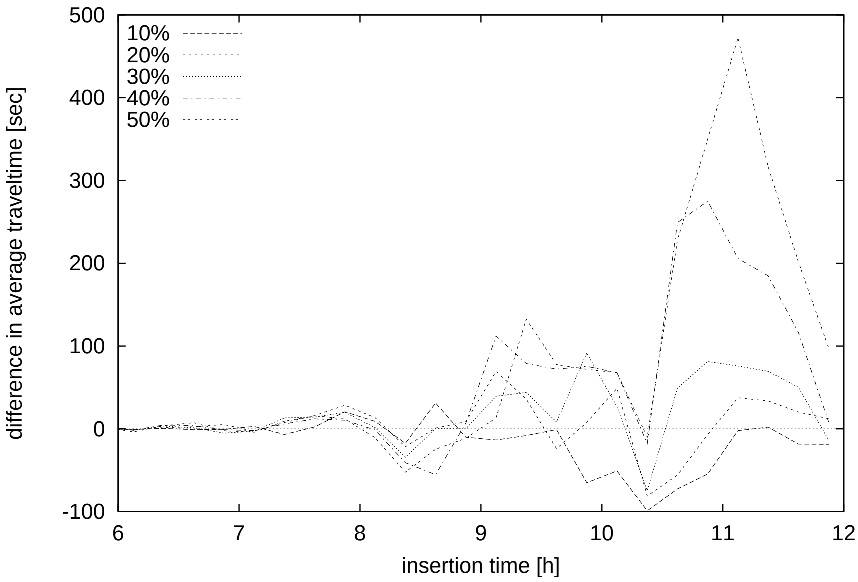

Figure 5.15: *Fairness of $h_{o-l} = 10[km]$ (iteration 20)* — Difference of average travel-time of non-subscribers compared to the base-case. Negative values denote an improvement.

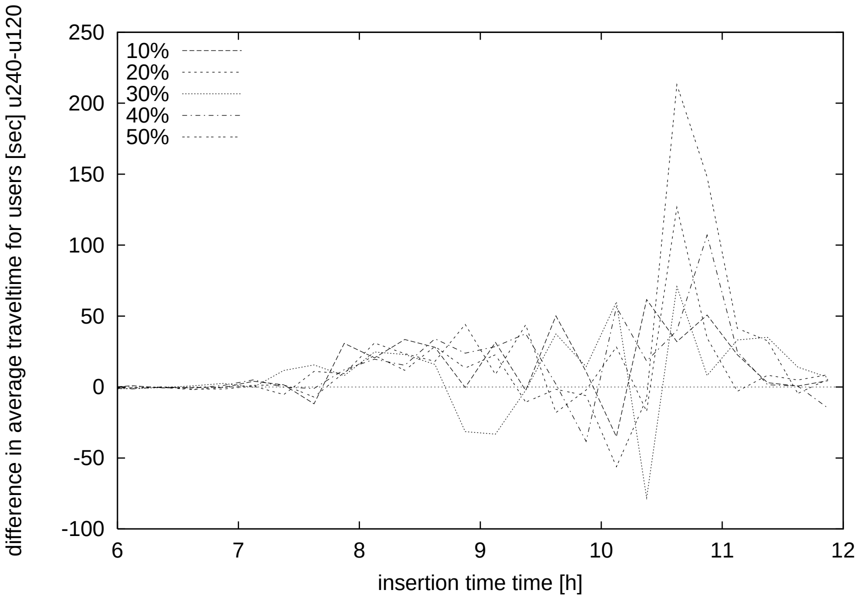

Figure 5.16: *Difference of average travel-time for different* u_{o-l}^{update}. — Difference of average travel-times for subscribers using $u_{o-l}^{update} = 240\,[sec]$ and $u_{o-l}^{update} = 120\,[sec]$. Negative values denote a benefit for $u_{o-l}^{update} = 120\,[sec]$.

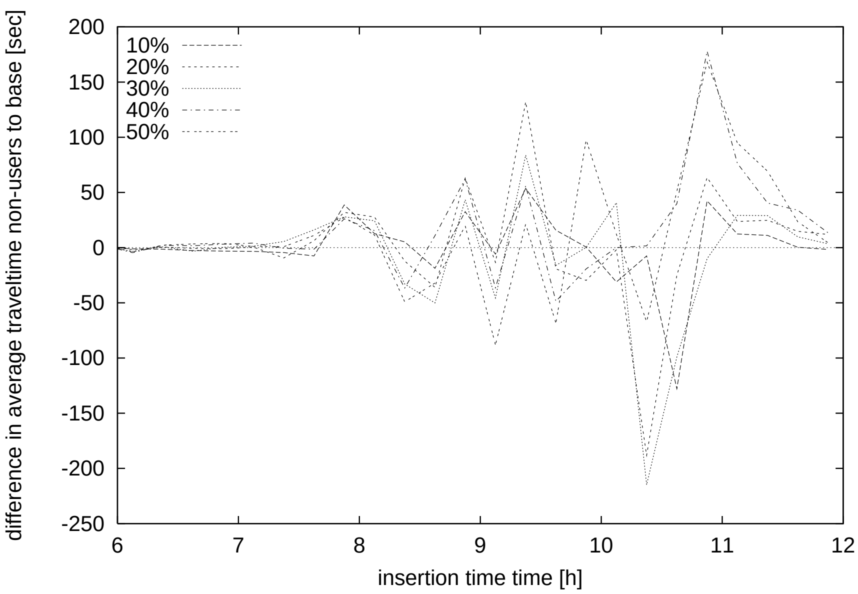

Figure 5.17: *Fairness of* $u_{o-l}^{update} = 240$ *(iteration 20)* — Difference of non-subscribers to base-case. Negative values denote improvement.

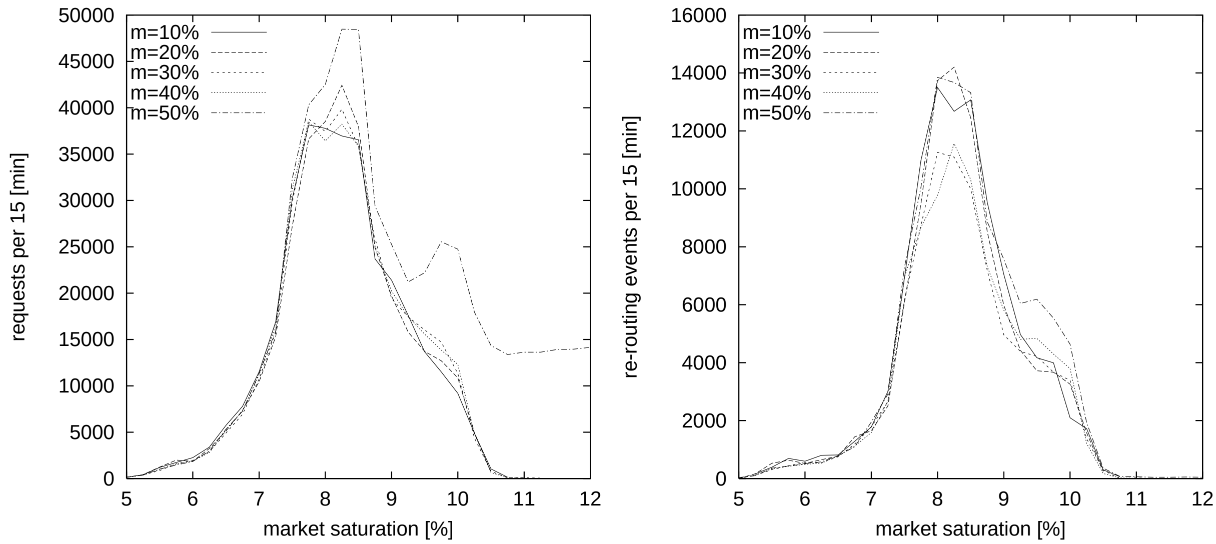

Figure 5.18: *Requests and re-routing events (iteration 30)* — LEFT: the number of potential requests derived from the number of vehicles in the system, and the actual number of requests scaled to a market saturation of 100%. RIGHT: the number of successful re-routing events with at least d_{r-r}^{min} relative improvement.

Accumulative Re-planning Fraction

After 20 iterations using random route re-planning selection, the fraction of plans that have never been re-planned is about 35% percent. As a consequence, 35% of all subscribers have never been re-planned either. It is reasonable to assume that a good fraction of these drivers actually do not use the online routing service in order to adapt to current stochastic traffic conditions, but because their route-plan is completely outdated. We have also seen that further iteration generally improves the traffic situation in the traffic area by distributing traffic more homogeneously. At the same time, by filling up links that have not been used yet, the iteration process withdraws alternative routes from the online-routing. Therefore, it is to be expected that the number of re-routing events decreases. Figure 5.18 shows this result for iteration 30. Compared to iteration 20 (see Figure 5.3) the peak request rate drops from 60,000 to 40,000. The number of successful re-routing events drops even more drastically from 27,000 to about 14,000. Surprisingly, for $m_{o-l} = 0.1 \ldots 0.2$ the quality of individual re-routing events improves. Figure 5.19 shows well-defined peaks at negative delays for these market saturations. Higher saturations show the same behavior as in iteration 20 (see Figure 5.8). At the same time the benefit of being re-routed decreases considerably. Figure 5.20 shows the difference of average travel-times of subscribers compared to non-subscribers. The best absolute improvement for $m_{o-l} = 0.1$ is also about 35 seconds compared to about 200 seconds for iteration 20 (see Figure 5.6). The benefit for higher saturations drops to about 5 seconds which is completely negligible. As for the fairness with respect to non-subscribers the online re-planning shows only minimal impact. Compared to iteration 20 (see Figure 5.10) the peaks (both positive and negative) are generally smaller.

Figure 5.22 shows the overall impact of re-routing on the number of executed routes and the average travel-time. As before in iteration 20 (see Figure 5.11) the situation improves slightly with increasing market saturation. In this case, however, the best results are already reached for $m_{o-l} = 0.3$. Above that, the variance increases noticeably. As mentioned before, the route configuration of this iteration is better adapted to the traffic pattern anyway: the number of executed routes has risen from 249,000 (20) to 256,000 (30), while the average travel-time has dropped from 710 seconds (20) to about 540 seconds (30). Note that for $m_{o-l} = 0.5$ there is an especially bad run with only 247,500 executed vehicles. The same run also causes the non-vanishing number of requests after 10:30 am in Figure 5.18.

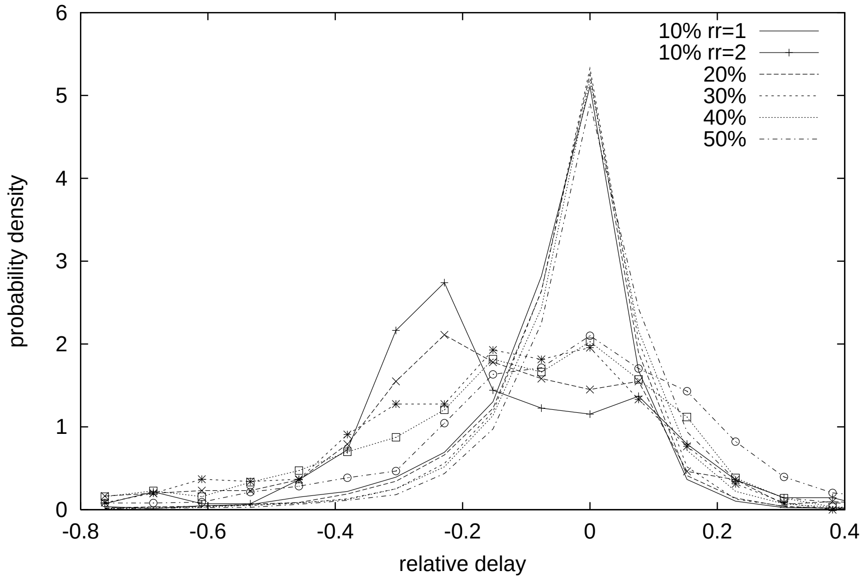

Figure 5.19: *Quality of re-routing suggestions (iteration 30)* — As with iteration 20 the curves for $rr = 1$ are basically independent of the market saturation. For $rr = 2$ the benefit decreases with growing market saturation.

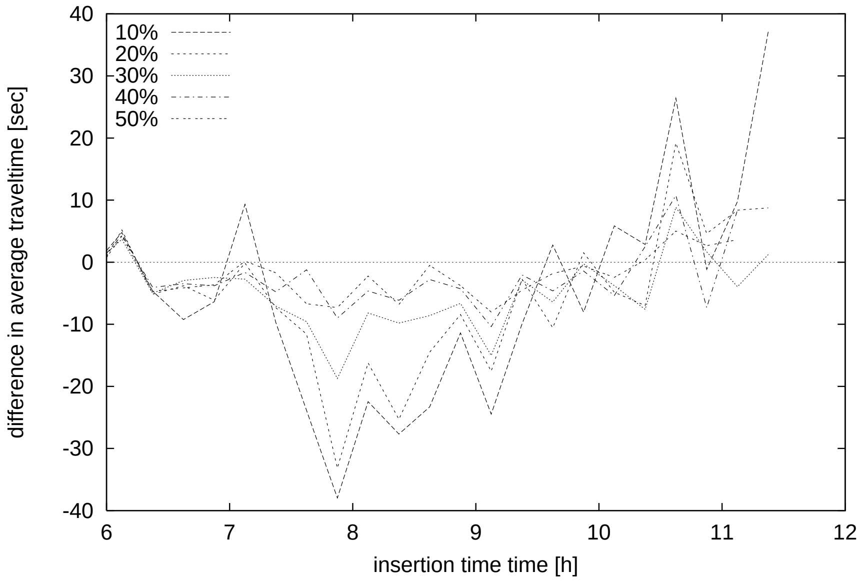

Figure 5.20: *Difference of average travel-times of subscribers to non-subscribers (iteration 30)* — Negative values denote shorter travel-times for subscribers.

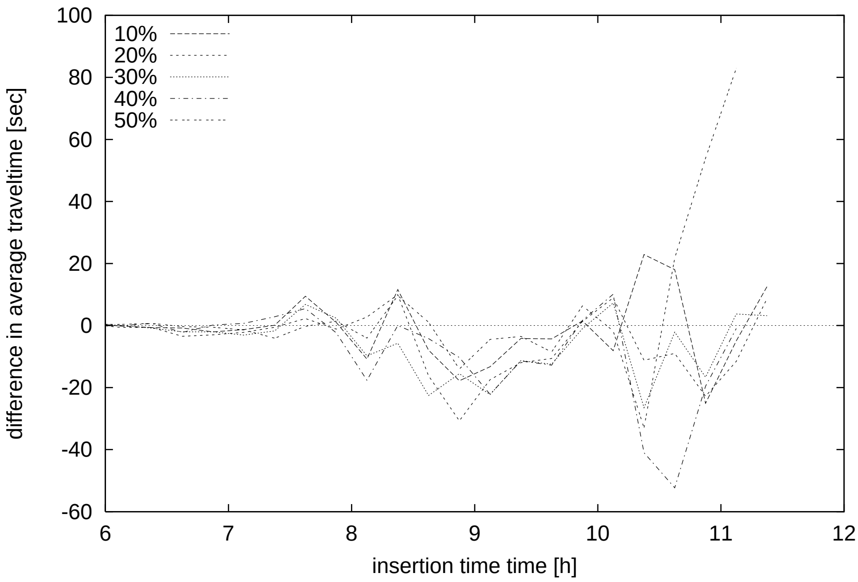

Figure 5.21: *Fairness (iteration 30)* — Difference of average travel-times of non-subscribers to base-case. Negative values denote an improvement.

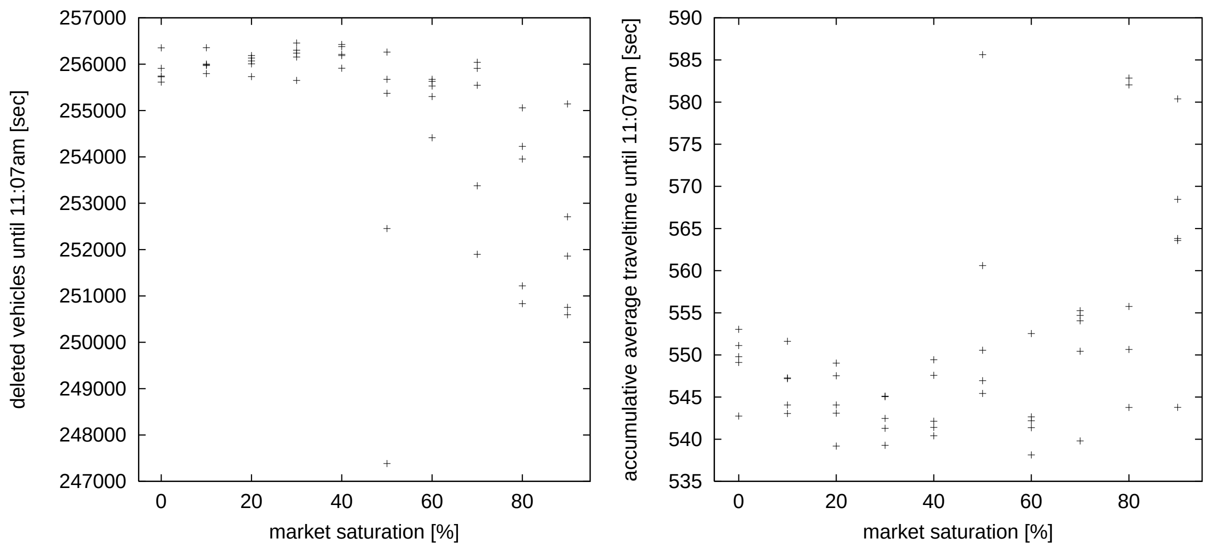

Figure 5.22: *Deletions and accumulative average travel-time (iteration 30)* — Both the number of executed routes (left) and the average travel-time (right) exhibit an improvement for market saturations up to 30%. Above that the improvement is negligible, but the variance increases considerably.

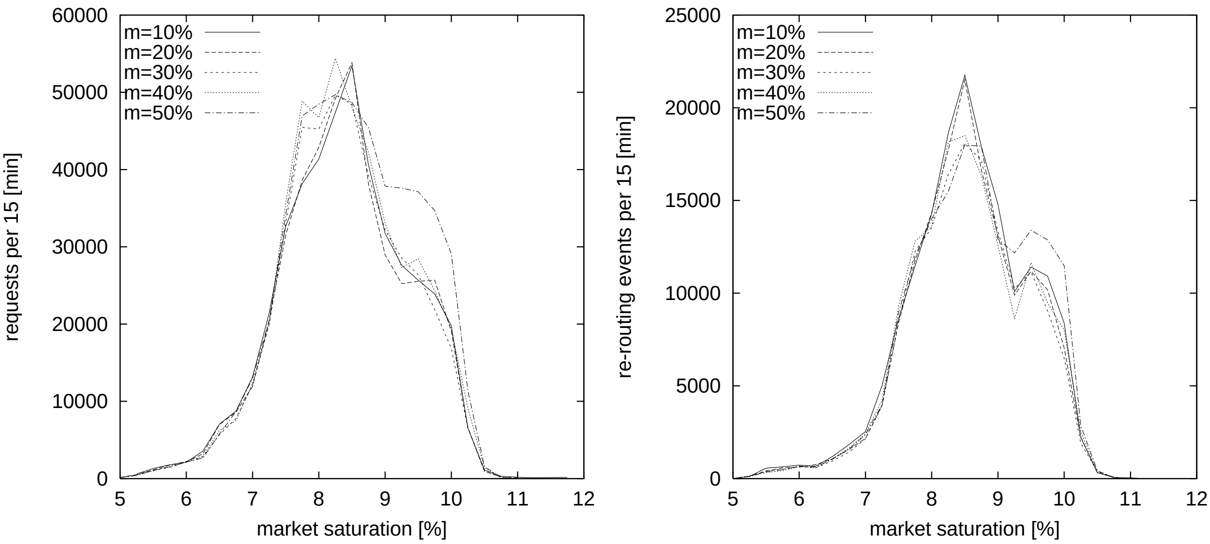

Figure 5.23: *Requests and re-routing events (iteration 20, weak congestion)* — LEFT: number of potential requests derived from the number of vehicles in the system, and the actual number of requests scaled to a market saturation of 100%. RIGHT: number of successful re-routing events with at least d_{r-r}^{min} relative improvement.

5.3.4 Non-recurrent congestion

For all experiments conducted so far, the traffic situation was only subject to stochastic variations. As a whole, the actual link travel-times could be expected to match those which were used by the external router. Now we introduce a *disturbance* into the system which could not be foreseen by the router: for one of the busiest links we will reduce the maximum speed from 4 [sites/sec] to 1 [site/sec] between 8:00 am and 9:00 am. Afterwards the old value will be restored. This is to model a temporary weak congestion which may have been caused by an accident or a construction site. We also use the term *weak congestion* for this scenario.

In a second setup we used a list of the most busiest links inside the study-area to determine ten links which were subjected to the same decrease in maximum speed as described above. Note that we chose the links to be sufficiently far apart to amplify the impact of the disturbance. The location of the links can be seen in Figure 4.21.

Weak congestion

From the non-vanishing number of potential requests in Figure 5.23 we conclude that with the additional disturbance the system has a certain chance of developing a grid-lock. Otherwise the number of requests does not differ noticeably from the undisturbed case. The number of successful re-routing events is even smaller (compare to Figure 5.3). Figure 5.24 shows that already online re-routing with $m_{o-l} = 0.1$ prevents the system from grid-locking: the variance in the number of executed routes drops from about 18,000 to practically zero. The beneficial effect is maintained up to 60%. Above that the variances start increasing again.

Heavy congestion

In the case of heavy congestion the number of requests is considerably higher than in any other case seen before. During the disturbance the number of requests peak at 170,000 per 15 minutes for 50%

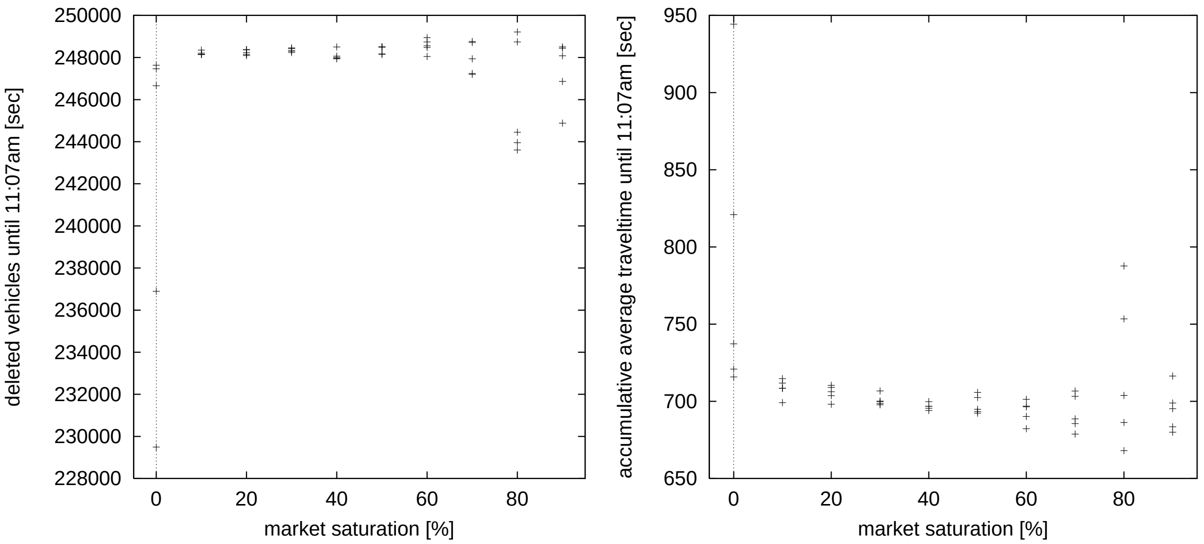

Figure 5.24: *Deletions and accumulative average travel-time (iteration 20, weak congestion)* — Already a market saturation of $m_{o-l} = 0.1$ dissolves the grid-locks caused by weak congestion.

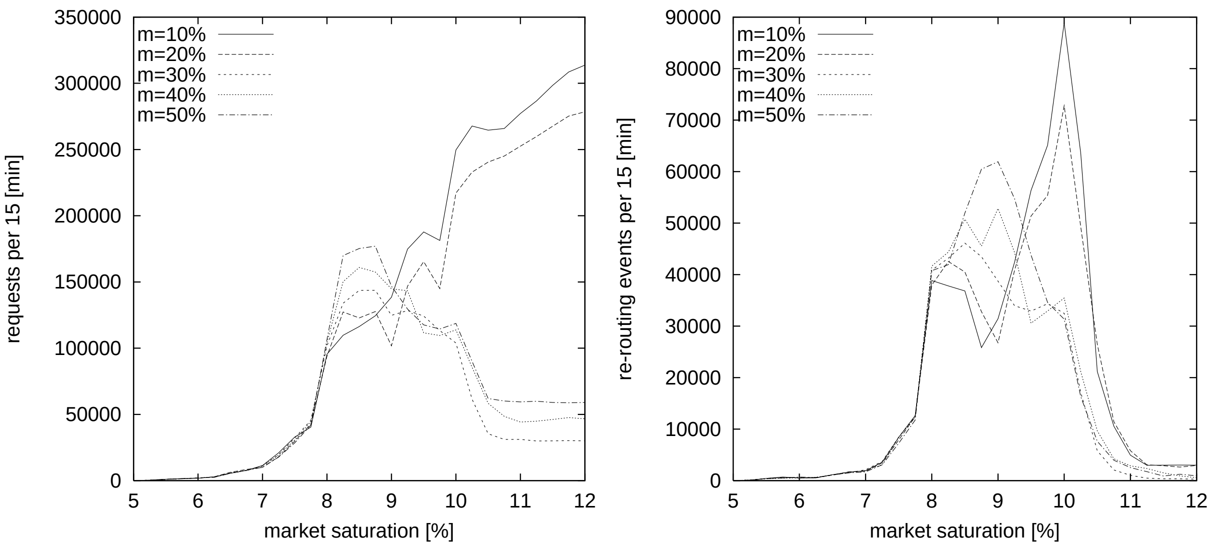

Figure 5.25: *Requests and re-routing events (iteration 20, heavy congestion)* — LEFT: the number of potential requests derived from the number of vehicles in the system, and the actual number of requests scaled to a market saturation of 100%. RIGHT: the number of successful re-routing events with at least d_{r-r}^{min} relative improvement.

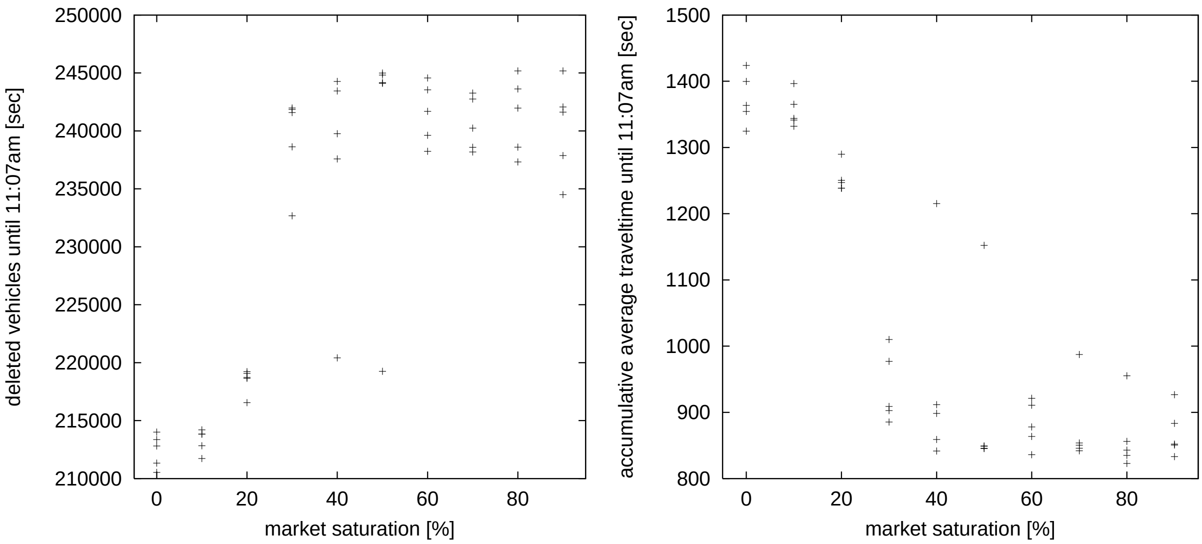

Figure 5.26: *Deletions and accumulative average travel-time (iteration 20, heavy congestion)* — A minimum market saturation of 30% is required to dissolve the grid-locks caused by heavy congestion.

(normalized to 100%). For market saturations of 10% and 20% the system grid-locks again. In this case all runs are affected. Increasing the saturation above 20% resolves the grid-locks. Note that the number of successful re-routing events shows a stronger dependency on market saturation than normal: while for a saturation of 10% there are 35,000 events per 15 minutes during the disturbance interval, for 50% there are up to 60,000. Figure 5.26 shows the beneficial effect of re-routing up to high market saturations: for $m_{o-l} = 0 \ldots 0.2$ the system grid-locks slightly with approx. 215,000 executed routes. For $m_{o-l} = 0.3 \ldots 0.5$ the situation improves, but occasionally there are still grid-locks. For $m_{o-l} = 0.6 \ldots 0.7$ the number of routes has stabilized at around 240,000. The same improvement can be stated for the average travel-time: starting at around 1400 seconds without online re-routing, it decreases to about 900 seconds for 70%.

Chapter 6

Parallel Computing

Although the performance of CPU's has increased tremendously over the last two decades, it becomes more and more evident that eventually technology is approaching the limits imposed by the laws of physics. The velocity of electronic signals is limited by the speed of light, so that the number of switching operations has a natural upper bound given by the area of the chip and its circumference. As a consequence, the cycle times of high-end CPU's (such as the CRAY series, see [34]) have only decreased by a factor of 2 over the last two decades[1]. One approach to increase the performance of a computer is to exploit the parallelism of algorithms by using several CPU's at the same time (i.e. [34, 36, 46, 56, 67, 68]). Ideally, p CPU's can reduce the computing time by a factor of p. Of course, this factor is almost never reached, although good parallel implementations on adequate hardware come very close.

In this chapter, we would like to outline some basic concepts of parallel computing as it is used on today's state-of-the-art computers. We start by introducing some major programming models, with a focus on distributed memory. In the second part, we describe the problem of load-balancing which proves to be decisive for the efficiency of distributed memory implementations. We conclude this chapter by computing an estimate for the maximally obtainable efficiency of the traffic simulator PAMINA. Moreover we will present benchmark results for runs of PAMINA II and PAMINA III.

6.1 Introduction

The reference for all considerations of parallel execution time is the execution time of the fastest known sequential algorithm running on one CPN[2]. Let T_{seq} be the time to solve a given problem on one CPN and $T_{par}(p)$ be the time required on a parallel computer using p CPN's of the same type. The efficiency $e(p)$ of a parallel application running on p CPN's is defined as

$$e(p) = \frac{T_{seq}}{pT_{par}(p)}.$$

The efficiency holds information about what fraction of the available computational power is still used for the application itself. It is the goal of any application to have $e = 1$ independent of the size of the parallel computer. Of course, in reality, the efficiency is below the optimal value and often quickly decreases with the number of CPN's.

[1] Cheaper CPU's such as those used in personal computers have increased by several orders of magnitude. This was, however, just a development to catch up with high-end CPU's.

[2] From now on, we will use the term CPN (short for computational node) instead of CPU for the smallest unit of our parallel computer.

Assuming that the time required on one CPN can be split into a constant part $T_s = const$ independent of the number of CPN's and a "perfectly" parallel part $T_p(p) = T_p(1)/p$ we obtain:

$$e(p) = \frac{T_{seq}}{pT_s + T_p}.$$

As p increases, the efficiency decreases to zero. This corresponds to Amdahl's Law which says that the efficiency of a parallel application is basically given by its sequential component.

The product of the number of CPN's and the efficiency

$$S(p) = e(p)p$$

is called the *speedup* of the parallel implementation. Ideally it scales linearly with p (assuming that $e(p) = const$). In real-world implementations, the necessity to synchronize intermediate results forces CPN's to "wait" for one another and to exchange information. Both the additional idle-time and the communication are the main reasons why $e(p)$ is often below one. We will give an estimate of $e(p)$ for the implementation of PAMINA II in Section 6.4.

6.1.1 Comparison of parallel algorithms

It is also possible to formulate algorithms for which the number P of desirable CPN's depends on the problem size n itself. If the sequential algorithm requires $T_{seq}(n)$ steps to solve the problem, and the parallel version requires $T_{par}(n)$ steps one prerequisite for a good parallel implementation is to demand

$$T_{seq}(n) \notin O(T_{par}(n))$$

That is to say that the parallel algorithm is definitely faster than the sequential counterpart by more than a constant factor.

Often, it is possible to increase computational performance by demanding prohibitively large number of CPN's (i.e. $P(n) \in N^3$ or even worse). In order to capture the costs inflicted by hardware, it is advisable to consider the product of runtime and the number of used CPN as additional criterion. In particular, a parallel algorithm is called optimal, if $T(n) \in O(\log^k n)$ for $k \geq 1$ and

$$P(n)T_{par}(n) = T_{seq}(n).$$

A good overview of parallel algorithms can be found in [46]. Of course, most of these considerations are purely theoretical, since the maximum number of simultaneously available CPN's is always limited by the current state of computer development.

6.2 Computers and networks

Most of the currently available computer systems can be described using a graph with vertices representing CPN's and edges representing communication links between the CPN's (see Figure 6.1). Some additional vertices may be required serving as multiplexer between communication links. Communication consists of transferring data packages (with a size of several bytes to a few kilobytes) from one CPN to another using available links in between. The performance of communication links is basically given by three values:

- The *latency* is the time required to initiate the communication. It is due to the underlying hardware, the software protocol, and the average rate of collisions on the network if there is simultaneous communication.

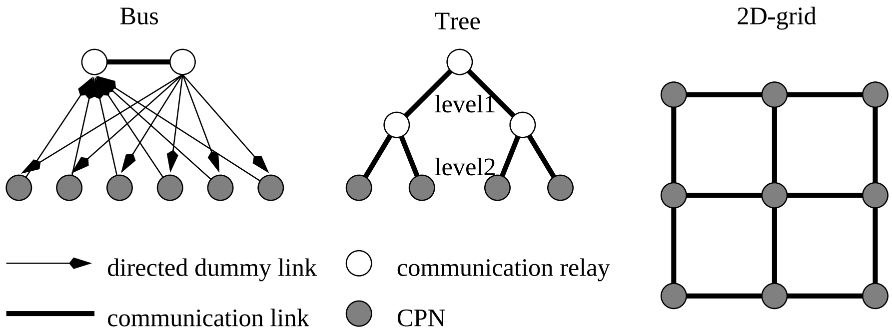

Figure 6.1: *Simple communication networks*

- The *bandwidth* defines how fast a message can be transmitted once the communication has been initiated. It is usually given in units of Megabits per second.
- If a message exceeds a maximum size (also called *packet size*) it has to be split into several smaller *packets*. The time required to re-initiate the communication for the new package corresponds to another type of latency. It is usually smaller than the initial latency.

A table of measurements for these values on common computer architectures can be found in [93].

On its way through a communication network the package has to go through a series of communication links. The maximum number of "hops" over all possible paths is called the *diameter* of the network. Large diameters are one of the main reasons for long latencies, while overloaded communication links are usually the cause for low bandwidths.

The simplest model of network is a bus where all CPN's share the same communication link. This implementation is often found in local area networks based upon Ethernet oder FDDI. In a bus system there is no notion of *local communication* since any communication possibly affects all other participants in a network. The diameter is constant for a bus system.

In a tree communication network the CPN's represent the leaves of a tree organized in layers. The inner vertices of the tree are multiplexer handling the communication between the layers. If p is the number of CPN's the diameter of the tree is $O(\log p)$, which is worse than the bus network. The tree network, however, allows simultaneous communication within sub-trees. Increasing the bandwidth of the links with increasing distance from the leaves allows bandwidth-preserving global communication. This configuration is often called a *fat tree*.

Another communication network that used to be very common is the two dimensional grid in which each CPN has communication links to its four neighbors. The diameter of the network is $O(\sqrt{p})$ which is even worse than above, but the system could be extended very easily by adding new rows or columns to the grid. At least one computer model (Intel Paragon) used the so-called *worm hole routing* to reduce the latency caused by the large diameter: a communication route (like a worm through the grid) was selected before the actual start of the communication. Afterwards all data could be transmitted as though there were only one link between sender and receiver.

There are other more complex versions of communications networks (see [56, 68]) such as the *Butterfly*, *Shuffle-Exchange*, *DeBruijn*, and hyper cube networks. They all have diameter $O(\log n)$.

6.2.1 Distributed memory

Probably, the most straight forward way to build a parallel computer is to use conventional hardware: in the simplest case CPN's are connected by standard communication networks. The so-called *workstation cluster* is one example (see [93]). Since each CPN has its own local memory (and no *direct* access to

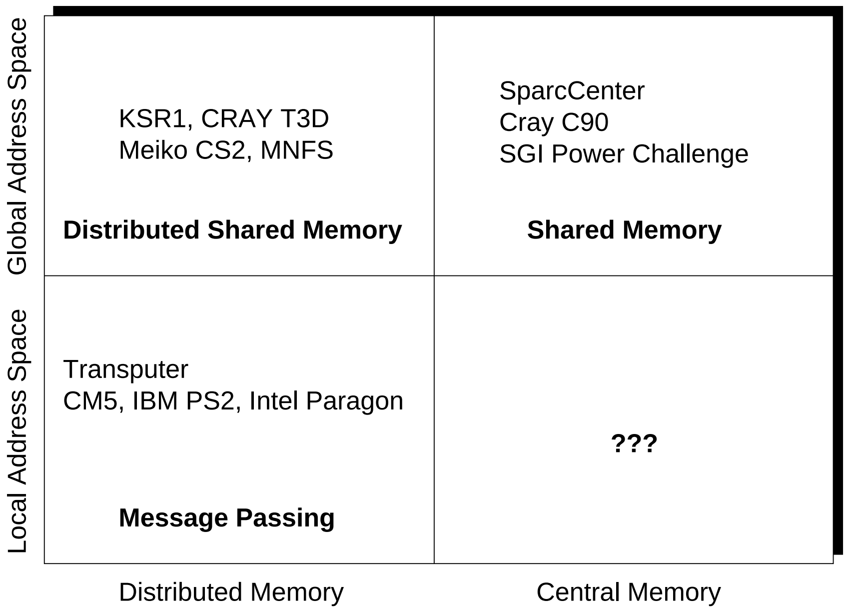

Figure 6.2: *Programming paradigms* — The original version of this figure was taken from [93].

memory located on other CPN's) this configuration is called *distributed memory* system (see Figure 6.2). The reason why many known distributed memory systems still use exotic hardware (Intel Paragon, IBM PS2) is due to the underlying high performance communication network.

The interface between the hardware level and the application level is realized by *message passing libraries*. The best known representatives are *PVM* (= Parallel Virtual Machine, see [97, 120]) and *MPI* (=Message Passing Interface, see [33]). Their main objective is to provide function calls for sending and receiving messages between CPN's. Moreover, they usually offer complex functions for data aggregation, disaggregation, and synchronization. The programming model is often called MIMD (=Multiple Instruction, Multiple Data) because each CPN can have an individual program and local data. In practice, however, it is common to generate only one executable (for all CPN's) that specializes in specific parts of the program depending on its local data configuration.

6.2.2 Other programming paradigms

Besides message passing there are several other programming models, some of which will be outlined next.

Shared memory

Recently, the *shared memory* architecture has gained a considerable market share. It is usually implemented in workstations with several (two to 64) CPN sharing the same address space. Although each processor may have a certain portion of the main memory locally and temporarily assigned to it, every other processor can access any other memory as though it were local[3]. The local memory is not necessarily continuous, but rather fragmented with a constant minimum fragment size. Non-local accesses are performed through a common bus connecting all processor units.

Write-conflicts are resolved without collision as long as they do not affect overlapping memory in the same fragment. For these cases the operating system supplies low-level routines which enable algorithms to serialize access to sensitive data-structures (see [93]). A concurrent read is usually permitted resulting in a CREW memory access model.

[3]So, *Shared Address Space* and *Distributed Address Space* actually describe the paradigms much better

The major advantage of a shared address space is that algorithms only require small structural changes from their sequential counterparts. The programmer only has to look for possible collisions of the CREW model and serialize the program flow locally (i.e. [11]). Of course, this approach does not automatically guarantee good efficiencies for a large number of CPN's since it does not consider the question of data locality. As in the distributed memory model, non-local data access generates communication (hidden from the programmer) on the underlying bus-system. Nevertheless, since the number of CPN's is usually small and the bus-systems are increasingly powerful, linear speedups are often possible with little effort.

Virtual shared memory

A variation of the shared memory paradigm are so-called *virtual shared memory* systems. The CPN's share the same address space as in shared memory, but the memory is physically distributed over all CPN's. Using a library that emulates the global address space any distributed memory machine can be used as a virtual shared memory machine. Pfenning [93] implemented a simple traffic simulation model using a virtual shared memory approach.

Vector computers

Vector computers are among the oldest parallel computer systems. They were used to compute the same series of instructions for a large number of data items. Applications are usually restricted to those that exhibit very regular data structures such as grid oriented simulation models and matrix operations. In these areas vector computers still offer the fastest implementations.

For the problem of traffic simulation which is highly irregular due the graph structure of the underlying traffic network vector computers are not well suited.

High level language support

In addition to the low-level programming paradigms presented above there is growing support at higher levels of program development. Auto-parallelizing compilers are able to detect simple parallelism implicitly or explicitly coded in conventional Fortran or C(++) source-code. They usually make use of *threads* (see i.e. [122]) to split the current single line execution into multiple threads of execution which are assigned to several CPN's on shared memory computers.

Also, there are extensions to high-level programming languages like HPF (=High Performance Fortran) or CC++ which supply additional compiler directives or language elements that can be used to describe the parallelism hidden in a program.

The operating system feature *remote procedure call* can be found on almost all Unix-platforms. They behave like a local procedure call, only that all parameters to the procedure are coded and transferred to a remote CPN. The result of the computation is also encoded and sent back to the originating CPN. The programming language SR [89], for example, supports remote procedure calls as a language component.

6.3 Load-balancing

One of the main objectives of parallel computing is to accept a problem instance, split it into several smaller sub-units and assign those to CPN's for parallel execution. At certain intervals — but at least once at the end of the execution phase — the sub-units have to exchange information to be able to continue or to produce a common result. These events are called *synchronization points*. The ratio of the time that CPN's can work independently of the time that they have to wait for synchronization is decisive for parallel efficiency. For reasons of clarity, we would like to differentiate between two extreme cases of dependencies between sub-units.

- In the first case, the sub-units (tasks) are almost independent and the waiting time for synchronization is short relative to the execution time. For a given number of tasks n and CPN's p (with $n \gg p$) the main objective is to distribute tasks in such a way that none of the CPN's is running idle. Note that the individual runtime of each task may vary considerably. This is usually done by maintaining a short queue of pending tasks for each CPN.

  Böhm and Speckenmeyer [12, 13] have shown that a fast SAT-solver can be implemented by redistributing tasks generated by the internal branching algorithm at certain intervals. With their parallelization scheme (called PLB for Precomputation-based Load-Balancing) they obtained efficiencies above 0.95 for a two-dimensional communication network up to 256 CPN's and efficiencies above 0.85 for a linear communication network of the same maximum size. Meisgen [65] applied the PLB algorithm to the parallel simulation of biological aging.

- If the execution time is very short and the data dependencies are strong, it is impossible to "queue up" tasks. Instead, it must be the goal to make the execution time of all tasks as equal as possible. Also, it does not make sense to have more parallel tasks than available CPN's since. Therefore, for the whole problem must be carefully split to yield exactly p tasks. Knecht and Kohring [52] used a parallel implementation for the simulation of granular materials with short-range interactions caused by inelastic collisions. Depending on the problem size up to 1000 synchronizations per second (on 128 CPN's) were necessary rendering the problem extremely susceptible to load imbalances.

Traffic simulation belongs to the second group because the rule-set of the CA-model has to be evaluated for each time-step. Depending on the so-called real-time-ratio, which determines how may simulation seconds are computed in one wall-clock second, there may be tens of synchronizations necessary per second.

From the examples mentioned above, it should be clear that the balancing of load is the key technique to achieve good efficiencies on parallel systems. It is useful differentiate the load-balancing schemes according to the time-scale at which they take place:

- If the load-balancing solely takes place before the start of the application, it is called *static*. This requires that the execution-time of all tasks are known beforehand and that they *do not* change during runtime. In some cases (i.e. on regular grids) the distribution of load may be trivial that it can be done *by inspection*, that is by examining the given problem geometry.

- If the load changes during runtime, it may be desirable to change the allocation of tasks to the CPN's. If the change is so slow that the extra-time required for balancing can be neglected compared to the execution time itself, it is called *quasi-dynamic* or *adiabatic*.

- If the load changes very quickly so that constant allocation corrections are required, it is necessary to consider the trade-off between letting the application run out-of-balance or spending too much time on finding a better allocation. In this case the load balancing is called truly *dynamic*.

In the following we would like to concentrate on the static and the quasi-dynamic balancing because they are relevant for the implementations of PAMINA II and PAMINA III.

For a given problem let $T_1 \ldots T_N$ be the tasks with execution time t_i. Assigning subsets of tasks to CPN's we will result in dependencies between CPN's which require communication. Let c_{ij} be the time task T_i and T_j will spend on communication if they end up on different CPN's. Further let L_k be the accumulative load generated by all tasks on CPN P_k computed as

$$L_k = \sum_{j \in P_k} l_j + \sum_{i \in P_k, j \notin P_k} c_{ij}$$

If L^{min} is the minimum, and L^{max} the maximum of all L_k, one way to define an optimal distribution is to demand

$$L^{max} - L^{min} \stackrel{!}{=} minimal.$$

6.3.1 Static balancing

We start out with the simplest allocation method: assign T_i to CPN P_j for $j = i \mathrm{mod} p$. This approach is called *cyclic mapping*. It works reasonably well if $n \gg p$ and the $c_{ij}(\ll l_i)$ are negligible. In this case it almost equivalent to *random mapping* where the assignment is done randomly. The latter may even work better if the tasks have correlations in their execution times.

In some cases there is additional structural information available. Since most real-world problems are defined in a two or three dimensional space, the tasks T_i are often associated with locations p_i. Moreover, many real-world dependencies are known to be local or at least short-range so that the c_{ij} are only non-zero if the corresponding locations p_i and p_j are close. The *recursive orthogonal bisection* iterative splits all available locations into smaller and smaller regions by using the X and Y (and Z) coordinates. The regions are also called *domains* which makes the whole process a *domain decomposition*. It yields good results, if the locations are homogeneously distributed and the c_{ij} are of comparable value. Note that due to the interval property of the coordinates the resulting domains are convex. PAMINA uses the orthogonal bisection for the initial distribution of the street network (see A.1.3).

Even more specialized is the *recursive spectral bisection* [95] which transfers the graph bisection onto a matrix eigenvalue problem. In contrast to the simple bisection it also works satisfactorily for heterogeneously distributed locations and/or communication dependencies c_{ij}. It is, however, considerably more computationally demanding because of the required matrix eigenvalue solution. An example can be found in [34].

A number of libraries (see i.e. [43, 96]) are available which implement one or several of these algorithms.

6.3.2 Quasi-dynamic balancing

The quasi-dynamic load-balancing during runtime is commonly split into two major phases (see [59]):

decision phase During this phase each CPN has to decide if load is to transferred to another CPN. The decision can be based upon (a) local data which is locally available from the CPN's neighbors in the communication network, or (b) global data, which was retrieved using global communication. Usually, either the receiver or the sender makes the decision in order to avoid collisions. But combined decisions are possible.

migration phase In the second phase, the CPN accepts or transfers load (or both). Again as before, it is useful to differentiate between local migration to or from neighbors and global migration to other CPN's in the network.

All in all, there is a variety of eight different strategies which are described by the code $SDSM_d$ where S must be replaces by the scope <u>*L*</u>*ocal* or <u>*G*</u>*lobal* and d must be replaced by the decision strategy <u>*s*</u>*ender*, <u>*r*</u>*eceiver*, or *sr* for combined decision. PAMINA uses a $LDLM_s$ strategy, which corresponds to a <u>L</u>ocal <u>D</u>ecision <u>L</u>ocal <u>M</u>igration method where the <u>s</u>ender decides how much load is being transferred.

Local methods

LDLM methods are also called *diffusion methods* because the migration of load along load gradients strongly resembles the flow of heat causes by temperature gradients. Their main advantage is that all communication is kept local which completely decouples load balancing in different regions of the network. Xu and Lau [130] have found an optimal set of diffusion parameters for the diffusion method in a regular mesh network.

If the load is locally well equilibrated small disturbances can trigger load transfer which may be reversed during the next migration. This ping-pong effect would cause additional undesired communication. A common remedy is to impose a minimum threshold for any migration. Unfortunately, this may result in a load gradient across the communication network. This is why local methods are often enhanced by global corrections.

Another question of the migration phase is what tasks are to transferred to neighboring CPN. In case of a local migration strategy the selection of tasks takes place along the boundary between CPN's. In principle, any task along the boundary can be chosen. It proves to be advantageous, however, to chose those tasks that are furthest away from the center of the domain to be transferred first. This way, the sending CPN "peels" off its excess at the surface (see [23]). Also, when selecting tasks, special care should be taken that those are chosen which have least dependencies with other tasks remaining on the CPN (see [50]).

Global methods

As we have seen load gradients are very common with local load balancing methods. One way out of this dilemma is to use global information. The most straight forward approach is to use a static load balancing strategy and apply it to the current configuration of execution times of all tasks. If afterwards all tasks are simply reassigned, the "dynamic" load-balancing is no more than a concealed iterated static load-balancing.

The aforementioned PLB [13] algorithm uses a prefix computation in the graph of the CPN topology to determine the optimal shift of load. Note that all transfers are done purely local. As a consequence, some CPN's may have to transmit *and* accept load during a balancing step. A similar global diffusion method was suggested by Horton [44].

6.4 Scaling behavior of the traffic simulation

Nagel and Schleicher [83] have investigated the parallel efficiency of traffic with periodic boundary conditions on several computer platforms. In this section we will consider whole traffic networks.

Technically, T_{seq} is the time needed to run the simulation on a single-CPN machine with the *fastest* available sequential algorithm. Due to the domain composition we can assume for traffic simulation T_{seq} to be $T_{par}(1)$, that is, we use the same algorithm in both cases. Furthermore, we do not associate $T(p)$ with the runtime of the whole simulation, but with the time needed to execute a single time-step.

Since we have a time-step driven simulation with implicit synchronization before each sub-time-step, $T(p)$ is dominated by the *slowest* CPN with execution time $T_{max}(p)$. Thus, for further investigation we define the efficiency as

$$e(p) = \frac{T(1)}{pT_{max}(p)} = \frac{T(1)}{p(T_{s,max}(p) + T_c(p))}.$$

In the following we will show how to compute $T_{max}(p)$ as a sum of communication time $T_c(p)$ and simulation time $T_{s,max}(p)$ which will be derived from parameters depending on the *traffic map*, the *computer hardware*, and some *simulation characteristics*.

Map parameters

In principal, the street network can be of any shape and any topological structure. In order to estimate the communication imposed by boundaries, however, it is useful to make the following assumptions about the graph $G = (N, E)$ associated with the street network:

- The graph is shaped like a square. We use a normalized area of 1.

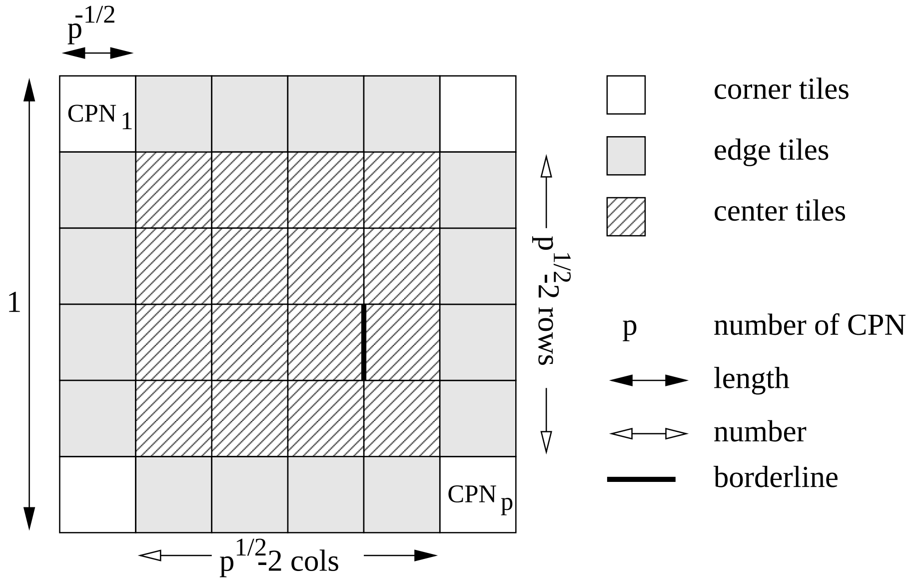

Figure 6.3: *Decomposition of map area into tiles of equal size*

- The density of vertices (or rather super-vertices; see below) is homogeneous.
- The graph is planar.

Suppose we have a map of n vertices and e edges, where each vertex represents an intersection or ramp of the street network and each edge the street segment between two vertices. We count both street segments directions as *one* edge. Further we identify N so-called *super-vertices* by removing all vertices of degree two from the network. The resulting E *super-edges* connect super-vertices which used to be connected by a path of normal edges.

We use the sum length L of all edges as the size of a map. The sequential simulation time $T(1)$ is assumed to be proportional to L which is true for the cellular automata implementation within a certain range of average edge lengths. During the initial distribution onto p CPN's, the whole network is split into p sub-nets of almost equal size. Let us first estimate how many neighbors one CPN has *on the average*: imagine a square area (see Figure 6.3) — representing the map — split into 2^i small tiles of equal size. The number of all relationships $N_n(p)$ between neighboring tiles is equivalent to the number of common borders with other tiles: two for each of the four corner tiles, tree for each of the $4(\sqrt{p}-2)$ edge tiles, and four for each of the $(\sqrt{p}-2)^2$ internal tiles. Note that $N_n(p)$ is only integer for even i. The average number of neighbors $n_n(p) = N_n(p)/p$ turns out to be

$$n_n(p) = \frac{8 + 12(\sqrt{p}-2) + 4(\sqrt{p}-2)^2}{p} \quad \in O(1).$$

The function is depicted on the left-hand side of Figure 6.4: after a steep increase for small numbers of CPN's it slowly approaches the value four of an infinitively large system of CPN's in which all tiles are internal. Note that — although the actual number of neighbors is slightly higher — $n_n(p)$ is independent of the map size.

Number of boundary edges

The number of neighbors $N_n(p)$ and the width $l_b(p) = \sqrt{1/p}$ of a border (see Figure 6.3) can be used to deduce the length of the complete *borderline* between CPN's: $L(p) = N(p)l_b(p)$. Since we assumed the

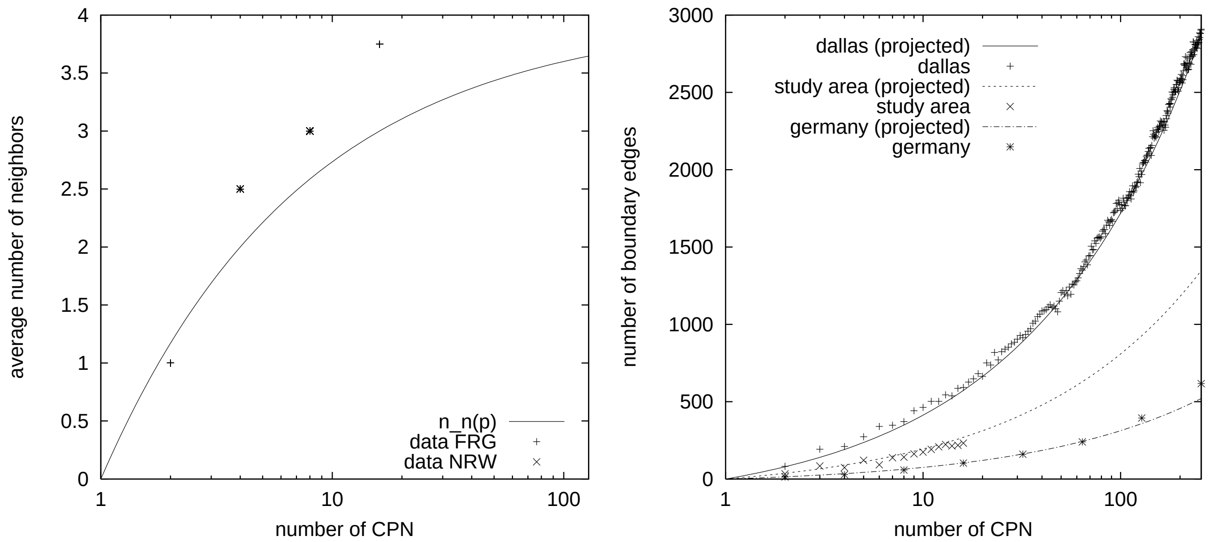

Figure 6.4: *Average number of neighbors per CPN and overall number of boundaries* — The functions $n_n(p)$ (left) and $B(p)$ (right) are plotted along with some results from actual orthogonal bisections.

graph to be homogeneous and planar, we obtain an super-edge density of

$$\varrho_E = \sqrt{N} a \frac{deg(G)}{2} = \sqrt{N} \frac{2E}{2N} = \frac{aE}{\sqrt{N}}$$

along any cut through the map area, where $deg(G)$ is the average degree of a vertex and a is a geometrical correction factor. The number of inter-CPN boundary edges $B(p)$ is equal to the length of the borderline times the super-edge density:

$$B(p) = \varrho_E L(p) = \frac{aE}{\sqrt{N}} N_n(p) l_b(p) \quad \in O(\sqrt{p})$$

We used the Dallas data to derive the correction factor $a \approx 2^{-1/2}$. On the right-hand side of Figure 6.4 $B(p)$ has been plotted for three different maps along with results from actual orthogonal bisections. We define the number of boundary edges per CPN as $b(p) := B(p)/p \in O(p^{-1/2})$.

Granularity

Ideally, all sub-nets of the street network are of equal size resulting in equal load on all CPN's. Technically, this is not possible since the traffic network does not allow an arbitrarily fine decomposition of its elements. A simple assumption is that the *atomic* size of the street network is proportional to the relative size e of an average street edge with respect to the whole network. We define the optimal granularity as $g_{opt} = 1/e$. This optimum, of course, can hardly be achieved. In particular, most dynamic load balancing strategies define a threshold for the minimum load difference between neighboring CPN's, which has to be exceeded before any transfer is done. Thus, we also consider a constant granularity g_{const}.

The value g refers to neighboring CPN's. In a network of p CPN's, the repeated granularity between the $d(p) = \sqrt{p}$ layers of d CPN's each from one side of the network to the other will result in a load imbalance depending with the number of layers. In order to estimate the load difference between the least and most heavily loaded CPN, we introduce a runtime correction $f_{grad}(p)$ for the most heavily loaded CPN. For a perfect system without granularity f_{grad} is zero. For a optimally balanced system f_{grad} is constant.

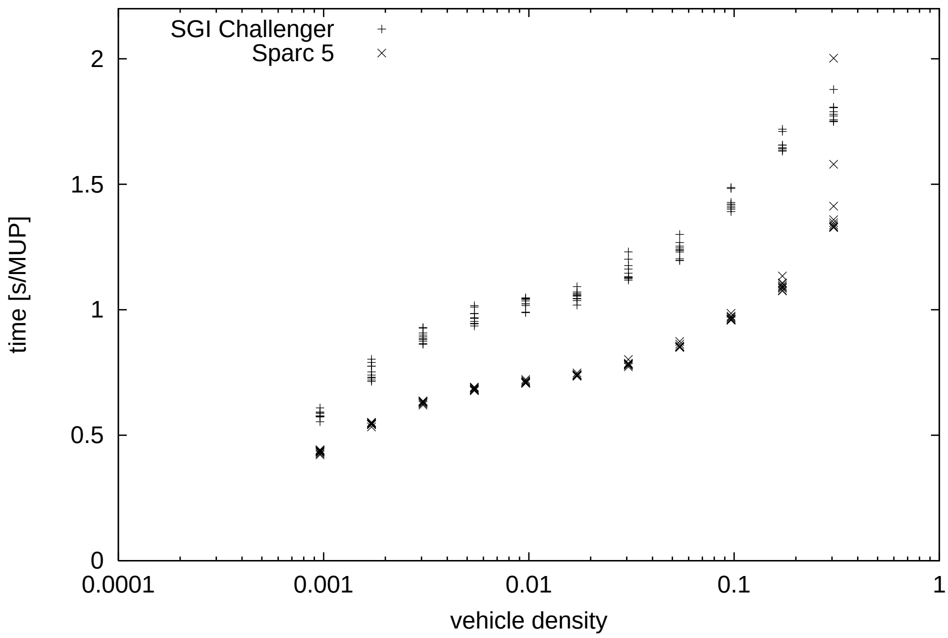

Figure 6.5: *Dependence of performance on vehicle density* — The performance of the CA-model only varies by a factor 2 in the density range 0.003 ... 0.3.

Moreover, we consider the cases of a linear and an exponential gradient:

$$f_{grad}(p) = \begin{cases} 0 & & \text{no granularity} \\ \frac{1}{2}g & \in O(1) & \text{optimal gradient} \\ \frac{1}{2}d(p)g & \in O(\sqrt{p}) & \text{linear gradient} \\ \frac{1}{2}((1+g)^{d(p)}-1) & \in O(2^{\sqrt{p}}) & \text{exponential gradient} \end{cases}$$

6.4.1 Hardware parameters

Hardware parameters describe the performance of the computer hardware with respect to both computation and communication.

Computational performance

Using the sequential time $T(1)$ required to execute one time-step and the size of the map L, we define the performance value $P = T(1)/L$. For a high-fidelity implementation, $T(1)$ slightly depends on the actual density of vehicle in the system (see Figure 6.5). In that case all measurements have to be carried out at the same overall density (e.g. $\varrho = 0.1$).

Communication performance

As far as the communication performance is concerned, we differ between the *application-level (a-l)* and *low-level (l-l)* performance. The application level performance refers to the time T_{c1}^{a-l} required to transfer one boundary to a neighbor and the corresponding latency T_{cl}^{a-l}, explicitly including the time for *retrieving* and *packing* data on the sending CPN as well as the time for *unpacking* and *storing* data on the receiving CPN. The achievable application-level bandwidth per CPN is considerably lower than the theoretical sustained bandwidth of the underlying communication network which we assume to be c_{net} bytes per second. We will see that the amount of simultaneous communication may actually exceed the physical bandwidth, so that we have to include the low-level communication time T_{c1}^{l-l}.

Communication network topology

Since we use domain decomposition as parallelization scheme, we have to *map* the resulting traffic sub-networks (tiles) onto the CPN's of the underlying communication network. In case of a bus-system this is trivial since *all* communication paths between CPN's overlap and are therefore equivalent. The drawback is that communication is *completely* serialized. We will see that the overall communication volume increases with the number of CPN's used in the computer network. The resulting effect is also known as *competition for bandwidth* (see [34]). Often, communication proves to be the bottleneck of the simulation in a bus-topology.

In case of the two-dimensional grid (with $\sqrt{p}$ rows of $\sqrt{p}$ CPN each) we assume that the mapping can be done in such a way that communication between a traffic sub-network (tile) and its neighbors will only result in serialized communication between its associated CPN and other CPN's within a certain constant distance[4]. Any other communication outside that radius is to be carried out in parallel.

We neglect the issue of *mapping* logical subnets onto *physical* CPN's handling the subnets. Monien [68, 66] and others [104] have investigated this problem in depth.

6.4.2 Simulation parameters

Since the traffic simulation is time-step driven, the simulation parameters mainly describe the coarse structure of a time-step, which is divided into n_{sub} sub-time-steps. We try to take an administrative overhead into account by defining a factor f_{over} denoting what fraction of the run time is wasted on administration if the normal number of sub-time-steps is assumed to be $n_{sub,0}$. If $T(1)$ is the time to execute with $n_{sub,0}$ sub-steps on one CPN then the actual optimal[5] execution time with n_{sub} sub-steps will be

$$T_s(p) = \frac{1}{p}\Big(1 + \frac{n_{sub} - n_{sub,0}}{n_{sub,0}} f_{over}\Big)T(1) = \frac{1}{p}\Big(1 + f_{adm}(n_{sub})\Big)T(1) \quad \in O(p^{-1})$$

Two other simulation parameters are the average length b_{size} of a single boundary in bytes and the corresponding message header length per neighbor b_{header} also in bytes. b_{size} will depend on the density of vehicles in the street network as well as the resolution of the vehicle representation. Also, if vehicles are routed, the length of the route-plan may be a decisive factor. b_{header} can be regarded as constant.

6.4.3 Combining components

Now that we have derived all parameters, we can combine them to estimate the execution time $T(p)$ for one time-step on p CPN's. The execution time $T_{s,max}(p)$ for simulation part on the slowest CPN amounts to the optimal execution time $T_s(p)$ corrected by the maximum load difference $f_{grad}(p)$:

$$T_{s,max} = \frac{1}{p}\left(1 + f_{adm}(n_{sub}) + f_{grad}(p)\right)T(1)$$

During the simulation, each sub-time-step will require a complete exchange of boundaries between neighboring CPN's. Let us first handle the application-level boundary communication. Assuming that communication of each CPN to its $n_n(p)$ neighbors is sequential, latency will require $n_n(p)T_{cl}$ and transmission $b(p)T_{c1}$ at application-level.

[4]The *distance* between two CPN's in the grid topology is defined as the number of inter-CPN connections a message has to cross while traveling from sender to receiver.

[5]At this point, we still neglect the load imbalance

So far the time required for communication is regarded as independent of the communication network saturation. This is only reasonable for peak communication rates below the performance of the network. Let us first define the total amount of *simultaneous* communication C_{bpt} during a sub-time-step. At this point we have to differentiate between the bus topology and the two-dimensional grid topology

$$C_{bpt}(p) = \begin{cases} N_n(p)b_{header} + B(p)b_{size} & = O(\sqrt{p}) \quad \text{bus} \\ n_n(p) \quad (\quad n_n(p)b_{header} + b(p)b_{size} \quad) & = O(1 + p^{-1/2}) \quad \text{2-D grid} \end{cases}$$

The additional factor $n_n(p)$ for the two-dimensional grid is included to estimate the simultaneous communication between nearest neighbors.

Assuming a maximum bandwidth of c_{bps} of the communication network, the new lower bound for the low-level transmission time of all boundaries is $C_{bpt}(p)/c_{net}$. We completely neglect latency for low-level communication. The total communication time is the sum of both low-level and application-level communication

$$T_c(p) = n_n(p)T_{cl} + b(p)T_{c1} + \frac{C_{bpt}(p)}{c_{net}}.$$

It proves to be practical to replace the absolute values of T_{cl}, $T_c(p)$, T_{c1}, and c_{net} by their relative values $t_{cl} = T_{cl}/T(1)$, $t_c(p) = T_c(p)/T(1)$, $t_{c1} = T_{cl}/T(1)$, $C_{net} = T(1)c_{net}$ with respect to the sequential execution time $T(1)$. Using this convention, the efficiency turns out to be

$$\boxed{e(p) = \left(\underbrace{n_n(p)t_{cl}}_{\text{latency}} + \underbrace{b(p)t_{c1} + \frac{C_{bpt}(p)}{C_{net}}}_{\text{bandwidth}} + \frac{1}{p}\left(1 + \underbrace{f_{adm(n_{sub})}}_{\text{overhead}} + \underbrace{f_{grad}(p)}_{\text{gradient}}\right)\right)^{-1}}$$

6.4.4 Measuring parameters

We describe how to to obtain the hardware parameters T_{cl}, T_{c1}, and b_{size}. In contrast to the preceding section, we now have to to select a specific implementation of the simulation. We will use PAMINA II as simulator and an excerpt of the *NRW* network as test-bed.

Boundary size

The average boundary size depends on the density of vehicles in the system and its resolution [103]. A boundary may be "empty" or contain as many vehicles as there are lanes on the corresponding inter-CPN street segment. A low resolution boundary only contains site information of the cellular automata, whereas a high resolution boundary contains site and vehicle information.

We measured at a low density ($\varrho = 0.01$), at a medium density ($\varrho = 0.05$), and a high density ($\varrho = 0.1$) which is close to the flow-maximum of the underlying traffic model. The measurements were carried out as follows: From one CPN a varying number (1 through 11) of complete boundary sets[6] were transferred to a neighboring CPN both for low resolution and high resolution. We measured the total amount of bytes as reported by the underlying communication library (PVM). The results from the curves in Figure 6.6 are summarized in table 6.1. High resolution boundaries show a stronger density dependency than low resolution boundaries. For the efficiency and runtime estimates of Section 6.4.5 we used the *time-step average* values as arithmetic mean of both resolutions.

The value b_{header} for the header size of a boundary transmission could not be obtained from the measurements since the intersections with the y-axis were sometimes negative. Therefore, we estimated a size of $b_{header} = 64[byte]$.

[6] A set of boundaries are all boundaries transferred between two neighboring CPN's before each sub-time-step

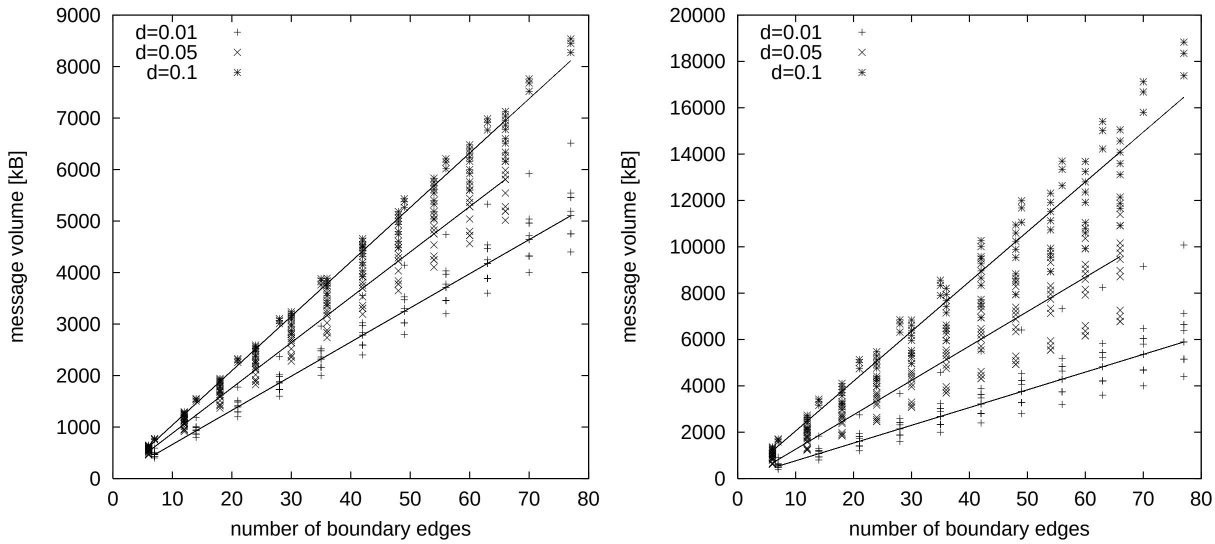

Figure 6.6: *Boundary communication volume* — LEFT: low resolution, RIGHT: high resolution

density	low resolution [byte/boundary]	high resolution [byte/boundary]	time-step average [byte/boundary]
0.01	66	77	72
0.05	88	148	118
0.10	106	215	160

Table 6.1: *Boundary communication volume*

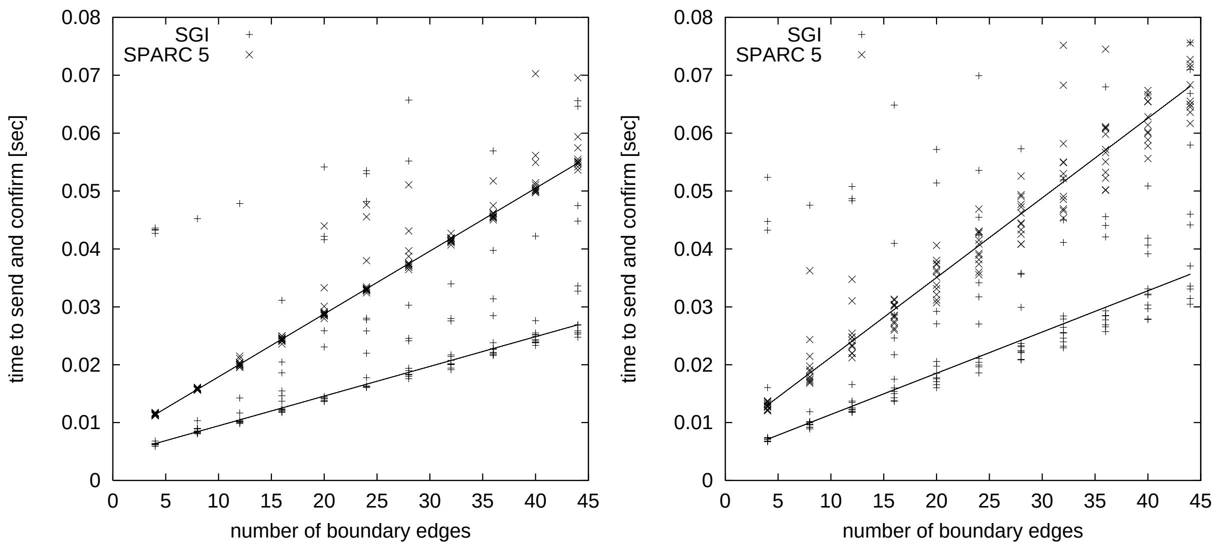

Figure 6.7: *Boundary communication performance* — LEFT: low resolution, RIGHT: high resolution)

High-level communication bandwidth and latency

The latency and bandwidth of boundary communication is dominated by the application-level encoding and decoding routines, if the number of sender CPN is small and the load of the communication network is low.

As above for the boundary volume, we used two CPN's between which we exchanged an varying number of boundary sets. This time, however, we measured the time required to encode the boundary set, transmit the set, decode the set on the remote CPN, and receive a confirmation message from the remote CPN. Thus, the intersection with the y-axis will be at *twice* the latency. The results from the curves in Figure 6.7 are summarized in table 6.2. Note that the measurements for the SGI Challenger — although very well defined with the lower bound transmission rates — show a lot of particularly bad exceptions. This is mainly due to the fact that the SGI Challenger was always loaded with about two jobs per CPN, while the SUN cluster was running idle.

machine	low resolution		high resolution	
	latency [ms]	transfer [ms]	latency [ms]	transfer [ms]
SGI-Challenger	2.16	0.257	2.16	0.356
Sparc-Cluster	3.52	0.543	3.77	0.688

Table 6.2: *Boundary communication performance*

6.4.5 Performance estimates

All curves presented in this section refer to the complete map of the German Autobahn network (FRG). The overall vehicle density was fixed at $\varrho = 0.1$. The administrative overhead per substep was assumed to be 2.5%.

Fixed problem size for bus architecture

Figure 6.8 shows what fraction of the execution-time is spent on which part of the simulation:

- The *raw simulation* curve mainly represents the traffic simulation itself, although this fraction includes the administrative overhead for multiple sub-time-steps. It is equivalent with the efficiency $e(p)$ of the simulation.
- The *load gradient* curve shows the loss of execution time due to the load gradient which builds up throughout the CPN network. We assumed a exponential gradient of 0.01 per layer.
- The *a-l communication* curve show the fraction spent on application-level communication with latency and bandwidth combined.
- Finally, the *l-l communication* curve shows the additional fraction of time spent on low-level communication due to the saturation of the underlying network through non-simultaneous communication.

The left-hand side of Figure 6.8 depicts the estimate for the SGI Challenger. We assumed the average sustained communication bandwidth on the internal bus system to be 5 [MB/s]. The efficiency is slightly above 0.9 for 16 CPN's, which is the configuration used in [103]. At approximately 150 CPN's the efficiency drops below 0.5, because the time spent on low-level communication quickly exceeds the raw simulation time itself. This is the major drawback of the bus-topology.

The right-hand side of Figure 6.8 reveals the same qualitative shape of the efficiency and low-level communication curves as those for the SGI Challenger. The crossover, however, is already reached at 30 CPN's, due to the lower sustained communication bandwidth of the Ethernet which was assumed to be 1.0 [MB/s].

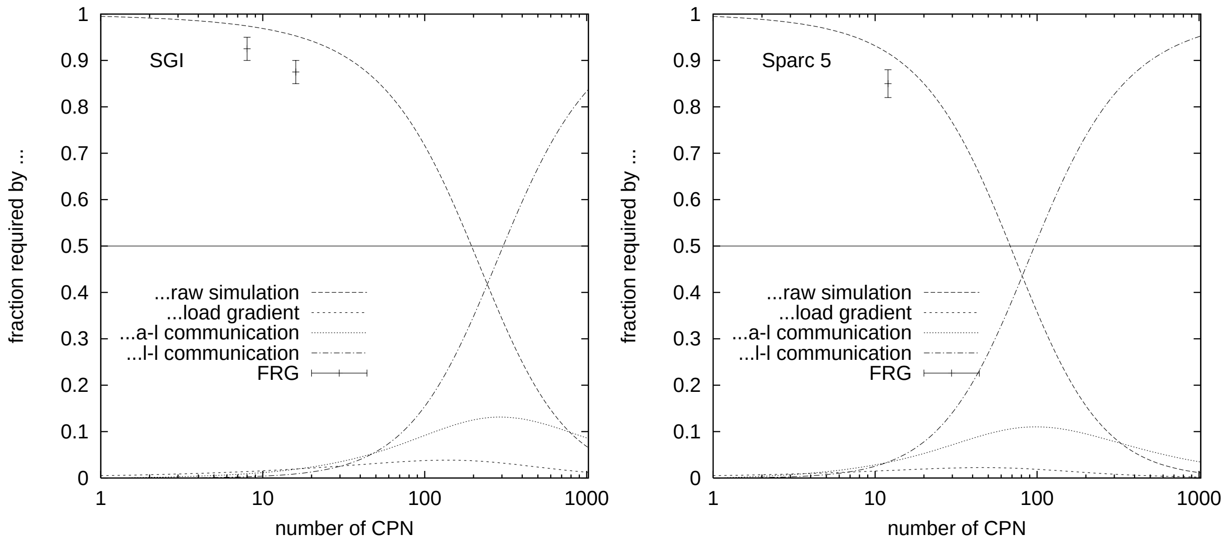

Figure 6.8: *Efficiency for bus topology* — The raw simulation fraction corresponds to the effiency of the simulation. Both figures contain points which were retrieved from the benchmarks of Section 6.4.6. Note that the actual efficiency is always lower than the theoretical efficiency.

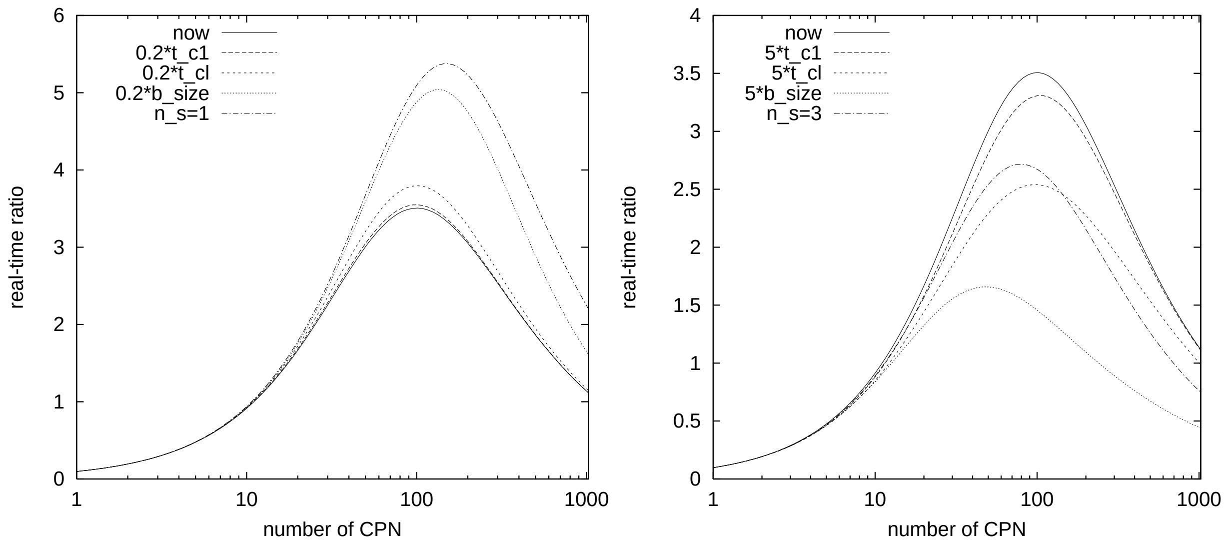

Figure 6.9: *Other software implementations* — Left: Better software implementations (in this order): current state, smaller latency, faster coding, smaller boundaries, only one sub time-step. Right: Worse software implementations: current state, longer latency, slower coding, larger boundaries, three sub time-steps.

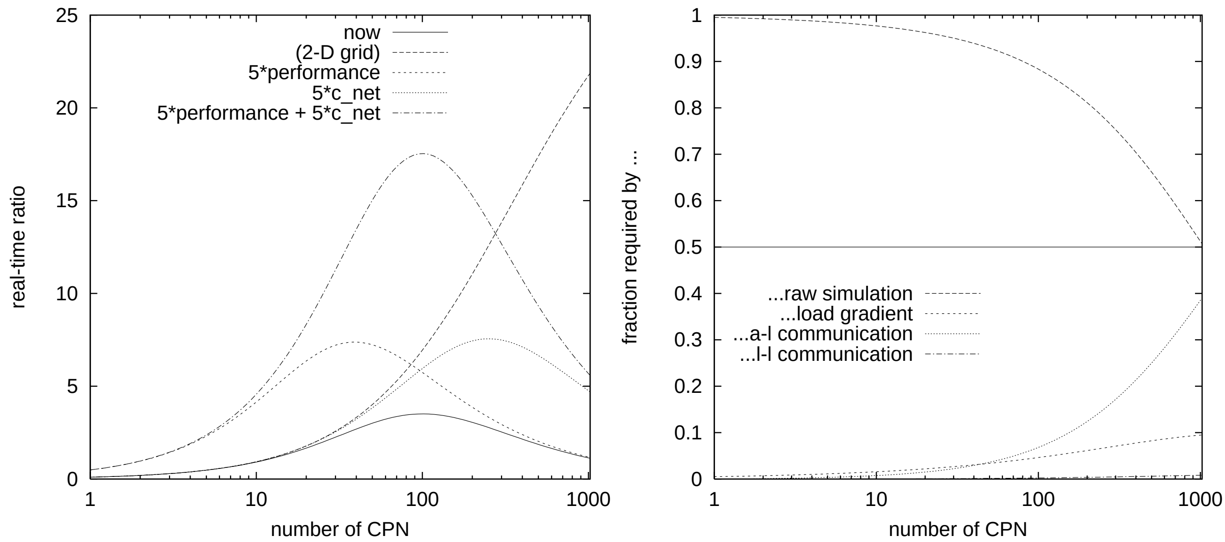

Figure 6.10: *Performance improvement through hardware tuning* — Improvements (in this order): current state, 2-D communication topology, higher CPU-performance, higher net bandwidth, higher CPU-performance in conjunction with higher net bandwidth. Right right-hand side depicts the partial fractions for a 2-D grid network.

Fixed problem size with software implementation tuning

Figure 6.9 depicts the obtainable real-time ratio if the software implementation were better or worse than in the current version, respectively. They refer to the Sparc-5 cluster with Ethernet. The following improvements were tested:

- the application-level communication rate t_{c1} is only a fifth,
- the application-level latency t_{cl} is only a fifth,
- the average boundary size b_{size} is only a fifth,
- the number of sub-time-steps is reduced to one.

Accordingly, the following disadvantageous modifications were tested:

- the application-level communication rate t_{c1} is five time as high,
- the application-level latency t_{cl} is five times as high,
- the average boundary size b_{size} is five times as high,
- the number of sub-time-steps is increased to three.

As could have been expected, the average boundary size b_{size} has most impact, since low-level communication proved to be the main bottleneck on the bus-topology. Nevertheless, no matter how many CPN's are available, it remains difficult to push performance beyond a real-time ratio of 3.

Fixed problem size with hardware tuning

After the software modifications we now look at hardware modifications. The left-hand side of Figure 6.10 depicts the estimates for the following improvements:

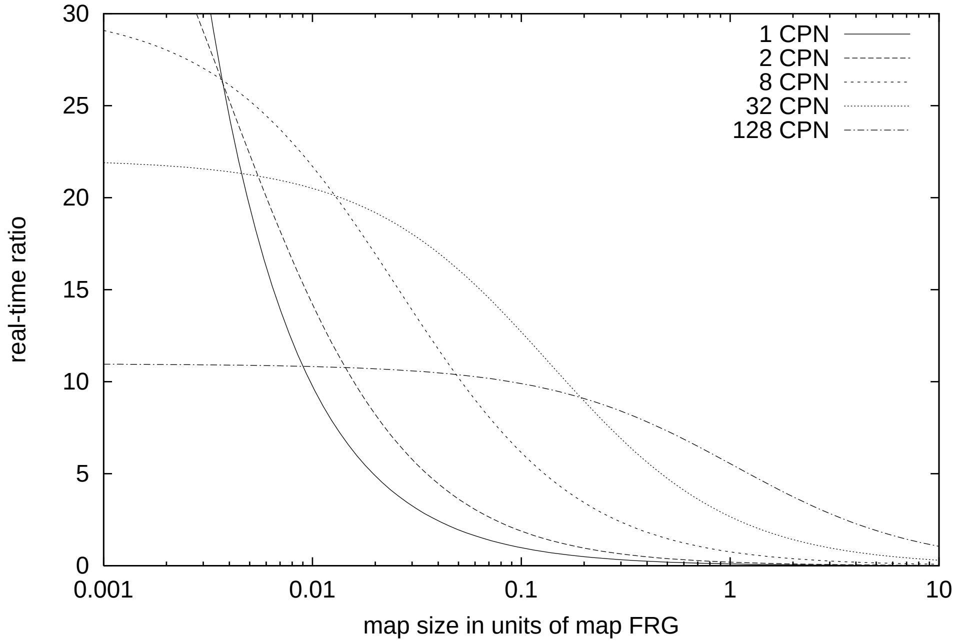

Figure 6.11: *Variable problem size on bus-topology*

- use a 2-D grid topology instead of the bus topology, while keeping all other hardware parameters,
- the per-CPN performance is five times as high,
- the low-level communication bandwidth is five times as high,
- the low-level communication bandwidth *and* per-CPN performance are five times as high.

Obviously, only switching to another communication topology is beneficial for large numbers of CPN's. Other modifications only yield a small increase of performance. The combined improvement of per-CPN performance and low-level communication gives a moderate speedup of factor 7 with approximately 50 CPN's.

The right-hand side of Figure 6.10 shows the potential advantage of a 2-D grid more clearly. Since there are no measurements for T_{cl} and T_{c1} available, we used those of the SPARC-5 workstation cluster. We assumed the bandwidth of local low-level communication to be as high as with the SGI-Challenger and the per-CPN performance to be a forth of a SPARC-5.

From the plot it is obvious that low-level communication does not play a role in a two-dimensional topology. With 128 CPN's efficiency is still at 0.85 with 15% loss equally split between dispersion and application-level communication. Only above 1024 CPN's the efficiency drops below 0.5.

Variable problem size on bus-network

The last plot depicts the obtainable real-time ratio for variable problem size in units of the map FRG. The number of super-nodes and super-vertices was scaled accordingly with ratio 403/280. Again we used the parameters for a cluster of Sparc-5 workstations connected through Ethernet. Note that on a system with 128 CPU's it is impossible to increase the real-time-ration beyond 11, no matter how small the map may be.

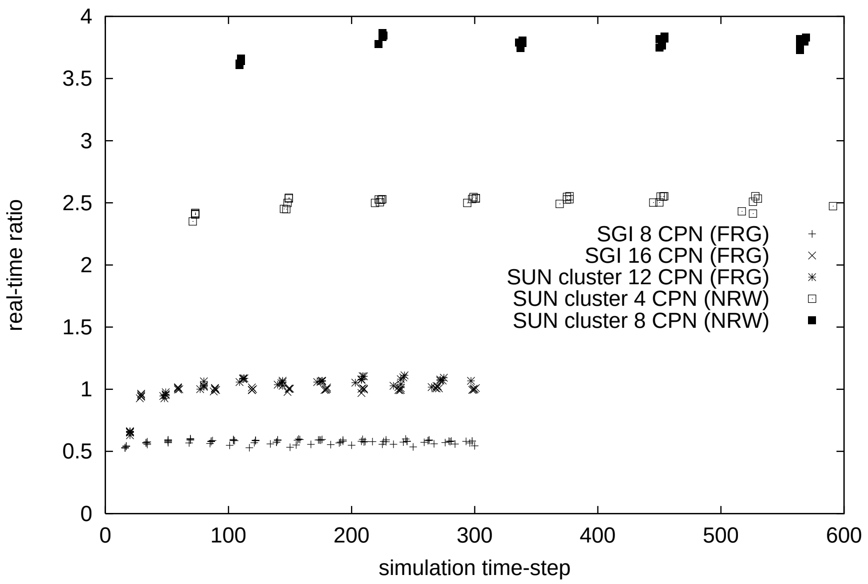

Figure 6.12: *Real-time ratio (German Autobahn network)*

6.4.6 Benchmarks of PAMINA II

We did measurements using different street network sizes on two different computer architectures, both with PVM (see [97]) as underlying communication library. One platform was a workstation cluster of 12 SUN SPARC-station (5, 10, and 20 with performances between 133 and 135.5 MIPS) connected through 10 Mbit Ethernet (PVM architecture `SUN4SOL2`). The other one was a 16 processor SGI Challenger (SC 900 XI) Shared Memory system (PVM architecture SGI64). On the latter we did not explicitly use the shared memory architecture, but the PVM architecture `SGI64` which is based on UNIX sockets for inter-process communication.

The networks consisted either of the whole German Motorway network (FRG) or an excerpt thereof, namely the sub-network of the federal state Nordrhein-Westfalen (NRW). Table B.1 gives an impression of their respective sizes.

Since the available data did not contain any information about street segment characteristics nor substructures of intersections we used the defaults $n_{lane} = 3$, $v_{max} = 5$, and $p_{brake} = 0.5$ on all segments, as well as $n_{transfer} = 1$, $l_{merge} = 5$, $l_{transfer} = 200[m] \equiv 26[site]$ for all intersections and ramps. The density of background traffic was set to $\varrho = 0.1$ for measurements on the SGI and $\varrho = 0.05$ on the workstation cluster. Note that we did not actually use route-plans since filling the network with the help of route-plans would take considerably longer and vary the computational load until a constant density is reached. As the performance only suffers less than 5% during vehicle insertion activity the measured values are also true for routed simulations. Figure 6.13 shows that after approximately 200 time-steps of load balancing the simulation reaches its optimal performance. Since there was not any other activity on the computer systems during the measurements, this load balancing activity is only due to the imbalances caused by the initial distribution: The work station cluster was running idle and the SGI was only loaded with jobs running at a high nice-level. Judging from the load reported by the UNIX-utility `top` the performance on the SGI may be improved by another 5% to 10% if the machine were not loaded otherwise.

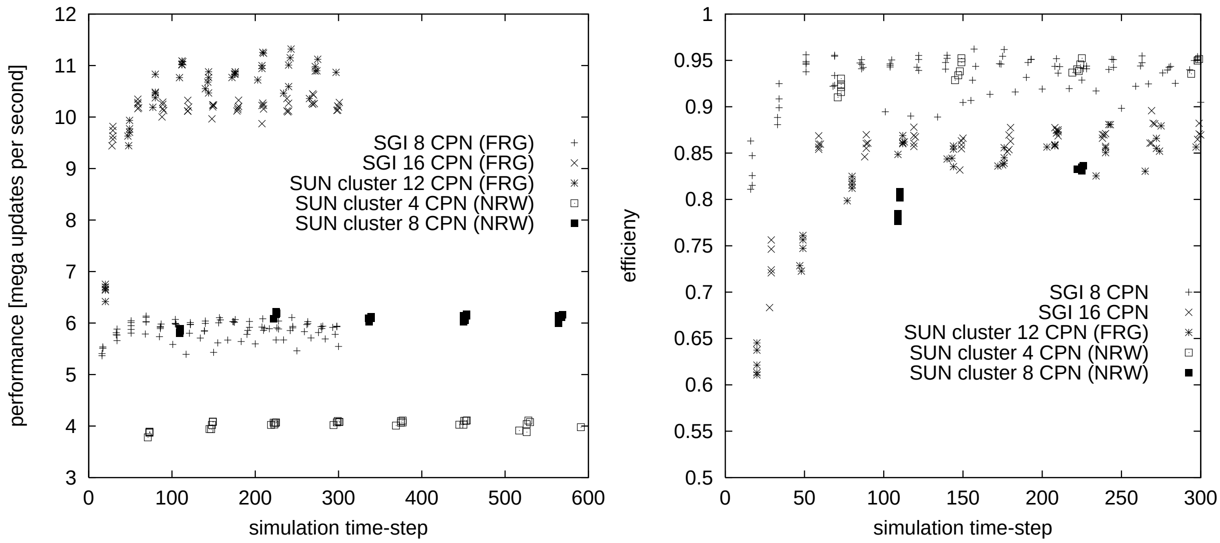

Figure 6.13: *Performance and efficiency (German Autobahn network)* — After a short dynamic load-balancing phase of approx. 200 time-steps, the simulation reaches its maximum performance (left-hand side) and maximum efficiency (right-hand side).

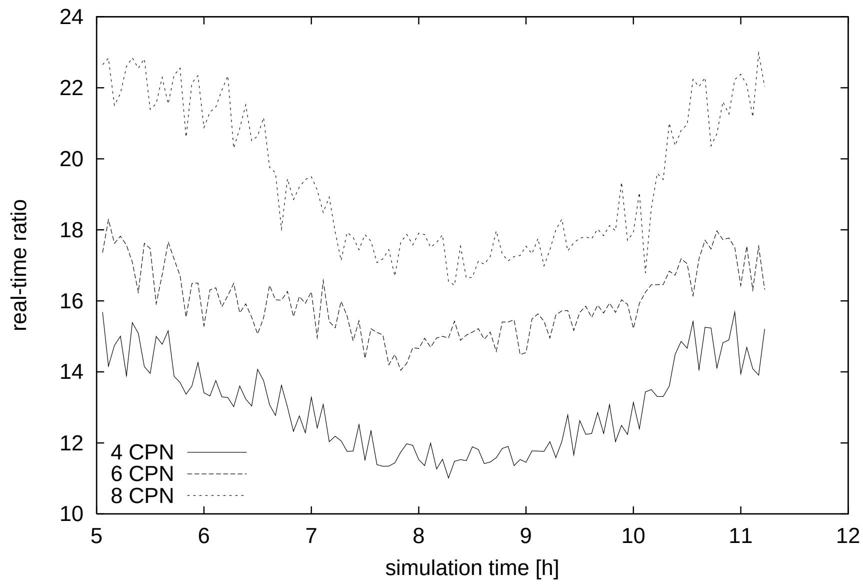

Figure 6.14: *Performance of PAMINA III* — The real-time ratio is plotted for 4, 6, and 8 CPU's on a Sparc Enterprise 2000 with 250 Mhz. The trough around 8:00 am is caused by the high vehicle density inside the study-area.

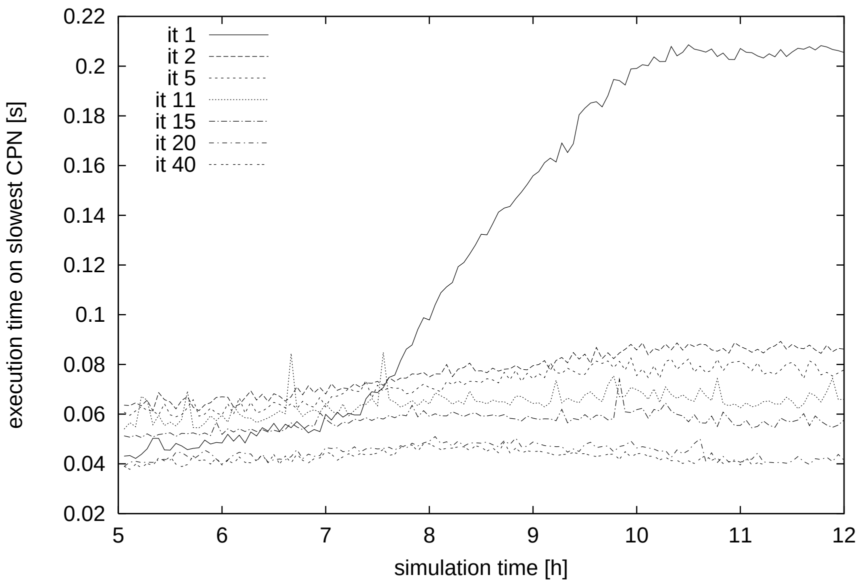

Figure 6.15: *Execution times with external load feedback* — The feedback visibly improves the performance of the simulation. After only two iterations the execution time is almost time-independent.

6.4.7 Benchmarks of PAMINA III

We used a route-plan of run 11 (iteration 60) to measure the real-time-ratio of PAMINA. The simulation ran on SUN Enterprise 2000 with 14 CPU's (250 MHz) and 2 Gigabytes of memory. Figure 6.14 shows the results for different numbers of CPU's (4, 6, 8). The maximally obtainable ratio for PAMINA is about 22 for 8 CPU's and early simulation hours while the study-area is still empty. The RTR drops to 18 during rush-hour at 8:30 am.

6.4.8 Static load balancing with feedback

In PAMINA III we implemented simple external feedback for the initial static load balancing. During run time we collect the execution time of each link and each intersection. The statistics are dumped to file every 1000 time-steps. For the next iteration run the file is fed back to the initial load balancing algorithm. Instead of using the link lengths as load estimate, now, the actual execution times are used as distribution criterion.

To verify the impact of this approach we monitored the execution times per time-step through-out the simulation period. Figure 6.15 depicts the results of run 17 for several iterations. For the initial iteration 1, the load balancer used the link lengths as criterion. The excution times are low until the first grid-locks appear around 7:30 am. The execution time increases fivefold from 0.04 [sec] to 0.2 [sec]. Already in iteration 2 the execution time is almost independent of the simulation time. Note that due to the equilibration the execution time for early simulation hours has increased from 0.04 [sec] to 0.06 [sec], but this effect is more than compensated later on.

The figure also contains plots for later iterations (11, 20, and 40). The improvement of execution times is mainly due to the route adaption process: all grid-locks have disappeared and the average vehicle density is much lower.

6.4.9 A remark on "super-linear" speedups

It is obvious that the parallel efficiency has to have 1 as upper bound. Otherwise for $e(p) > 1$ the product

$$T_{emu}(p) = pT_{par}(p) = \frac{T_{seq}}{e(p)}$$

would be less than T_{seq}. Since $T_{emu}(p)$ represents the time required by a single CPN to emulate (by sequential execution) the p-CPN machine, this would contradict the assumption that we chose the fastest available sequential algorithm as reference. We could simply declare the emulation the fastest sequential algorithm solving the problem.

In practical implementations, super-linear speedups (with efficiencies above one) have been detected, though. We would like to mention two causes for these observations:

Hardware When measuring the runtime for an implementation, one generally assumes that a problem with size n and complexity $T(n)$ will cause a runtime proportional to $T(n)$. This disregards the fact that a larger n will increase the memory-requirements. The more memory is required, however, the more likely data is stored in "slower" memory types (including virtual swap memory) which results in a considerably decreased performance.

Thus, for a problem of fixed size, distributing the problem onto several CPN's will also reduce the amount of memory required on each CPN and speed up computation. In some cases this outweighs additional computation imposed by the distribution.

Software Parallel implementations often differ principally from their sequential counterparts, that is to say, instead of converting the sequential algorithm into a parallel version according to some canonical rules, a completely different "ansatz" is chosen. The comparison of runtimes is only fair, then, if the results can be compared to be identical.

Many applications, however, do not yield exact result, since they have built-in stochasticity provided by random generators. Reproducing "exact" results would require any random sequence of the single-CPN application to be reproduced on the multi-CPN machine. Obviously, this would result in an enormous coding and validation effort. Therefore, applications use local random generators hoping that results will be at least equivalent if not identical.

Another aspect relates to the processing of input data which is usually stored in sequential fashion on files. To preserve a good performance data is read in such a way to optimize access time often neglecting the arbitrariness of the chosen access order. Algorithms which rely on input order may fail to reproduce the results of the sequential version.

The traffic model presented here is subject to the problems described above although none of them really changes the output in a noticeable way.

Chapter 7

Summary, Discussion and Outlook

In this work we presented different aspects of modern traffic simulation. Starting with the simulation of traffic on a single link, we extended the model to include network traffic. The simulator PAMINA was used to create self-consistent route-sets which finally served as basis for the online re-routing experiments.

Chapter 2

The traffic on a two-lane road exhibits new characteristics compared to the simple single-lane model. As in real-world traffic the passing of vehicles is a major component of the driving dynamics. Chapter 2 contained a rule extension with three parameters defining a basic lane changing process. The symmetric version treats left-to-right and right-to-left changes equally. The asymmetric version favors the right lane over the left lane, which is to mimic driving guidelines on highways in many countries. Fundamental diagrams of flow-density relationship are similar to those found in real-world traffic. We could also show that the artifact of ping-pong lane-changes could be significantly reduced by introducing an additional lane-changing probability.

The look-back parameter was found to be essential for the continuity of traffic flow. Reducing the parameter from the usual v_{max} sites to zero destroyed the laminar flow, especially in the asymmetric case. In real-world traffic, egoistic lane-changing maneuvers increase the risk of accidents considerably. Thus — though the cellular automaton rule does not model traffic accidents explicitly – the amount of flow disruption can be taken as a measure how safe a modeled traffic state is.

Chapter 3

In Chapter 3 the two-lane model was used as a basis for the micro-simulator PAMINA. By combining simple network elements to composite structures, we were able to model both highway networks (or Autobahn networks, respectively) and city traffic. The complete Autobahn network of Germany with 75,000 kilometers of road-way was used for a feasibility test. Also, the simulation already included a feature that became more and more important in later versions: the execution of individual route-plans. At this point, however, the route-plans were still chosen at random.

The first tests of PAMINA using route plans were done in the context of the TRANSIMS Dallas/Fort Worth case-study. Both the network and the plan-set were preliminary at that point: The network did not include any local streets; the plansets were "unbalanced" in the sense that drivers did not attempt to avoid congestion.

A first series of runs investigated the influence of different levels of fidelity. We defined the "fidelity" of the micro-simulator by activating/deactivating speed-limits and/or traffic lights. We measured the impact of these parameters by comparing the actual travel-times of the vehicles to those which were estimated by the route-planner. We found that running the simulation with activated speed-limits and deactivated traffic lights yielded most similarities. Although in these runs we did not expect the micro-simulation to reproduce expected travel times, the comparison was useful to see if *in general* the microsimulation generated faster or slower travel times than the planner; and which fidelity level of the micro-simulation changed this to what degree.

During the simulation of the study-area, we also encountered the phenomenon of so-called grid-locks. These configurations occurring at high densities consisted of vehicles that permanently prevented each other from proceding any further. They were due to the fact that vehicles — so far — were not able to modify their routes. The number of grid-locks depended on the fidelity of the simulation: without speed-limits and without traffic-lights the simulation never grid-locked. The same was true for activated speed-limits. Activating traffic-lights had a stronger impact on the dynamics: the system grid-locked independent from the use of speed-limits. This underlines that in the city-network the performance is mainly defined by the throughput at intersections. Reducing the speed-limit on links has a small effect since at medium and high densities the average velocity is comparable to the speed-limit anyway.

We introduced an additional parameter q_r for adjusting the red-phase of the traffic-lights. By varying the parameter between 0 and 1 we could continuously move the fidelity from one without traffic-lights to one with fully active traffic-lights. At $q_r = 0.6 \ldots 0.65$ we noticed a transition: below $q_r = 0.6$, no run developed a grid-lock. Above $q_r = 0.65$ all runs grid-locked. For the runs within the interval, the grid-locking depended on the stochasticity of the simulation.

Chapter 4

Up to this point, the route-sets used for PAMINA originated from TRANSIMS. In Chapter 4 we chose a more consistent setting: both the micro-simulation and its feedback were obtained from the *same* micro-simulator PAMINA. Due to the high execution speed of PAMINA and additional improvements of the data exchange between router and route-converter it was possible to reduce the run-time considerably. Compared to TRANSIMS which required about 8.5 hours for one iteration of *micro-simulation — route-converter — planner*, PAMINA could process the computation in about 35-45 minutes. This allowed us to run extensive experiments for route adaption with micro-simulation feedback. We defined four simple parameters which were expected to have an influence on the relaxation of iterative process: the initial route-set, the selection scheme for routes to be re-planned, and the re-planning fraction. In the experiments conducted here it turned out that none of the parameters actually had a strong impact on the final results. Especially the independence from the initial route-set is very advantageous, since otherwise special care would have to be taken to start the iteration from a *good* route-set. We found, however, that the computational speed of the relaxation could largely be reduced by using a linear-age[1] approach for selection. The accumulated re-planning fraction[2] f_{acc} should be at least two to allow for sufficiently stable results, which were measured by contemplating the overall travel-time of all vehicles during the simulation period. It will be interesting to conduct at least one run with an accumulative re-planning $f_{acc} \gg 3$ to see if the relaxation has really come to an end and what the final sum-travel-time will be at that point.

We discovered two artifacts in the iteration process: (a) The number of vehicles that were fed as input into the micro-simulation decreased by about 10%, because the planner re-planned more and more routes *avoiding the study-area* due to its highly increased link travel-times. By introducing a level-0 correction factor c for all links outside the simulation area it was possible to reverse this effect to a certain extent. In a linear correction run the number of plans was reduced by only 5%, in another one with correction $\sqrt{c}$ the reduction was about 7.5%. This dependency gives rise to the assumption that a factor of c^q with $q \in [0.5, 2]$ can be used to arbitrarily adjust the number of vehicles inside the study-area. It also became

[1] In this case, the probability of a route to be re-planned increases linearly with its age, which is the number of iterations since its last re-planning.

[2] The accumulated re-planning fraction corresponds to the sum of all individual re-planning fractions used up to a given iteration.

obvious that the level-0 correction should only be applied *after* the iteration has been successfully relaxed. That way, one could avoid the disadvantageous loss of vehicles outside the time-window which occurred during early stages of the level-0 correction iteration.

Another artifact was that of up to 10,000[3] vehicles queuing up at the boundaries of the study-area until the end of the simulation period. This effect was caused by insufficient feedback of waiting time on feeding links to the planner. We used a correction in which the average waiting time for entering a link was *added* to the travel-time on the link itself. The results are promising: after an accumulated re-planning fraction $f_{acc} > 1.5$, none of the iteration runs using the correction had any vehicles waiting in queues at the end of the simulation period.

Although route-sets could be assumed to be well relaxed after sufficiently large accumulated re-planning fractions ($f_{acc} > 2.0$), the values for sum travel-time and the number of vehicles in the study-area still exhibited fluctuations between subsequent iterations. It will be interesting to compare the amplitude of these fluctuations to the corresponding fluctuations of the *link hit-rates* which can be computed from the travel-time estimates provided by the planner. It gives a hint at how *many* vehicles are expected to be on a link, provided that the link travel-times were not disturbed stochastically. As a preparation, this data has already been collected for most iteration runs.

We continued by conducting a comparison between three micro-simulators working on the same test-bed. The TRANSIMS micro-simulator, PAMINA, and the low-fidelity simulator VSS were used to produce their own self-consistent route-sets. The results for the sum travel-time and the number of vehicles matched qualitatively, but there were quantitative differences. VSS yielded results *below* those of PAMINA. This could easily be attributed to the fact that VSS simulates the whole Dallas - Forth Worth area instead of the study-area only. Traffic jams outside the area which cannot occur in PAMINA or TRANSIMS give rise to a lower number of vehicles inside the area. The curves of the TRANSIMS runs were equivalent for small accumulation re-planning fractions ($f_{acc} \approx 1.0$). Later during the iteration ($f_{acc} \approx 3.0$) the PAMINA curves eventually dropped below TRANSIMS curves. From this we can assume that the TRANSIMS route-set was still not sufficiently relaxed. In future comparisons one should see to it that only well relaxed route-sets are used.

So far the comparisons concentrated on values which had been aggregated in both space and time. In order to investigate more detailed differences between the micro-simulators, we also considered less aggregated data. First, we looked at the time-dependent travel-speed into the study-area for both PAMINA and TRANSIMS. As before, the curves showed a qualitative similarity: starting from high traveling speeds during early morning hours, the speed drops to reach a minimum during the rush hour. Afterwards they recover again. The absolute variation of the average speeds, however, is larger for PAMINA than it is for TRANSIMS. At a first glance, this may be surprising since the TRANSIMS runs has a smaller average vehicle density in the study-area during rush-hour and the speed limits are computed the same way. In future the PAMINA runs may have to be repeated using the same deceleration probability p_d=0.2 for the CA rule set as TRANSIMS. Originally, $p_d = 0.3$ had been selected which reduces the ability to accelerate fast. Especially at medium and high densities, lower p_d have a direct impact on the sustained throughput.

This effect also raises the question of how to generally define the speed-limit for the simulation links. Due to the granularity of CA velocity the limit can only be adjusted in steps of approx. 24 [km/h]. Using the deceleration probability to fine-tune the maximum speed of CA also changes its behavior at low velocities.

In a second experiment we compared the turn counts for a selected number of intersections inside the study-area. The similarity between the TRANSIMS run and PAMINA are satisfactory. In general, the counts are lower for PAMINA due to the lower vehicle density in the study-area for TRANSIMS. Both simulations, however, show considerable differences compared to *actual* traffic counts provided by the NCTCOG. Probable reasons are: (a) The O-D matrix which served as input for the activity list (year 1990) did not match the street network (year 1996) that was used for the simulation. Changes of the infrastructure can be expected to result in equivalent changes in the O-D matrix. (b) Both simulations consider insufficient penalty for making turns at intersections. A more realistic representation for the

[3] Compared to approx. 15,000 vehicles in the study-area during rush hour.

"inconvenience of turns" should reduce the excess number of counts compared to reality. This will be part of future work. All in all, it must be noted the approaches decribes in this thesis yielded some of the first simulation results ever, that were realistic enough to be compared to real-world data.

Chapter 5

In Chapter 5 we used PAMINA with the previously computed route-sets to conduct an investigation of online re-routing. We assumed that the fleet of drivers can be split into two separate groups: those who have access to an online route guidance system, and those without access. The fraction of drivers with access is called *market saturation*, the drivers themselves *subscribers*, all others *non-subscribers*.

One sample run was done as follows: at a constant update interval all subscribers tried to estimate their remaining travel-time. They were allowed to use the *current* link-travel times of all links of remaining portion of their route-plan up to a certain *planning horizon*. If the current estimate was worse (by a certain fraction) than the original estimate provided by the router, a new route was computed using a standard shortest path Dijkstra algorithm limited to the planning horizon. If the new route was sufficiently better than the current the driver accepted. This was called a *re-routing event*.

The first experiment was done using a route-set with an accumulated re-planning fraction of $f_{acc} = 1.0$. The street conditions were not changed with respect to the one that route-set is based on. The results can summarized as follows:

(i) During rush hour about half of all subscribers regularly try to obtain a new estimate. Half of those receive a new (potentially better) route.

(ii) Upon arrival at their destination subscribers have a (up to 28%) lower average travel-time than if they had not been re-routed. The advantage decreases with increasing market saturation: With a saturation of 50% the gain diminishes to about 6%. This effect is disadvantageous to the providers of the re-routing service. On the one hand one would like to reach high market penetration in order to use any installed hardware more cost-effectively. On the other hand, it is reasonable to assume that the *incentive* to buy a guidance system (or to subscribe to a service) will increase with a potential benefit.

(iii) The quality of the re-routing process improves if a vehicle is re-routed more than once. Quality was measured by comparing the actual arrival time of a subscriber to another run in which the route guidance system had been switched off. This suggests how re-routing schemes may have to work: upon receiving a new route the driver starts into the general direction of prescribed by the suggestion. Subsequent requests will refine the route as the driver proceeds through the network.

(iv) Up to a market saturation of 40%, re-routing has a beneficial effect on both subscribers and non-subscribers. This was determined by looking at the number of executed routes (overall throughput through the study-area) and the overall average travel-time. For a market saturations of 40% the reduction of average travel-time was about 6%. For higher saturations the situation has an overall negative effect: although *in principal* some runs showed an improvement, *in general* the variance of travel-times is much larger. Re-routing at high market saturation in undisturbed networks makes the system more unstable.

(v) Increasing the update-interval has a small negative effect. This is quite intuitive, since the shortest path search is based upon the *current* condition of the links and not on a projection of the *future* condition. Increasing the update interval also widens this gap.

We re-did exactly the same parameters but used a route-set with $f_{acc} = 1.5$. As mentioned before, route-sets of later iterations used the traffic network more efficiently. Presumably, online re-routing is less likely to produce better routes in a more relaxed network. The results support this assumption: (a) the advantage of subscribers over non-subscribers diminishes by a factor of five, (b) there is no overall benefit for all drivers for small market saturations but, as before, a larger variance for higher saturations.

The question is now: how well relaxed is a real-world traffic system? If it resembles the case $f_{acc} = 1.5$, it will be very difficult to profit from the re-planning scheme presented here. Some improvement may be brought about by trying to extrapolate the travel-time on links for future times. This could be done by referring to data bases of standard traffic configurations forecast by the micro-simulation for the planning area, such as the day of the week or weather conditions. This approach, however, must be handled with care since it will be difficult to reproduce the startint point of a forecast as an evolution of a given route-set.

In the last experiments we tried to investigate a more realistic traffic configuration by looking at *non-recurring* congestion. We disturbed one link between 8:00 am and 9:00 am by reducing its maximum speed given in CA units from the usual 4 to 1 [site/sec]. In reality this may have been caused by a traffic accident or a road construction. The important factor is that *none of the drivers knew* of this obstruction beforehand. Therefore, there was no way that the external router could have adapted to the situation. Simulation results show that the reduced throughput on one link already reduces the overall throughput noticeably by about 8%. A modest re-routing alleviates the problem, but as before increasingly high market saturations worsened the situation again.

We increased the number of disturbed links from one to ten for the same time interval. This time the system had considerably lower throughput for market saturations up to 20%. Above that, online planning improved the throughput considerably. Even for higher market saturations, the positive trend continues. We obtain increasingly *lower* average travel-times up to a market saturation of 90%. In this respect, the heavily disturbed traffic system significantly differs from all other cases above. It could be argued that ten *simultaneous* disturbances are exaggerated for such a small area. But even if reality lies somewhere between one and ten disturbances, it clearly shows that within the current framework the amount of *non-recurrent* congestion appears to be a key prerequisite for the success of an intelligent route guidance system in terms of throughput. Possible benefits for drivers not familiar with the area and the congestion structure have not been investigated. In fact, all drivers in the simulation correspond to "experienced" drivers.

Chapter 6

In the last chapter we shifted the focus to the computational aspects of traffic simulation. The implementation of PAMINA is based upon a domain decomposition of the street network. By looking at simple map parameters (i.e. the number of intersections and links) and hardware parameters (i.e. communication bandwidth and latency) we were able to derive an upper limit for the theoretical parallel efficiency. It turns out that on a bus communication network which is implemented in many of today's shared memory computers, the simulation reaches high efficiencies for medium-sized computers (less than 64 CPU's). Further improvement can only be achieved on computers based upon a two-dimensional communication grid or similar scalable communication structures.

The next step in optimizing the iterative adaptive process described in Chapter 4 will be to integrate the components route-planner, route-converter, and micro-simulator into a compatible parallelization scheme. So far, both the route-planner and the route-converter are only available in a sequential version and the data exchange between route-converter and micro-simulator is still strictly serialized. Moreover, the micro-simulator will have to be adapted to link with parallelized versions of emission evaluation models and ultimately climate simulations models (see ref[25]).

The Appendix A contains a description of the Parallel Toolbox which was used for the parallelization with a domain decomposition approach and PVM as message library. The first implementation was run on a 16-CPU SGI Power-Challenge, but the approach also permitted the port to a workstation-cluster of 12 SUN Sparc5 which yielded approximately the same performance. It was the first time that a traffic network of the size of the German Autobahn network could be simulated in real-time. The toolbox also handles the dynamic load-balancing and allows the dynamic insertion and deletion of CPN's during the run-time of the simulation.

It should be noted that in spite of increasingly more powerful computers, traffic simulation still requires a huge amount of computation time. For the results of Chapters 4 and 5 alone we needed about 630 hours of parallel execution on 6 CPU's and 130 hours of sequential execution on one CPU. If the simulation had not been parallelized, the overall sequential execution time on one CPU with 250 [MHz] would have amounted to 3900 hours. This is equivalent to about *5.5 months of continuous computation.*

Appendix A

PAMINA

In the previous chapter we have outlined the domain decomposition as the basic approach for parallelizing a traffic simulation on distributed memory systems. In this chapter we will give a concrete overview of the actual implementation (A.1.1) of PAMINA. The description will concentrate on PAMINA II since it is the only[1] version that provides dynamic load-balancing. The parallelization scheme which is based upon the Parallel Toolbox will be described in section A.1.2. The chapter will be concluded with an overview of the run-time statistics that can be collected in PAMINA II. They include both computational statistics (i.e. run-time profiling) and traffic statistics (i.e. link characteristics vehicle count, density, and average velocity).

A.1 Implementation

A.1.1 General overview

The implementation of PAMINA II is bases upon defining descendent C++ classes of the C++ base classes provided in the Parallel Toolbox Version 1.0. A detailed description can be found in [100]. However, we would like to outline how the traffic simulation interacts with the toolbox and its basic functional elements (refer to Figure A.1). The PAMINA source code splits into three major parts: (a) the underlying CA model, which is the most compact module containing only about 500 lines of code, (b) the graphics support module with about 7.000 lines, (c) handling of the traffic network elements, the route-plans, and statistics with about 17.000 lines of code. The latter also contains the interface to the Parallel Toolbox which by itself has about 29.000 lines of code. The toolbox uses the commonly available message library PVM 3.3.1 as high-level communication interface to the underlying computer hardware. PAMINA II and the Parallel Toolbox 1.0 have been ported to several platforms, such as Sun Solaris, SGI Irix, DEC Alpha, IBM RS6000, and PC Linux.

The source-code of the toolbox and the micro-simulation cannot not be discussed in detail. We would like to refer interested readers to the web page

http://www.zpr.uni-koeln.de/~mr/PAMINA/

Among other things it contains:

- instructions on how to retrieve demo versions of both the Parallel Toolbox and the PAMINA micro-simulation,

[1] Dynamic load-balancing will be added to PAMINA III in the near future.

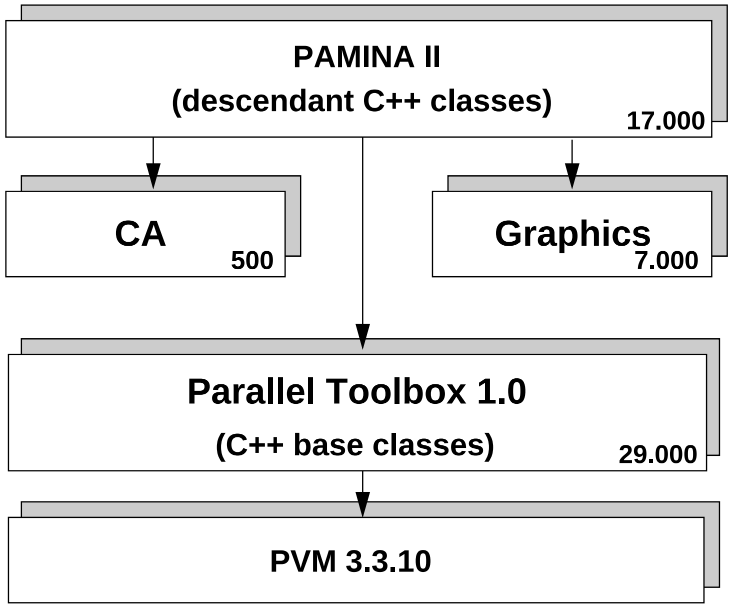

Figure A.1: *Software implementation structure of PAMINA II* — The parallel toolbox serves as an interface between the traffic application classes in PAMINA and the message passing library PVM. The figures in the lower right corners denote the number of source-code lines in each module. Note that most of the programming effort was required for coding the network implementation.

- instructions on how to retrieve the source-code of both applications,
- a commented object hierarchy[2] of all C++ classes,
- a list of all options available for the configuration file,
- an overview of all features that are new in PAMINA III, and
- a description of all statistics files.

A.1.2 Parallelization

The inherent structure of a traffic micro-simulation favors a *domain decomposition* as the general approach to parallelization:

- The street network can easily partitioned into tiles of equal or almost equal size. A realistic measure for size is not the number of net elements (nodes and segments), but the CA grid lengths associated with those elements (see Figure A.2 for a schematic view of tiles and Figure 3.6 for a concrete example). Tiles are then assigned to processors.
- The range of interdependencies between network elements are restricted to the interaction range of the CA. All current rule sets have an interaction range of either $v_{max} \equiv 35[m]$ or $2v_{max} \equiv 70[m]$ which is a short distance compared to the average length of the edge segments (e.g. $484[site] \equiv 3630[m]$ for map FRG, see Table B.1) in a motorway network. In a city street network which has somewhat shorter segments on the average, there may be links which are to short to hold at least 10 grid sites. We artificially enlarge the links to contain the minimum number of sites. In any case, the most straightforward approach is to cut the network at the middle of street segments.

[2]The object hierarchy was created using the tool DOC++ [24].

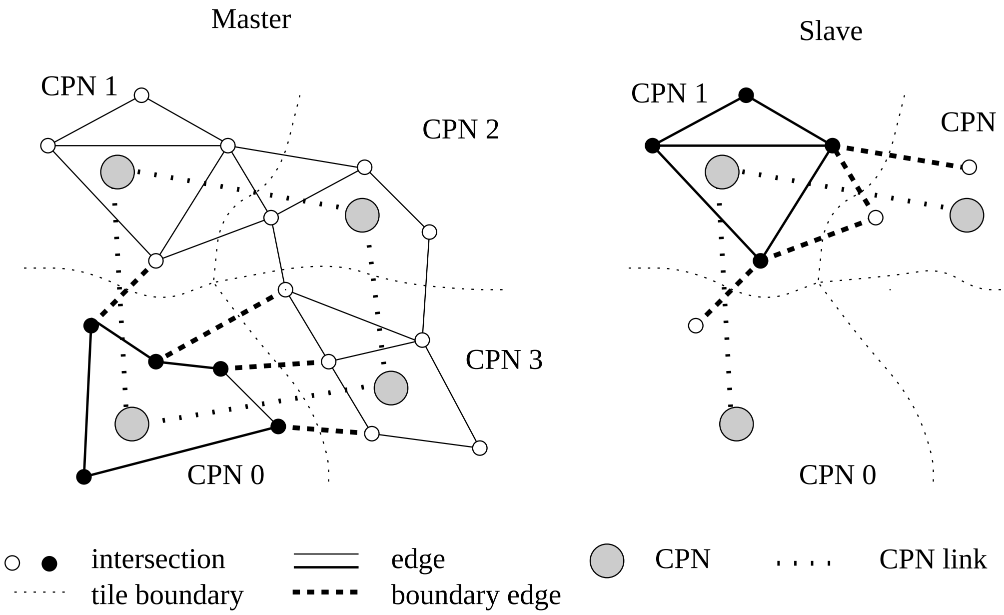

Figure A.2: *Geometric distribution of a street-network* — LEFT: Traffic network on the master: inactive representation of the complete network. Active representation of the local sub-net. RIGHT: Traffic network on a slave (here CPN 1): only active representation of the local sub-net.

- As a consequence of the distribution the tiles exchange *boundary information* containing all vehicle data necessary for the execution of the traffic rule set, resulting in local communication between neighboring tiles. Obviously, there is no real counterpart for a boundary in the original traffic simulation. It is an artifact of the parallel implementation. We differentiate between two resolutions of vehicle data: (a) the *primary vehicle data* only contains information on the location of vehicle, which is required for the CA rule set. (b) The *secondary data* contains all other information about the vehicle, such as maximum velocity and route-plan.

The traffic links and intersections are mapped onto the structural elements supplied by the toolbox. These are *nodes*, *edges*, and *boundaries*:

- The *node* class was used to represent exactly one node of the traffic network. The toolbox guarantees that nodes exactly reside on one CPN. This is advantageous for the substructures (see 3.3.3) associated with a node: all elements can assume that other related elements of the same substructure reside on the same CPN. If an incident edge happens to be split (see below), at least the half of the edge next the node can be assumed to be local.

- The *edge* was used to represent a bi-directional multi-lane street segment. For each direction a multi-lane CA grid was used. In contrast to nodes, an edge may be duplicated by the toolbox in case that the incident nodes reside on different CPN. Such a so-called *boundary-edge* or *inter-CPN edge* is split exactly in the middle. A discrete CA grid of odd length l has to be handled with care by assigning $\lfloor l/2 \rfloor$ sites to one and $\lceil l/2 \rceil$ sites to the other CPN after breaking their symmetry. On one CPN the first half is active and on the other CPN the second half.

- Exactly in the middle *boundaries* are retrieved from the grids and transferred to the remote CPN. They contain information about the state of the traffic system close the split point.

A.1.3 Initial domain decomposition

The toolbox handles the initial distribution and subsequent load balancing if requested. The geometric node locations are used to perform a recursive orthogonal bisection of the traffic network.

For the initial distribution of a network with nodes $n_1 \dots n_N$ onto CPN's $C_1 \dots C_C$ we have to make three assumptions:

- We have performance values for each CPN given in arbitrary but proportional values $S_1 \dots S_C$ where larger values denote better performance.
- Each node n_i has an estimated load l_i associated with it which is derived from the complexity of the node itself and all its incident edges. In particular, we use the number of CA grid–sites on transfer lanes at junctions as a measure for the nodes, and the number of CA grid-sites on traffic links as a measure for the edges. In PAMINA III the actual execution time of the previous iteration is used individually for each network element (also see 6.4.8).
- Each node n_i has an Euclidean location (x_i, y_i).

Recursive algorithm

The algorithm is defined recursively for a set of nodes $n_p \dots n_q$, a set of CPN's $C_a \dots C_b$, and recursive depth d.

1. If the number of CPN's $b - a + 1$ is one, assign all nodes to this CPN.
2. Otherwise, split CPN's in halves $C^l = \{C_a, \dots, C_{\lfloor a+b/2 \rfloor}\}$ and $C^r = \{C_{\lceil a+b/2 \rceil}, \dots, C_b\}$ with sum performances $S^l = \sum_{i=a}^{\lfloor (a+b)/2 \rfloor} S_i$ and $S^r = \sum_{i=\lceil a+b/2 \rceil}^{b} S_i$.
3. Sort nodes according to their x-coordinates for even depth d and according to their y-coordinates for odd depth d.
4. Split nodes at node n_j $(p < j \leq q)$ into two sets $N^l = \{n_p, \dots, n_{j-1}\}$ with load $L^l = \sum_{i=p}^{j-1} l_i$ and $N^r = \{n_j, \dots, n_q\}$ with load $\sum_{i=j}^{q} l_i$ in such a way that their load ratio is equivalent to the ratio of the performance values:
$$\frac{L^l}{L^r} \overset{!}{\simeq} \frac{S^l}{S^r}$$
Due to the granularity of the l_i exact equality may not be achieved. In this case, j has to be chosen in such a way that the inequality in minimized.
5. Split the subsets C^l with nodes N^l and C^r with nodes N^r recursively.

Corrections

Since no topological aspects are considered, the resulting tiles may not be connected anymore. Nevertheless, each CPN can re-establish a single connected component by casting off all superfluous, not connected components to neighbors and keeping the largest one only.

If the area of the map is not square, the first split of the nodes should be parallel to the *shorter* edge of the rectangle. This way, unnecessarily narrow stripes are avoided.

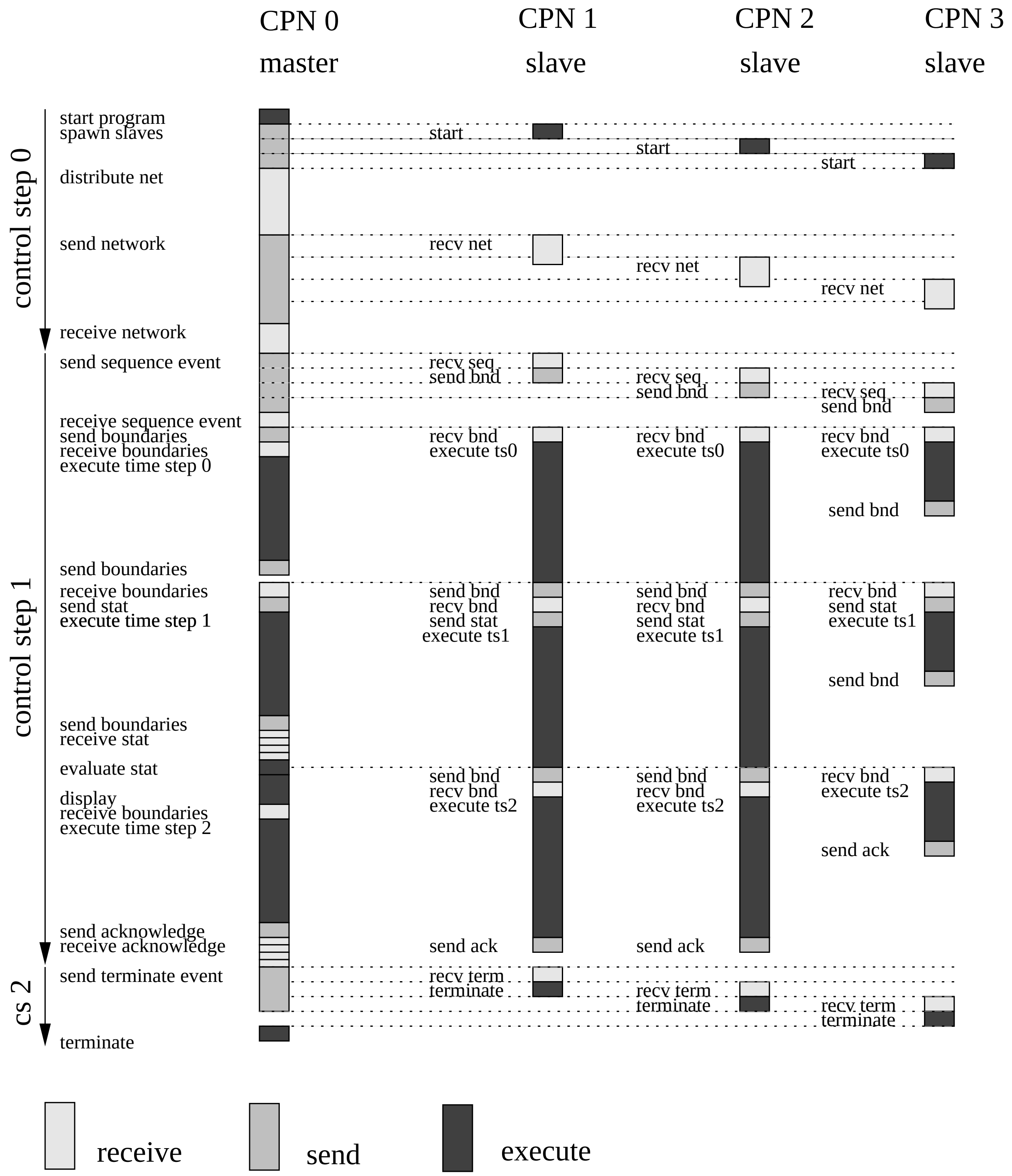

Figure A.3: *Timing of a simulation run*

A.1.4 Simulation control

The toolbox uses a master-slave algorithm as control logic for program execution. The master process is started first, usually directly by the user. It spawns several slaves on the parallel computer architecture. For the simulation itself the master also operates as a slave. It does, however, also retain some additional functionality.

The following enumeration is supposed to convey an idea of the main steps executed during a simulation run. See Figure A.3 schematic overview of such a run on four CPN's.

1. The user initializes PVM on the future master CPN by calling the interactive PVM-shell `pvm`. He adds all CPN's manually or starts a script to do this automatically. The user starts the application executable which is the master process.

   PAMINA III also supports MPI as message passing library, in which case the user simply calls the application executable with one additional command line parameter (`-np`) defining the number of CPN's to be used.

2. The master starts all other instances of the program on the slave CPN's. They will enter the main message loop and wait for messages.

3. The master reads the network structure from the input data files.

4. The master distributes the network and sends out messages to the slave CPN's with encoded network elements.

5. The master sends an event to all CPN's to start a simulation sequence of a given number of time-steps. All slaves send out the boundaries for the first time-step to the neighboring CPN's. During that sequence the master mainly functions as a slave.

6. The slaves enter the main loop:

   - They wait for the arrival of all boundaries from their neighbors.
   - If necessary, statistical data (e.g. idle time statistics) is sent out.
   - If idle-time statistics are available local load balancing is done.
   - They execute a time-step.
   - Unless the end of the sequence is reached they send out the boundaries for the next time-step.

   The master has several additional functions:

   - If available, global statistical data from the slaves is processed and displayed.
   - The X-Windows event queue is checked if there is need to update the graphics output.
   - The PVM environment is checked for changes of the CPN topology. If necessary, CPN's may be removed from the topology or new CPN's may be added to the topology.

7. The master stops the simulation by sending an termination event to all slaves.

8. The master and the slaves stop execution.

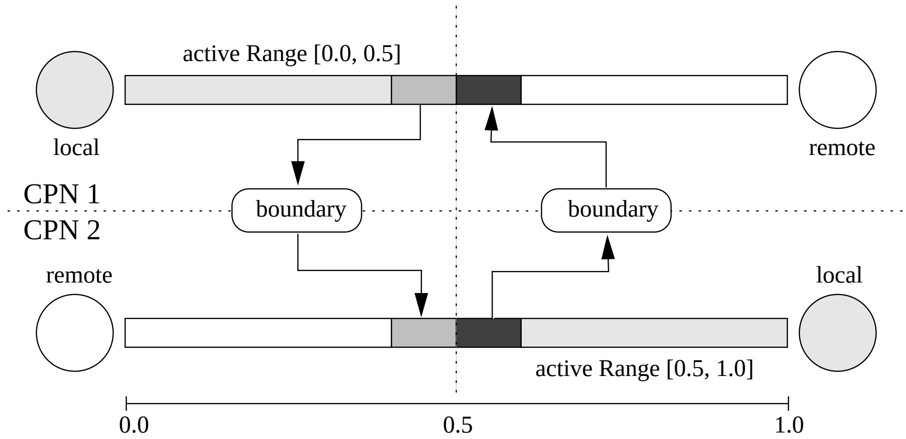

Figure A.4: *Exchanging boundaries*

A.1.5 Boundaries

After the distribution of the nodes of the network there will be edges crossing CPN boundaries which are called *boundary edges*. They are cut exactly in the middle so that each associated CPN computes half of the edge. Note that, therefore, boundary edges exist twice (see Figure A.4). Before either CPN can execute a time-step it obtain information about the objects on the remote CPN. We differentiate between *primary* and *secondary* vehicle data. Primary data only contains information about the *location* of the any objects in the boundary area, which is sufficient for evaluating the CA rule set. Secondary data also contains information about (or rather copies of) the vehicles themselves.

As for the width of the boundary we have to transmit information about all vehicles within the *interaction range*. This requires encoding and decoding of all vehicle data that are located within a range of v_{max} sites from a boundary. The information is transferred to the remote CPN. Over there, it is given to the duplicate of the edge which appends this information to the local data stored on the edge. The additional data allows the execution of the time-step.

The actual size (byte-wise) of boundaries can be optimized by taking advantage of specific characteristics of the CA rule set. If the local density is high, that is, the boundary is located within a traffic jam, there may be more than one vehicle per lane. The CA rules, however, only refer to the *immediate* predecessor or successor on each lane, reducing the maximum number of vehicles in a boundary to one per lane. Moreover, only the *primary* vehicle data is needed and not the secondary data including the route-plan. This is true, at least for the first sub-time-step. In the second sub-time-step all vehicle data is needed to guarantee a consistent vehicle migration across CPN boundaries, which will be described next.

Consistent handling of boundary objects

Boundaries contain regions in which the same objects are handled by both CPN's (see Figure A.5). In order to guarantee consistence of the simulation it is necessary that both instances of the same object behave exactly *identically*. In case of a deterministic simulation this is, of course, no problem. In case that decisions of objects depend on random numbers it is necessary to make the random generator[3] part of the object and to pass it together with the object data to the remote CPN. In the CA model the stochasticity of the motion of a vehicle is completely determined by a single bit which is set with a probability of p_{dec}. In PAMINA II this bit is included whenever secondary boundary information is transferred.

[3] or rather: values necessary to reproduce the random sequence on the remote CPN, which could be for example the current seed of the generator

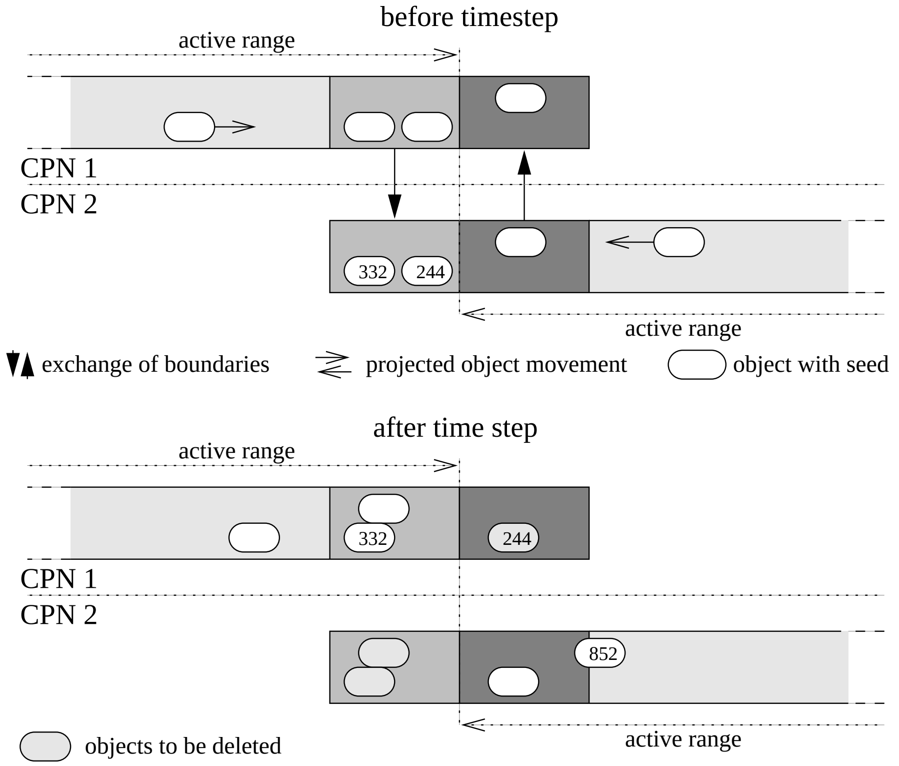

Figure A.5: *Consistent Handling of Boundary Objects*

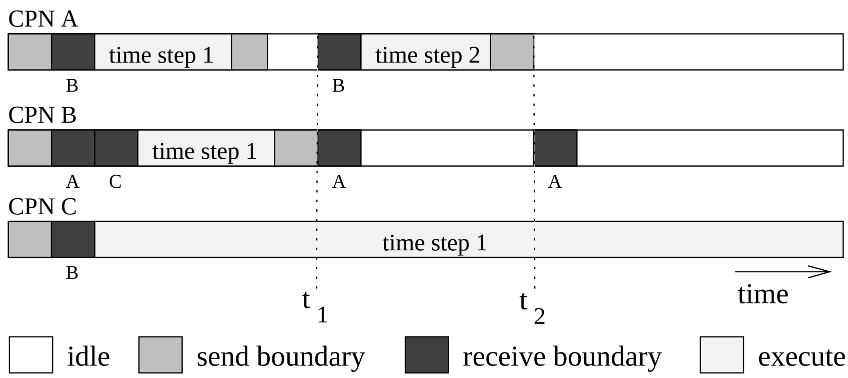

Figure A.6: *Timing of boundaries*

During a time-step the objects on an an edge usually change positions, that is, some will probably leave the remote boundary and enter the normal active part of the edge whereas others will do vice versa (see Figure A.5). After the time-step all objects have to be deleted that still remain in boundaries supplied by remote CPN's (object 244 for CPN 1 and objects 332 and 567 for CPN 2). Likewise all objects that have entered the local active part of the edge have to be permanently inserted (object 567 for CPN 1 and object 244 for CPN 2). Note that object 332 has to be deleted, too, although it would be at the correct location for the next time-step. Leaving it in the edge, however, would lead to collision with the boundary for the next time-step which will contain another copy of that object.

A.1.6 Timing of a simulation time-step

The simulation uses a parallel update with a global time-step. However, synchronization of all CPN is only performed after a so-called *simulation–sequence* comprising approximately 10-20 time–steps. In between, there is only a weak synchronization through the exchange of boundaries. Between neighboring CPN's there may be a difference in time-steps of $\Delta t = \pm 1$ as displayed in Figure A.6: due to slow execution of time step 1 on CPN C, CPN B has received boundaries from CPN A for time step 2 *and* already for time-step 3. The toolbox buffers those early boundaries automatically.

The global time–step is used to guarantee consistent collection of statistical data: Although partial results from the CPN may not be collected at the same physical wall-clock time due to a potential time-step gradient (see [83]), they always belong to the same logical time-step. The master CPN takes care of combining partial results.

Each global time-step is subdivided into two sub-time-steps. The first sub-time-step is used for lane changing, while the second sub-time-step is used for forward motion. Each sub-time-step requires the exchange of boundaries between CPN, although they are of different resolution: the first time-step only requires the transfer of primary vehicle data, while the second sub-time-step also comprises the secondary data.

Each sub-time-step is subdivided into a preparation phase (P) and an execution phase (E) preceded by the implicit local synchronization (IS) through boundary exchange as summarized in Table A.1.

A.1.7 Dynamic load-balancing

During the execution of the simulation a quasi-dynamic load-balancing is performed in PAMINA II. The implemented method corresponds to a *local decision, local migration* ($LDLM_S$, see 6.3.2) strategy applied to the network nodes. Incident edges are transferred or split accordingly. When a part of a local network has to be migrated, nodes are sequentially transferred along the boundaries with the node furthest away

sub-time-step	IS/P/E	Action
1	IS	exchange primary vehicle data, gather statistics
1	P	CONN, DR, MR, resolve dead-locks
1	E	lane change
2	IS	exchange all vehicle data
2	P	CONN, ER, AR
2	E	motion, migration

Table A.1: *Timing of a simulation time-step*

from the center of the subnetwork being selected first. As an optional restriction only those nodes can be selected that maintain one connected component on the CPN.

Measuring load

During the simulation each CPN keeps track of the time it spends on different tasks:

execution time is the time used for execution of the micro-simulation itself. It includes the time to update intersections and links, insertion and deletion of vehicles, and handling of route-plans.

idle time is the time the CPN spends idle. In particular, for both PVM and MPI it is the time spent in blocking receive calls.

graphics time is the time spent on retrieving and displaying graphics. As a rule this value will be non-zero only for the master.

boundary time is the spent on retrieving, sending, and receiving boundaries.

work time is everything else.

In addition to those five timers, each CPN also computes the estimated load of the residing subnet l and a performance P. The load is measured in the same units that were used for the initial load-balancing (see A.1.3).

$$P = \text{corrected load} = \frac{\text{estimated load}}{\text{execution time}} = \frac{l}{t}$$

This value is stored in a queue of constant length. The minimum value

$$P^{min} = \min_Q P$$

of all values in the queue Q is used as a measure of realistic performance. In certain intervals each CPN uses the neighbor statistics mechanism to propagate this P^{min} information to its neighbors together with l. Since the performance of the CA only weakly depends on the number of vehicles in a grid we use a value proportional to the number of grid sites handled by a segment as a measure for its computational load. Measurements (see Figure 6.5) confirm that the time required for updating one million sites [MUP] only varies by a factor of two for densities between $\varrho = 0.003 \ldots 0.3$.

Selecting amount of transfer

As soon as a CPN receives a complete set of performance values from its neighbors, the following steps are executed. Assume a CPN having n neighbors $N_1 \ldots N_n$ with corrected performance minimum $P_1^{min} \ldots P_n^{min}$ and estimated loads $l_1 \ldots l_n$. The local execution time is t_0 and the local load l_0. If the performance exceeds that of neighbor N_j by a more than minimum percentage t_{min} the CPN will try to migrate a part of its network to that neighbor.

To compute the exact amount let us assume that after the transfer the execution times of the two CPN's t_0 and t_j should be equal. If l_t is the amount of load to be transferred one obtains

$$t_0 := \frac{l_0 - l_t}{P_0^{min}} \stackrel{!}{=} \frac{l_j + l_t}{P_t^{min}} =: t_j$$

After isolation of l_t one obtains

$$l_t = \frac{l_0 P_j^{min} - l_j P_0^{min}}{P_0^{min} + P_j^{min}}.$$

This l_t is the optimal amount of load that should be transferred. Unfortunately the CPN does not know about the other neighbors of N_j which might migrate load to N_j, too. So l_t is corrected by a factor c which depends on the number of neighbors n_j of N_j, for example $c = \frac{1}{n_j}$, so that the effective load amounts to

$$l_t^{eff} = \frac{1}{n_j} \frac{l_0 P_j^{min} - l_j P_0^{min}}{P_0^{min} + P_j^{min}}.$$

Thus, c serves as dampening factor to prevent the system from over-compensating.

Now, the load-balancing algorithm tries to migrate load to the neighbor it has the largest load difference with.

Selecting time of transfer

The talk mechanism provides a means to perform a locally synchronized exchange of data during load-balancing. It ensures that the time-steps of the sending and the receiving CPN's match exactly before the migration is started. A so-called *talk* between CPN A from which it *originates* and *answering* CPN B is executed as follows:

1. CPN A calls requests a talk from the toolbox passing three parameters:
   - the ID of the other CPN,
   - the *Talk-ID*, a unique number by which the "topic to be talked about" is identified,
   - a ΔT relative to the current time-step T_{now} after which the talk is to be established.
2. CPN A continues execution until the requested time-step $T_{talk} = T_{now} + \Delta T$ is reached. In between CPN B has received the talk request. It also continues execution.
3. Now, let us assume both CPN's A and B have reached the same time-step T_{talk}. The toolbox prompts CPN A to start the actual talk.
4. CPN A starts talking to CPN B in a ping–pong manner until either of them terminates the talk.
5. Both continue their normal operation.

There are some aspects which need special attention:

- If both CPN's issue equivalent talk requests with the same *Talk-ID* and for the same effective time-steps, a collision occurs. In this case one of the talk requests is canceled by the toolbox. The Ids of the CPN's and the time-step are used to determine which one.

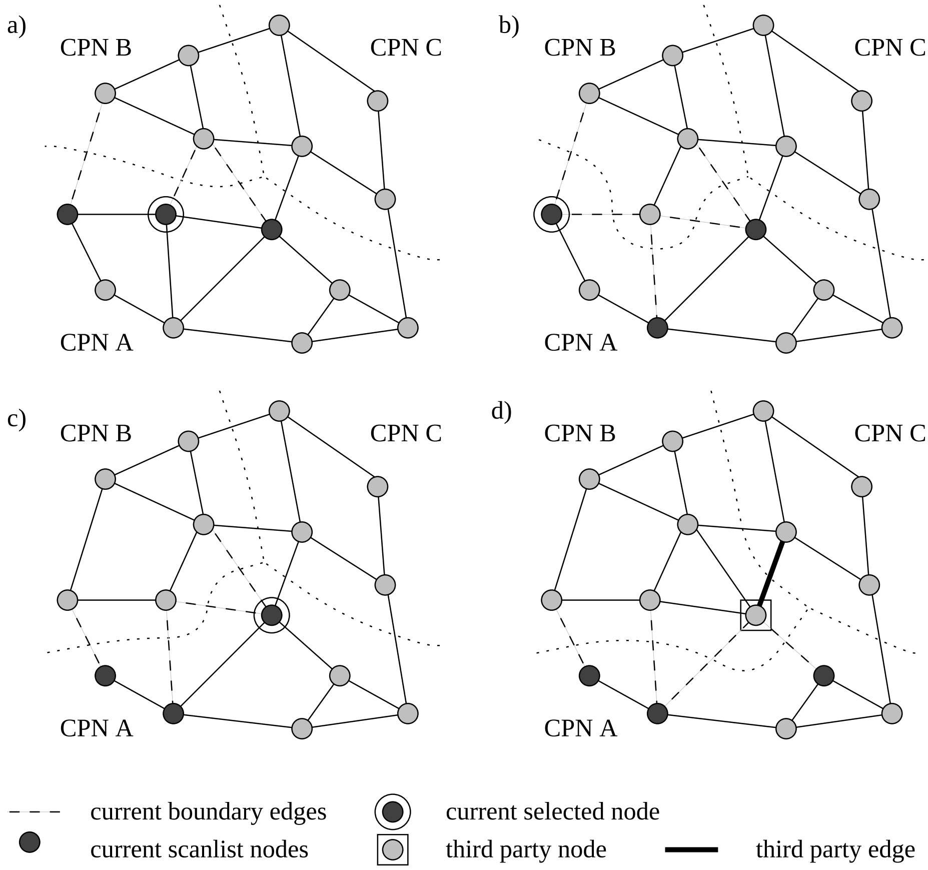

Figure A.7: *Selecting topology* — The parts (a) through (d) depict the iterative selection of three nodes for migration from CPN A to CPN B.

- A CPN might issue more than one request for the same time-step with more than one CPN using one or more talk Ids. In such a case no assumption can be made in which order the talks are initiated except that requests for the same time-step with the same CPN and different talk Ids are handled exactly in the order they were requested.
- There is only one active talk per CPN at any time.
- During the talk neither of the CPN performs any computation. Therefore the talk protocols should be kept to be as short as possible.

Selecting traffic network

At this point we assume that we know *how much* load is to be migrated and that the talk mechanism has established a communication channel to the destination CPN. It is now necessary to determine *what* topology is to be transferred. The toolbox takes an iterative approach (see Figure A.7):

1. CPN A goes through all boundary edges it has in common with CPN B[4]. All local nodes that are reached by those boundary edges are added to a scan-list[5].

[4]If CPN's A and B have at least one boundary edge in common, there is a CPN link between the CPN-node of A and the CPN-node of B. This CPN link contains a list of all boundary edges that both CPN's have in common.

[5]Some nodes may be reached by more than one boundary edge. They are only added once.

2. Among all nodes in the scan list, the one closest to the center of mass of the sub-net is selected for migration.
3. All edges leading from a local node to the selected node are marked to be migrated. All local nodes reached by the incident edges of the selected node are added to the scan-list unless they are already part of it. The selected node is removed from the scan-list.
4. Repeat steps 2 and 3 until the desired amount of topology is reached.

Some nodes which were transferred may be referenced by other CPN's. Those which will be called *third party nodes*. The same applies to edges referencing a node which is owned by another CPN. Those are called *third party edges*. In step (d) of Figure A.7 there is one node on CPN C referencing one of the nodes to be migrated from A to B and one edge between A and C that will connect B and C after migration. In such a case two measures are necessary:

- CPN C has to be informed about the change of ownership of the third party node.
- CPN A has to see to it that boundaries sent by CPN C for the third party edge will be forwarded to CPN B. This is necessary since there is no synchronization between CPN C and CPN A or B. So if CPN C is fast compared to CPN A it might have already sent out boundaries for topology that is no longer residing on CPN A.

The toolbox handles this problem by keeping track of edges that have just been migrated to another CPN. If a boundary arrives which refers to an edge no longer residing on the local CPN it is *forwarded* to the new owner of the edge. Since the maximal time-step gap between neighboring CPN is one, any forwarding information can be discarded two time-steps after the corresponding migration.

A.1.8 Fault tolerance

Dynamic insertion of CPN's

The toolbox allows the dynamic insertion of a CPN during the run time of the simulation. This option is advantageous in a local area network since the complete set of workstations in the network may not be available at the start of the simulation. It is the responsibility of the user to prompt the insertion of CPN's. The following list summarizes the most important steps of an insertion.

1. The user adds another CPN through the PVM-shell.
2. The master receives a predefined PVM message containing information about the new CPN CPN_N added and initiates a global synchronization for all CPN's at the end of the current control step.
3. The master uses the idle time data of the most recent time-steps to determine which CPN has least idle-time. Let CPN_A be that CPN.
4. The master determines which neighbor CPN_B of CPN_A has least idle time.
5. The master prompts CPN_A to transfer a *single* node having a common boundary edge with CPN_B to CPN_N.
6. CPN_A informs CPN_B about the transfer with the new node having the status of a third party node and possibly third party edges described in A.1.7.
7. The master initiates the next control step and the simulation continues.
8. Over the next few time-steps load-balancing is blocked for the new CPN. Then, its neighbors will start migrating nodes through regular load-balancing until its load is balanced with those of its neighbors.

Dynamic deletion of CPN's

In contrast to the insertion of a CPN which can be seen as optional feature, the ability to dynamically delete a CPN CPN_D is much more helpful in real-world computer networks. This time, however, the event will probably be triggered by the shutdown mechanism of the UNIX operating system. In a normal shutdown on a CPN each process receives the defined signal `SIGHUP` prompting the process to terminate gracefully before the signal `SIGKIL` is issued. The toolbox uses the signal to remove the CPN from the CPN topology before the CPN is completely shut down. Note that this will probably require a longer delay period between the two signals than defined by default. This can be changed by the system manager.

1. CPN_D receives the signal SIGHUP.
2. CPN_D refuses to accept any road network from its neighbors, but instead tries to migrate everything down to a *single* node.
3. CPN_D informs the master when the node count has reached one.
4. At the end of the current control step the master initiates a global synchronization.
5. The master picks out one neighbor CPN_A of CPN_D and prompts CPN_D to transfer the remaining node to CPN_A.
6. The master terminates the slave process on CPN_D.
7. The master initiates the next control step and simulation continues.
8. Over the next time-steps the neighbors of the deleted CPN will migrate load to their neighbors (and so on) through regular load balancing until the excess load is distributed over the whole CPN topology.

A.2 Background Traffic

In addition to the network traffic generated by routes it is possible to select a certain density of non-routed vehicles as background. These vehicles are generated automatically during startup and are homogeneously distributed across the network. Their CA behavior is exactly equivalent to that of routed vehicles except at network nodes: at terminators non-routed vehicles are not removed from the network but reinserted into the opposite direction using connectors. Therefore non-routed vehicles do not "see" terminators at all. Also, they completely disregard ramps. As for intersections of degree three and four, a certain ratio of non-routed vehicles behave as though they had temporary route-plans: With a given turning probability a vehicle is marked and later absorbed at the associated absorption range.

Note that for each vehicle this probability is only applied once even though it may remain on the marking range for more than one time-step. This is to maintain a density-independent exit behavior.

As a side effect the background traffic can be used to check the consistency of the simulation especially for the distributed version: running the simulation without routes, that is, with background traffic only, the number of vehicles is constant. Inconsistencies due to the intersection or parallel functionality can thus easily be detected since those usually result in a loss of vehicles or additional (phantom) vehicles.

Appendix B

Traffic

B.1 Route aging

Three different route selection schemes were tried for the route re-planning in Chapter 4. We assume a stationary distribution $f(a)$. If $g(a)$ is the probability for a route of age a to be re-planned, the following equation has to hold:

$$f(a+1) = f(a) - g(a)f(a)da \tag{B.1}$$

Isolation of $g(a)$ yields

$$\frac{f'(a)}{f(a)} = g(a).$$

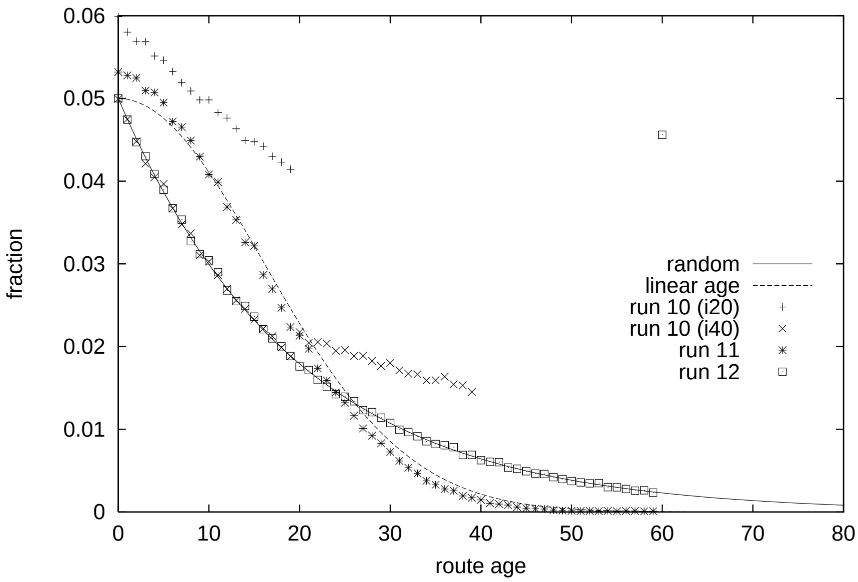

Figure B.1: *Age distribution of route-plans* — The theoretical curves for random and linear age selection are overlaid by actual distributions retrieved from runs 10, 11, and 12.

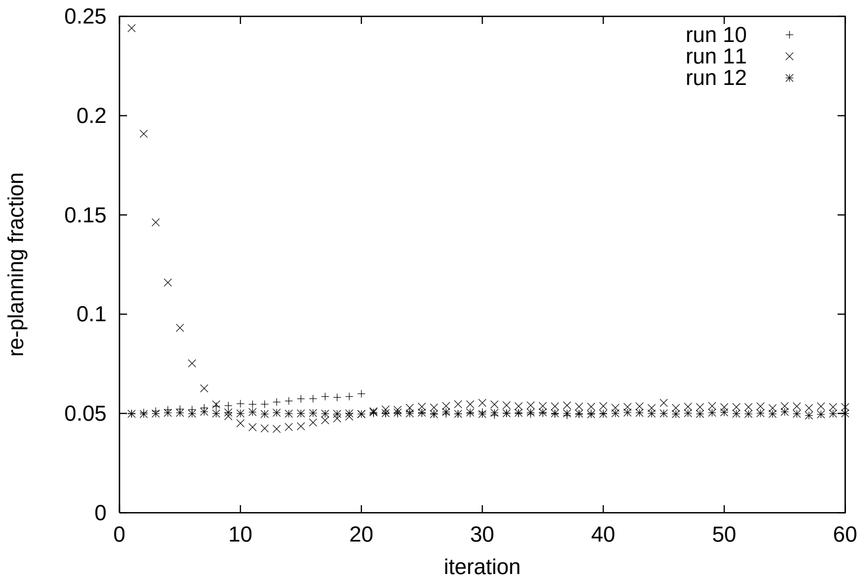

Figure B.2: *Re-planning fraction with respect to iteration number* — The linear age selection starts out with fractions considerably above $f_r = 0.05$. Only when the age distribution is stationary, the re-planning fraction approaches its correct value.

Subsequent integration yields

$$f_{lin}(a) = e^{\int g(a)da}.$$

The most straight forward is the *random selection* method in which all routes are equally likely to be picked independent of their age, that is $g(a) = f_r$. We obtain a decreasing exponential function

$$f_{rnd}(a) = f_r(1 - f_r)^a.$$

Figure B.1 depicts this function and counts which were retrieved from the 60 iterations of run 11. Up to iteration 59 the points exactly overlay the curve. Iteration 60 has a fraction of 0.43 representing all routes that have never been re-planned.

For run 10 during the first 20 iterations 5% of all the initial route-plan were re-planned. Afterwards, all plans had been re-planned at least once. The figure shows a linear decay which is centered around the average fraction of $f_r = 0.05$. After another 20 iterations with random selection, the points match the exponential curve up to iteration 20. The iterations 20 through 40 show a higher fraction than the exponential distribution but no exceptionally high fraction for the last one as in run 11.

For the linear age selection we use $g(a) = ca$ and $f_{lin}(0) = f_r$ we obtain a normal distribution

$$f_{lin}(a) = f_r e^{ba^2}$$

Figure B.1 shows f_{lin} for $f_r = 0.05$ and the actual fractions from run 11. Note that the actual distribution is slightly higher for small a. This is due to the fact that equation B.1 is only for correct for continuous age distributions. The iterative process, however, does a discrete update.

Figure B.2 shows the re-planning fractions for runs 10, 11, and 12. Run 12 maintains its value exactly at $f_r = 0.05$ throughout all iterations. Run 10 has a linear increase until iteration 20 due to the additional random selection of $f_r = 0.01$ which overlays the reduced selection of $f_{red} = 0.05$. For run 11 the curve starts out at $f_r = 0.24$ for the first iteration. It drops below $f_r = 0.05$, but eventually levels off slightly above the desired value. The high re-planning fractions are caused by the initial age of all route-plans, which were set to be $a = 60$ at the beginning of the iteration. If we had used the actual age $a = 0$, the first iterations would have negligible re-planning fractions.

B.2 Map sizes

	NRW	FRG	Dallas (local streets)
nodes	549	3,307	2292
edges	1,160	6,860	6124
terminators	19	46	
ramps	349	1,568	
intersections (degree=3)	39	176	
intersections (degree=4)	21	58	
nodes (degree≠2)	79	280	
transfer segments (TS)	1,720	7,440	
lane-kilometer	11,712	74,844	2,276
sites	1,561,600	9,979,200	303,500
average edge length [sites]	448	484	50

Table B.1: *Network Sizes*

B.3 Computing the effective travel-time

ALGORITHM ComputeEffectiveTravelTime (tt)
BEGIN
$d := 1$
$tt_{eff} := 0$
$i := \lfloor T/t_b \rfloor$
$T_b := i * t_b$
WHILE $d > 0$
BEGIN
IF $T + d * tt(i) > T_b + t_b$ **THEN**
$tt_{eff} := tt_{eff} + T_b + t_b - T$
$d := d - (T_b + t_b - T)/tt(i)$
ELSE
$tt_{eff} := d * tt(i)$
$d := 0$
$i := i + 1$
END
RETURN tt_{eff}
END

Algorithm1: Improving $tt(i)$

List of Figures

List of Tables

Bibliography

[1] M. Van Aerde, B. Hellinga, M. Baker, and H. Rakha. INTEGRATION: An overview of traffic simulation features. *Transportation Research Records*, in press.

[2] R.K. Ahuja, Th.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice Hall, 1993.

[3] A. Alvarez, J. J. Brey, and J. M. Casado. A simulation model for traffic flow with passing. *Transpn. Res. B*, 24:193–202, 1988.

[4] D. Anson, C.L. Barrett, M. Marathe, K. Nagel, and M. Stein. A theoretical study of some routing algorithms. Technical report, TSA-DO/SA LANL, 1997. LA-UR 97-1333.

[5] C.L. Barrett. personal communication.

[6] C.L. Barrett, S. Eubank, K. Nagel, S. Rasmussen, J. Riordan, and M. Wolinsky. Issues in the representation of traffic using multi-resolution cellular automata. Technical report, Los Alamos, 1995. Unclassified Report LA-UR 95-2658.

[7] C.L. Barrett and M.A. Wolinsky. Incident-induced flow anomaly analysis and detection testbed final report: Design and concept demonstration. Technical report, Los Alamos National Laboratory, TSA-DO/SA, 1995.

[8] R.J. Beckman, K.A. Baggerly, and M.D. McKay. Creating synthetic baseline populations. *Transportation Research, A*, 30, No.6:415–429, 1996.

[9] E. Ben-Naim, P. L. Krapivsky, and S. Redner. Kinetics of clustering in traffic flows. *Phys. Rev. E*, 50(2):822, 1994.

[10] T. Benz. The microscopic traffic simulator AS (Autobahn Simulator). In A. Pave, editor, *European Simulation Multiconference*, page 486. The Society for Computer Simulation, Istanbul, 1993.

[11] R. Berrendorf, M. Gerndt, Z. Lahjomri, and T. Priol. A comparison of shared virtual memory and message passing techniques based on a finite element application. Technical report, Forschungszentrum Jülich, Germany, 1994. KFA-ZAM-IB-9410.

[12] M. Böhm and E. Speckenmeyer. A fast parallel SAT–solver — efficient workload balancing. *Annals of Mathematics and Artificial Intelligence*, 17:381–400, 1996.

[13] M. Böhm and E. Speckenmeyer. Precomputation-based load balancing. In F. Hoßfeld, E. Maehle, and E. W. Mayer, editors, *Proceedings of the 4th PASA Workshop,1996*, pages 177–194. World Scientific, 1997.

[14] D. Braess. Über ein Paradoxon der Verkehrsplanung. *Unternehmensforschung*, 12:256–268, 1968.

[15] J.G. Brankov, V.B. Priezzhev, A. Schadschneider, and M. Schreckenberg. The Kasteleyn model and a cellular automaton approach to traffic flow. preprint, 1995.

[16] G.D.B. Cameron and C.I.D. Duncan. PARAMICS – Parallel microscopic simulation of road traffic. *J. Supercomputing*, in press, 1996.

[17] CASim. URL http://rs1.comphys.uni-duisburg.de/OLSIM/.

[18] G.-L. Chang, H.S. Mahmassani, and R. Herman. Macroparticle traffic simulation model to investigate peak-period commuter decision dynamics. *Transp.Res.Rec.*, 1005:107–121.

[19] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest path algorithms: theory and experimental evaluation. *Math.Prog.*, 73:129–174, 1996.

[20] M. Cremer and J. Ludwig. A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, 28:297–303, 1986.

[21] A.J. Cuesta, F.C. Martínez, J.M. Molera, and A. Sánchez. Phase transitions in two-dimensional traffic flow models. *Phys. Rev. E*, 48(6):R4175–R4178, 1993.

[22] C.F. Daganzo. The cell transmission model: A dynamic representation of traffic consistent with the hydrodynamic theory. University of California, Berkeley CA 94720, 1993.

[23] P. Diniz, S. Plimpton, B. Hendrickson, and R. Leland. Parallel algorithms for dynamically partitioning unstructured grids. In *SIAM 95*, pages 615–620, SanFrancisco, February 1995.

[24] DOC++. URL http://www.zib.de/visual/software/doc++/.

[25] H. Elbern. Parallelization and load balancing of a comprehensive atmospheric chemistry transport model. *Atmos.Env.*, 31:3561–3574, 1997.

[26] R.H.M. Emmerink, K.W. Axhausen, P. Nijkamp, and P. Rietveld. The potential of information provision in road transport networks with non-recurrent congestion. Technical report, Tinbergen discussion paper TI 94-30, Tinbergen Institute, 1994.

[27] R.H.M. Emmerink, K.W. Axhausen, P. Nijkamp, and P. Rietveld. Effects of information in road transport networks with recurrent congestion. *Transportation Science*, 22:21, 1995.

[28] J. Esser. *Simulation von Stadtverkehr auf der Basis zellularer Automaten.* PhD thesis, University of Duisburg, Germany, 1997.

[29] J. Esser and M. Schreckenberg. Deriving traffic state information in urban road networks with microscopic simulations based on local measurements. Technical report, Fachbereich Physik-Technologie, Gerhard-Mercator-Universität Duisburg, Germany, 1997. submitted.

[30] J. Esser and M. Schreckenberg. Efficiency of route guidance systems in urban road networks. Technical report, Fachbereich Physik-Technologie, Gerhard-Mercator-Universität Duisburg, Germany, 1997. submitted.

[31] J. Esser and M. Schreckenberg. Microscopic simulation of urban traffic based upon cellular automata. Technical report, Fachbereich 11 / Diskrete Mathematik, Gerhard-Mercator-Universität Duisburg, Germany, 1997. to be published in Int.J.Mod.Phys.C.

[32] M. Van Aerde et al. *INTEGRATION – Release 2, User's Guide*, 1995.

[33] Message Passing Interface Forum. MPI: A message-passing interface standard. Technical report, University Of Tennessee, Knoxville, Tennessee, 1994.

[34] I. Foster. *Designing and Building Parallel Programs.* Addison Wesley, 1995.

[35] B.L. Fox. Data structures and computer science techniques in Operations Research. *Oper. Res.*, 26 No. 5:686–717, 1978.

[36] G.C. Fox, R. D. Williams, and P. C. Messina. *Parallel computing works!* Morgan Kaufmann, 1994.

[37] FRESIM User's Guide, Version 4.5, 1994.

[38] J. Freund and T. Pöschel. A statistical approach to vehicular traffic. *Physica A*, 219(1–2), 1995.

[39] Bundesminister für Verkehr, editor. *Verkehr in Zahlen.* Deutsches Institut für Wirtschaftsforschung, 1995.

[40] Ch. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. Technical report, Center for Parallel Computing, Universiy of Cologne, Germany, 1997.

[41] Ch. Gawron and P. Oertel. The PlayTraffic traffic simulation. Technical report, Traffic Group at the Center of Parallel Computing, University of Cologne, 1996.

[42] D.L. Gerlough and M.J. Huber. *Traffic Flow Theory.* Special Report No. 165. Transportation Research Board, National Research Council, Washington, D.C., 1975.

[43] B. Hendrickson and R. Leland. The CHACO user's guide: version 2.0. Technical report, Sandia National Laboratories, Albuquerque, NM, 1994. SAND94-2692.

[44] G. Horton. A multi-level diffusion method for dynamic load balancing. *Parallel Computing*, 19:209–218, 1993.

[45] T.-Y. Hu and H.S. Mahmassani. Day-to-day evoluation of network flows under real-time information and reactive signal control. *Transpn. Res.-C*, 5 No.1:51–69, 1997.

[46] J. JáJá. *An Introduction to Parallel Algorithms.* Addison Wesley, Reading, Massachusetts, 1992.

[47] T. Kelly. ATIS at rush hour: Adaptation and departure time coordination in iterated commuting. Technical report, Artificial Intelligence Lab, University of Michigan, USA, 1997.

[48] T. Kelly. Driver strategy and traffic system performance. *Physica A*, 235(3–4):407–416, 1997. A draft is available on the Web at `http://ai.eecs.umich.edu/people/tpkelly/papers/`.

[49] T. Kelly and K. Nagel. Characteristics of fastest paths in congested and uncongested simulated traffic networks. LA-UR 97-4453, submitted to Int.J.Mod.Phys.C, 1997.

[50] B.W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[51] A. Kiahashemi. Start-Ziel Matrizen für Verkehrsflüsse. Master's thesis, University of Cologne, 95?

[52] R. Knecht and G.A. Kohring. Dynamic load balancing for the simulation of granular materials. Technical report, Forschungszentrum Jülich, Germany, 1994. KFA-ZAM-IB-9428.

[53] S. Krauß, P. Wagner, and Ch. Gawron. A continuous limit of the Nagel-Schreckenberg model. *Phys. Rev. E*, 54:3707, 1996.

[54] S. Krauß, P. Wagner, and Ch. Gawron. Metastable states in a microscopic model of traffic flow. *Phys. Rev. E*, 5:5597, 1997.

[55] A. Latour. Simulation von Zellularautomaten–Modellen für Mehrspurverkehr. *Schriftliche Hausarbeit im Rahmen der Ersten Staatsprüfung*, 1993.

[56] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, 1992.

[57] W. Leutzbach and F. Busch. Spurwechselvorgänge auf dreispurigen BAB-Richtungsfahrbahnen. Technical report, Institut für Verkehrswesen, Universität Karlsruhe, 1984.

[58] M.J. Lighthill and G.B. Whitham. On kinematic waves (part ii): A theory of traffic flow on long crowded roads. *Proc. R. Soc. London*, A229:317–345, 1955.

[59] R. Lüling, B. Monien, and F.Ramme. Load balancing in large networks: A comparative study. In *3rd IEEE Symposium On Parallel And Distributed Processing*, pages 686–689, 1991.

[60] H.S. Mahmassani, R. Jayakrishnan, and R. Herman. Network traffic flow theory: Microscopic simulation experiments on supercomputers. *Transpn. Res. A*, 24A (2):149, 1990.

[61] M. Marathe, D. Anson, M. Stein, M. Rickert, K. Nagel, and C.L. Barrett. Engineering the route planner for the Dallas case study. In preparation.

[62] A.D. May. *Traffic Flow Fundamentals*. Prentice Hall, 1990.

[63] D. McArthur. The *PARAMICS* Model: Present and Future Directions. Technical report, SIAS Ltd., Edinburgh, 1994.

[64] M. McDonald and M. A. Brackstone. Simulation of lane usage characteristics on 3 lane motorways. In *Proceedings of the 27th International Symposium on Automotive Technology and Automation (ISATA)*, 1994.

[65] F. Meisgen. Dynamic Load Balancing for Simulations of Biological Aging. *International Journal of Modern Physics C*, 8(3):575–582, June 1997.

[66] B. Monien, R. Dieckmann, and R. Lüling. Communication throughput of interconnection networks. In *Proc. of MFCS '94*. Springer LNCS, 1994.

[67] B. Monien, R. Diekmann, R. Feldmann, R. Klasing, R. Lüling, K. Menzel, Th. Römke, and U.-P. Schroeder. Efficient use of parallel & distributed systems: From theory to practice. In J. van Leeuwen, editor, *Trends in Computer Science*. Springer, 1995.

[68] B. Monien, R. Feldmann, R. Klasing, and R. Lüling. Parallel architectures: Design and efficient use. Technical report, Department of Computer Science, University of Paderborn, Germany, 1993. A preliminary version of this report appeared as part of the proceedings of STACS 1993.

[69] T. Nagatani. Effect of traffic accident on jamming transition in traffic-flow model. *J. Phys. A*, 26(19):L1015, 1993.

[70] T. Nagatani. Dynamical jamming transition induced by a car accident in traffic-flow model of a two-lane roadway. *Physica A*, 202:449, 1994.

[71] T. Nagatani. Traffic jam and shock formation in stochastic traffic-flow model of a two-lane roadway. *J.Phys.Soc.Jpn*, 63:52, 1994.

[72] T. Nagatani. Bunching of cars in asymmetric exclusion models for freeway traffic. *Phys.Rev.E*, 51:992, 1995.

[73] T. Nagatani. Self-organization and phase transition in traffic-flow model of a two-lane roadway. *J.Phys.A*, 26:781–787, 93.

[74] K. Nagel. Personal communication.

[75] K. Nagel. *High–speed Microsimulations of Traffic Flow*. PhD thesis, Universität zu Köln, 1994.

[76] K. Nagel. Individual adaptation in a path-based simulation of the freeway network of Northrhine-Westphalia. *Int.J.Mod.Phys.C*, 7 No.6:883–892, 1996.

[77] K. Nagel. Particle hopping models and traffic flow theory. *Phys. Rev. E*, 53:4655, 1996. Los Alamos Unclassified Report 95-2908.

[78] K. Nagel. From particle hopping models to traffic flow theory. *Transportation Research Board meeting*, submitted, 1998.

[79] K. Nagel and C.L. Barrett. Using microsimulation feedback for trip adaptation for realistic traffic in Dallas. *Int.J.Mod.Phys.C*, 8 No.3:505, 1997. LA-UR 97-1334.

[80] K. Nagel and M. Paczuski. Emergent traffic jams. *Phys. Rev. E*, 51:2909, 1995.

[81] K. Nagel and S. Rasmussen. Traffic at the edge of chaos. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 222–235, Cambridge, MA, 1994. MIT Press.

[82] K. Nagel, S. Rasmussen, and C.L. Barrett. Network-traffic as a self-organized critical phenomenon. In F. Schweitzer, editor, *International Conference "Self-organization of complex structures: From individual to collective dynamics"*, 1995.

[83] K. Nagel and A. Schleicher. Microscopic traffic modeling on parallel high performance computers. *Parallel Computing*, 20:125–146, 1994.

[84] K. Nagel and M. Schreckenberg. A cellular automaton model for freeway traffic. *J. Physique I*, 2:2221, 1992.

[85] K. Nagel and M. Schreckenberg. Traffic jam dynamics in stochastic cellular automata. In J.I. Soliman and D. Roller, editors, *Proceedings of the 28th International Symposium on Automotive Technology and Automation (ISATA)*, page 531, Croydon, England, 1995. Automotive Automation Ltd, Croydon, England. Paper No. 95ATS089.

[86] K. Nagel, P. Stretz, M. Pieck, S. Leckey, R. Donelly, and C.L. Barrett. TRANSIMS traffic flow characteristics. Technical report, TSA-DO/SA, Los Alamos National Lab, 1997. LA-UR 97-3530.

[87] NRW-FVU. URL http://www.zpr.uni-koeln.de/Forschungsverbund-Verkehr-NRW/.

[88] United States Department of Transportation. URL http://www.dot.gov.

[89] R.A. Olsson, G.R. Andrews, M.H. Coffin, and G.M. Townsend. SR: A language for parallel and distributed programming. Technical Report TR 92–09, Department of Computer Science, University of Arizona, Tuscon, 1992.

[90] PARAMICS. URL http://www.epcc.ed.ac.uk/epcc-projects/PARAMICS/.

[91] D. Park and L.R. Rilett. Identifying multiple and reasonable paths in transportation networks: A heuristic approach. *Transpn.Res.Board*, in press, 1997.

[92] H.J. Payne. FREFLO: A macroscopic simulation model of freeway traffic. *Transportation Research Record 722*, 1979.

[93] J.T. Pfenning. *Beiträge zum Einsatz von "Workstation Clustern" als Parallel-Rechner*. PhD thesis, University of Cologne, 1994.

[94] M. Ponzlet and P. Wagner. Validation of a CA-Model for Traffic Simulation of the Northrhine-Westphalia Motorway Network. In *The 24th European Transport Forum, Proceedings of Seminar D&E*, volume P 404-1, 1996.

[95] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *J. Mat. Anal. Appl.*, 11(3):430–452, 1990.

[96] R. Preis and R. Diekmann. The PARTY partitioning library, User Guide, Version 1.1. Technical report, Department of Mathematics and Computer Science, University of Paderborn, Germany, 1996.

[97] PVM. URL http://www.epm.ornl.gov/pvm/pvm_home.html.

[98] A.K. Rathi and A.J. Santiago. The new NETSIM simulation program.

[99] M. Rickert. Simulation zweispurigen Verkehrsflusses auf der Basis zellularer Automaten. Master's thesis, Universität zu Köln, 1994.

[100] M. Rickert. Parallel Toolbox 1.0. Technical report, Center for Parallel Computing, Cologne, Germany, and TSA-DO/SA, Los Alamos National Lab, USA, 1995.

[101] M. Rickert and K. Nagel. Experiences with a simplified microsimulation of the Dallas/Fort Worth area. *Int.J.Mod.Phys. C*, 8 No.3:483–503, 1997.

[102] M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 231:534, 1996.

[103] M. Rickert and P. Wagner. Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. *Int.J.Mod.Phys. C*, 7:133–153, 1996.

[104] Th. Römke, M. Röttger, U.-P. Schneider, and J. Simon. On efficient embeddings of grids into grids in PARIX. Technical report, Paderborn Center for Parallel Computing and Department of Mathematics and Computer Science, University of Paderborn, Germany, 1194.

[105] A. Schadschneider and M. Schreckenberg. Cellular automaton models and traffic flow. *Journal of Physics A*, 26, 1993.

[106] M. Schreckenberg, A. Schadschneider, K. Nagel, and N. Ito. Discrete stochastic models for traffic flow. *Phys. Rev. E*, 51:2939, 1995.

[107] H. Schütt. Entwicklung und Erprobung eines sehr schnellen, bitorientierten Verkehrssimulationssystems für Straßennetze. Technical report, Schriftenreihe der AG Automatisierungstechnik TU Hamburg–Harburg, 1991.

[108] T. Schwerdtfeger. *Makroskopisches Simulationmodell für Schnellstraßennetze mit Berücksichtigung von Einzelfahrzeugen (DYNEMO)*. PhD thesis, TH Karlsruhe, 1987.

[109] K. Scott, G. Pabon-Jiminez, and D. Bernstein. Finding alternatives to the best path. *Transpn.Res.Board*, 76th Annual Meeting, 1997. Preprint 970682.

[110] D. Serwill. *DRUM - Modellkonzept zur dynamischen Routensuche und Umlegung*. PhD thesis, Institut für Stadtbauwesen, RWTH Aachen, 1994.

[111] Y. Sheffi. *Urban Transportation Networks*. Prentice Hall, 1985.

[112] H.P. Simão and W.B. Powell. Numerical methods for simulating transient, stochastic queueing networks. *Transportation Science*, 26:296, 1992.

[113] P.M. Simon. Personal communication.

[114] P.M. Simon and K. Nagel. A simplified cellular automaton model for city traffic. submitted to Phys.Rev.E, 1997.

[115] Z. Sinuany-Stern, E. Stern, Z. Sfaradi, and E. Holm. The effect of information on commuters' behaviour: A comparative micro-simulation approach. *Eur. J. Oper. Res.*, pages 455–470, 96.

[116] R. Smith, R. Beckman, D. Anson, and M. Williams. TRANSIMS, the TRansportation ANalysis and SIMulation System. Technical report, Los Alamos National Laboratory, 1995. LA-UR 95-1664.

[117] U. Sparmann. Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen. Technical report, Forschung Straßenbau und Straßenverkehrstechnik, 1978. Heft 263.

[118] A. Stathopoulos and T. Parmaksizoglou. A time-dependent traffic assignment procedure using transient o/d matrices. Technical report, Laboratory of Railways and Transport, Dept. of Transportation Planning and Engineering, National University of Athens, Greece, 1989.

[119] D. Stauffer. Computer simulations of cellular automata. *J. Phys. A*, 24:909–927, 1991.

[120] V.S. Sunderam. PVM: A framework for parallel distributed computing. *Concurrency — Practice and Experience*, 2:315–339, Dec. 1990.

[121] TRANSIMS. URL http://www-transims.tsasa.lanl.gov/.

[122] Th. Ungerer. *Parallelrechner und Parallele Programmierung.* Spektrum, Akademischer Verlag, 1997.

[123] SMARTEST Project URL. http://www.its.leeds.ac.uk/smartest/.

[124] D. van Vliet and M. Hall. SATURN - a simulation-assignment model for the evaluation of traffic management schemes. *Traffic Engineering and Control, London*, 4, 80.

[125] P. Wagner. Personal communication.

[126] P. Wagner, K. Nagel, and D.E. Wolf. Realistic multi-lane traffic rules for cellular automata. *Physica A*, 234:687, 1997. LA-UR 96-2586.

[127] D. Watling and T. van Vuren. The modeling of dynamic route guidance systems. *Transpn.Res.-C*, 1 No. 2:159–182, 1993.

[128] R. Wimmershoff. URL http://www.rrz.uni-koeln.de/RRZK/Autoren/WF/traffic/OneLaneCA.html.

[129] S. Wolfram. Theory and applications of cellular automata. *Singapore: World Scientific*, 1988.

[130] C.-Z. Xu and F.C.M. Lau. Optimal parameters for load balancing with the diffusion method in mesh networks. *Parallel Processing Letters*, 4(1&2):139–147, June 1994.

Erklärung

Ich versichere, daß ich die von mir vorgelegte Dissertation selbständig angefertigt, die benutzten Quellen und Hilfsmittel vollständig angegeben und die Stellen der Arbeit — einschließlich Tabellen, Karten und Abbildungen —, die anderen Werken im Wortlaut oder dem Sinn nach entnommen worden sind, in jedem Einzelfall als Entlehnung kenntlich gemacht habe; daß diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; daß sie abgesehen von unten angegebenen 'Teilpublikationen noch nicht veröffentlicht worden ist sowie, daß ich eine solche Veröffentlichung vor Abschluß des Promotionsverfahrens nicht vornehmen werde. Die Bestimmungen dieser Promotionsordnung sind mir bekannt. Die von mir vorgelegte Dissertation ist von Prof. Dr. Achim Bachem betreut worden.

Köln, den 6. Dezember 1997

Teilveröffentlichungen

- A. Bachem, K. Nagel, and M. Rickert. Ultraschnelle mikroskopische Verkehrssimulationen. In R. Flieger and R. Grebe, editors, *Parallele Datenverarbeitung Aktuell TAT*, 1994.
- K. Nagel, C. Barrett, and M. Rickert. Parallel traffic micro-simulation by cellular automata and application for large scale transportation modeling. Technical report, TSA-DO/SA, Los Alamos National Lab, New Mexico, USA, and Santa Fe Institute, Santa Fe, New Mexico, USA, and Center for Parallel Computing, University of Cologne, Germany, 1996.
- K. Nagel, M. Rickert, and C.L. Barrett. Large scale traffic simulations. In J.M.L.M. Palma and J. Dongarra, editors, *Vector and Parallel Processing – VECPAR'96*, page 380. Springer, 1997.
- M. Rickert and K. Nagel. Experiences with a simplified microsimulation of the Dallas/Fort Worth area. *Int.J.Mod.Phys. C*, 8 No.3:483–503, 1997.
- M. Rickert, K. Nagel, M. Schreckenberg, and A. Latour. Two lane traffic simulations using cellular automata. *Physica A*, 231:534, 1996.
- M. Rickert and P. Wagner. Parallel real-time implementation of large-scale, route-plan-driven traffic simulation. *Int.J.Mod.Phys. C*, 7:133–153, 1996.
- M. Rickert, P. Wagner, and Ch. Gawron. Real-time traffic simulation of the German Autobahn Network. In *Proceedings of the 4th PASA Workshop.* World Scientific, Singapore, 1996.

Lebenslauf Marcus Rickert

Geburtsdatum	26. Oktober 1967
Geburtsort	Bensberg (jetzt Bergisch Gladbach), Deutschland
Staatsangehörigkeit	deutsch
Eltern	Maria Magdalena Rickert (geborene Würtz) und Wolfgang Rickert
1974-1978	Besuch der Grundschule *Alte Wipperführter Straße* in Köln
1978-1987	Besuch des *Johann–Gottfried–Herder Gymnasiums* in Köln
3. Jun. 1987	Abitur
1. Okt. 1987	Immatrikulation an der Universität zu Köln, Studiengang Diplomphysik
28. Sep. 1989	Vordiplom in Physik
Seit 1989	Studentischer Mitarbeiter am Mathematischen Institut der Universität zu Köln und später am Zentrum für Paralleles Rechnen
2. Mai 1994	Diplom in Physik unter Anleitung von Prof. Dr. Achim Bachem
Jun. 1994 bis Jan. 1995	Aufenthalt am Los Alamos National Lab (USA) als wissenschaftlicher Mitarbeiter in der Gruppe TSA-DO/SA
Jan. 1995 bis Dez. 1997	Stipendiat im Graduiertenkolleg "Scientific Computing" an der Universität zu Köln
Jul. bis Okt. 1995	Zweiter Aufenthalt am Los Alamos National Lab
Jun. bis Okt. 1996	Dritter Aufenthalt am Los Alamos National Lab
Jun. bis Sep. 1997	Vierter Aufenthalt am Los Alamos National Lab