Torsten Bruns

Trajektorienplanung mittels Diskretisierung und kombinatorischer Optimierung am Beispiel des autonomen Kreuzungsmanagements für Kraftfahrzeuge

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen National-bibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar

Satz und Gestaltung: Torsten Bruns

Hersteller: Verlagshaus Monsenstein und Vannerdat OHG
Druck · Buch · Verlag
Münster

Printed in Germany

HEINZ NIXDORF INSTITUT
Universität Paderborn
Regelungstechnik und Mechatronik
Prof. Dr.-Ing. habil. Ansgar Trächtler

Trajektorienplanung mittels Diskretisierung und kombinatorischer Optimierung

am Beispiel des autonomen Kreuzungsmanagements für Kraftfahrzeuge

zur Erlangung des akademischen Grades eines
DOKTORS DER INGENIEURWISSENSCHAFTEN (Dr.-Ing.)
der Fakultät für Maschinenbau
der Universität Paderborn

genehmigte

DISSERTATION

von

Dipl.-Wirt.-Ing. Torsten Bruns

aus Bad Pyrmont

Tag des Kolloquiums: 13. Mai 2011
Referent: Prof. Dr.-Ing. habil. Ansgar Trächtler
Korreferent: Prof. Dr. math. Friedhelm Meyer auf der Heide

Vorwort

Die vorliegende Dissertation entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am *Lehrstuhl für Regelungstechnik und Mechatronik (RtM)*. Ihre Wurzeln gehen auf den Sonderforschungsbereich *Massive Parallelität: Algorithmen, Entwurfsmethoden und Anwendungen* (SFB 376) zurück, der im Jahre 1995 etabliert wurde und zum Ende des Jahres 2006 ausgelaufen ist.

Meinem Betreuer, dem Leiter des Lehrstuhls für Regelungstechnik und Mechatronik, Prof. Dr.-Ing. habil. Ansgar Trächtler, danke ich für die Mitwirkung bei der Arbeit. Gegen Ende des Jahres 2005 war er es, der die entscheidenden Impulse gesetzt hat, die letztlich zu der vorliegenden Arbeit geführt haben. Er hat mich durch sein Interesse, durch wertvolle Hinweise und Anregungen, Diskussionen und schließlich die Begutachtung meiner Arbeit fortdauernd unterstützt.

Ebenfalls bedanken möchte ich mich bei Prof. Dr. math. Friedhelm Meyer auf der Heide, dem Sprecher des SFB 376, für die Übernahme des Korreferats und die Durchsicht der Arbeit.

Des verstorbenen Herrn Prof. Dr.-Ing. Joachim Lückel möchte ich an dieser Stelle in besonderer Dankbarkeit gedenken. Bereits in den frühen 1990er Jahren hat Herr Prof. Dr.-Ing. Joachim Lückel die Forschung im Bereich des *Kreuzungsmanagements* etabliert und stetig vorangetrieben. In seiner Eigenschaft als früherer Leiter des Lehrstuhls und Vorgänger von Prof. Dr.-Ing. habil. Ansgar Trächtler hat er mir die Möglichkeit der Promotion eröffnet.

Bei meinen Kollegen am Lehrstuhl bedanke ich mich für die kooperative und sehr angenehme Arbeitsatmosphäre sowie für die vielen inspirierenden Gespräche. In diesen Dank möchte ich auch alle Studenten mit einbeziehen, die mich bei meiner Arbeit unterstützt haben. Besonders hervorheben möchte ich hier Philip Sommer, der mich als studentische Hilfskraft und im Rahmen seiner Bachelorarbeit unterstützt hat und dabei stets sehr gute Arbeit leistete.

Ein ganz großes Dankeschön geht auch an Frau Annette Bökamp-Gros für die sorgfältige Durchsicht des Manuskripts.

Ganz herzlich bedanken möchte ich mich auch bei meinen Eltern, Christa und Josef Bruns, die mir Vieles in meinem Leben ermöglicht und mich stets unterstützt haben.

Lügde, im Mai 2011 *Torsten Bruns*

Inhaltsverzeichnis

Zusammenfassung

In der vorliegenden Arbeit wird ein Verfahren vorgestellt, mit dem für niedrigdimensionale dynamische Systeme optimale und kollisionsfreie Trajektorien berechnet werden können. Im Rahmen der Optimierung können weiterhin unterschiedliche Zielgrößen berücksichtigt werden, die ihren Ausdruck allerdings in Form einer Wunsch- bzw. Soll-Trajektorie finden müssen. Die Modellierung erfolgt auf Basis einer systematischen Diskretisierung der prinzipiell kontinuierlichen Systeme und ihrer Umwelt, so dass Verfahren aus dem Bereich der kombinatorischen Optimierung verwendet werden können. Dieser Ansatz ermöglicht eine Abschätzung und Skalierung des Berechnungsaufwands, was wiederum eine gute Eignung für die Anwendungen unter Echtzeitbedingungen bedeutet.

Das Verfahren wird am Beispiel des autonomen Kreuzungsmanagements eingeführt und bewertet: Für autonome Fahrzeuge, die als lineare dynamische Systeme 2. Ordnung modelliert werden, werden kollisionsfreie Trajektorien für die Überquerung einer Kreuzung bzw. eines beliebigen Verkehrsknotenpunktes berechnet. Bei der Berechnung werden Optimierungszielgrößen wie Dauer, Komfort und Kraftstoffverbrauch berücksichtigt.

Prinzipiell kann das Verfahren auf beliebige gleichartige Anwendungen übertragen werden, etwa auf die Trajektorienplanung von Robotern mit gemeinsamem Arbeitsraum oder in einer unbekannten Umwelt.

Summary

The thesis presents a procedure of computing optimal and collision-free trajectories for low-dimensioned dynamical systems. In the context of the optimization, different objectives can additionally be taken into account; these, however, have to be described in the shape of desired resp. reference trajectories. The modelling is effected on the basis of a systematic discretization of the systems, which are basically continuous ones, and their environment, thus enabling the use of methods from the domain of combinatorial optimization. This approach allows estimation and scaling of the computational effort and is thus well suited for applications under real-time conditions.

The procedure is introduced and evaluated using the example of the autonomous intersection management: For autonomous vehicles, modelled as linear dynamical systems of 2nd order, collision-free trajectories are computed in view of their crossing an intersection resp. any desired traffic junction. The computation takes into account target values for the optimization, such as time, comfort, and fuel consumption.

In principle, the procedure can be transferred to any desired application of a similar type, e.g., the trajectory planning of robots sharing the same workspace or within an unknown environment.

1 Einführung

1.1 Motivation und Gliederung

In den frühen 1990er Jahren zeichnete sich die zunehmende Durchdringung elektronischer Komponenten im Bereich der Fahrzeugtechnik immer deutlicher ab. Diese Entwicklung ermöglichte vielfältige neue Funktionen und legte den Grundstein für *Autonomes Fahren*. Haben Fahrzeuge die Fähigkeit, autonom – also unabhängig vom Fahrer – zu handeln, so liegt der Gedanke an ein Szenario nicht fern, in dem Fahrzeuge eigenständig Kreuzungen durchfahren und Lichtsignalanlagen obsolet sind. Insbesondere vor dem Hintergrund des ständig zunehmenden Verkehrsaufkommens und der prinzipiellen technischen Machbarkeit gewann eine mögliche Umsetzung dieses Szenarios in die Realität im Laufe der Zeit immer mehr an Attraktivität. Die Berechnung kollisionsfreier Trajektorien für sämtliche Fahrzeuge im Kreuzungsbereich muss in einem solchen Szenario allerdings in Echtzeit erfolgen, also innerhalb weniger Millisekunden. Dies war eine Forderung, die seinerzeit problematisch erschien, da umfangreiche und komplexe Berechnungen zu erwarten waren, die von einem einzelnen Rechner aller Voraussicht nach nicht in Echtzeit durchgeführt werden konnten. In der Folge entstand die Idee, die Berechnungen von einem *Parallelrechner* durchführen zu lassen. Zu diesem Zweck sollten die Steuergeräte sämtlicher Fahrzeuge im Kreuzungsbereich zu einem massiv parallelen Rechnersystem vernetzt werden, um eine Lösung für die Kreuzungsüberquerung in adäquater Zeit massiv parallel berechnen zu können.

Im Jahre 1995 trat der Sonderforschungsbereich *Massive Parallelität: Algorithmen, Entwurfsmethoden und Anwendungen* (SFB 376) der Universität Paderborn mit dem Ziel an, Methoden und Techniken für die optimale Ausnutzung des Leistungspotenzials massiv paralleler Rechnersysteme zu entwickeln. Eine bedeutende Rolle in diesem Rahmen spielte die Problematik, sehr komplexe Berechnungen, die von einem einzelnen Rechner nicht in ausreichender Zeit abgearbeitet werden können, zu parallelisieren und auf mehrere

Rechner zu verteilen, um sie auf diesem Wege dennoch in adäquater Zeit ausführen zu können. Diese Thematik adressierte exakt die Anforderungen der Umsetzung des zuvor beschriebenen Szenarios der autonomen Kreuzungsdurchfahrt. Dies wiederum führte in der Folge dazu, dass das *Dezentrale autonome Kreuzungsmanagement für Kraftfahrzeuge* als Teilprojekt C1[1] im Bereich der Anwendungsbeispiele in den SFB 376 aufgenommen wurde.

Die vorliegende Arbeit beschreibt den Entwicklungsstand des Kreuzungsmanagements bis zum Ende des Projektes im Dezember 2008. Zunächst wird ein Überblick über die Entwicklung seit der Etablierung des SFB 376 gegeben. In der Folge wird ein neu entwickelter Lösungsansatz für die Problemstellung des Kreuzungsmanagements im Detail vorgestellt. Die ersten Ideen dieses völlig neuartigen Ansatzes entstanden gegen Ende 2005 und wurden Anfang 2006 veröffentlicht ([3]). Die auf diesen ersten Ideen basierenden Konzepte haben jedoch seitdem umfangreiche Modifikationen erfahren, so dass sie mit dem hier vorgestellten neuen Verfahren, das bisher in [4] und [5] veröffentlicht wurde, nicht mehr viel gemein haben.

Lösungsvorschläge anderer Forschungseinrichtungen, die im Bereich des Kreuzungsmanagements aktiv sind, bestehen im Wesentlichen darin, die autonome Kreuzungsüberquerung als *Multi-Agenten-System* zu modellieren. Von diesem Ansatz wird hier ausdrücklich Abstand genommen; vielmehr soll in dieser Arbeit gezeigt werden, dass die Anwendung bekannter Verfahren aus dem Bereich der kombinatorischen Optimierung den agentenbasierten Ansätzen an Effizienz und Leistungsfähigkeit überlegen ist.

Die zeitliche Entwicklung im Kreuzungsmanagement ist in Abbildung 1.1 dargestellt. Es wird unterschieden nach Verfahren, die im Rahmen des SFB 376 an der Universität Paderborn entstanden sind, und nach den bekanntesten Verfahren, die von anderen Forschungseinrichtungen entwickelt wurden. Die früheren Verfahren aus dem Umfeld des SFB 376 werden in Abschnitt 1.2 kurz erläutert. Im Anschluss daran erfolgen in Abschnitt 1.3 eine Betrachtung der bekanntesten Aktivitäten anderer Forschungseinrichtungen sowie eine kurze Bewertung. In Abschnitt 1.4 wird die Idee des neuesten und in dieser Arbeit im Detail vorgestellten Ansatzes kurz skizziert. Der Ansatz basiert auf Verfahren aus dem Bereich der kombinatorischen Optimierung, die in Kapitel 2 eingeführt werden. In den Kapiteln 3, 4 und 5 wird dann der neue Ansatz

[1] Ursprünglich war das Projekt allgemeiner gehalten und hatte daher den Titel *"Vernetzung von autonomen mechatronischen Systemen zu selbstorganisierenden Gesamtsystemen"*.

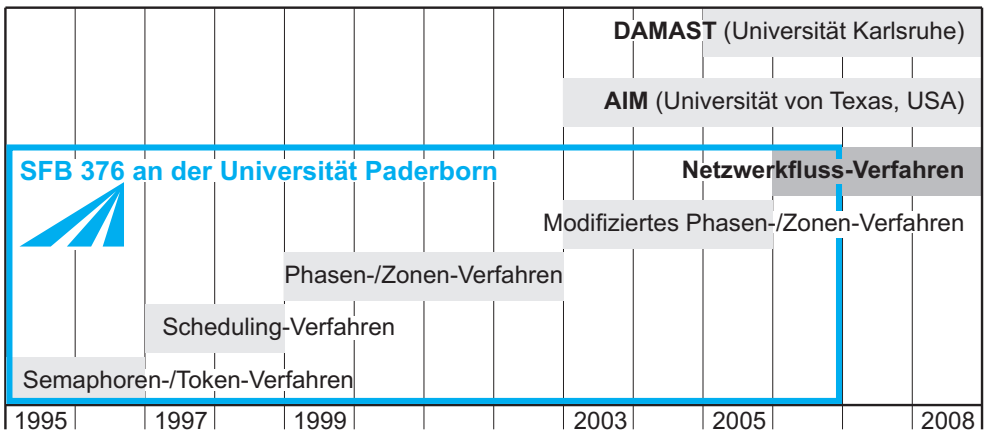

Abbildung 1.1: Zeitlinie beim Kreuzungsmanagement

detailliert beschrieben. Die Arbeit endet mit Kapitel 6, in dem die Ergebnisse zusammengefasst werden sowie ein Ausblick auf mögliche zukünftige Aktivitäten und weitere Einsatzbereiche der vorgestellten Methodik gegeben wird.

1.2 Entwicklung des Kreuzungsmanagements im SFB 376

Der im Rahmen dieser Arbeit vorgestellte neue Lösungsansatz unterscheidet sich grundlegend von seinen Vorgängern. Während die älteren Verfahren des Kreuzungsmanagements entweder keinen Optimierungsansatz bieten oder aber auf Optimierungsverfahren der nichtlinearen, beschränkten Optimierung zurückgreifen, verwendet der hier vorgeschlagene neue Ansatz erstmalig Optimierungsverfahren aus dem Bereich der diskreten Mathematik. Diese Optimierungsverfahren bieten gerade im Hinblick auf Echtzeitanforderungen enorme Vorteile gegenüber Verfahren der nichtlinearen Optimierung, von denen im Rahmen des Kreuzungsmanagements im Wesentlichen sogenannte Abstiegsverfahren wie das Gradienten- oder Quasi-Newton-Verfahren zum Einsatz gekommen sind. Bei Verfahren auf Basis von diskreter Mathematik kann in der Regel eine Abschätzung über die Anzahl der erforderlichen Berechnungen getroffen werden und damit auch über die maximale

Zeit, in der das gesuchte Optimum spätestens gefunden wird. Bei der Gruppe der Abstiegsverfahren hingegen sieht es so aus, dass in der überwiegenden Zahl der Fälle eine Konvergenz des Verfahrens zum gesuchten globalen Optimum nicht garantiert werden kann. Es kann weder ausgeschlossen werden, dass das Verfahren in einem lokalen Optimum abbricht, noch, dass es nicht konvergiert, insbesondere bei gleichzeitiger Optimierung mehrerer Zielgrößen. Folglich kann a priori auch keine Aussage über die erforderliche Dauer der Optimierung getroffen werden, was bei Echtzeitanwendungen ein großes Problem darstellen kann. Um die Vorteile des neuen gegenüber den älteren Verfahren zu verdeutlichen, soll im Folgenden ein kurzer Überblick über die älteren Verfahren gegeben werden. Dazu zählen eine kurze Beschreibung der generellen Lösungsidee und des Optimierungsansatzes, sofern vorhanden, und eine kurze Nennung der wesentlichen Vor- und Nachteile.

1.2.1 Semaphoren-Token-Verfahren

Die erste Lösungsvariante für das Problem des Kreuzungsmanagements wurde maßgeblich von Naumann und Rasche entwickelt [27, 21, 31]. Es wurde eine einfache Kreuzungsgeometrie zugrunde gelegt, bei der sich zwei Straßenverläufe orthogonal schneiden. Weiterhin wurde die vereinfachende Annahme getroffen, dass alle Fahrzeuge über identische geometrische Abmessungen verfügen. Die Fahrzeuge folgen während der Kreuzungsüberquerung festen Fahrspuren bzw. *Routen*, die durch Linien symbolisiert werden. Die Schnittpunkte dieser Linien sind potenzielle Kollisionspunkte bzw. Mittelpunkte von kritischen Zonen, innerhalb derer sich zu jedem Zeitpunkt jeweils nur ein Fahrzeug befinden darf. Die Größe der kritischen Zonen und die Fahrzeugabmessungen sind so aufeinander abgestimmt, dass die Fahrzeugabmessungen innerhalb der Abmessung einer kritischen Zone bleiben. Abbildung 1.2 zeigt die Kreuzung mit den möglichen Fahrspuren und den zugehörigen kritischen Zonen für zwei Fahrzeuge, welche die Kreuzung von West nach Ost bzw. von Süd nach Nord überqueren wollen. Wie zu sehen ist, kann es in den Zonen 3, 4 und 7 zu einer Kollision der beiden betrachteten Fahrzeuge kommen.

Jedes Fahrzeug kennt aufgrund seiner geplanten Route die kritischen Zonen, die es durchfahren muss. Ein Fahrzeug darf die Kreuzung erst dann passieren, wenn es über Aufenthaltsberechtigungen für sämtliche zu passierenden kritischen Zonen verfügt. Probleme können beim gleichzeitigen Zugriff mehrerer Fahrzeuge auf identische kritische Zonen bzw. auf die zugehörigen

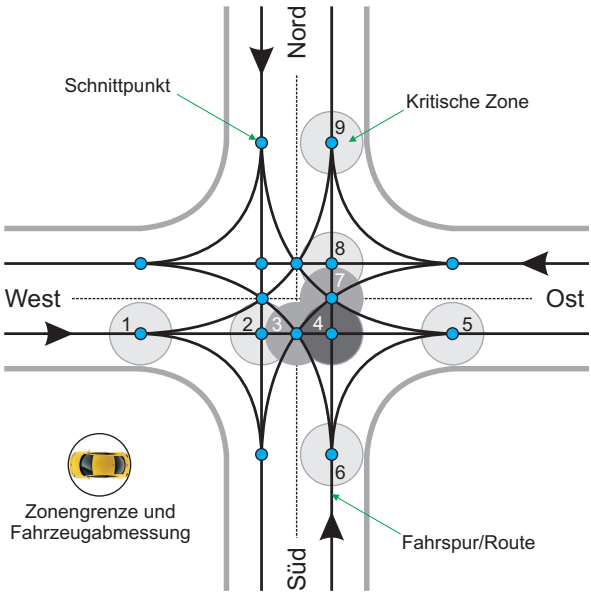

Abbildung 1.2: Prinzipdarstellung des Semaphoren-Token-Verfahrens

Aufenthaltsberechtigungen auftreten. Ein analoges Problem, das *Dining Philosophers Problem*, wurde in den 1960er Jahren von dem niederländischen Mathematiker DIJKSTRA formuliert. Bei diesem Problem sitzen Philosophen an einem runden Tisch. Die Philosophen sind ausschließlich damit beschäftigt, sich zu unterhalten oder zu essen. Zwischen je zwei Philosophen liegt eine Gabel. Damit ein Philosoph essen kann, benötigt er zwei Gabeln, sowohl die links als auch die rechts neben ihm. Ein Problem tritt genau dann auf, wenn alle Philosophen gleichzeitig zur rechten Gabel greifen. In diesem Fall kann niemand mehr die linke Gabel aufnehmen. Es kommt zu einer *Verklemmung* bzw. zu einem sogenannten *Deadlock*. Ein weiteres Beispiel ist eine Straßenkreuzung, an der die Regel *"Rechts vor Links"* gilt. Kommen aus allen vier Richtungen gleichzeitig Fahrzeuge, die darauf warten, dass das jeweils rechte Fahrzeug zuerst fährt, so kommt es ebenfalls zu einem *Deadlock*.

In [37] wird für das *Dining Philosophers Problem* als Lösung ein mit doppelten Semaphoren arbeitender Algorithmus angegeben. Der Zugriff auf die Gabeln wird dabei durch Semaphore geregelt. Der Algorithmus gestattet zu jedem Zeitpunkt jeweils nur einem Philosophen, entweder beide Gabeln aufzunehmen oder zurückzulegen. Über ein zweites Semaphorenfeld mit jeweils einem

Semaphor je Philosoph wird ausgeschlossen, dass zwei benachbarte Philosophen gleichzeitig den Wunsch äußern zu essen. Die prinzipielle Vorgehensweise dieses Algorithmus wurde auf das Kreuzungsmanagement übertragen [31]. Die Fahrzeuge entsprechen dabei den Philosophen, die kritischen Zonen im Kreuzungsbereich den Gabeln. Die Zugangsberechtigungen für die kritischen Zonen werden mittels eines Semaphorenfeldes verwaltet. Insgesamt 16 Semaphore, jeweils ein Semaphor je kritische Zone (vgl. Abbildung 1.2), bilden dieses (erste) Semaphorenfeld. Die Funktion des zweiten Semaphorenfeldes wird durch die Anwendung des Token-Ring-Prinzips erfüllt: Sämtliche Fahrzeuge, die sich im Kreuzungsbereich aufhalten, werden in einem Kommunikationsnetzwerk zusammengeschlossen, dessen Basis das Token-Ring-Prinzip ist. In einem Token-Ring kreist ein spezielles Bitmuster, das sogenannte Token, zwischen den angeschlossenen Netzwerkrechnern bzw. Netzwerkknoten umher. Das Token wird stets von einem Knoten an den nächsten weitergereicht. Selbst im Leerlauf geben die Stationen das Paket fortwährend weiter. Möchte ein Knoten Daten übertragen, muss er zunächst das Token erlangen und für die Dauer des Übertragungsvorgangs festhalten. Mit dem Token-Ring-Prinzip wird sichergestellt, dass jederzeit nur ein Knoten Daten versendet. Im Falle des Kreuzungsmanagements transferiert das Sende-Token den kompletten Semaphoren-Satz durch das Kommunikationsnetzwerk. Ist ein Fahrzeug im Besitz des Tokens, so reserviert es die benötigten Semaphore, sofern diese frei sind, überquert die Kreuzung und gibt die Semaphore im Anschluss wieder frei. Der Nachweis der Kollisionsfreiheit bei diesem Prinzip wurde mit der Formulierung des Problems mit Hilfe einer speziellen Form von Petri-Netzen, den Prädikate-Transitionen-Netzen, erbracht [31, 30].

Der oben beschriebene Lösungsansatz legt noch nicht fest, in welcher Reihenfolge die Fahrzeuge auf die benötigten Semaphore zugreifen dürfen. Bei der Bildung der Reihenfolge sollten die folgenden Anwendungsfälle berücksichtigt werden:

- Ein Fahrzeug soll nicht übermäßig lange an der Kreuzung warten.
- Eine längere Kolonne soll nicht notwendigerweise abbremsen, um einem Einzelfahrzeug Vorfahrt zu gewähren.
- Rettungsfahrzeuge sollen möglichst immer Vorfahrt haben.

Der gewählte Ansatz besteht darin, die Semaphoren-Zuteilung prioritätsgesteuert, unter Berücksichtigung der oben genannten Anwendungsfälle, vorzu-

nehmen. Die folgende Gleichung enthält eine heuristische Prioritätsberechnung für das Fahrzeug k:

$$P_k = k_1 \cdot s + k_2 \cdot v + k_3 \cdot t_W + k_4 \cdot \sum_{i=1}^{n} P_i. \tag{1.1}$$

Die Faktoren k_1 bis k_4 sind Gewichtungsfaktoren für die Erzwingung eines bestimmten Verhaltens der Gesamtheit aller Fahrzeuge. Mit s wird die Weglänge bezeichnet, die Fahrzeug k bereits im Kreuzungsbereich zurückgelegt hat, v entspricht der Geschwindigkeit und t_W der Wartezeit des Fahrzeugs k im Kreuzungsbereich. P_i ist die Priorität eines Fahrzeugs $i \neq k$, das sich in einer Schlange, bestehend aus n Fahrzeugen, hinter Fahrzeug k befindet. Rettungsfahrzeuge werden in diesem Ansatz berücksichtigt, indem man ihnen erlaubt, sich selbst eine hohe Priorität zuzuteilen.

Das Semaphoren-Token-Verfahren schließt Kollisionen zwischen Fahrzeugen aus. Es ermöglicht allerdings keine Vorausschau bzw. Vorausberechnung einer durchsatzoptimalen Vorfahrtsreihenfolge. Darüber hinaus sind auch keine direkten Angriffspunkte für eine Optimierung von Zielgrößen wie Fahrkomfort oder Kraftstoffverbrauch gegeben. Diese Unzulänglichkeiten haben zu der Entwicklung des Scheduling-Verfahrens geführt, das im folgenden Abschnitt vorgestellt wird.

1.2.2 Scheduling-Verfahren

Dieses Verfahren wurde ebenfalls von Rasche und Naumann vorgeschlagen [29, 28]. Unter *Scheduling* bzw. Zeitablaufsteuerung versteht man die Zuordnung von gegebenen, mengen- und terminmäßig spezifizierten Aufträgen zu Ressourcen und die Bestimmung der zeitlichen Reihenfolge, in der die zu einem bestimmten Zeitpunkt an einer Ressource wartenden Aufträge bearbeitet werden sollen. Als Ergebnis wird ein Plan (*Schedule*) aufgestellt. In [37] werden Anforderungen an ein Scheduling-Verfahren für Computer-Betriebssysteme vorgestellt. Diese Anforderungen lassen sich von ihrer Grundproblematik her auch auf das Kreuzungsmanagement übertragen. Die Analogien sind in der folgenden Tabelle 1.1 gemäß [27] zusammengefasst: Die in [37] vorgestellten Lösungsansätze für das Scheduling von Prozessen können nicht in ihrer Gesamtheit auf das Kreuzungsmanagement übertragen werden; sie dienen aber als Basis für die im Folgenden dargestellte Lösung gemäß [29]

Anforderung	Betriebssystem-Scheduler	Kreuzungsmanagement
	Prozesse Prozessoren	Fahrzeuge kritische Zonen
Fairness	Jeder Prozess erhält einen gerechten Anteil der Prozessorzeit	Jedes Fahrzeug erhält die zum Passieren der kritischen Zone erforderliche Zeit
Effizienz	Der Prozessor ist immer vollständig ausgelastet (wenn genügend Prozesse vorhanden sind)	Die kritischen Zonen werden immer befahren (wenn genügend Fahrzeuge vorhanden sind)
Verweilzeit	Die Wartezeit auf die Ausgabe von Stapelaufträgen wird minimiert	Die Durchfahrtszeit eines Einzelfahrzeugs wird minimiert
Durchsatz	Maximierung der ausgeführten Aufträge je Zeitintervall	Maximierung der Fahrzeuge, die je Zeitintervall die Kreuzung passieren

Tabelle 1.1: Analogien zwischen Prozessor-Scheduling und Kreuzungsmanagement

für das Kreuzungsmanagement. Die Kreuzung wird mit jeweils einer Fahrbahn je Richtung modelliert. Der Mittelpunkt der Kreuzung besteht aus vier kritischen Zonen. In jeder Zone darf sich immer nur ein Fahrzeug aufhalten. Tritt ein Fahrzeug in den Einflussbereich des Kreuzungsmanagements ein, so wird ein Geschwindigkeitsprofil bzw. eine Trajektorie für die Kreuzungsdurchfahrt berechnet. Bei der Berechnung können Zielgrößen wie Fahrkomfort, Kraftstoffverbrauch oder Durchfahrtszeit berücksichtigt werden. Dem aus der Berechnung resultierenden Geschwindigkeitsprofil muss das Fahrzeug folgen. Anhand des Profils lassen sich im Vorfeld die Zeitfenster bestimmen, innerhalb derer sich ein Fahrzeug innerhalb einer kritischen Zone befindet. Diese Zeitfenster werden in einen Fahrplan (Schedule) eingearbeitet, der so beschaffen sein muss, dass sich die Zeitfenster nicht überschneiden. Abbildung 1.3 zeigt diesen Sachverhalt. Fahrspuren von Fahrzeugen, die gleiche kritische Zonen durchlaufen, müssen sich nicht zwangsläufig schneiden. Beispielsweise durchfährt ein von West nach Nord fahrendes Fahrzeug (WN) die Zonen 3-2-1. Ein von Ost nach Süd fahrendes Fahrzeug (OS) durchfährt die Zonen 1-4-3. Beide Fahrzeuge durchfahren somit die Zonen 1 und 3. Trotzdem kann es zu keiner Kollision kommen, da sich die Fahrspuren der beiden Fahrzeuge nicht schneiden. Beziehungen dieser Art werden in einer Kollisionsmatrix verwaltet und bei der Erstellung des Schedules entsprechend berücksichtigt.

In dem dargestellten Schedule (vgl. Abbildung 1.3) überschneiden sich bei-

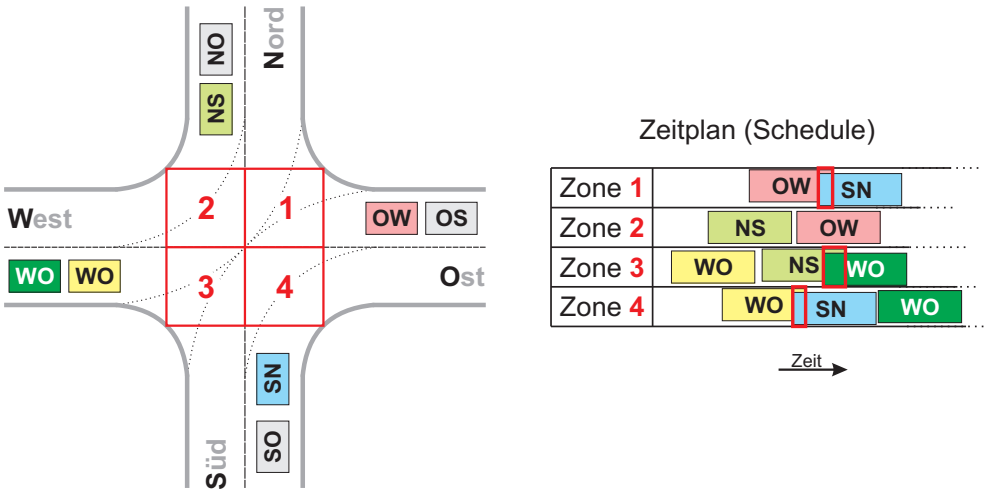

Abbildung 1.3: Prinzipdarstellung des Scheduling-Verfahrens

spielsweise bei Zone 1 noch die Zeitfenster von Fahrzeug OW (Ost nach West) und Fahrzeug SN. Dies ist nicht zulässig, da es dann zu einer Kollision kommen kann. Mit einem speziell auf diese Problematik zugeschnittenen Gradientenverfahren, werden die Überdeckungen auf einen zulässigen Bereich minimiert [29]. Dies erfolgt durch eine Variation des Geschwindigkeits- bzw. Beschleunigungsprofils. Der Schedule kann auch als Warteschlange für die Benutzung der Ressourcen *Kritische Zonen* interpretiert werden. Fahrzeuge, deren Zeitfenster für das Befahren kritischer Zonen derart optimiert wurden, dass Kollisionen mit in der Warteschlange vor ihnen befindlichen Fahrzeugen ausgeschlossen sind, werden in der Warteschlange fest einsortiert. Ihre Geschwindigkeitsprofile werden eingefroren und können nicht mehr beeinflusst werden.

Da sich in einer Zone immer nur ein Fahrzeug aufhalten darf, beträgt der Mindestabstand zwischen zwei Fahrzeugen auch genau eine Zonenlänge [29, 28]. Dieser Mindestabstand beschränkt den Durchsatz der Kreuzung. Man kann dem entgegenwirken, indem man die Zonen verkleinert bzw. den Kreuzungsbereich in mehrere, kleinere Zonen aufteilt. Mit zunehmender Anzahl von Zonen steigt jedoch der Optimierungsaufwand für die Geschwindigkeitsprofile exponentiell an. Ein weiteres Manko besteht darin, dass die unterschiedliche räumliche Ausdehnung der Fahrzeuge (PKW, LKW) nicht berücksichtigt wird. Aufgrund dieser Nachteile wurde das realitätsnähere Phasen-/Zonen-

Verfahren entwickelt.

1.2.3 Phasen-/Zonen-Verfahren

Innerhalb des von Deppe und Neuendorf vorgeschlagenen Phasen-/Zonen-Verfahrens [24] wird die Kreuzungsdurchfahrt in mehrere Phasen aufgeteilt. Diese Phasen dauern je Fahrzeug so lange an, wie sich das Fahrzeug in der zugehörigen Zone der Kreuzung befindet. Einen Überblick über die Zonen der Kreuzung gibt Bild 1.4. Fährt ein Fahrzeug in den Einflussbereich des

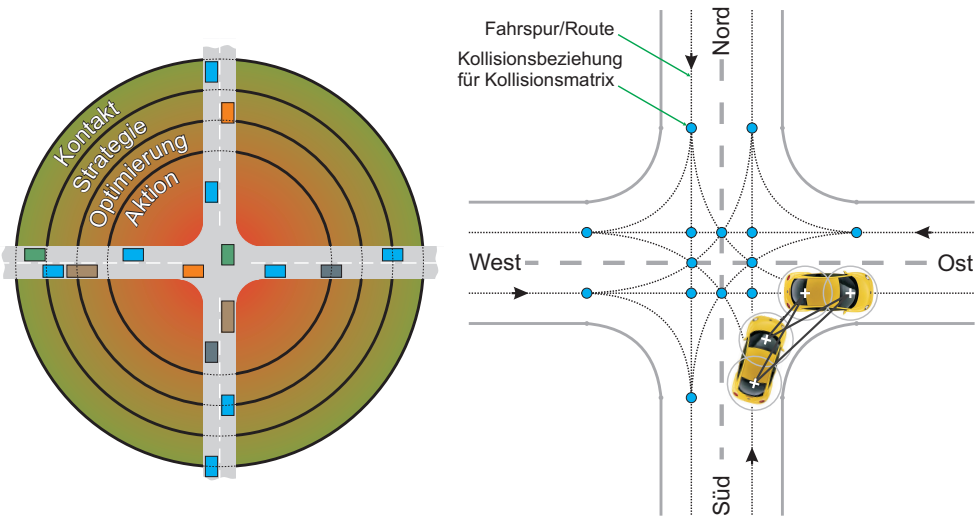

Abbildung 1.4: Prinzipdarstellung des Phasen-/Zonen-Verfahrens

Kreuzungsmanagements ein, so befindet es sich in der Kontaktphase[2]. In dieser Phase wird dem Fahrer die Kontrolle über das Fahrzeug entzogen. Die Fahrdynamikregler für eine autonome Lateral- und Longitudinalbewegung übernehmen die Kontrolle. Das Fahrzeug wird auf eine konstante Geschwindigkeit gebracht, und der Fahrer hat die Möglichkeit, seinen Richtungswunsch anzugeben. Das Fahrzeug wird in ein Ad-hoc-Kommunikationsnetzwerk eingebunden und tauscht Informationen mit den anderen Verkehrsteilnehmern aus. An die Kontaktphase schließt sich die Strategiephase an. In dieser Phase

[2]Diese Phase ist prinzipiell bei allen vorgestellten Verfahren vorhanden, da die in ihr stattfindenden Vorgänge Voraussetzung sind für eine autonome Kreuzungsüberquerung.

wird schnell ein suboptimales Geschwindigkeitsprofil berechnet. Dabei werden potenzielle Kollisionspartner berücksichtigt. Mit diesem Geschwindigkeitsprofil kann das Fahrzeug somit die Kreuzung bereits sicher überqueren. Das Profil dient weiterhin als Startprofil für die anschließende Optimierung in der Optimierungsphase. In dieser Phase beginnt ein auf den Fahrzeugen verteilt rechnender Optimierer für kontinuierliche Mehrgrößen-Parameter-Optimierung [7] damit, bessere Geschwindigkeitsprofile zu suchen, welche die Abstände zwischen den Fahrzeugen verringern und dadurch den Durchsatz an der Kreuzung erhöhen. In dieser Phase können prinzipiell auch Zielgrößen wie Komfort oder Kraftstoffverbrauch berücksichtigt werden, allerdings nur auf Kreuzungsebene und nicht fahrzeugindividuell. Während der Aktions- oder Handlungsphase werden von den Fahrzeugen die Verzögerungs-, Durchfahrts- und Beschleunigungszone durchfahren. In der Verzögerungszone werden die Geschwindigkeitsprofile aus Strategie- und Optimierungsphase umgesetzt. In der folgenden Durchfahrtszone wird der zentrale Kreuzungsbereich von allen Fahrzeugen mit konstanter Geschwindigkeit durchfahren. Die Höhe dieser Geschwindigkeit richtet sich nach der Fahrspur mit dem geringsten Kurvenradius bzw. nach der sich daraus ergebenden höchsten Querbeschleunigung. Konstante Geschwindigkeit aller Fahrzeuge in der Durchfahrtszone gewährleistet Kollisionsfreiheit in der sich anschließenden Beschleunigungszone. Aus diesem Grund muss die Beschleunigungszone innerhalb der Strategie- und der Optimierungsphase nicht betrachtet werden [24]. In der Beschleunigungsphase werden die Fahrzeuge letztlich wieder auf ein höheres Geschwindigkeitsniveau gebracht.

Die Bestimmung der Vorfahrts- bzw. Durchfahrtsreihenfolge erfolgt mit Hilfsmitteln aus der Graphentheorie [25]. Mit dem Beginn der Kontaktphase wird das Fahrzeug in den sogenannten Kollisionsgraphen eingefügt (Bild 1.5). In diesem Graphen werden Fahrzeuge als Knoten modelliert. Die Kollisionsbeziehungen ergeben sich wiederum aus den Schnittbeziehungen der einzelnen Fahrspuren und werden, ähnlich wie beim Scheduling-Verfahren, in einer Kollisionsmatrix gespeichert. Jedes in den Kreuzungsbereich einfahrende Fahrzeug wird mit allen potenziellen Kollisionspartnern gemäß der Kollisionsmatrix über gerichtete Kanten (Pfeile) verbunden. Es entsteht dadurch ein azyklischer, gerichteter Graph mit einer Ausrichtung von der Kreuzungsmitte nach außen. Während der Strategiephase wird mit Hilfe schneller Simulationen der Vorgänge in der Aktionsphase ermittelt, wie stark jedes Fahrzeug für seine jeweiligen Vorgänger abbremsen muss. Bei der mit der Berechnung verbundenen Kollisionsprüfung werden die geometrischen Abmessungen des

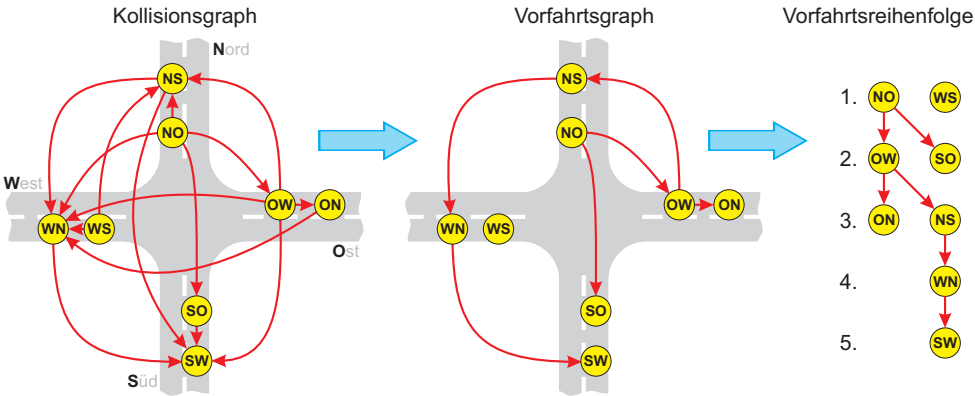

Abbildung 1.5: Ermittlung der Vorfahrtsreihenfolge

Fahrzeugs berücksichtigt: Den Fahrzeugen werden zu diesem Zweck entsprechend ihrer Länge mehrere Kreisflächen mit festem Durchmesser überlagert (vgl. Bild 1.4). Während einer Simulation wird ständig anhand der Positionen der Fahrzeuge, der zum jeweiligen Fahrzeugmittelpunkt relativen Positionen der Kreisflächenmittelpunkte und dem fest vorgegebenen Radius der Kreisflächen überwacht, ob sich Kreisflächen unterschiedlicher Fahrzeuge schneiden. Die als eines der Ergebnisse der Simulationen errechneten Verzögerungswerte werden an den Pfeilen des Kollisionsgraphen als Gewicht vermerkt. Anschließend wird der Kollisionsgraph in den Vorfahrtsgraphen transformiert. Dazu wählt jedes Fahrzeug den Vorgänger aus, für den es am meisten verzögern muss. Alle anderen Pfeile werden eliminiert. Die konkrete Vorfahrtsreihenfolge ergibt sich dann aus der sogenannten topologischen Sortierung des Vorfahrtsgraphen [25]. Die errechneten Verzögerungswerte werden weiterhin in der Strategiephase dazu genutzt, um initiale Geschwindigkeitsprofile für die einzelnen Fahrzeuge zu bestimmen. Die daraus resultierende Lösung garantiert bereits Kollisionsfreiheit, ist jedoch noch suboptimal. In der Optimierungsphase wird versucht, diese initiale Lösung zu verbessern.

Das Phasen-/Zonen-Verfahren erhöht gegenüber den zuvor vorgestellten Verfahren den Durchsatz. Der wesentliche Grund dafür besteht darin, dass der Abstand zwischen den Fahrzeugen gegenüber dem Scheduling-Verfahren verkleinert werden kann. Weiterhin können nun unterschiedliche geometrische Abmessungen von Fahrzeugen (PKW, LKW) berücksichtigt werden. Die größte Einschränkung des Verfahrens besteht darin, dass unterschiedliche Fahrprofile der Fahrzeuge nur sehr begrenzt realisiert werden können. Dies wieder-

um führt dazu, dass sowohl unterschiedlichen fahrdynamischen Eigenschaften als auch unterschiedlichen Fahrzeugprioriäten nicht ausreichend Rechnung getragen werden kann. Um diese Einschränkung aufzuheben, wurde die Idee entwickelt, das Verfahren bzgl. des Durchfahrens der Aktionszone dahingehend zu modifizieren, dass jedes Fahrzeug für sich ein sogenanntes Wunschfahrprofil generiert, mit dem es die Kreuzung passieren würde, sofern es auf keine anderen Verkehrsteilnehmer Rücksicht nehmen müsste. Rettungsfahrzeuge könnten innerhalb kürzester Zeit die Kreuzung passieren. Komfortorientierte Fahrer oder LKW, die aufgrund ihrer fahrdynamischen Eigenschaften die Kreuzung nur mit geringer Geschwindigkeit durchfahren können, würden mit entsprechend geringerem Geschwindigkeitsniveau die Kreuzung passieren. Diese Überlegungen führten zum modifizierten Phasen-/Zonen-Verfahren.

1.2.4 Modifiziertes Phasen-/Zonen-Verfahren

Im Rahmen dieses von Neuendorf vorgeschlagenen Verfahrens wird das komplexe Problem der autonomen Kreuzungsüberquerung aller Fahrzeuge in Anlehnung an die Strukturierungssystematik für mechatronische Systeme [19, 40] in mehrere, weniger komplexe Teilprobleme zerlegt. Die Strukturierungssystematik dient dazu, komplexe Systeme modular und hierarchisch zu gliedern, um auf diese Weise Komplexität beherrschbar zu machen. Im Falle des Kreuzungsmanagements sind dafür lediglich die zwei Strukturierungselemente *Autonomes Mechatronisches System* (AMS) und *Vernetztes Mechatronisches System* (VMS) erforderlich. Das Problem des Kreuzungsmanagements wird gemäß Bild 1.6 in drei Teilprobleme zerlegt. In der Strategiephase

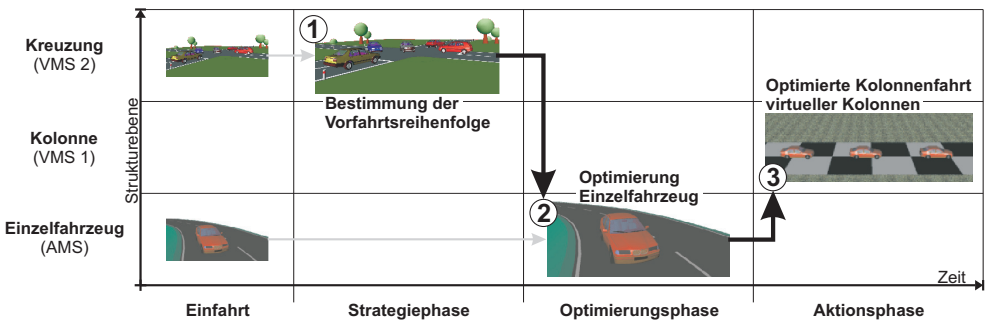

Abbildung 1.6: Strukturierung der Optimierungsaufgaben

wird mit dem oben vorgestellten, graphenbasierten Verfahren eine Vorfahrtsreihenfolge festgelegt (erstes Teilproblem, ① in Abbildung 1.6). In der Optimierungsphase erzeugt ein lokal auf jedem Fahrzeug arbeitender Mehrgrößen-Parameter-Optimierer ein Wunschfahrprofil [22] (zweites Teilproblem, ② in Abbildung 1.6). Mit diesem Profil würde das jeweilige Fahrzeug die Kreuzung passieren, sofern keine anderen Verkehrsteilnehmer im Sinne potenzieller Kollisionspartner berücksichtigt werden müssen. Zu diesem Zweck arbeitet der Optimierer mit einem Modell der Kreuzung und einem Modell des Fahrzeugs. Mit diesen Modellen werden Kreuzungsdurchfahrten des Fahrzeugs mit unterschiedlichen Geschwindigkeitsprofilen simuliert. Die Parameter der Optimierung sind Stützstellen des Geschwindigkeitsprofils für die jeweilige Route. Folgende Zielgrößen können während des Optimierungsprozesses berücksichtigt werden:

- **Kraftschlussbeanspruchung**: Diese Zielgröße setzt die während des Abbiegens auftretenden Zentrifugalkräfte in Relation zu den Reibkräften zwischen Fahrzeug und Fahrbahn. Eine entsprechende Berücksichtigung stellt sicher, dass das Fahrzeug während des Abbiegens nicht ausbricht.
- **Durchfahrtszeit**: Insbesondere bei Rettungsfahrzeugen ist eine Minimierung dieser Zielgröße relevant.
- **Energieverbrauch**: Ökonomische Fahrer können mit dieser Zielgröße ihren Kraftstoffverbrauch minimieren.
- **Quer- und Längsbeschleunigung**: Komfortorientierte Fahrer sind bestrebt, mit diesen Zielgrößen ein bestimmtes Niveau nicht zu überschreiten.

Prinzipiell kann mit der Generierung des Wunschfahrprofils schon begonnen werden, sobald die Informationen über die Kreuzungsgeometrie bekannt sind, es muss jedoch spätestens am Ende der Optimierungsphase vorliegen. Aus der Vorfahrtsreihenfolge werden sogenannte virtuelle Kolonnen gebildet, indem aufeinanderfolgende Fahrzeuge, die in der Regel nicht örtlich hintereinander her fahren, über einen Kolonnenregler miteinander verbunden werden [23]. Die Regelgröße folgt aus der aktuellen Entfernung aufeinanderfolgender Fahrzeuge zu dem gemeinsamen Kollisionspunkt ihrer Fahrspuren bzw. Routen. Der Kolonnenregler muss *Kolonnenstabilität* gewährleisten. Das heißt, er muss für alle Fahrzeuge innerhalb einer Kolonne Kollisionsfreiheit garantieren. Die Kollisionsfreiheit muss unabhängig sein von der Anzahl der

Fahrzeuge innerhalb der Kolonne und vom Fahrmanöver des Kolonnenführungsfahrzeugs. Die kollisionsfreie Fahrt virtueller Kolonnen bildet das dritte Teilproblem (③ in Abbildung 1.6). Wunschfahrprofile werden vom Kolonnenregler insofern berücksichtigt, als dem Wunschfahrprofil gefolgt wird, wenn auf keine vorfahrtsberechtigten Fahrzeuge gewartet werden muss.

Untersuchungen unterschiedlicher Kreuzungssituationen haben gezeigt, dass der bisherige Algorithmus für die Bestimmung der Vorfahrtsreihenfolge bei gleichzeitiger Realisierung unterschiedlicher Fahrprofile nur bedingt geeignet ist. Der Algorithmus arbeitet prinzipiell nach dem FCFS-Prinzip (*First-Come, First-Served*). Die Fahrzeuge werden in der Reihenfolge ihres Kreuzungseintrittes in den Kollisionsgraphen eingebunden. Diese Reihenfolge entspricht im Wesentlichen auch der späteren Vorfahrtsreihenfolge. Solange die Fahrzeuge über annähernd identische Fahrprofile verfügen, liefert der vorgestellte Algorithmus gute Ergebnisse. Bei stark unterschiedlichen Fahrprofilen ist dies aber nicht mehr der Fall. Fährt beispielsweise ein Bus vor einem Rettungsfahrzeug in die Kreuzung ein und fahren beide Fahrzeuge nicht unmittelbar hintereinander, so würde dem Bus vor dem Rettungsfahrzeug das Passieren der Kreuzung ermöglicht, auch wenn das Rettungsfahrzeug aufgrund seiner fahrdynamischen Eigenschaften und seines auf hohem Niveau befindlichen Geschwindigkeitsprofils noch vor dem Bus die Kreuzung passieren könnte. Aus dem Versuch heraus, die graphenbasierte Vorfahrtsbestimmung an diese Problematik anzupassen, ist ein völlig neuartiges Verfahren entstanden, das die Vorgänge im Kreuzungsbereich als Netzwerkflussproblem formuliert und den Schwerpunkt der vorliegenden Arbeit bildet.

1.3 Stand der Technik

Auch außerhalb des SFB 376 haben sich Forscher und Entwickler mit dem Problem der autonomen Kreuzungsüberquerung beschäftigt und verschiedene Lösungsvorschläge veröffentlicht. Gemein ist allen Ansätzen, dass sie die Situation im Kreuzungsbereich als sogenanntes Multi-Agenten-System zu modellieren versuchen. Zu den Pionieren in diesem Bereich gehörten Zou und Levinson [41] sowie Dresner und Stone [11]. Insbesondere die Arbeiten der beiden Letztgenannten haben in der Fachwelt große Beachtung gefunden und wurden von anderen Forschergruppen aufgegriffen und erweitert: Vasirani und Ossowski schlagen als Erweiterung ein Lernverfahren vor, das

Gruppen von Agenten ermöglicht, ihre Fahrprofile bzw. Geschwindigkeitsverläufe aufeinander abzustimmen bzw. zu koordinieren [39]. Schepperle und Böhm erweiterten die Arbeiten von Dresner und Stone dahingehend, dass die Zeitfenster für das Passieren der Kreuzung an die jeweiligen Fahrzeuge bzw. deren Fahrer(agenten) im Rahmen von Auktionen versteigert werden, um so den unterschiedlichen Wertschätzungen von Wartezeiten, die sich bei den Fahrern zum Teil stark unterscheiden, Rechnung zu tragen [33]. Im Folgenden sollen die zwei bedeutendsten Ansätze kurz vorgestellt werden. Zum einen handelt es sich dabei um das AIM-Projekt von Dresner und Stone. Darauf aufbauend wird dann der Ansatz von Schepperle und Böhm vorgestellt. Zum Abschluss folgt eine Bewertung der Zweckmäßigkeit von Ansätzen, die auf die Modellierung der Vorgänge im Kreuzungsbereich als Multi-Agenten-System abzielen.

1.3.1 Das AIM-Projekt der Universität von Texas

Etwa seit 2003 gibt es im Fachbereich für Computerwissenschaften (Laboratorium für Künstliche Intelligenz) der Universität von Texas in Austin ein Projekt mit der Bezeichnung *Autonomous Intersection Management* (AIM)[3] [12]. Der dort verfolgte Ansatz besteht in der Modellierung der Vorgänge im Kreuzungsbereich als Multi-Agenten-System. Die Begründung dafür besteht im Wesentlichen in der Annahme, dass Verkehr seinem Wesen nach stets einem Multi-Agenten-System entspricht. Die zur Zeit noch komplett menschlichen Verkehrsteilnehmer entsprechen den Agenten, wobei ihre Interaktionen durch Gesetze, Verkehrszeichen und -signale (Ampeln) restringiert werden. Aufgrund der begrenzten Reaktionsfähigkeit der Menschen besitzt das System eine Vielzahl von Zeitpuffern, zum Beispiel bei Lichtphasen von Ampelanlagen. Insgesamt wird das bestehende System mit der Verkehrsregelung durch Ampeln und Schilder als ineffizient angesehen. Eine deutliche Effizienzsteigerung erhofft man sich, wenn menschliche Aufgaben durch (intelligente) Software-Agenten im Rahmen eines Multi-Agenten-Systems übernommen werden, vorausgesetzt, die Fahrzeuge verfügen über autonome Fahrfunktionen. Als weiterer Grund für die Wahl eines Multi-Agenten-Ansatzes wird die gute Eignung für dezentrale Probleme angeführt. Verkehrsführung und -planung werden als zu komplex für einen einzelnen Computer angesehen. Weiterhin würde sich in diesem Zusammenhang die Frage stellen, wie

[3] Weitere Informationen sind unter **http://www.cs.utexas.edu/users/kdresner/aim/** verfügbar.

zu verfahren ist, wenn dieser zentrale Computer einmal ausfallen würde. Im Folgenden werden die wesentlichen Eigenschaften und Besonderheiten dieses Ansatzes kurz zusammengefasst:

Die Kernidee hinter der Vision einer autonomen Kreuzungsdurchfahrt besteht darin, dass Fahrzeuge im Vorfeld ein Zeitfenster exklusiv für sich reservieren und die Kreuzung dann exakt innerhalb dieses Zeitfensters passieren. Jede Kreuzung verfügt zu diesem Zweck über einen Kreuzungsagenten bzw. Kreuzungsmanager (*Intersection Manager*), während jedes Fahrzeug mit einem Fahreragenten (*Driver Agent*) ausgestattet ist. Fahreragenten kommunizieren über standardisierte Protokolle mit dem Kreuzungsmanager. Sobald ein Fahrzeug in den Kreuzungsbereich[4] einfährt, versucht dessen Fahreragent ein Zeitfenster beim Kreuzungsmanager für die Kreuzungsdurchfahrt zu reservieren[5]. Zu diesem Zweck teilt der Fahreragent dem Kreuzungsmanager charakteristische Daten seines Fahrzeugs mit, zum Beispiel die Fahrzeugabmessungen und die maximalen Beschleunigungs- und Verzögerungsvermögen. Mit diesen Daten führt der Kreuzungsmanager Simulationen einer Kreuzungsdurchfahrt für das jeweilige Fahrzeug durch und überprüft, ob es während dieser simulierten Durchfahrten zu Kollisionen mit anderen Fahrzeugen kommt. Falls es zu keinen Kollisionen kommt, so wird das Zeitfenster für das Fahrzeug reserviert. Ein Fahrzeug darf die Kreuzung erst passieren, wenn ihm ein Zeitfenster vom Kreuzungsmanager zugewiesen wurde. Der Kreuzungmanager führt zwei Simulationen durch. Während der ersten Simulation wird das Fahrzeug mit seinem maximalen Beschleunigungsvermögen auf die maximal zulässige Geschwindigkeit im Kreuzungsbereich beschleunigt. Sollten während dieser Simulation Kollisionskonflikte mit anderen Fahrzeugen auftreten, so wird eine zweite Simulation durchgeführt, während derer die Kreuzung mit konstanter Geschwindigkeit durchfahren wird. Diese Geschwindigkeit darf jedoch ein Minimum nicht unterschreiten, um andere Fahrzeuge nicht unnötig lange warten zu lassen. Kann auch in dieser Simulation keine Einplanung erfolgen, so muss das Fahrzeug eine gewisse Zeit warten (*Timeout*), bevor es eine erneute Reservierungsanfrage stellen darf. Diese Systematik wurde gewählt, um eine Überlastung des Kreuzungsmanagers zu vermeiden und jedem Fahrzeug eine Einplanung zu ermöglichen. Die Schwachstellen dieser Systematik sind offensichtlich: Sie scheint nicht besonders effizient und keinesfalls optimal zu sein.

[4] Beim AIM wird derzeit ein Areal von 250m × 250m zugrunde gelegt.

[5] Grundlage ist das FCFS-Prinzip (*First-Come, First-Served*), das seinem Wesen nach prinzipiell dem FIFO-Prinzip (*First-In, First-Out*) entspricht.

Die Fahrzeugbewegung wird anhand eines einfachen Modells für nichtholonome Bewegungen in der Ebene beschrieben, das lediglich kinematische Beziehungen berücksichtigt. Die Geschwindigkeit v und der Lenkwinkel δ werden vorgegeben und je Zeitintervall der Simulation konstant gehalten. Die Position in der Ebene (x, y) und die Orientierung ψ des Fahrzeugs ergeben sich dann gemäß der folgenden Differentialgleichungen

$$\begin{aligned} \dot{x} &= v \cdot \cos\psi, \\ \dot{y} &= v \cdot \sin\psi, \\ \dot{\psi} &= v \cdot \frac{\tan\delta}{L}. \end{aligned} \tag{1.2}$$

Bei L handelt es sich um den Radstand bzw. den Achsabstand des Fahrzeugs.

Die Kollisionsvermeidung wird realisiert, indem der Durchfahrtsbereich mit einem Raster überzogen und in Zonen (*Tiles*) eingeteilt wird, wie es in Abbildung 1.7 dargestellt ist. Jede Zone darf zu einem beliebigen Zeitpunkt nur von einem Fahrzeug besetzt werden. Bei der Einplanung von Fahrzeugen werden Zonen für bestimmte Zeitpunkte bzw. Zeitfenster reserviert. Um Ungenauigkeiten aufgrund der Zeitdiskretisierung und daraus resultierende Gefahren abzufangen, arbeitet der Algorithmus mit sogenannten Sicherheitspuffern (*Safety Buffers*). Es werden statische Puffer (*Static Buffers*), Zeitpuffer (*Time Buffers*) und eine Mischung aus beiden (*Hybrid Buffers*) verwendet. Statische Puffer bestehen aus einem fixen Abstand in Metern, bei Zeitpuffern hängt der Abstand von der aktuellen Geschwindigkeit ab. Bei der Reservierung von Zonen wird die um die jeweiligen Puffer erweiterte ursprüngliche Fahrzeuggeometrie verwendet (vgl. Abbildung 1.7). Diese Systematik der Kollisionsvermeidung findet lediglich im Innenbereich der Kreuzung statt. Wie die Kollisionsvermeidung im Ein- und Ausfahrtsbereich erfolgt, geht aus den Veröffentlichungen nur ansatzweise hervor. Ein Ansatz im Ausfahrtsbereich besteht darin, dass Fahrzeuge, die Randzonen (*Edge Tiles*) belegen, mit einem größeren Zeitpuffer versehen werden. Auf diese Weise wird das Potenzial für Kollisionen mit nachfolgenden Fahrzeugen, die auf die gleiche Spur einbiegen, reduziert.

Der hier skizzierte Lösungsansatz berücksichtigt von Anfang an Fußgänger, Fahrradfahrer und konventionelle Fahrzeuge ohne Fahreragenten. In dem Szenario werden Ampelanlagen simuliert, an deren Signale sich Verkehrsteilnehmer ohne autonome Software-Agenten halten müssen. Während Grünphasen

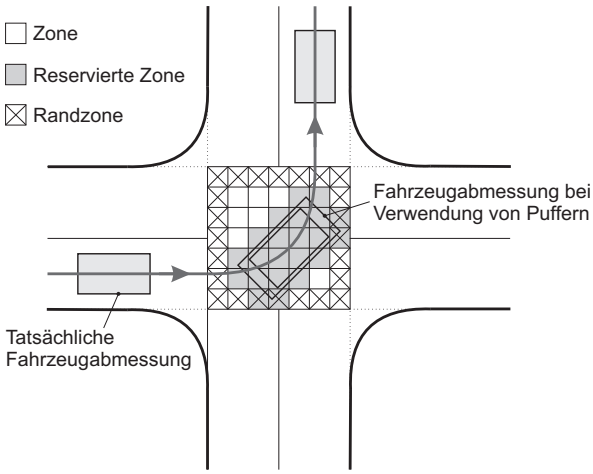

Abbildung 1.7: Kollisionsvermeidung beim AIM

für bestimmte Routen werden die zugehörigen Zonen gesperrt und können nicht exklusiv durch Fahreragenten reserviert werden. Fahrzeuge mit Fahreragenten können während Grünphasen ganz normal die Kreuzung passieren, wie andere Verkehrsteilnehmer auch. Zusätzlich dürfen sie auch während eigener Rotphasen fahren, sofern es keine Konflikte durch besetzte Zonen auf ihrer Route gibt.

1.3.2 Der Multi-Agenten-Ansatz der Universität Karlsruhe

Im Jahr 2005 wurde das Projekt *Driver Assistance using Multi-Agent Systems in Traffic* (DAMAST)[6] gestartet. In dem Projekt kooperieren das Institut für Programmstrukturen und Datenorganisation (IPD) der Universität Karlsruhe und das Unternehmen INIT AG mit Hauptsitz ebenfalls in Karlsruhe. Ziel von DAMAST ist es, das Potenzial fahrzeuggebundener Agententechnologie im Straßenverkehr zu erkunden und dadurch Ansatzpunkte für eine Mobilitätssteigerung und bessere Ausnutzung der Verkehrswege und ihrer Knotenpunkte aufzuzeigen. Das dem Projekt zugrunde liegende Szenario entspricht dem klassischen Kreuzungsmanagement: Es werden an Kreuzungen keine Ampeln mehr benötigt, weil sich sämtliche Fahrzeuge im Kreuzungsbereich gemeinsam auf Art und Abfolge ihrer Kreuzungsdurchfahrt einigen.

[6] Weitere Informationen sind unter **http://www.ipd.uni-karlsruhe.de/~damast/** verfügbar.

Der im Rahmen des Projekts erarbeitete Lösungsansatz geht von der Grundannahme aus, dass Fahrer die Wartezeit an Kreuzungen unterschiedlich bewerten. Beispielsweise würde jemand, der einen dringenden Termin wahrnehmen möchte und Gefahr läuft, sich zu verspäten, eine Wartezeit von n Sekunden an einer Kreuzung stärker bewerten als jemand, der sich gerade auf dem Heimweg befindet und nicht in Eile ist. Diese individuellen und unterschiedlichen Bewertungen sollen als Basis herangezogen werden, um eine Vorfahrtsreihenfolge an der Kreuzung zu bilden, die den Gesamtnutzen aller Verkehrsteilnehmer steigert. Auch hier werden wieder autonome Fahrfunktionen der Fahrzeuge und die Möglichkeit der Kommunikation untereinander vorausgesetzt. Die prinzipielle Systematik wird in die folgenden vier Phasen eingeteilt [33] und soll den Ansatz erweitern, der im AIM-Projekt vorgestellt wurde:

1. Fahrzeuge fahren in die Kreuzung ein, und ihre Fahreragenten nehmen Kontakt mit dem Kreuzungsmanager auf.
2. Der Kreuzungsmanager weist den Fahreragenten ein initiales Zeitfenster zu, innerhalb dessen sie die Kreuzung ohne Kollisionen überqueren können.
3. Die initialen Zeitfenster werden zwischen Fahreragenten gehandelt, um ihren jeweiligen Nutzen zu erhöhen. Der Handel kann in unterschiedlichen Formen erfolgen und wird durch den Kreuzungsmanager überwacht.
4. Das jeweilige Fahrzeug passiert die Kreuzung innerhalb des jeweils aktuell zugewiesenen Zeitfensters.

Ferner werden in [33] einige Varianten für die Ermittlung der Vorfahrtsreihenfolge in Ansätzen diskutiert und entsprechend der obigen vier Phasen gegliedert. Basis für die Berechnungen der ersten Varianten ist die Zeit T_t^j, die von Fahrzeug j benötigt wird, um die Kreuzung zu passieren. Die Differenz zwischen der schnellstmöglichen Durchfahrtszeit, wenn also das Fahrzeug j auf keine anderen Verkehrsteilnehmer Rücksicht nehmen muss, und der Durchfahrtszeit aufgrund des aktuell zugeteilten Zeitfensters wird als Wartezeit T_w^j bezeichnet. Diese Wartezeit wird von den Fahrzeugen bzw. ihren Fahrern unterschiedlich bewertet. Fahrzeug j bewertet diese Wartezeit mit dem Faktor v^j. Weiterhin wird jedem Fahrzeug die Nutzenfunktion

$$u^j = b^j - v^j \cdot T_w^j \tag{1.3}$$

zugeordnet, wobei es sich bei b^j um das Budget von Fahrzeug j handelt. Es wird darauf hingewiesen, dass es prinzipiell für die Systematik egal ist, ob es sich bei dem Budget um reales Geld oder um Einheiten einer beliebigen Währung handelt, die vom Kreuzungsmanager ausgegeben werden. Eine Möglichkeit für die Ermittlung der Vorfahrtsreihenfolge besteht nun darin, dass innerhalb von Phase 2 den Fahrzeugen nach dem FIFO-Prinzip (*First-In, First-Out*) ein Zeitfenster für das Passieren der Kreuzung durch den Kreuzungsmanager zugewiesen wird. Diese initialen Zeitfenster werden nun von den Fahreragenten in Phase 3 unter Verwendung der jeweils vorhandenen Budgets gehandelt. Ziel ist es, den eigenen Nutzen gemäß Gleichung (1.3) zu erhöhen. Gleichzeitig wird auch automatisch der Gesamtnutzen aller Fahrzeuge im Kreuzungsbereich erhöht, da ein Handel nur zustande kommt, wenn beide Handelspartner dadurch einen Nutzenzuwachs zu erwarten haben.

Ein weiterer Ansatz für die Ermittlung der Vorfahrtsreihenfolge wird als *Initial Time-Slot Auction* (ITSA) bezeichnet. In dieser Variante werden aufeinanderfolgende Zeitfenster ΔT durch den Kreuzungsmanager an Fahrzeuge versteigert, die in den Kreuzungsbereich einfahren und noch nicht über ein Zeitfenster verfügen. An der Versteigerung dürfen nur Fahrzeuge teilnehmen, die ausschließlich Fahrzeuge vor sich haben, die alle schon über ein Zeitfenster verfügen. Die Auktionen haben die Besonderheit, dass sämtliche an ihr beteiligten Fahreragenten ein für die jeweils anderen Fahreragenten nicht sichtbares Angebot $v \cdot \Delta T$ abgeben. Den Zuschlag für das jeweilige Zeitfenster bekommt der Fahreragent mit dem höchsten Gebot; er muss jedoch nur den Preis des zweithöchsten Gebots zahlen[7]. Mit dieser Systematik soll erreicht werden, dass die Fahreragenten ein Gebot mit ihrer tatsächlichen Wertschätzung abgeben. Die Dauer der Auktionen ist prinzipiell nach oben nicht beschränkt, so dass es vorkommen kann, dass am Ende der Auktion das versteigerte Zeitfenster bereits in der Vergangenheit liegt. Diese Tatsache wiederum lässt weder Effizienz noch Optimalität des Ansatzes vermuten. Weiterhin wird in der Basisvariante von ITSA keine obere Schranke für die Wartezeit einzelner Fahrzeuge garantiert. ITSA wurde auch mit der Modifikation getestet, dass nachfolgende Fahrzeuge ihre Vorgänger bei der Auktion unterstützen können, damit diese den Zuschlag erhalten und die Nachfolger davon profitieren, da sie selbst dadurch die Möglichkeit erhalten, schneller die Kreuzung zu passieren. Diese Variante fördert jedoch nicht ein Verhalten, bei dem Fahreragenten ein Gebot abgeben, das ihrer tatsächlichen Wertschät-

[7] Zweitpreis-Auktion oder Vickrey-Auktion, nach dem US-amerikanischen Ökonom William Spencer Vickrey.

zung entspricht, da sie darauf hoffen können, dass nachfolgende Fahrzeuge sie ausreichend unterstützen. Von ITSA existieren verschiedene Varianten, die nur die Phasen 1, 2 und 4 der oben genannten Gliederung berücksichtigen. Als weitere Modifikation wurde angedacht, einen Handel gemäß Phase 3 mit den bereits ersteigerten Zeitfenstern zu erlauben.

Die unterschiedlichen Varianten wurden an einer Kreuzung gemäß Abbildung 1.8 untersucht. Bei dieser Kreuzung gibt es in jeder Himmelsrichtung

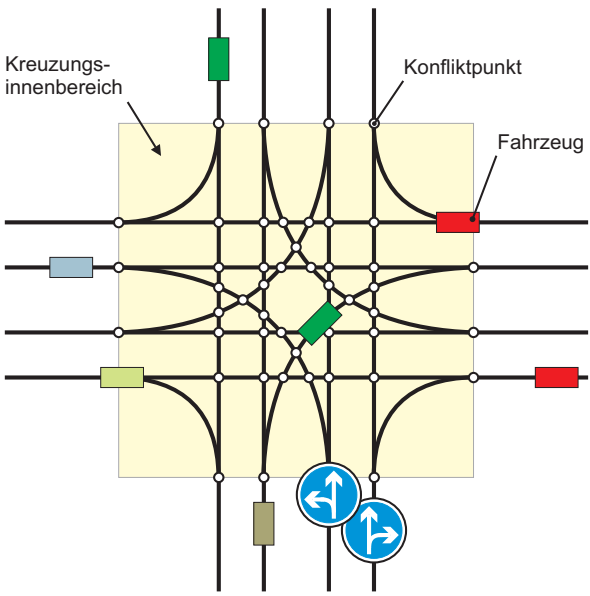

Abbildung 1.8: Modell der Kreuzung für die Evaluation

zwei parallele Ein- und Ausfahrten. Bei den Einfahrten sind auf der rechten Fahrspur Geradeausfahren und Rechtsabbiegen erlaubt, auf der linken Fahrspur entsprechend Geradeausfahren und Linksabbiegen. Um die Kreuzung zu durchfahren, müssen je Route bzw. Fahrspur ca. 500 m zurückgelegt werden; ein Geschwindigkeitslimit von 50 km/h darf dabei nicht überschritten werden. Im Kreuzungsinnenbereich darf sich zu jedem Zeitpunkt nur ein Fahrzeug aufhalten. Für das Durchfahren dieses Bereichs werden jedem Fahrzeug 4 Sekunden eingeräumt. Damit ist die Kapazität der Kreuzung auf max. 900 Fahrzeuge pro Stunde beschränkt. Dieser Wert erscheint sehr gering, wenn man bedenkt, dass bei einer gewöhnlichen Kreuzung mit Lichtsignalanlage und einer Einfahrt je Himmelsrichtung bereits eine Kapazität von ca. 1800

Fahrzeugen pro Stunde erreichbar ist (vgl. [13]). Ein Grund für die geringe Kapazität liegt sicher in der unzureichenden Berücksichtigung der Fahrzeugdynamik und einer auf den Fahrzeugabmessungen beruhenden Kollisionsprüfung, die darin ihren Ausdruck findet, dass sich zu jedem Zeitpunkt nur ein Fahrzeug im Kreuzungsinnenbereich aufhalten darf. Ein weiterer Grund ist in dem Verfahren für die Bestimmung der Vorfahrtsreihenfolge zu sehen, das keine Optimalität gewährleisten kann, worauf in Abschnitt 1.3.3 noch ausführlicher eingegangen wird. Als wesentliches Ergebnis der Untersuchung wird herausgestellt, dass die durchschnittliche gewichtete Wartezeit durch die vorgestellten Verfahren reduziert werden kann, die durchschnittliche Wartezeit aber kaum verändert wird. Diese Aussage impliziert, dass der Durchsatz an einer Kreuzung mit dem vorgestellten Verfahren nicht signifikant erhöht werden kann, sondern lediglich die Wartezeit von eiligen Fahrern zu weniger eiligen innerhalb gewisser Grenzen verschoben werden kann.

1.3.3 Analyse und Bewertung

Um die Tragfähigkeit eines agentenbasierten Ansatzes beurteilbar zu machen, werden zuvor die wichtigsten Eigenschaften eines Software-Agenten dargestellt. Nach [18], [32] und weiteren Veröffentlichungen aus dem Forschungsfeld der Künstlichen Intelligenz sind dies die Folgenden:

- **Autonomie**: Der Agent handelt innerhalb bestimmter Grenzen autonom und ohne Eingriff des Menschen.
- **Proaktivität**: Der Agent versucht ein gegebenes Ziel zu erreichen, indem er Aktionen plant bzw. auswählt und durchführt.
- **Reaktivität**: Der Agent reagiert in angemessener Weise auf Änderungen seiner Umwelt.
- **Soziale Fähigkeiten**: Hiermit ist im Wesentlichen die Fähigkeit gemeint, mit anderen Agenten zu kommunizieren bzw. zu interagieren.
- **Lernfähigkeit**: Der Agent lernt aufgrund zuvor getätigter Entscheidungen und Beobachtung seiner Umwelt und passt sein Verhalten an.

Laut Schneeberger [18] müssen die ersten vier dieser Eigenschaften erfüllt sein, damit von einem Agenten bzw. von einem Agentensystem gesprochen werden kann. Agenten können grob in zwei Kategorien unterteilt werden,

in *reaktive Agenten* und *kognitive Agenten*, wobei die Grenzen hier fließend bzw. auch Mischformen möglich sind. Während kognitive Agenten über eine eigene Wissensbasis verfügen und ein Modell der Umwelt in einer eigenen Datenstruktur verwalten, verfügen reaktive Agenten prinzipiell nicht über eigenes Wissen, sondern reagieren direkt auf Basis ihrer Wahrnehmungen. Ihre Architektur bietet Vorteile bei dynamischen Anwendungen, bei denen nicht viel Zeit für eine Reaktion bleibt. Reaktive Agenten verfügen über die ersten vier Eigenschaften der obigen Liste, während kognitive Agenten zusätzlich noch lern- und anpassungsfähig sind. Bei den Agenten innerhalb der oben vorgestellten Projekte AIM und DAMAST handelte es sich innerhalb der ersten Implementierungen um relativ einfach strukturierte reaktive Agenten. In aktuellen Arbeiten ist jedoch eine Tendenz zur Ausstattung der Agenten mit kognitiven Fähigkeiten zu erkennen. So schlagen Dresner und Stone in [12] vor, dass die Fahreragenten auf Basis der aktuellen Verkehrslage ihr Verhalten in Bezug auf Reservierungsanfragen beim Kreuzungsmanager zwischen *pessimistisch* und *optimistisch* ändern sollen. Je optimistischer ein Fahreragent wird, desto früher liegt das Zeitfenster, das er für sich zu reservieren versucht.

Ein *Multi-Agenten-System* (MAS) kann allgemein als System angesehen werden, in dem mehrere Einheiten (Agenten) interagieren, häufig mit dem Ziel, kollektiv bzw. gemeinsam ein komplexes Problem zu lösen. Diese beteiligten Einheiten können gleichartig oder verschieden sein. Multi-Agenten-Systeme gibt es sowohl in der Biologie als auch in der Technik. Man spricht in diesem Zusammenhang auch von *verteilter Intelligenz* (Biologie) bzw. *verteilter künstlicher Intelligenz* (Technik). Ein biologisches Multi-Agenten-System ist zum Beispiel eine Ameisenpopulation auf Futtersuche. Beobachtet man die Ameisen bei ihrer Futtersuche, so stellt man fest, dass nach einiger Zeit optimale bzw. kürzeste Wege vom Nest zu allen Futterquellen existieren. Mathematisch gesehen, handelt es sich bei dem Gebilde um einen sogenannten *minimal spannenden Baum* bzw. um einen Graphen, bei dem kürzeste Wege von einem Startknoten zu allen anderen Knoten des Graphen existieren. Die Futterquellen entsprechen dabei den Knoten und die Ameisenstraßen den Kanten. Eine einzelne Ameise ist nicht in der Lage, einen minimal spannenden Baum zu bestimmen, das Zusammenspiel aller Ameisen hingegen lässt diese Fähigkeit *emergieren*. Eine weitere wichtige und entscheidende Eigenschaft des Szenarios besteht darin, dass die zu lösende Problematik mit einer unbekannten Umwelt verknüpft ist. Die Ameisen wissen a priori nicht, wo sich die Futterquellen befinden, sondern müssen eine unbekannte Umwelt sukzessive

explorieren und nach Futterquellen durchsuchen. Zu diesem Zweck bewegen sie sich einerseits (bedingt) zufällig in der unbekannten Umwelt, kommunizieren ihre dabei gewonnenen Erkenntnisse aber andererseits über das Setzen von Duftmarken an ihre Artgenossen, deren zufällige Bewegung dadurch innerhalb gewisser Bahnen gelenkt wird (bedingte Zufälligkeit).

Ein weiteres Beispiel für ein biologisches Multi-Agenten-System ist in Abbildung 1.9.1 dargestellt. Vier Fahrer stehen mit ihren Fahrzeugen an ei-

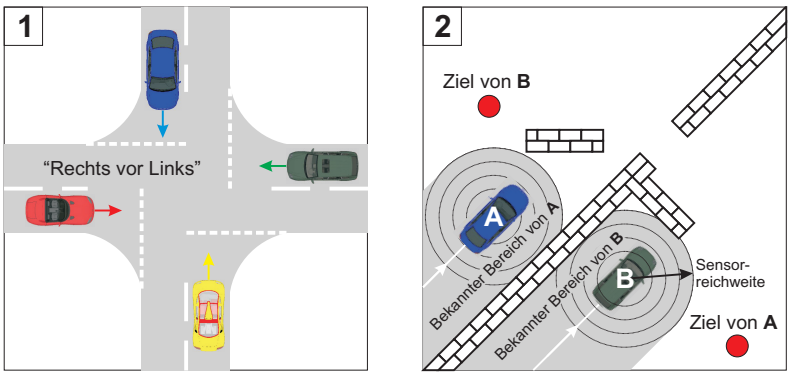

Abbildung 1.9: Agenten und Situationen

ner Kreuzung ohne Verkehrszeichen oder Lichtsignalanlagen. Die in diesem Fall in vielen Ländern gültige Regel *"Rechts vor Links"* führt zu einer *Deadlock*-Situation, da sich rechts von jedem Fahrzeug ein Fahrzeug befindet. In der Realität ist zu beobachten, dass sich in so einer Situation die Agenten (Menschen) mit Handzeichen oder anderen Gesten verständigen, um eine Lösung herbeizuführen. Diese Verständigung ist allerdings mit einem relativ hohen Zeitbedarf verbunden, der den Durchsatz an einem solchen Verkehrsknoten stark limitiert. Genau aus diesem Grunde wurden Verkehrszeichen und Lichtsignalanlagen eingeführt, die nichts anderes sind als Regeln. Regeln (und Gesetze) schränken immer die Freiheiten einzelner Individuen ein, können andererseits aber das Gemeinwohl oder den Nutzenzuwachs jedes betroffenen Individuums signifikant steigern. Je mehr über die Umwelt bekannt ist, desto detaillierter können a priori Regeln zum Erreichen eines gewünschten Ist-Zustands formuliert werden. In dem Szenario einer autonomen Kreuzungsüberquerung sind sämtliche erforderlichen Informationen über die Umwelt bekannt: Die Kreuzungsgeometrie ist bekannt, ebenso die Positionen der Fahrzeuge sowie ihre jeweiligen gewünschten Routen mit dem

zugehörigen gewünschten Fahrprofil. Wenn alle Umweltparameter sowie das gewünschte Ziel bekannt sind, dann kann im Vorfeld eine optimale Strategie auf Basis mathematischer Methoden entwickelt werden, an die sich alle betroffenen Individuen im betrachteten System strikt zu halten haben. Die beteiligten Individuen sind dann reine Befehlsempfänger ohne den geringsten Entscheidungsspielraum. Dies steht jedoch im Widerspruch zu den oben aufgezählten Eigenschaften eines Agenten. Im Falle des Ameisenbeispiels wären die exakten Positionen und Größen der Futterquellen bekannt. Auf dieser Basis würde eine optimale Strategie entwickelt, die für jede Ameise eine detaillierte Marschroute und die zu transportierende Nahrung festlegen würde. Die Ausbeutung der Nahrungsquellen würde dann deutlich schneller und effizienter durchgeführt werden können. Jede Ameise müsste sich dann aber streng deterministisch und einer Maschine gleich verhalten.

Das Multi-Agenten-System gemäß Abbildung 1.9.2 besteht aus den rein technisch realisierten Agenten A und B, die sich in einer unbekannten Umwelt befinden und den Auftrag haben, Zielpunkte auf der jeweils anderen Seite einer Mauer anzufahren. Ein Agent kann im Rahmen der Ausführung dieses Auftrags unterschiedliche zielführende Suchstrategien anwenden, in deren Rahmen unterschiedliche Ereignisse auftreten können, auf die in angemessener Form reagiert werden muss. In diesem Fall lässt sich aufgrund der unbekannten Umwelt keine optimale Strategie im Voraus berechnen. Beispielsweise kann nicht a priori gesagt werden, wann ein Agent die Öffnung in der Mauer findet und ob dann, im Falle eines Durchfahrtsversuchs, die Öffnung nicht durch den zweiten Agenten versperrt wird.

Der Hauptunterschied in den beschriebenen Szenarien liegt in der Fragestellung, ob die relevante Umwelt bekannt oder unbekannt ist. Bei vollständig bekannter Umwelt kann im Vorfeld mit mathematischen Methoden eine optimale Strategie entwickelt werden, die einem agentenbasierten Ansatz stets überlegen sein wird, da sie im Vorfeld bekannt ist und nicht erst *erlernt* werden muss. Im Falle des AIM kann man sich leicht vorstellen, dass die Möglichkeit, nahezu beliebig Reservierungs- und Stornierungsaufträge für Zeitfenster zu initiieren, nur zu einer suboptimalen Ausnutzung des Kreuzungsinnenbereichs führen kann. Ähnlich sieht es bei dem DAMAST-Ansatz aus: Die Dauer einer Versteigerung kann nicht garantiert werden, so dass ersteigerte Zeitfenster in der Vergangenheit liegen können. *Deadlocks* können nicht ausgeschlossen werden, Interessen nachfolgender Fahrzeuge werden nur unzureichend berücksichtigt usw. In dem Szenario des autonomen Kreuzungsmanagements ist die relevante Umwelt vollständig bekannt: Sämtliche Fahrzeuge

wissen, wo sie prinzipiell fahren können und wie sie fahren wollen. Weiterhin ist bekannt, wann und wo potenzielle Kollisionspartner zu berücksichtigen sind. Auf dieser Basis kann nach einem festen mathematischen Algorithmus eine optimale Strategie für die eigene Kreuzungsüberquerung bestimmt werden. Die Art, wie dies erfolgt, wird in den folgenden Kapiteln ausführlich dargelegt. Agentenbasierte Ansätze hingegen sind in einer unbekannten oder dynamischen Umwelt überlegen, wenn auf plötzlich oder unerwartet auftretende Ereignisse reagiert werden muss. Dies wäre beispielsweise der Fall, wenn beim Kreuzungsmanagement ein Unfall auftreten würde, durch den eine oder mehrere Fahrspuren plötzlich gesperrt wären. Aber auch solche Fälle können zum Teil vorausgesehen und in einer entsprechenden Strategie im Vorfeld berücksichtigt werden.

1.4 Die Idee der Interpretation als Netzwerkflussproblem

Viele praktische Probleme, die auf den ersten Blick völlig unterschiedlich erscheinen, erweisen sich oft als gleichartig, wenn man sie nur genügend abstrahiert bzw. auf eine Abstraktionsebene projiziert, die das jeweilige Problem auf seinen Kern oder eigentlichen Charakter reduziert und alles andere ausblendet.

Die Formulierung eines praktischen Problems als *Netzwerkflussproblem* (Kapitel 2) entspricht beispielsweise einer solchen Abstraktionsebene. Die unterschiedlichsten Problemstellungen können als Netzwerkflussprobleme formuliert werden. Ein Beispiel ist die Routenplanung mit einem Navigationssystem, bei dem Städte oder Ortschaften die Knotenpunkte eines Wegenetzes bzw. Netzwerks bilden. Die interessierende Fragestellung könnte in diesem Fall beispielsweise sein, wie ein Fahrzeug am besten von der Stadt A zur Stadt B durch das Wegenetz *fließt*. Ein völlig anderes Problem, dessen Kern jedoch derselbe ist, besteht in der Beantwortung der Fragestellung, wie Notausgänge in einem Bürogebäude zu planen sind, um sicherzustellen, dass es bei einem Brand in einer vorgegebenen Zeit evakuiert werden kann. In diesem Fall bilden die Flure und Notausgänge des Gebäudes ein Netzwerk, an dessen Knotenpunkten sich ein oder mehrere Flure treffen. Die in dem Netzwerk *fließenden* Objekte entsprechen den zu evakuierenden Personen, die eine

bestimmte maximale *Flussgeschwindigkeit* aufweisen und von ihren Büroräumen zu einem zentralen, vor dem Gebäude befindlichen Sammelplatz gelangen müssen. Dabei ist zu beachten, dass Flure und Knotenpunkte bestimmte maximale Kapazitäten aufweisen, die nicht überschritten werden dürfen. Im Vergleich zu dem zuvor skizzierten Problem der Routenplanung sind hier also zusätzliche Kapazitätsschranken zu berücksichtigen, was aber letztlich nichts am Kern des Problems ändert.

Das Problem der autonomen Überquerung einer Kreuzung bzw. eines beliebigen Verkehrsknotenpunktes ist in seinem Kern nicht anders als die beiden zuvor beschriebenen Probleme: Die Möglichkeiten eines Fahrzeugs zum Passieren eines Verkehrsknotenpunktes werden als *Routen* bezeichnet. Routen schneiden einander oder liegen quasi übereinander auf demselben Fahrstreifen: Ein in die Kreuzung einfahrendes Fahrzeug hat die 3 Routenmöglichkeiten, geradeaus zu fahren und links oder rechts abzubiegen, wobei diese 3 Routen bis zum Kreuzungsinnenbereich auf demselben Fahrstreifen liegen. Für eine Kreuzung mit sowohl 4 Einfahrts- als auch 4 Ausfahrtsfahrstreifen ergeben sich dann insgesamt 12 Routenmöglichkeiten. Jeder Route lässt sich eine Folge von Orten oder Positionen auf der Kreuzungsebene zuordnen, die nacheinander von Fahrzeugen auf dieser Route besetzt werden und die zu jedem Zeitpunkt nur von einem Fahrzeug besetzt werden dürfen. Die Routen mit ihren zugeordneten Ressourcen in Form von diskreten Orten bzw. Zonen zu diskreten Zeitpunkten können somit als Netzwerk aufgefasst werden. Durch dieses Netzwerk ist für jedes Fahrzeug ein Weg zu planen. Die Fahrzeuge entsprechen dabei den durch das Netzwerk *fließenden* Objekten, deren Fluss unter gewissen Nebenbedingungen zu optimieren ist.

2 Diskrete Optimierungsmethoden

Im Jahre 1735 wurde von Leonhard Euler das sogenannte *Königsberger Brückenproblem* formuliert [38]: Das Zentrum von Königsberg wurde seinerzeit durch den Fluss Pregel in insgesamt 4 Bereiche unterteilt, die miteinander über insgesamt 7 Brücken verbunden waren. Aus diesem Sachverhalt leitete Euler die Fragestellung ab, ob ein Rundgang durch das Stadtzentrum realisierbar sei, bei dem alle 7 Brücken genau einmal überquert werden. Euler schloss seinerzeit einen solchen Rundgang aus und begründete dies damit, dass für jeden Weg, der zu einem der Bereiche führt, auch ein entsprechender Weg vorhanden sein muss, der auch wieder von diesem Bereich fortführt. Die Folge daraus ist, dass die Anzahl der Brücken, über die ein Bereich mit einem anderen Bereich verbunden ist, stets gerade sein muss. Diese Bedingung ist jedoch beim *Königsberger Brückenproblem* nicht erfüllt. Aufgabenstellungen wie diese sind prädestiniert für die Lösung mit Methoden aus dem Bereich der Graphentheorie. Im beschriebenen Beispiel werden die Bereiche des Stadtzentrums durch *Knoten* symbolisert und die Brücken durch *Kanten*, welche die Knoten zu einem *Graphen* verbinden.

Leonhard Euler wird häufig als Begründer der Graphentheorie genannt. Er wurde jedoch seinerseits durch Ausführungen von Gottfried Wilhelm Leibniz aus dem Jahre 1679 inspiriert. Leibniz nannte die Disziplin, die von Euler wiederentdeckt wurde und später in der Graphentheorie aufging, die *Geometrie der Lage* [38].

Viele algorithmische Probleme, insbesondere Optimierungsprobleme aus dem Bereich der diskreten Mathematik, können auf Graphen zurückgeführt und mit Methoden der Graphentheorie effizient gelöst werden, so auch das im vorherigen Kapitel skizzierte Problem der Trajektorienplanung. In diesem Kapitel werden die Grundlagen dafür zur Verfügung gestellt. Zuerst werden die grundlegenden Eigenschaften von Graphen kurz erläutert; auf dieser Grundlage wird dann ein Algorithmus für die Suche sogenannter *kürzester Wege* von

einem Startknoten zu allen anderen Knoten des Graphen vorgestellt. Die Bezeichnung *kürzester Weg* ist hier im weiteren Sinne zu verstehen; es kann sich dabei tatsächlich auch um einen kostenminimalen Weg handeln, was durch die Ausführungen in den folgenden Abschnitten noch deutlicher werden wird. Im Anschluss daran wird die *Dynamische Programmierung* nach BELLMAN detailliert eingeführt. Es wird gezeigt, dass sie prinzipiell der Suche eines kürzesten Weges entspricht und dass Probleme, die gemäß der *Dynamischen Programmierung* formuliert wurden, mit den im Folgenden eingeführten Algorithmen für die Suche kürzester Wege gelöst werden können.

Letztlich werden in diesem Kapitel die Methoden eingeführt und erläutert, die es erlauben, das komplexe Problem des Kreuzungsmanagements gemäß der *Dynamischen Programmierung* zu formulieren bzw. zu strukturieren und im Anschluss mit einem modifizierten Standard-Verfahren für die Suche kürzester Wege zu lösen.

2.1 Grundlagen der Graphentheorie

In der Graphentheorie [9] wird ein beliebiger Graph G durch eine Menge V von Knoten (*Vertices*) und eine Menge E von Kanten (*Edges*) definiert. Kanten verbinden Knoten miteinander und stellen dadurch eine Art von Transportweg dar, denn nur über Kanten kann ein *Objekt* von einem Knoten zu einem anderen gelangen bzw. *fließen*. Werden zwei Knoten $i, j \in V$ durch die Kante e verbunden, so schreibt man auch $e := [i, j]$. Prinzipiell können Kanten gerichtet oder ungerichtet sein. Gerichtete Kanten werden durch Pfeile symbolisiert. Ein Graph, dessen Kanten gerichtet sind, wird auch als *Digraph* (*Directed Graph*) bezeichnet. Innerhalb eines Digraphen ist ein Objektfluss nur in Richtung der Pfeile zulässig. Bei einem azyklischen Digraph existiert kein Knoten v_i, von dem aus ein Weg entlang der Pfeile des Graphen wieder zu v_i führt. Ist jedoch solch ein Weg vorhanden, so spricht man von einem geschlossenen Weg bzw. von einem Zyklus. Weiterhin können Kanten über Kosten und/oder Kapazitätsschranken verfügen, wobei Kapaziätsschranken bei dem hier behandelten Problemen nicht relevant sind. Kosten werden in der Kostenmenge C zusammengefasst und stellen eine Bewertung dar für den Fluss eines Objektes über eine Kante. Der Begriff *Kosten* ist hier also im weiteren Sinne zu verstehen: Es kann sich bei den *Kosten* tatsächlich auch um Entfernungen handeln oder aber um Gütemaße, die auf Basis einer beliebig

vorgegebenen Gütefunktion berechnet werden, um den Übergang vom Knoten i zum Knoten j, symbolisiert durch die Kante $e := [i, j]$, zu bewerten. In der Literatur werden *kostenbewertete Digraphen* häufig auch als *Netzwerke* bezeichnet [9].

Ein anschauliches Beispiel für ein Netzwerk liefert Abbildung 2.1. Das in dem

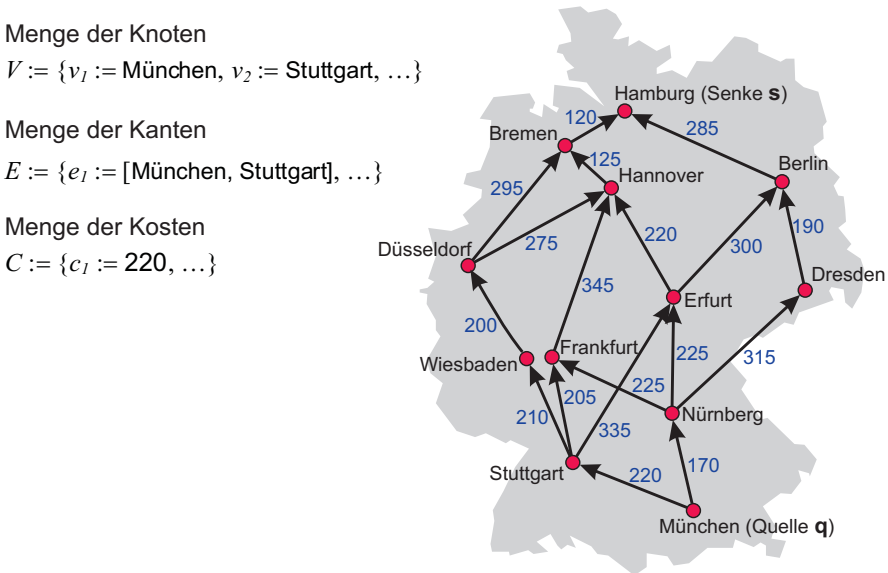

Abbildung 2.1: Beispiel für ein Netzwerk

Bild dargestellte Netzwerk ist gegeben durch den Graphen $G = (V, E, C)$. Die Knotenmenge V besteht aus Städten, die Kantenmenge E aus Autobahnverbindungen zwischen den Städten und die Kostenmenge C aus den Längen [km][1] der einzelnen Autobahnverbindungen. Ein typisches Optimierungsproblem in diesem Netzwerk besteht nun beispielsweise darin, einen kürzesten Weg von München nach Hamburg zu finden oder, anders ausgedrückt, ein Objekt von der *Quelle* **q** zur *Senke* **s** des Graphen zu minimalen Kosten[2] fließen zu lassen; in diesem Fall spricht man auch von einem *q-s-Flussproblem* [9]. Derartige Probleme lassen sich mit Standard-Verfahren für die Suche *kürzester Wege* lösen, was im Folgenden näher erläutert wird.

[1] Entgegen von DIN-Empfehlungen werden (physikalische) Einheiten innerhalb dieser Arbeit durchgehend in eckigen Klammern dargestellt.

[2] *Minimale Kosten* ist in diesem Beispiel also gleichbedeutend mit *kürzestem Weg*.

2.2 Kürzeste-Wege-Algorithmen

Die Algorithmen, die in diesem Abschnitt angesprochen werden, arbeiten auf kostenbewerteten Digraphen $G = (V, E, C)$ mit n_V Knoten und n_E Kanten. Sie berechnen von einem beliebigen Startknoten $a \in V$ kürzeste (bzw. kostenminimale) Wege zu allen anderen Knoten $i \in V$ mit $i \neq a$ des Graphen. Nachdem der Lauf eines solchen Algorithmus abgeschlossen wurde, verfügt jeder von a aus erreichbare Knoten i innerhalb des Graphen G über ein sogenanntes *Label* mit den Informationen Distanz D[i] zu Knoten a und Routenvorgänger R[i] von Knoten i auf dem kürzesten Weg von Startknoten a zu Knoten i. Über R[i] kann nun rekursiv rückwärts schreitend von jedem beliebigen erreichbaren Knoten i der entsprechende kürzeste Weg von Knoten a zu Knoten i bestimmt werden. Führt man dies für sämtliche von a aus erreichbaren Knoten durch, so erhält man eine baumartige Struktur mit einer Wurzel, in der sämtliche Wege beginnen. Diese Wurzel entspricht dem Startknoten a. Man spricht deshalb in diesem Fall auch von einem *Wurzelbaum* und bei den Algorithmen entsprechend von *Baumalgorithmen*.

Die Gruppe der Baumalgorithmen wird in [9] in die beiden Kategorien *Label-Correcting-* und *Label-Setting*-Verfahren unterteilt (abgekürzt: LC- und LS-Verfahren). Bei den LC-Verfahren wird das Label eines Knotens während der Laufzeit des Algorithmus unter Umständen mehrmals korrigiert, während bei den LS-Verfahren jeder Knoten nur einmal überprüft und sein jeweiliges Label dabei gesetzt wird. Dem unter Umständen erhöhten Überprüfungsaufwand bei den LC-Verfahren steht das Erfordernis eines Sortierverfahrens und entsprechender Sortieraufwand bei den LS-Verfahren gegenüber. Welcher Algorithmus am günstigsten ist, hängt stark von den Eigenschaften des Graphen und von der jeweiligen Implementierung des Baumalgorithmus ab. Sämtliche Baumalgorithmen haben jedoch gemein, dass es sich bei ihnen um Iterationsverfahren handelt. Die wichtigsten Algorithmen aus der Kategorie der LS-Verfahren sind die Algorithmen von Dijkstra [8] und Dantzig [6]. Im Rahmen dieser Arbeit werden jedoch ausschließlich Algorithmen aus dem Bereich der LC-Verfahren verwendet, da sie einerseits sehr leicht zu implementieren sind und andererseits, im Hinblick auf die charakteristischen Eigenschaften der verwendeten Graphen (vgl. Kapitel 5), auch als geeigneter erscheinen. Prinzipiell muss hier noch eine genaue Analyse und ein Vergleich von Verfahren aus den Kategorien der LC- und LS-Verfahren erfolgen. Aus zeitlichen Gründen steht diese Analyse noch aus bzw. ist nicht mehr Bestandteil der vorliegenden Arbeit. Bei den LC-Verfahren ist an erster Stelle

der Algorithmus von Richard Bellman und Lester Ford zu nennen. Gelegentlich wird auch vom Moore-Bellman-Ford-Algorithmus gesprochen, da auch Edward Moore zu seiner Entwicklung beigetragen hat [17]. Im Folgenden wird zuerst der Bellman-Ford-Algorithmus vorgestellt. Auf dieser Basis wird dann gezeigt, wie sich die Effizienz des Algorithmus durch die Modifikationen von Moore [20] bei einer Vielzahl von Problemstellungen teilweise deutlich steigern lässt. Nur der Vollständigkeit halber angemerkt sei an dieser Stelle noch, dass bei dem Bellman-Ford-Algorithmus, im Gegensatz zu den bekanntesten Algorithmen aus dem Bereich der LS-Verfahren [8, 6], die Gewichtungen der Kanten auch negative Werte annehmen dürfen. In diesem Fall müssen jedoch Zyklen negativer Länge ausgeschlossen werden, da man sonst keine sinnvollen Ergebnisse erhält. Diese Eigenschaft spielte jedoch keine Rolle für die hier getroffene Wahl zu Gunsten eines LC-Verfahrens. Der Pseudo-Code des Bellman-Ford-Algorithmus ist im Listing 2.1 abgedruckt.

In den Zeilen 100 bis 104 erfolgt die Initialisierung der Datenfelder. Es wird ein Startknoten $a \in V$ ausgewählt, der zu sich selbst die Entfernung 0 besitzt. Die Entfernung D[i] aller anderen Knoten $i \in V \wedge i \neq a$ zum Startknoten a wird zu Beginn auf ∞ gesetzt. In einem Schleifendurchlauf gemäß den Zeilen 108-115 werden sämtliche Kanten $[i, j] \in E$ des Graphen, die im Knoten i beginnen und im Knoten j enden, dahingehend überprüft, ob sich die Distanz D[j] des Knoten j zum Startknoten a reduzieren lässt, wenn man den Knoten j über den Knoten i erreicht. Ist dies der Fall, so werden im Label des Knotens j sowohl der neue optimale Vorgängerknoten i (R[j]:=i) als auch die neue kürzeste Entfernung (D[j]:=D[i]+c(i,j)) gespeichert. Im ersten Durchlauf dieser Schleife besitzen sämtliche Knoten des Graphen, mit Ausnahme des Startknotens a, die Distanzinformation D[i]=∞. Das bedeutet, dass Korrekturen des Labels im ersten Schleifendurchlauf nur möglich sind bei Knoten, die unmittelbar durch eine Kante mit dem Startknoten a verbunden sind, also unmittelbare Nachfolger von Knoten a sind. Im zweiten Schleifendurchlauf werden dann die unmittelbaren Nachfolger der Nachfolger von a überprüft, und im k-ten Schleifendurchlauf werden entsprechend kürzeste Wege mit maximal k Kanten berechnet. Ein Weg ohne Zyklen kann maximal n_V Knoten und $n_V - 1$ Kanten enthalten. Die innere Schleife gemäß den Zeilen 108-115 muss somit $k = n_V - 1$ mal durchlaufen werden. Wenn sich jedoch im n_V-ten Durchlauf nochmals eine Verkürzung eines Weges durch einen Weg mit nunmehr n_V Kanten realisieren lässt, so belegt dies das Vorhandensein eines Zyklus mit negativer Länge. Diese Überprüfung erfolgt im Code-Segment

```
100   Wähle Startknoten a ∈ V;
101   Distanz D[i] von Knoten a zu Knoten i := ∞ für i=1,...,n_V;
102   Distanz D[a] := 0;
103   Routenvorgänger R[i] von Knoten i := ∅ für i=1,...,n_V;
104   k := 1;
105
106   while (k < n_V)
107   {
108      for (jede Kante [i,j] aus Kantenmenge E)
109      {
110         if (D[i] + c(i,j) < D[j]))
111         {
112            D[j] := D[i] + c(i,j);
113            R[j] := i;
114         }
115      }
116      k := k+1;
117   }
118
119   for (jede Kante [i,j] aus Kantenmenge E)
120   {
121      if (D[i] + c(i,j) < D[j]))
122      {
123         Ausgabe: "Zyklus mit negativer Länge gefunden!"
124      }
125   }
```

Listing 2.1: Pseudo-Code des Bellman-Ford-Algorithmus

der Zeilen 119-125. Die Laufzeit des Bellman-Ford-Algorithmus ist immer identisch mit seiner *Worst-Case-Komplexität* $O(n_V n_E)$.

Durch den Vorschlag von Moore [20], erreichte Knoten zu markieren und nur jeweils diese Knoten in einem Iterationsschritt einer Überprüfung zu unterziehen, kann in vielen Fällen der Rechenaufwand signifikant reduziert werden, wenn auch die Worst-Case-Komplexität identisch bleibt. Der im Folgenden vorgestellte Algorithmus wird als *"FIFO - Kürzeste Wege"*-Algorithmus bezeichnet [9]. Er basiert auf dem Vorschlag von Moore, wurde jedoch in der Form des Listings 2.2 erstmalig von Braess [2] implementiert. FIFO (*First-In, First-Out*) bedeutet dabei, dass markierte Knoten am Ende einer Warteschlange gespeichert werden und in einem Iterationsschritt jeweils der erste Knoten innerhalb der Warteschlange überprüft wird.

```
100   Wähle Startknoten a ∈ V;
101   Distanz D[i] von Knoten a zu Knoten i := ∞ für i=1,...,n_V;
102   Distanz D[a] := 0;
103   Routenvorgänger R[i] von Knoten i := ∅ für i=1,...,n_V;
104   Erster Knoten SK in Warteschlange := a;
105
106   do
107   {
108      for (alle Knoten j, die Nachfolger sind von Knoten SK)
109      {
110         if (D[SK] + c(SK,j) < D[j]))
111         {
112            D[j] := D[SK] + c(SK,j);
113            R[j] := SK;
114            if (Knoten j noch nicht in Warteschlange) then
115            {
116               Füge Knoten j am Ende der Warteschlange ein;
117            }
118         }
119      }
120      if (mehr als ein Knoten in Warteschlange) then
121      {
122         setze SK auf nächsten Knoten;
123      }
124      lösche den ersten Knoten aus der Warteschlange;
125   } while (Schlange nicht leer)
```

Listing 2.2: Pseudo-Code der FIFO-Implementierung

Zu Beginn erfolgt bei der FIFO-Implementierung die Initialisierung der Datenfelder in identischer Form wie beim Bellman-Ford-Algorithmus. Hinzu kommt jedoch, dass eine Warteschlange implementiert wird, in der zu Beginn lediglich der Startknoten a vorhanden ist. Die Hauptschleife des Algorithmus (Zeilen 106-125) wird so lange ausgeführt, bis die Warteschlange leer ist. Die Knoten in der Warteschlange werden nacheinander überprüft. Dies bedeutet, dass der erste Knoten i der Warteschlange entnommen und geprüft wird, ob seine Nachfolger bzw. sämtliche Knoten j, die direkt über eine Kante von i aus erreicht werden können, auf diesem Wege besser erreicht werden können als zuvor. Wenn sich die Distanz D[j] von Knoten j zu Startknoten a durch einen Weg über den Knoten i reduzieren lässt, so wird das Label des Knotens j in bekannter Weise geändert. Weiterhin wird der Knoten j am

Ende der Warteschlange aufgenommen, sofern er noch nicht in dieser vorhanden ist. Nachdem sämtliche Nachfolger von Knoten i bearbeitet wurden, werden i aus der Warteschlange gelöscht und der nächste Knoten der Warteschlange auf die beschriebene Art überprüft. Der Algorithmus endet, wenn die Warteschlange leer ist. Die Implementierung gemäß Listing 2.2 deckt keine Zyklen negativer Länge auf. Eine dahingehende Erweiterung lässt sich aber leicht einfügen, indem man protokolliert, wie oft ein Knoten bereits in die Warteschlange aufgenommen wurde: Da bei einem Graphen mit n_V Knoten maximal $n_v - 1$ Kanten in einem beliebigen Knoten des Graphen enden können, liegt ein Zyklus negativer Länge vor, sobald ein Knoten zum n_V-ten Mal in die Warteschlange aufgenommen wurde.

Der Algorithmus gemäß Listing 2.2 wird nun beispielhaft am Wegenetz gemäß Abbildung 2.1 angewendet. Startknoten und somit erster Knoten der Überprüfung ist die Stadt **München**. Diese Stadt hat die beiden Nachfolger **Stuttgart** und **Nürnberg**. Beide Städte haben zu Beginn die Distanzinformation ∞. Die Entfernung lässt sich also verkürzen; die Labels werden entsprechend korrigiert (D[Stuttgart]:=220, ...), und beide Städte werden in die Warteschlange aufgenommen (Abbildung 2.2). Nachdem die Nachfolger von **München** nun

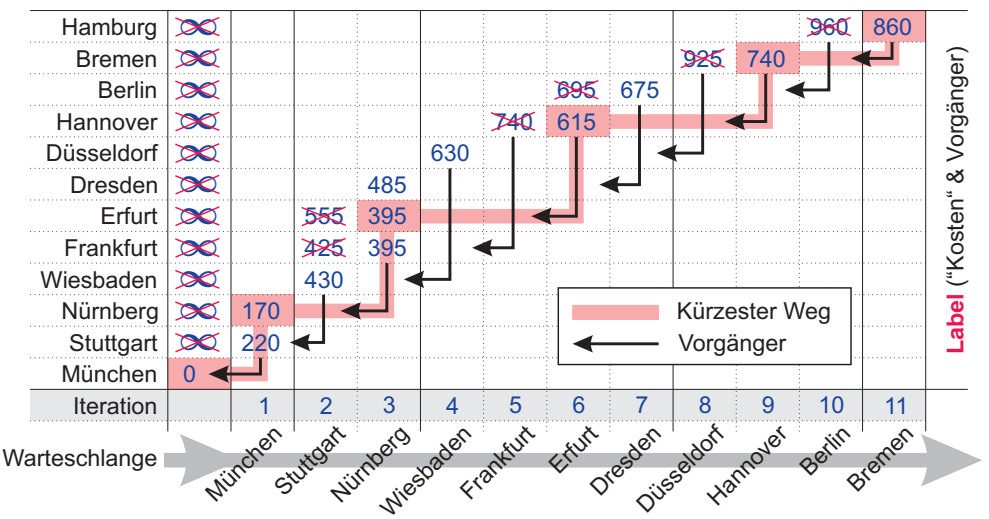

Abbildung 2.2: Ergebnisse der FIFO-Variante

überprüft wurden, werden der nächste Knoten (**Stuttgart**) der Warteschlange entnommen und seine Nachfolger analysiert. Dies sind die Städte **Wiesba-**

den, **Frankfurt** und **Erfurt**. Da auch diese Städte zu Beginn die Distanzinformation ∞ haben, werden auch hier die Labels korrigiert (D[Wiesbaden]:=430, D[Frankfurt]:=425, ...) und die Städte in die Warteschlange aufgenommen. Der nächste Knoten in der Warteschlange ist nun die Stadt **Nürnberg**, von der aus die Städte **Frankfurt**, **Erfurt** und **Dresden** erreichbar sind. **Frankfurt** wurde bereits zuvor über **Stuttgart** erreicht (D[Frankfurt]:=425), der Weg von **München** über **Nürnberg** ist jedoch kürzer (D[Nürnberg]+c(Nürnberg,Frankfurt)=170+225=395). Folglich wird das Label von **Frankfurt** korrigiert (D[Frankfurt]:=395). Da **Frankfurt** bereits in der Warteschlange vorhanden ist, erfolgt keine erneute Aufnahme. Das beschriebene Vorgehen wird nun solange wiederholt, bis die Warteschlange leer ist. Das Ergebnis des Vorgehens zeigt Abbildung 2.2. Sucht man nun den kürzesten Weg von **München** zu einer der anderen Städte, so muss man lediglich die *Zeile* der betreffenden Stadt suchen, dort die *Spalte* mit dem aktuell gesetzten Label wählen und über die Vorgängerbeziehung (Pfeil) zu der unmittelbar vorher auf dem kürzesten Weg liegenden Stadt übergehen.

Was den Rechenaufwand betrifft, so ist festzustellen, dass im konkreten Beispiel des Wegenetzes gemäß Abbildung 2.1 bei der Implementierung des ursprünglichen Bellman-Ford-Algorithmus (Listing 2.1) der Rechenaufwand identisch ist mit der Worst-Case-Komplexität des Algorithmus. Bei den $n_V = 12$ Knoten und $n_E = 18$ Kanten des betrachteten Beispiels bedeutet dies $12 \cdot 18 = 216$ Überprüfungen. Wird ein Algorithmus gemäß der Ausführungen von Moore/Braess implementiert (Listing 2.2), so müssen im Beispiel insgesamt 18 Kanten bzw. Zielknoten auf eine Wegverkürzung hin überprüft werden. Der Rechenaufwand lässt sich also für das Beispiel gemäß Abbildung 2.1 um mehr als 90 % reduzieren.

2.3 Die Dynamische Programmierung nach Bellman

Die *Dynamische Programmierung*, im Operations Research häufig auch als *Dynamische Optimierung* bezeichnet, ist ein Optimierungsverfahren aus dem Bereich der diskreten Mathematik, mit starker Anlehnung an das Teilgebiet der Graphentheorie. In den folgenden Abschnitten wird zuerst das Grundprinzip der Dynamischen Programmierung erläutert. Richard Bellman selbst spricht davon, dass man sie, von einem gewissen Standpunkt aus gesehen,

mehr als eine bestimmte Art zu denken als ein festes System von mathematischen Regeln und Vorschriften auffassen kann [1]. Der Grund für diese Aussage liegt wohl darin, dass vielfach die mit der Dynamischen Programmierung verbundene Herausforderung bei der Lösung eines Optimierungsproblems nicht in der mit der Lösungsberechnung verbundenen Mathematik bzw. Algorithmik gesehen wird, sondern vielmehr in der adäquaten Modellierung der Problemstellung, so dass die Methode der Dynamischen Programmierung überhaupt erst auf das gegebene Problem anwendbar wird [10].

In den folgenden Abschnitten wird das Grundprinzip erläutert; danach wird auf die beiden Varianten der Dynamischen Programmierung eingegangen, die *Vorwärts-Rechnung* und die *Rückwärts-Rechnung*. Beide Varianten führen prinzipiell zum selben Ergebnis. Durch die adäquate Wahl einer dieser Varianten, je nach gegebener Problemstellung, kann die Anzahl der erforderlichen Berechnungen reduziert werden. Im Bereich der Regelungstechnik hat man bisher der Rückwärts-Rechnung, dort auch als Rückwärts-Rekursion bezeichnet, die größte Beachtung geschenkt. Es wird sich jedoch zeigen, dass für die mit dem Kreuzungsmanagement verbundenen Optimierungsaufgaben die Vorwärts-Rechnung besser geeignet ist.

2.3.1 Das Grundprinzip der Dynamischen Programmierung

Grundvoraussetzung für die Anwendung der Dynamischen Programmierung ist das Vorhandensein eines mehrstufigen Entscheidungsprozesses Markov'scher Art [1]. Dies wird im Folgenden anhand eines dynamischen Systems in Zustandsraumdarstellung erläutert[3]. Ausgangspunkt dabei bildet ein zeitdiskretes, lineares und zeitinvariantes Systems der Form

$$\underline{x}_{k+1} = \underline{\Phi}\ \underline{x}_k + \underline{H}\ \underline{u}_k, \tag{2.1}$$

mit der Transitionsmatrix $\underline{\Phi}$ und der Eingangsmatrix $\underline{H}$, wie es beispielsweise in [14] eingeführt wird. Werden die Komponenten des Eingangsvektors $\underline{u}_k$ für die Dauer $\Delta t = t_{k+1} - t_k$ im Zeitintervall $[t_k, t_{k+1})$ konstant gehalten, so verhält sich das zeitdiskrete System gemäß (2.1) exakt genauso wie das entsprechende System in zeitkontinuierlicher Darstellung, wobei Δt beliebig gewählt werden kann. Das Gleichungssystem (2.1) wird verwendet, um

[3] Detailliertere Informationen zu kontinuierlichen Systemen in Zustandsraumdarstellung sowie zu der Transformation dieser Systeme in den zeitdiskreten Bereich werden in Abschnitt 3.2 gegeben.

auf Basis des Systemzustands $\underline{x}_k$ und nach Festlegung des konstanten Systemeingangs $\underline{u}_k$ zum Zeitpunkt t_k den Systemzustand $\underline{x}_{k+1}$ zum Zeitpunkt t_{k+1} zu berechnen. Der Index k kann vor diesem Hintergrund als *Entscheidungsstufe* interpretiert werden, auf der jeweils $\underline{u}_k$ festzulegen ist. Der zukünftige Zustand $\underline{x}_{k+1}$ hängt dann ausschließlich ab von dem gegenwärtigen Zustand $\underline{x}_k$ und dem gegenwärtigen Eingangsvektor $\underline{u}_k$. Diese Konstellation entspricht den Eigenschaften eines sogenannten MARKOV-Prozesses[4] 1. Ordnung, die jeder Systemdarstellung gemäß (2.1) gemein sind. Diese MARKOV-Eigenschaft kann auch folgendermaßen interpretiert werden: Der zukünftige Systemzustand hängt lediglich von der Gegenwart ab, nicht aber von der Vergangenheit bzw. seiner Vorgeschichte. Auf dieser Basis lässt sich nun leicht das Optimalitätsprinzip von BELLMAN ableiten.

Das Optimalitätsprinzip von Bellman

Optimierungsprobleme in Verbindung mit dynamischen Systemen gemäß Gleichung (2.1) erfüllen die Forderung der Dynamischen Programmierung nach dem Vorhandensein eines mehrstufigen Entscheidungsprozesses MARKOV'scher Art. Die diskreten Zeitpunkte $t_0, t_1, ..., t_k, t_{k+1}, ..., t_n$, in denen der Systemeingang $\underline{u}_k$ festgelegt und für die Dauer $\Delta t = t_{k+1} - t_k$ konstant gehalten wird, entsprechen dabei den Entscheidungsstufen. Die Folge $\underline{u}_0, \underline{u}_1, ..., \underline{u}_{n-1}$ von Entscheidungen in einem Entscheidungsprozess mit n Stufen, die auch als *Strategie* oder *Politik* bezeichnet wird, führt dazu, dass das vorliegende dynamische System nacheinander, ausgehend vom initialen Zustand $\underline{x}_0$, die Zustände $\underline{x}_1, \underline{x}_2, ..., \underline{x}_n$ annimmt, die wiederum als Punkte auf einer Trajektorie im Zustandsraum des Systems zu interpretieren sind.

Um nun überhaupt eine optimale Trajektorie bestimmen zu können, müssen sämtliche potenziell möglichen Trajektorien betrachtet und hinsichtlich eines Gütekriteriums bzw. mittels einer Gütefunktion bewertet werden. Die verwendete Gütefunktion muss dabei die Eigenschaft der sogenannten *Additivität* aufweisen, damit die Dynamische Programmierung angewendet werden kann. Additivität bedeutet hier konkret, dass die Auswirkungen der Zustandsübergänge auf die jeweilige Gütefunktion an den einzelnen Entscheidungsstufen über alle Entscheidungsstufen aufsummiert werden müssen, damit die Gesamtgüte eines Zustands auf einer gegebenen Entscheidungsstufe beurteilt werden kann. Prinzipiell entspricht dies exakt den Gegebenheiten bei der

[4] Sehr gebräuchlich ist auch die Bezeichnung MARKOV-Kette.

Berechnung *kürzester Wege* (Abschnitt 2.2). Ein Problem, das gemäß der Dynamischen Programmierung formuliert wurde, kann deshalb stets auf das Problem einer Kürzeste-Wege-Suche transformiert werden, was im Folgenden noch im Detail gezeigt werden wird.

Aus der Eigenschaft der Additivität kann ferner das Optimalitätsprinzip von Bellman direkt abgeleitet werden: Jede optimale Trajektorie $\underline{x}_0^*, \underline{x}_1^*, ..., \underline{x}_n^*$ wird durch die optimale Strategie bzw. Politik $\underline{u}_0^*, \underline{u}_1^*, ..., \underline{u}_{n-1}^*$ realisiert (vgl. Abbildung 2.3). Wählt man auf der optimalen Trajektorie einen beliebi-

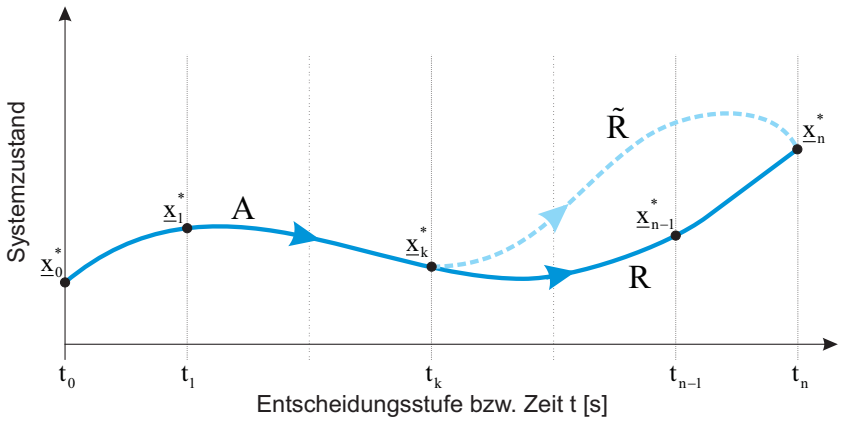

Abbildung 2.3: Herleitung des Optimalitätsprinzips nach [15]

gen Zwischenzustand $\underline{x}_k^*$ aus, so werden dadurch sowohl die optimale Trajektorie als auch die optimale Strategie in zwei Teile geteilt. Man erhält in dem Fall die Anfangsstrategie $A = \underline{u}_0^*, \underline{u}_1^*, ..., \underline{u}_{k-1}^*$ und die Reststrategie $R = \underline{u}_k^*, \underline{u}_{k+1}^*, ..., \underline{u}_{n-1}^*$. Nimmt man nun an, eine prinzipiell beliebige Reststrategie $\tilde{R}$ sei günstiger als die Reststrategie R, so folgt daraus aufgrund der oben geforderten Additivitätseigenschaft für eine Gütefunktion J die Beziehung

$$J(A) + J(\tilde{R}) \quad < \quad J(A) + J(R) \tag{2.2}$$

bzw.

$$J(A, \tilde{R}) \quad < \quad J(A, R). \tag{2.3}$$

Dies ist jedoch ein Widerspruch zu der Voraussetzung, dass (A, R) eine optimale Strategie ist. Diese Argumentationsweise gilt nicht nur für Reststrategien, sondern generell für beliebige Teilstrategien der optimalen Gesamtstrategie [15].

Man kann sich diesen Sachverhalt auch an dem anschaulichen Beispiel aus Abschnitt 2.1 verdeutlichen (Abbildung 2.1): Hat man einen optimalen bzw. kürzesten Weg J^* von der Stadt A zur Stadt B gefunden, der über die Städte X und Y führt, so muss auch der Weg zwischen den Städten X und Y optimal sein bzw. der kürzesten Verbindung dieser beiden Städte entsprechen. Wäre dies nicht der Fall, so könnte J^* verkürzt werden, indem man zwischen den Städten X und Y einen kürzeren Teilweg verwenden würde. J^* wäre in diesem Fall dann aber nicht optimal.

Letztlich lässt sich auf Basis dieser Beispiele das Optimalitätsprinzip von Bellman formulieren:

> ***Die Gesamtstrategie kann nur dann optimal sein, wenn jede Teil- bzw. Reststrategie optimal ist, ganz gleich, von welchem Zwischenzustand sie ausgeht.***

Grundlegende Voraussetzung für die Anwendung des Optimalitätsprinzips ist, wie bereits oben erwähnt, das Vorhandensein eines mehrstufigen Entscheidungsprozesses im Sinne von mehreren Entscheidungen, die nacheinander zu treffen sind. Die Formulierung eines Optimierungsproblems als Dynamische Programmierung macht genau dies, nämlich ein Optimierungsproblem als mehrstufigen Entscheidungsprozess zu formulieren. Das Problem des Auffindens der optimalen Gesamtstrategie kann dann auf Basis des Optimalitätsprinzips auf das Auffinden von optimalen Teilstrategien reduziert werden. Dadurch wird die Komplexität des Optimierungsproblems stark reduziert. In [35] wird dieses Prinzip auch als *"Teile und herrsche"* bezeichnet, das für eine Vielzahl von Algorithmen in der Informatik die Basis bildet.

2.3.2 Die Varianten der Dynamischen Programmierung

Die Varianten und die Systematik der Dynamischen Programmierung werden im Folgenden an einem einfachen System erster Ordnung veranschaulicht[5], das durch die folgende Systemgleichung gegeben ist:

$$x_{k+1} = x_k + u_k, \tag{2.4}$$

mit

[5] Das Beispiel basiert auf dem Wegenetz und den Kostenbewertungen, mit denen Schneider und Mikolcic [34] die Dynamische Programmierung einführen.

$$
\begin{array}{lll}
u_k = \{-1, 0, 1\}, & \text{falls} & |x_k| \leq 2 \\
u_k = \{-1, 0\}, & \text{falls} & x_k = 3 \\
u_k = \{0, 1\}, & \text{falls} & x_k = -3.
\end{array}
$$

Das System wird über $n = 6$ Entscheidungsstufen hinweg betrachtet. Jede Entscheidungsstufe k ($k = 0, ..., n-1$) entspricht dabei einem diskreten Zeitpunkt t_k, an dem der Systemeingang u_k geändert wird. Die Entscheidung der jeweiligen Stufe besteht in diesem Fall somit in der Wahl des Systemeingangs u_k. Je nachdem, wie u_k gewählt wird, wird sich auch der Systemzustand x_{k+1} beim Erreichen der Entscheidungsstufe $k+1$ ändern. Die Zustandsübergänge auf den einzelnen Entscheidungsstufen werden mit einer Güte- bzw. Kostenfunktion bewertet. Auch hier gilt wieder die für die Anwendung der Dynamischen Programmierung erforderliche Additivität der Güte- bzw. Kostenwerte. In Abbildung 2.4 wird der beschriebene Sachverhalt noch einmal graphisch veranschaulicht (vgl. [34]). Auf der Abszisse sind die Entscheidungsstufen aufgetragen, an denen der Systemeingang u des Systems vorzugeben ist. Zwischen zwei aufeinanderfolgenden Stufen können Zustandsübergänge stattfinden. Die Ordinate definiert den Systemzustand $x \in \{-3, -2, ..., 3\}$. Die einzelnen diskreten Zustände werden in der Abbildung durch Punkte symbolisiert, die im Weiteren als *Knoten* bezeichnet werden. Die jeweils möglichen Zustandsübergänge werden durch Geraden bzw. Verbindungslinien zwischen zwei Knoten dargestellt, die im Weiteren als *Kanten* bezeichnet werden. Die Zahlenwerte auf den Kanten entsprechen den Kosten für die Zustandsübergänge bzw. dem Einfluss auf die Kostenbewertung des zugehörigen Weges durch die Knotenanordnung, wobei im Falle eines dynamischen Systems gemäß Gleichung (2.1) der Begriff *Trajektorie* synonym für den Begriff *Weg* verwendet werden kann. Die Abbildung 2.4 kann auch als topologische Repräsentation eines Wegenetzes, wie desjenigen aus Abbildung 2.1, interpretiert werden. Die Kosten- bzw. Gütewerte entsprechen in diesem Fall den Weglängen bzw. Entfernungen von einem Knoten der Entscheidungsstufe k zu einem Knoten der Stufe $k+1$. In diesem Fall besteht die Aufgabe darin, einen kürzesten Weg von Knoten A zu Knoten B durch das Netzwerk zu bestimmen. Im Falle der Interpretation der Abbildung 2.4 als zeitabhängige Darstellung des Zustandes eines dynamischen Systems besteht die resultierende Aufgabe in dem Auffinden einer optimalen Trajektorie, die das System von dem Zustand A in den Zustand B überführt. Für die Lösung dieser Aufgabe mittels der Dynamischen Programmierung bestehen prinzipiell zwei Möglichkeiten: die sogenannte Vorwärts-Rechnung und die Rückwärts-Rechnung. Beide Möglichkeiten führen selbstverständlich zu demselben Ergebnis; allerdings kann

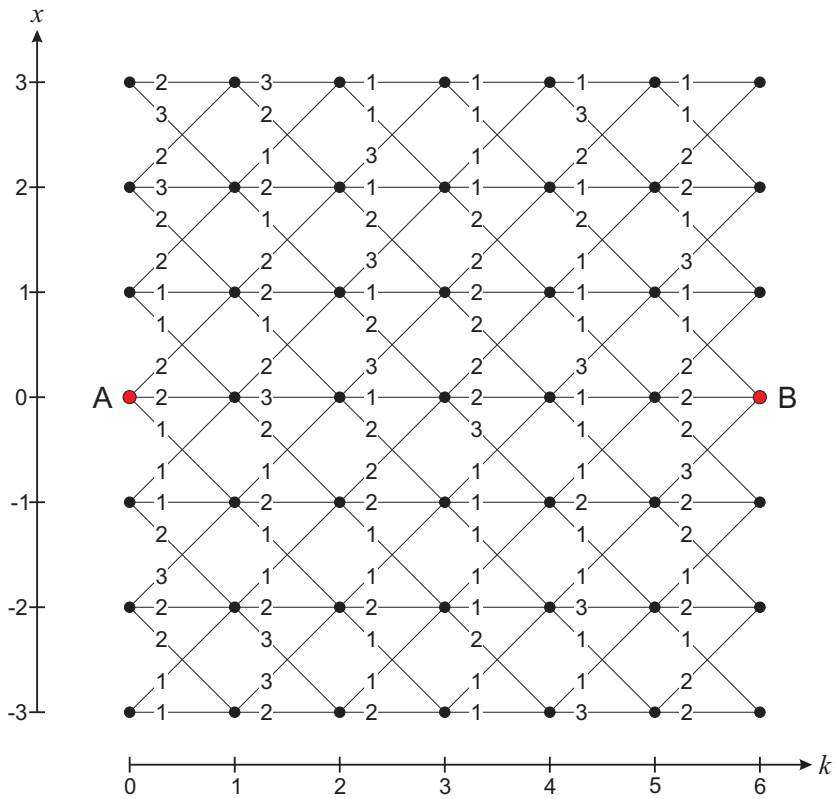

Abbildung 2.4: *"Wegenetz"* gemäß [34] als Ausgangssituation

je nach Konstellation des Optimierungsproblems die eine oder die andere Möglichkeit mit einer signifikanten Reduktion des Berechnungsaufwands verbunden sein. Im Folgenden wird zuerst die Rückwärts-Rechnung am Beispiel des zuvor formulierten Optimierungsproblems vorgestellt, da diese Variante im Bereich der Regelungstechnik bisher die breiteste Anwendung gefunden hat. Darauf aufbauend, wird dann zum Vergleich die Vorwärts-Rechnung an dem gleichen Beispiel demonstriert.

Die Rückwärts-Rechnung

Innerhalb der Regelungstechnik wurde bisher in fast allen oder sogar allen Fällen die Rückwärts-Rechnung bzw. die Rückwärts-Rekursion angewendet, da die zugrunde liegende Problemstellung in der Regel folgender Art war:

Gegeben ist ein dynamisches System gemäß Gleichung (2.1) bzw. (2.4). Das System befindet sich zu Beginn in dem prinzipiell beliebigen Anfangszustand $\underline{x}(t_0 = 0)$, im Folgenden als $\underline{x}_0$ bezeichnet (Zustand A im Beispiel), und soll auf optimale Weise (Abschnitt 2.3.1) in einen gewünschten Endzustand $\underline{x}_E$ (Zustand B im Beispiel) überführt werden.

In diesem Fall bietet sich die Rückwärts-Rechnung an, die in dem gewünschten Endzustand (bzw. -knoten) startet und dann rekursiv, von hinten nach vorne schreitend, sämtliche möglichen Anfangszustände (bzw. -knoten) erreicht. Der komplette erreichbare Zustandsraum wird unter Ausnutzung der Additivitätseigenschaft der Kosten- bzw. Gütefunktion sukzessive, in der Zukunft beginnend und in der Gegenwart endend, exploriert. Letztlich erhält man optimale Trajektorien, die auch als *kürzeste Wege* interpretiert werden können, von sämtlichen möglichen Anfangsknoten zum gewünschten Endknoten.

Bezogen auf das obige Beispiel bedeutet dies, dass man, beginnend mit dem gewünschten Endknoten B ($x = 0$ auf der Stufe $k = 6$ oder kürzer: $x_6 = 0$), prüft, wie man von den Knoten der jeweils vorherigen Entscheidungsstufe ($k = n - 1, ..., 0$) auf optimalem Wege zu dem Endknoten B gelangt. Sämtliche Entscheidungsstufen werden dabei sukzessive von hinten nach vorn abgearbeitet. Für die Entscheidungsstufe bei $k = 5$ sind dies die drei Knoten $x_5 = 1$, $x_5 = 0$ und $x_5 = -1$, denn nur von ihnen aus lässt sich der gewünschte Endknoten B erreichen (vgl. Abbildung 2.5). Da es keine Wahlmöglichkeit gibt, wie von diesen drei Knoten aus der Endknoten B erreicht werden kann, sind alle drei Wege von diesen Knoten zum Endknoten B automatisch optimale Wege. Die kumulierten Kosten für die jeweiligen optimalen und zum Endknoten B führenden Wege sind in Abbildung 2.5 über dem zugehörigen Anfangsknoten des Weges in einem Kreis notiert und hängen von den Kostenwerten der jeweiligen Kanten ab. Optimale Knotenübergänge zwischen zwei aufeinander folgenden Entscheidungsstufen sind weiterhin durch Pfeile dargestellt. Betrachtet man nun den Knoten $x_4 = 1$, so gibt es genau zwei Möglichkeiten, um den Endknoten B zu erreichen. Eine Möglichkeit führt über den Knoten $x_5 = 1$, die andere Möglichkeit über den Knoten $x_5 = 0$. Um vom Knoten $x_4 = 1$ zum Knoten $x_5 = 0$ zu gelangen, fallen Kosten in Höhe von einer Kosteneinheit an. Um von dort dann in den Endknoten B zu gelangen, fallen weitere Kosten in Höhe von 2 Kosteneinheiten an. Der gesamte Weg wird damit also mit $1 + 2 = 3$ Kosteneinheiten bewertet. Bei der anderen Variante, über $x_5 = 1$, fallen Kosten in Höhe von jeweils einer Kosteneinheit an, um von dem Knoten $x_4 = 1$ in den Knoten $x_5 = 1$ und

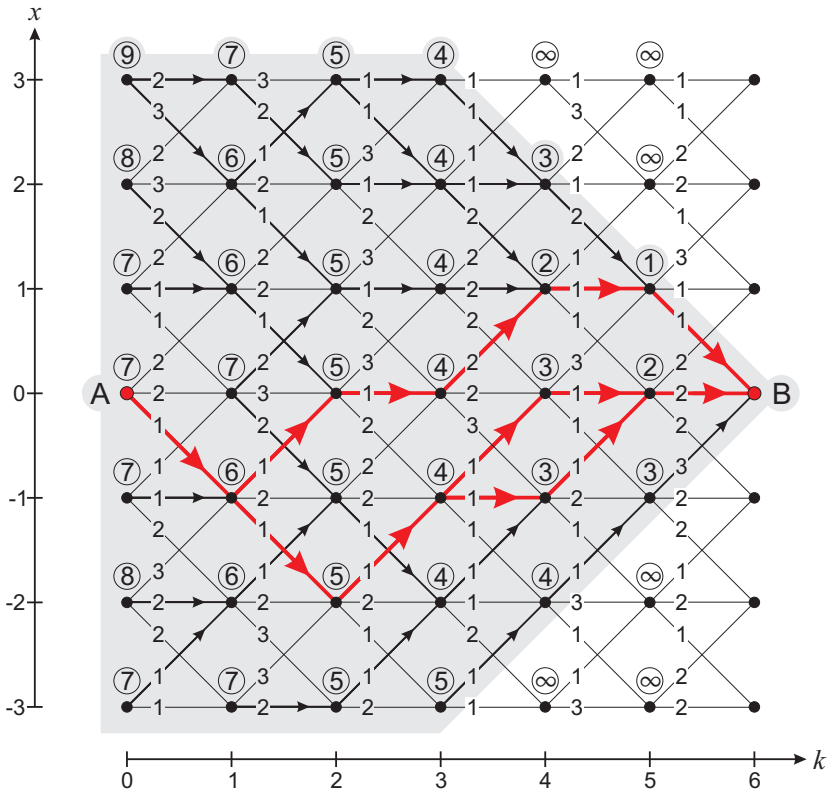

Abbildung 2.5: Rückwärts-Rechnung mit festem Endpunkt

von dort dann in den Endknoten B zu gelangen. Für den gesamten Weg $x_4 = 1 \rightarrow x_5 = 1 \rightarrow B$ fallen somit Kosten in Höhe von $1 + 1 = 2$ Kosteneinheiten an. Dieser Weg ist also der günstigere der beiden Alternativen. Es werden deshalb ein Pfeil vom Knoten $x_4 = 1$ zum Knoten $x_5 = 1$ gezogen und die Gesamtkosten des optimalen Weges mit Anfangsknoten $x_4 = 1$ und Endknoten B über dem Anfangsknoten in einem Kreis notiert. Die Werte in den Kreisen entsprechen also den Gesamtkosten der optimalen Wege, die im zugehörigen Knoten beginnen und im Endknoten B enden. Die Dynamik des Systems gemäß Gleichung (2.4) folgt somit der optimalen Trajektorie, wenn sie den Pfeilen folgt. Die beschriebene Auswertungssystematik wird für sämtliche Knoten einer Entscheidungsstufe durchgeführt; erst dann wird zur vorausgehenden Entscheidungsstufe übergegangen.

Sehr interessant und bemerkenswert ist dabei das Potenzial für eine Parallel-

verarbeitung, denn die Auswertungen für die Knoten bzw. diskreten Zustände einer Entscheidungsstufe sind völlig unabhängig voneinander: Kann das betrachtete System auf einer beliebigen Entscheidungsstufe n verschiedene Zustände annehmen, so können die Berechnungen für die jeweilige Entscheidungsstufe prinzipiell auf n Rechner verteilt und parallel durchgeführt werden. Vor dem Übergang zur nächsten Entscheidungsstufe müssen allerdings die Ergebnisse der parallelen Berechnungen *synchronisiert* werden.

Zu einem Knoten einer Entscheidungsstufe k können mehrere Pfeile aus der vorherigen Entscheidungsstufe $k-1$ führen. Dies ist der Fall, wenn es mehrere Knoten auf der Entscheidungsstufe $k-1$ gibt, von denen aus der Übergang zum betrachteten Knoten auf der Stufe k gleich bewertet wird. Es können auch mehrere Pfeile aus einem Knoten weg führen. In diesem Fall gibt es mehrere optimale Wege bzw. mehrere Wege mit identischer Kostenbewertung, die von dem betrachteten Knoten zum gewünschten Endknoten B führen. Ein Beispiel dafür ist der Knoten $x_3 = -1$ in Abbildung 2.5. Für diesen Knoten existieren zwei Wege zum Endknoten B, die beide die optimale Kostenbewertung von 4 Kosteneinheiten besitzen. Der eine Weg führt über den Knoten $x_4 = 0$, der andere über den Knoten $x_4 = -1$.

Sobald man die Auswertungen im Rahmen der Rückwärts-Rechnung bis zur Stufe $k = 0$ nach obigem Schema durchgeführt hat, kennt man auch die optimalen Wege für sämtliche Knoten dieser Stufe. Das System gemäß Gleichung (2.4) kann sich dann also in einem beliebigen Anfangszustand befinden, und völlig unabhängig von diesem Zustand ist bereits die optimale Trajektorie bzw. Strategie bekannt. Man muss lediglich dem Pfeil folgen, der in dem jeweiligen Knoten bzw. Zustand beginnt. Führt man dies für den Anfangszustand A durch, so ergeben sich die folgenden drei optimalen Trajektorien, die in Abbildung 2.5 jeweils durch die dickeren Pfeile dargestellt sind:

	x_0	$\rightarrow$	x_1	$\rightarrow$	x_2	$\rightarrow$	x_3	$\rightarrow$	x_4	$\rightarrow$	x_5	$\rightarrow$	x_6
(1)	0		-1		0		0		1		1		0
(2)	0		-1		-2		-1		0		0		0
(3)	0		-1		-2		-1		-1		0		0

Die Besonderheit der Rückwärts-Rechnung mit festem Endknoten und variablem Anfangsknoten besteht darin, dass optimale Strategien und somit auch kürzeste Wege von sämtlichen Anfangsknoten zu dem festen Endknoten bestimmt werden. Bei einem dynamischen System gemäß Gleichung (2.1)

wird der Zustandsraum aus der Zukunft kommend bzw. rückwärts schreitend exploriert. Dadurch entstehen in der Regel in der Nähe des Endzustands Bereiche des Zustandsraumes, von denen aus der Endzustand nicht mehr erreichbar ist. Diese Zustände werden daher mit ∞ bewertet. In Abbildung 2.5 ist der erreichbare Zustandsraum grau hinterlegt. Die Rückwärts-Rechnung kann anstatt mit festem Endknoten auch mit variablem Endknoten durchgeführt werden. In diesem Fall wird bei einem dynamischen System der gesamte Zustandsraum untersucht, mit dem Ergebnis, dass optimale Trajektorien von jedem beliebigen Anfangszustand zu jedem beliebigen Endzustand bekannt sind.

Die Vorwärts-Rechnung

Die Vorwärts-Rechnung mit festem Anfangsknoten und variablem Endknoten geht von einem festen Anfangsknoten aus, der nach n Entscheidungen in einen beliebigen Endknoten überführt werden soll. Das Vorgehen wird anhand von Abbildung 2.6 erläutert. Beginnend mit dem Anfangsknoten A auf Stufe $k = 1$, wird analysiert, über welche Kanten die jeweiligen Knoten auf der nächsten Stufe $k + 1$, jeweils vom Anfangsknoten A aus, in optimaler Weise erreicht werden können. Für die Stufe $k + 1$ sind dies die drei Knoten $x_1 = 1$, $x_1 = 0$ und $x_1 = -1$. Diese Knoten können unmittelbar und nur auf einem Wege vom Anfangsknoten A aus erreicht werden. Es handelt sich also um optimale Wege, und folglich ist ein Pfeil von A zu diesen drei Knoten zu ziehen. Bei den Knotenübergängen von der Stufe $k = 1$ zur Stufe $k = 2$ sieht dies anders aus. Beispielsweise kann der Knoten $x_2 = 0$ über alle drei von A aus erreichbaren Knoten x_1 der vorherigen Stufe $k = 1$ erreicht werden. Gelangt man von A aus über den Knoten $x_1 = 1$ in den Knoten $x_2 = 0$, so wird dieser Weg mit $2+1 = 3$ Kosteneinheiten bewertet. Im Falle des Weges über den Knoten $x_1 = 0$ sind es $2 + 3 = 5$ Kosteneinheiten, und bei dem Weg über den Knoten $x_1 = -1$ sind es $1+1 = 2$ Kosteneinheiten. Dies sind alle existierenden Möglichkeiten, wie man, vom Anfangsknoten A ausgehend, zum Knoten $x_2 = 0$ gelangen kann. Die optimale Strategie führt somit über den Knoten $x_1 = -1$, da hier die geringsten Kosten in Höhe von 2 Kosteneinheiten anfallen. Es wird somit ein Pfeil vom Knoten $x_1 = -1$ zum Knoten $x_2 = 0$ gezogen. Weiterhin werden über dem Knoten $x_2 = 0$ die Gesamtkosten in einem Kreis notiert, die anfallen, wenn man dem optimalen Weg von A zu diesem Knoten folgt. Dieses Vorgehen wird auf sämtliche Knoten der Stufe $k = 2$ angewendet. Danach wird zu den Knoten der nächsten Stufe übergangen und immer so

weiter, bis zuletzt die Stufe $k = 6$ erreicht ist. Die Kosten über den Knoten

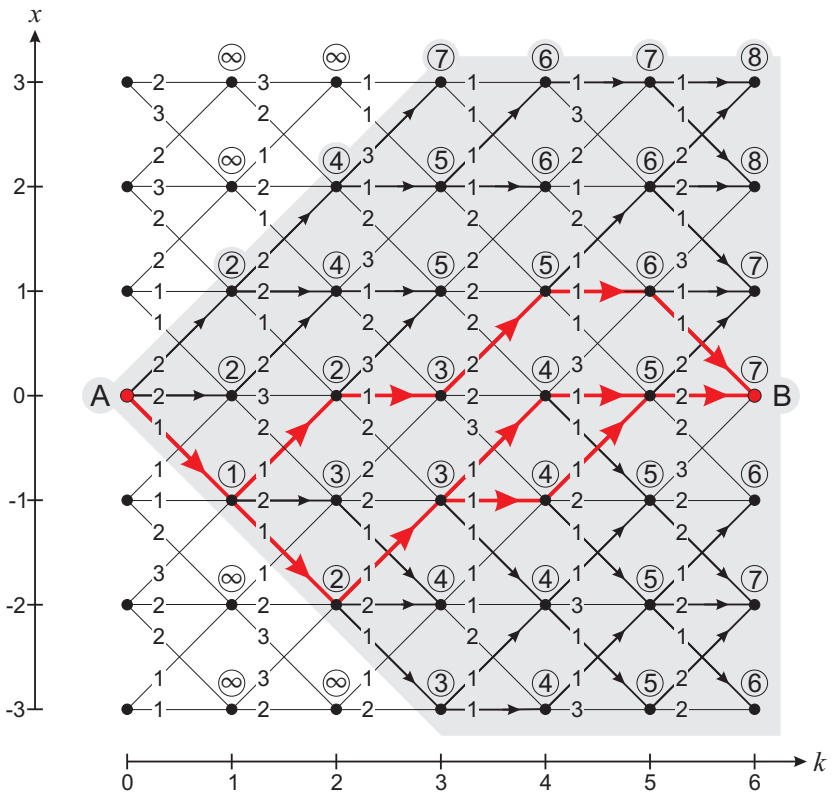

Abbildung 2.6: Vorwärts-Rechnung mit festem Endpunkt

der letzten Stufe $k = 6$ entsprechen den Gesamtkosten der kürzesten Wege, die entstehen, wenn man der optimalen Strategie, vom Knoten A ausgehend, zu dem jeweils betrachteten Endknoten x_6 folgt.

Möchte man nun nach Durchführung der Vorwärts-Rechnung die optimale Strategie bzw. den kürzesten Weg vom Knoten A zum Knoten B bestimmen, so startet man im Knoten B ($k = 6$) und verfolgt dann von dort aus die Pfeile zurück, die jeweils von einem Knoten der vorherigen Stufe $k-1$ in den jeweils betrachteten Knoten der Stufe k führen. In Abbildung 2.6 sind diese Pfeile dicker dargestellt. Man erkennt schließlich, dass man durch dieses Vorgehen dieselben Wege erhält, die sich bei der Rückwärts-Rechnung ergeben.

Auch bei der Vorwärts-Rechnung lassen sich die Berechnungen je Entschei-

dungsstufe massiv parallelisieren. Eingehende Pfeile und Gesamtkosten von Wegen zu Knoten der Stufe k sind lediglich abhängig von den Gesamtkosten und Knotenübergängen der Stufe $k-1$, nicht jedoch von Größen derselben Stufe k. Auch hier können somit die Berechnungen für n Knoten (bzw. diskrete Zustände des Zustandsraumes) je Entscheidungsstufe parallel auf n verschiedenen Rechnern durchgeführt werden.

Die Vorwärts-Rechnung mit festem Anfangsknoten und variablem Endknoten kann ebenfalls auch als Vorwärts-Rechnung mit variablem Anfangsknoten und variablem Endknoten durchgeführt werden. Die Ergebnisse und der Rechenaufwand wären in diesem Fall identisch mit den Ergebnissen und dem Rechenaufwand bei der Rückwärts-Rechnung mit variablem Endknoten und variablem Anfangsknoten. Geht man jedoch von der beschriebenen Variante mit festem Anfangsknoten aus, so sieht man, dass im Falle eines Systems gemäß Gleichung (2.1) der komplette Zustandsraum erst nach und nach exploriert wird. Je näher sich die Entscheidungsstufe an dem festen Anfangszustand befindet, desto mehr Zustände derselben Entscheidungsstufe sind in der Regel nicht vom Anfangszustand aus erreichbar. Dies kann je nach Problemstellung mit einer mehr oder weniger signifikanten Reduktion der Anzahl von Berechnungen verbunden sein. In Abbildung 2.6 sind die erreichbaren Knoten bzw. Bereiche des Zustandsraumes wieder grau hinterlegt dargestellt.

Im weiteren Verlauf dieser Arbeit wird deutlich werden, dass die Vorwärts-Rechnung mit festem Anfangs- und variablem Endpunkt prädestiniert ist für die Anwendung auf die Problemstellungen des autonomen Kreuzungsmanagements.

Weiterhin sollte an dieser Stelle bereits klar geworden sein, dass es sich bei der Dynamischen Programmierung prinzipiell um nichts anderes handelt als um eine Methodik für die Suche kürzester Wege auf azyklischen gerichteten Graphen bzw. auf Netzwerken. Das einführende Beispiel gemäß Abbildung 2.1 in Abschnitt 2.1 kann somit auch gemäß der Dynamischen Programmierung formuliert werden. In diesem Fall entsprechen die Entscheidungsstufen den Wahlmöglichkeiten, die man in einer beliebigen Stadt hat, bezüglich der als nächstes zu erreichenden Stadt. Befindet man sich beispielsweise in der Stadt **Stuttgart**, so ist diese Stadt einer Entscheidungsstufe zuzuordnen, auf der man für diese Stadt bzw. für den zugehörigen Knoten die Entscheidungsmöglichkeiten hat, als nächstes nach **Wiesbaden**, **Frankfurt** oder **Erfurt** zu fahren und diese Städte dann auf der nächsten Entscheidungsstufe zu erreichen. Abbildung 2.7 stellt dies für sämtliche Städte des einführenden Beispiels dar und zeigt

weiterhin, wie der Graph gemäß Abbildung 2.1 umstrukturiert werden kann, um den Zusammenhang mit der Dynamischen Programmierung offensichtlich zu machen. Die Anwendung der Dynamischen Programmierung führt dann

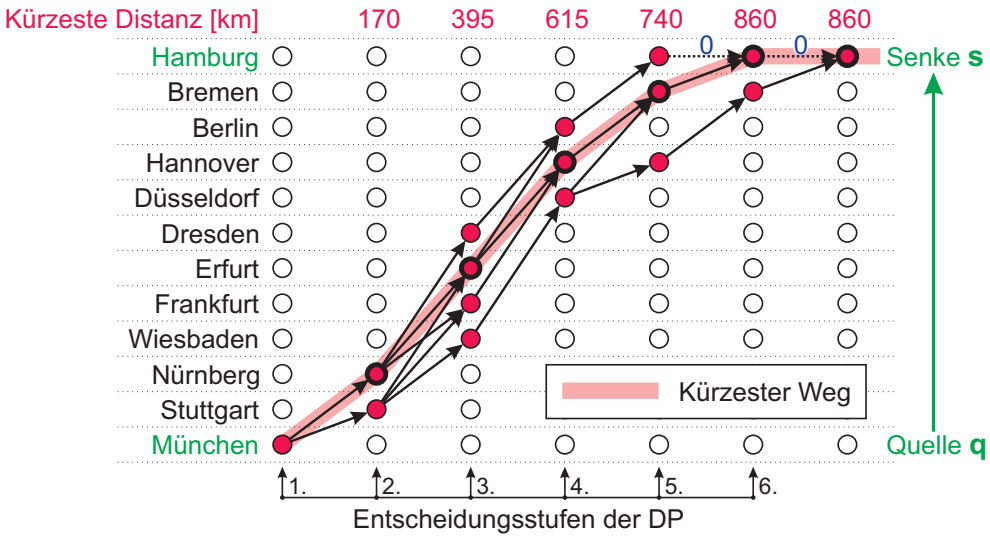

Abbildung 2.7: Strukturierung des Wegenetzes gemäß Abbildung 2.1 nach der Systematik der Dynamischen Programmierung

zu denselben Ergebnissen, die bereits in Abbildung 2.2 dargestellt wurden. In diesem Fall ist die Dynamische Programmierung als Vorwärts-Rechnung mit festem Anfangsknoten und variablem Endknoten aufzufassen. Das Ziel **Hamburg** ist zwar fest, es kann aber nach unterschiedlichen Entscheidungsstufen erreicht werden. Im dargestellten Beispiel existieren prinzipiell 7 Endknoten, die der Stadt **Hamburg** zuzuordnen sind. Von diesen Knoten sind allerdings nur 3 Knoten erreichbar, nämlich nach der 4., 5. und 6. Entscheidungsstufe.

2.3.3 Vor- und Nachteile der Dynamischen Programmierung

Im Falle eines dynamischen Systems gemäß (2.1) offenbaren sich die Nachteile der Dynamischen Programmierung in der Tatsache, dass der Rechenaufwand mit den Dimensionen von Zustands- und Eingangsvektor exponentiell

ansteigt. Das Beispielsystem gemäß Gleichung (2.4) weist nur eine Zustandsvariable x_k sowie einen skalaren Systemeingang u_k auf. Weiterhin wurden die Randbedingungen gesetzt, dass x_k nur insgesamt $a = 7$ diskrete Zustände und u_k nur $b = 3$ diskrete Werte annehmen dürfen. Da zusätzlich noch für $|x_k| = 3$ die Werte von u_k auf zwei Möglichkeiten begrenzt werden, ergeben sich schließlich je Entscheidungsstufe $3 \cdot 7 - 2 = 19$ Wege bzw. theoretische Möglichkeiten für Zustandsübergänge, die analysiert werden müssen. Bei $n = 6$ Entscheidungsstufen und sowohl variablem Anfangs- als auch variablem Endzustand ergeben sich somit insgesamt 114 mögliche Zustandsübergänge.

Stellt man sich nun vor, das zu untersuchende System habe die Ordnung 2 bzw. der Systemzustand sei durch den Zustandsvektor $\underline{x} = [x_1, x_2]^T$ definiert, so wird die Ordinate (x-Achse) aus dem Beispielsystem zu einer Ebene mit den Achsen x_1 und x_2. Wenn nun beide Zustandsvariablen gleichermaßen mit $a = 7$ diskretisiert werden, wie zuvor die Zustandsvariable x, so ergibt sich je Ebene bereits ein Raster mit $a^2 = 49$ diskreten Zuständen. Gleiches gilt bei entsprechender Dimensionserweiterung des Systemeingangs. Allgemein kann der Rechenaufwand bzw. die Komplexität O mit der Formel

$$O = n \cdot a^p \cdot b^q \tag{2.5}$$

abgeschätzt werden, wobei die folgenden Definitionen verwendet werden:

n : Anzahl der Entscheidungsstufen,
a : Anzahl der Diskretisierungswerte je Zustandsgröße,
p : Systemordnung bzw. Anzahl der Zustandsgrößen,
b : Anzahl der Diskretisierungswerte je Eingangsgröße,
q : Anzahl der Eingangsgrößen.

Einen Eindruck über das Anwachsen der erforderlichen Berechnungen im Rahmen der Dynamischen Programmierung vermittelt Tabelle 2.1. Der Tabelle liegt die Annahme zugrunde, dass die Dynamische Programmierung 10 Entscheidungsstufen umfasst und sowohl jede Eingangs- als auch Zustandsgröße jeweils 10 diskrete Werte annehmen kann ($n = a = b = 10$). Bei einem System 4. Ordnung und 2 Eingangsgrößen müssen dann bereits 10 Millionen unterschiedliche Zustandsübergänge analysiert werden.

Es gibt jedoch mehrere Ansätze, die je nach Problemstellung dafür geeignet sind, die Anzahl der Berechnungen signifikant zu reduzieren. Zu diesen Ansätzen zählen beispielsweise die Folgenden:

Dimension u \ Dimension x	1	2	3	4	5
1	100	1.000	10.000	100.000	1.000.000
2	1.000	10.000	100.000	1.000.000	10.000.000
3	10.000	100.000	1.000.000	10.000.000	100.000.000

Tabelle 2.1: Beispiele für Rechenaufwand

- Da die Systemordnung eine wesentliche Rolle beim Anwachsen der Berechnungen spielt, sollte sie prinzipiell so gering wie möglich angesetzt werden. Bei der Systembeschreibung mittels physikalischer Beziehungen kann allerdings häufig der Einfluss einzelner Zustandsgrößen im Hinblick auf die Modellgenauigkeit a priori nicht exakt eingeschätzt werden. Die Systemanalyse im Rahmen einer Ordnungsreduktion [16] gibt hier Aufschluss, inwieweit eine *kompaktere* Gestaltung des (physikalischen) Modells bei Einhaltung der geforderten Modellgenauigkeit realisierbar ist.

- Unter Umständen lässt sich der Zustandsraum bzw. der (zulässige) Bereich des Zustandsraums, in dem die Dynamische Programmierung sinnvolle Ergebnisse liefert, a priori begrenzen. Dies ist beispielsweise beim Kreuzungsmanagement der Fall.

- Bei *stetigen* Optimierungsproblemen kann die sogenannte *Iterative Dynamische Programmierung* angewendet werden, die auf der wiederholten Ausführung der Dynamischen Programmierung mit unterschiedlichen Diskretisierungen beruht. Anfangs wird die Dynamische Programmierung mit einem groben Raster (bzgl. n, a und b) durchgeführt. Die erforderlichen Berechnungen können schnell erfolgen, jedoch ist die aus ihnen resultierende Trajektorie entsprechend ungenau, gibt aber Hinweise auf die ungefähre Lage der gesuchten Trajektorie. Um die gefunden Trajektorie wird nun eine "Schlauch" gelegt, in dem die Dynamische Programmierung erneut, aber nun mit einem feineren Raster durchgeführt wird. Dieser Vorgang wird so lange wiederholt, bis eine (optimale) Trajektorie mit ausreichender Genauigkeit gefunden ist. Da es sich beim Kreuzungsmanagement um ein *unstetiges* Optimierungsproblem handelt, kann diese Methode dort nicht angewendet werden. Die Uns-

tetigkeiten resultieren aus der Verteilung der von anderen Fahrzeugen besetzten Knoten im Kreuzungsbereich, was in den folgenden Kapiteln noch deutlich werden wird.

- Die Dynamische Programmierung bietet gute Möglichkeiten für eine Parallelverarbeitung bzw. für die parallele Ausführung der Berechnungen auf mehreren Recheneinheiten (vgl. Abschnitt 2.3.2). Dies reduziert zwar nicht die Anzahl der Berechnungen, bietet aber das Potenzial für eine signifikante Verkürzung der Berechnungszeit. Die Option der Parallelisierung ist auch beim Kreuzungsmanagement gegeben.

- Eine weitere Reduktion der Berechnungen kann durch den Einsatz von Heuristiken erreicht werden. Dabei handelt es sich im Prinzip um sinnvolle Schätzungen von Bereichen, innerhalb derer die optimale Trajektorie zu vermuten ist. Da mit Schätzungen gearbeitet wird, gibt es dann allerdings keine Garantie mehr dafür, dass tatsächlich die optimale Trajektorie gefunden wird. In der Regel kann eine zulässige Lösung innerhalb kurzer Zeit bestimmt werden, die allerdings nicht unbedingt der optimalen Lösung entspricht. Auch diese Möglichkeit für die Reduktion der Berechnungen kann im Kreuzungsmanagement genutzt werden.

Ein weiterer Punkt, der bereits bei der Einführung der Iterativen Dynamischen Programmierung indirekt angesprochen wurde, betrifft die Lösung kontinuierlicher Optimierungsprobleme. Kontinuierliche Probleme müssen bei Anwendung der Dynamischen Programmierung zuvor diskretisiert werden. Je nach Wahl der Diskretisierung wird dann die optimale Lösung des diskretisierten Problems von der tatsächlich optimalen Lösung, also von der Lösung des kontinuierlichen Problems, abweichen. Mit feiner werdender Diskretisierungsschrittweite sinkt diese Abweichung, gleichzeitig erhöht sich allerdings der Rechenaufwand. Hier besteht ein Zielgrößenkonflikt, der einen auf die jeweilige Problemstellung zugeschnittenen Kompromiss erforderlich macht. Bei stetigen Problemen kann die Iterative Dynamische Programmierung diesen Konflikt entschärfen.

Nachdem die Nachteile der Dynamischen Programmierung dargelegt und einige Möglichkeiten aufgezeigt wurden, wie man diesem entgegenwirken kann, wird nun auf die wesentlichen Vorteile eingegangen:

Echtzeitfähigkeit

Der maximale Zeitbedarf T_{DP} der Dynamischen Programmierung kann prinzipiell beliebig exakt bestimmt werden, da durch die Formulierung des Optimierungsproblems (Systemordnung, Entscheidungsstufen, Diskretisierung etc.) schon im Vorfeld feststeht, wie viele mögliche Zustandsübergänge maximal analysiert werden müssen. Nach T_{DP} Zeiteinheiten steht fest, ob es eine zulässige Lösung für das Problem gibt und, falls ja, wie sie aussieht. Weiterhin lässt sich T_{DP} beeinflussen, beispielsweise durch die Wahl der Diskretisierung oder durch die oben bereits aufgeführten Maßnahmen. Die Dynamische Programmierung kann somit in gewissen Grenzen an die zur Verfügung stehenden Rechenkapazitäten adaptiert werden, um Optimierungsaufgaben in einer vorgeschriebenen Zeit zu lösen. Vor diesem Hintergrund ist sie für den Einsatz in Verbindung mit Echtzeitsystemen ideal geeignet. Optimierungsmethoden der nichtlinearen, kontinuierlichen und beschränkten Optimierung wie beispielsweise Gradienten- und Quasi-Newton-Verfahren haben unter diesem Aspekt deutliche Nachteile, denn es lässt sich in der Regel nicht a priori sagen, wann bzw. ob das gewählte Verfahren konvergiert!

Auffinden des globalen Optimums wird garantiert

Sofern die Dynamische Programmierung eine Lösung für das jeweilige Problem ermitteln konnte, ist diese Lösung automatisch auch die optimale Lösung für die gegebene Konstellation des Optimierungsproblems. Bei kontinuierlichen Problemen ist allerdings eine adäquate Diskretisierung Voraussetzung dafür, dass das gefundene Optimum des diskretisierten Systems auch in unmittelbarer Nähe des (tatsächlichen) Optimums des kontinuierlichen Systems liegt. Der Grund für das Auffinden des globalen Optimums liegt darin, dass die Dynamische Programmierung den gesamten zulässigen Zustandsraum systematisch und sukzessive exploriert und analysiert. Bei vielen Verfahren aus dem Bereich der nichtlinearen Optimierung kann, falls das Verfahren gegen eine Lösung konvergiert, häufig nicht gesagt werden, ob die gefundene Lösung tatsächlich auch dem globalen Optimum entspricht oder ob es sich nur um ein lokales Minimum bzw. Maximum handelt.

Gute Eignung für unstetige Probleme

Viele Optimierungsverfahren, insbesondere Verfahren, die mit Ableitungen bzw. Gradienten der Zielfunktionen arbeiten, reagieren äußerst empfindlich auf Unstetigkeiten in den Verläufen der Zielfunktionen. Bei der Optimierung dynamischer Systeme kann dies der Fall sein, wenn bestimmte Bereiche des Zustandsraumes auf einer Entscheidungsstufe nicht erlaubt sind und sich dieser Sachverhalt nicht durch einen direkten funktionalen Zusammenhang beschreiben lässt. Das Kreuzungsmanagement ist dafür ein Beispiel, da durch andere Fahrzeuge belegte Knoten im Zustandsraum des aktuell zu optimierenden Fahrzeugs als unzulässige Zustände auftreten, durch die keine Trajektorie führen darf. Dies wird in Kapitel 5 noch ausführlich erläutert. Für die Dynamische Programmierung stellt dies allerdings kein Problem dar, da ohnehin der gesamte zulässige Zustandsraum sukzessive analysiert wird. Unstetigkeiten in Form von einzelnen, *stochastisch verteilten*, unzulässigen Zuständen in einem prinzipiell zulässigen Bereich des Zustandsraums können einfach berücksichtigt werden, indem man die zugehörigen Zustandsübergänge schlichtweg ignoriert.

Abschließend lässt sich noch anmerken, dass es prinzipiell eine Vielzahl unterschiedlicher Optimierungsverfahren gibt, die alle ihre Vor- und Nachteile besitzen und für bestimmte Optimierungsprobleme jeweils besser geeignet sind als die anderen Verfahren. Die wesentlichen Vorteile der Dynamischen Programmierung wurden ausreichend dargelegt, und es wird im weiteren Verlauf dieser Arbeit noch deutlich werden, dass sie für die Lösung der Optimierungsprobleme im Rahmen des Kreuzungsmanagements in hervorragender Weise geeignet ist, insbesondere im Hinblick auf die Echtzeitanforderungen.

3 Modellierung als Netzwerkflussproblem

Die dem Kreuzungsmanagement inhärente Problemstellung kann leicht zu einem Netzwerkflussproblem abstrahiert werden. Die Fahrzeuge im Kreuzungsbereich bewegen sich auf Fahrspuren[1], die einander schneiden. Aus diesen Schnittbeziehungen resultieren Positionen bzw. Zonen innerhalb der Kreuzungsebene (xy-Ebene), die zu mehreren Fahrspuren gleichzeitig gehören und im Folgenden als Schnittmengen der Fahrspuren bezeichnet werden. Die Elemente der Schnittmengen entstehen durch einander direkt schneidende bzw. kreuzende Fahrspuren, wie in Abbildung 3.1 dargestellt, oder aber durch die Tatsache, dass Fahrspuren im Ein- und Ausfahrtsbereich der Kreuzung direkt übereinander liegen. Beispielsweise teilen sich bei einer Kreuzung gemäß Ab-

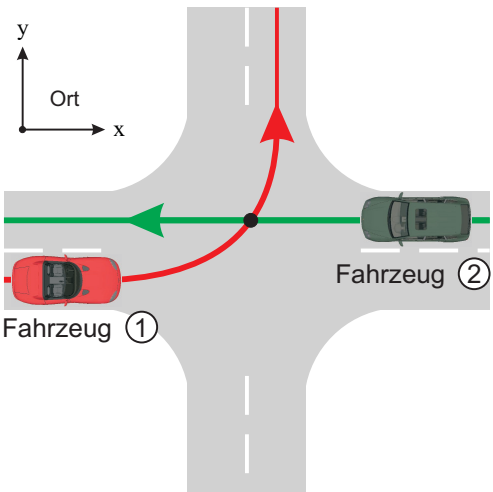

Abbildung 3.1: Typische Problemstellung

[1]Der Begriff Fahrspuren ist hier im Sinne von *Routen* bzw. Passiermöglichkeiten zu verstehen.

bildung 3.1 Geradeausfahrer, Links- und Rechtsabbieger, die aus derselben Richtung in die Kreuzung einfahren, denselben Einfahrtsbereich.

Durch die Elemente der Schnittmengen werden die Fahrspuren zu einem Netzwerk verknüpft. Fahrzeuge, die in die Kreuzung einfahren, werden als in das Netzwerk eintretende Objekte betrachtet, die auf ihrem Weg durch das Netzwerk Ressourcen belegen. Die Ressourcen des Netzwerks entsprechen Positionen bzw. Zonen im Kreuzungsbereich, die zu jedem Zeitpunkt jeweils nur von einem Fahrzeug belegt werden dürfen. Durch dieses Netzwerk ist nun für jedes Fahrzeug ein Weg zu planen, der optimal ist bezüglich zuvor definierter Zielgrößen.

Wie der Name *Kreuzungsmanagement* bereits andeutet, bestand die ursprüngliche Aufgabenstellung darin, den Verkehrsfluss an Kreuzungen zu optimieren. Die in dieser Arbeit vorgestellte Systematik ist allerdings für beliebige andere Verkehrsknotenpunkte ebenfalls geeignet. Bei der Erläuterung der Systematik wird im Folgenden jedoch größtenteils, stellvertretend für sämtliche Verkehrsknotenpunkte, eine Kreuzung gemäß Abbildung 3.1 herangezogen.

Das Lösen von Netzwerkflussproblemen ist dem Bereich der kombinatorischen Optimierung zuzuordnen (Kapitel 2). Aus diesem Grund müssen die prinzipiell kontinuierlichen Vorgänge im Kreuzungsbereich diskretisiert werden. Wie dies im Detail erfolgt, wird in den folgenden Abschnitten dargelegt.

3.1 Modellierung der Verkehrsknotenpunkte

Die beispielhaft betrachteten Verkehrsknotenpunkte (Kreuzungen und Kreisverkehre) verfügen über jeweils $n_f = 4 \cdot 3 = 12$ Fahrspuren: Ein Fahrzeug kann über 4 unterschiedliche Einfahrten in den Knotenpunkt hineinfahren und dann zwischen 3 möglichen Ausfahrten wählen. Bei einer Kreuzung, für die im Folgenden die Modellierung erläutert wird, entsprechen diese Wahlmöglichkeiten dem Geradeausfahren, dem Links- und dem Rechtsabbiegen. Sämtliche Fahrspuren sind in der Menge $F = \{1, ..., n_f\}$ zusammengefasst. Für die Fahrspur f gilt somit $f \in F$.

Die Position $p_f \in \mathbb{R}^2$ in der Kreuzungsebene eines sich auf der Fahrspur f (Länge $s_{max,f}$) bewegenden Fahrzeugs lässt sich zunächst durch kontinuierliche Funktionen[2] des zurückgelegten Weges $s_f \in \mathbb{R}_{\geq 0}$ auf der jeweiligen

[2]Konkret wurden in der Arbeit bikubische parametrische Splines verwendet.

Fahrspur beschreiben (vgl. Abbildung 3.2):

$$p_f(s_f) := (x_f(s_f), y_f(s_f)), \quad \text{mit} \quad 0 \leq s_f < s_{max,f}, \tag{3.1}$$

wobei $s_f = 0$ für den Beginn und $s_f = s_{max,f}$ für das Ende der Fahrspur im Kreuzungsbereich steht.

Die Wegkoordinate s_f wird nun für die Fahrspur f gemäß $s^*_{0,f} < s^*_{1,f} < ... < s^*_{h,f} < ...$ diskretisiert (Index h), wobei der Abstand $\Delta s^*_f = s^*_{h,f} - s^*_{h-1,f}$ nicht äquidistant sein muss und dies insbesondere im Zentrum des Verkehrsknotenpunktes auch nicht ist. Auf diese Weise entsteht die Menge

$$S_{D,f} = \{s^*_{h,f}\}, \quad h = 0, ..., n_{s^*,f}, \tag{3.2}$$

in der sämtliche Wegstützstellen der Fahrspur f zusammengefasst sind. Durch die Diskretisierung des Weges ergibt sich gemäß (3.1) weiterhin die Menge

$$P_f = \{p_{h,f} | p_{h,f} := (x_f(s^*_{h,f}), y_f(s^*_{h,f}))\}, \quad h = 0, ..., n_{s^*,f}, \tag{3.3}$$

von diskreten Positionen (Abbildung 3.2). Jeder dieser Positionen (oder auch

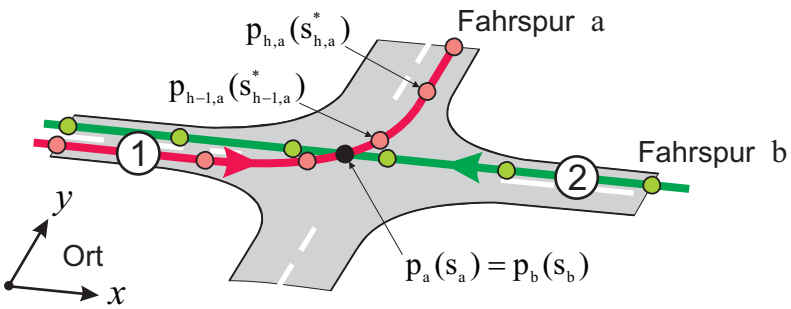

Abbildung 3.2: Diskretisierung des Ortes

Knoten) $p_{h,f}$ wird eine Zone $z_{h,f}$ zugeordnet ($p_{h,f} \mapsto z_{h,f}$), die einem kontinuierlichen Wegbereich auf der jeweiligen Fahrspur entspricht. Auf Basis der durchgeführten Wegdiskretisierung werden diese Zonen folgendermaßen definiert:

$$z_{h,f} = \left\{ s_f \,\middle|\, \frac{s^*_{h-1,f} + s^*_{h,f}}{2} \leq s_f < \frac{s^*_{h,f} + s^*_{h+1,f}}{2} \right\}, \tag{3.4}$$

$$\text{für} \quad h = 1, ..., (n_{s^*,f} - 1),$$

$$z_{0,f} = \left\{ s_f \,\middle|\, 0 \leq s_f < \frac{s^*_{0,f} + s^*_{1,f}}{2} \right\} \quad \text{und}$$
$$z_{n,f} = \left\{ s_f \,\middle|\, \frac{s^*_{n-1,f} + s^*_{n,f}}{2} \leq s_f < s_{max,f} \right\}, \quad n = n_{s^*,f}.$$

Die Mengen

$$Z_f = \{z_{h,f}\}, \quad h = 0, ..., n_{s^*,f}, \tag{3.5}$$

$S_{D,f}$ und P_f sind demnach paarweise *bijektiv* zueinander.

Sofern zwei Fahrspuren $a, b \in F$ mit $a \neq b$ einander schneiden, so existieren gemäß (3.1) Positionen, für die

$$p_a(s_a) = p_b(s_b) \tag{3.6}$$

gilt. Über die Wegkoordinaten s_a und s_b können diese Positionen eindeutig einer Zone und somit auch einer diskreten Position bzw. einem Knoten der jeweiligen Fahrspur zugeordnet werden:

$$p_a(s_a) = p_b(s_b) \mapsto \begin{cases} z_{h_a,a} \mapsto p_{h_a,a} \in P_a \\ z_{h_b,b} \mapsto p_{h_b,b} \in P_b. \end{cases} \tag{3.7}$$

Da sich in sämtlichen Zonen $z_{h,f}$ zu jedem Zeitpunkt nur ein Fahrzeug aufhalten darf, muss das bisherige Modell noch um die Dimension Zeit erweitert werden. Zu diesem Zweck wird eine Diskretisierungsschrittweite $\Delta t^* \in \mathbb{R}_{>0}$ vorgegeben, aus der die Menge

$$T_D = \{t^*_i | t^*_i := i \cdot \Delta t^*\}, \quad i = 0, ..., n_{t^*}, \tag{3.8}$$

diskreter Zeitpunkte (Index i) resultiert.

Die h-te Position $p_{h,f}(s^*_{h,f}) \in P_f$ in der Kreuzungsebene und auf der Fahrspur f, zum diskreten Zeitpunkt t^*_i betrachtet, wird als Knoten $\alpha_{f,h,i}$ definiert. Auf dieser Basis wird die Menge

$$A_f = \{\alpha_{f,h,i} | \alpha_{f,h,i} := (f, s^*_h, t^*_i),\ s^*_h \in S_{D,f},\ t^*_i \in T_D\},\ f \in F \tag{3.9}$$

gebildet, die sämtliche Knoten der Fahrspur f enthält und eindeutig beschreibt. Prinzipiell kann die Menge A_f auch als *Kreuzprodukt* der Mengen $S_{D,f}$ und T_D ausgedrückt werden:

$$A_f = S_{D,f} \times T_D \tag{3.10}$$

Die Kreuzung in ihrer Gesamtheit ist dann durch die dreidimensionale Knotenanordnung

$$A = \bigcup_{f=1}^{n_f} A_f \tag{3.11}$$

gegeben. In Abbildung 3.3 ist dies anschaulich für einige Knoten der Mengen A_a und A_b dargestellt.

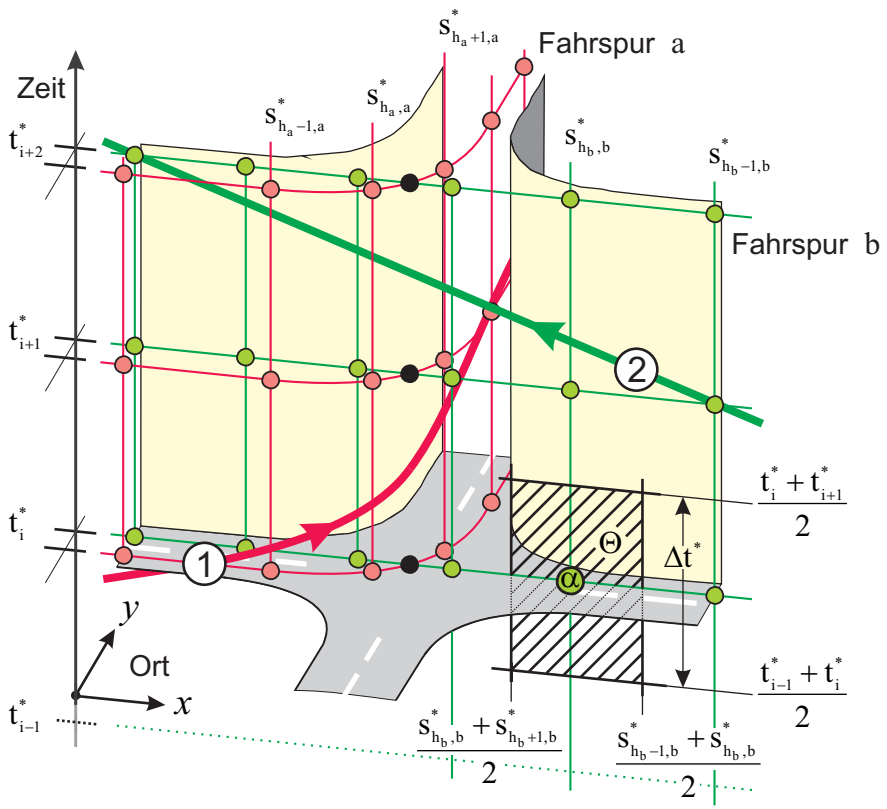

Abbildung 3.3: Diskretisierung von Ort und Zeit

Die kontinuierlichen Bewegungen $s_f(t)$ der Fahrzeuge im Kreuzungsbereich bzw. ihre Trajektorien (Abbildung 3.3, ① und ②) müssen nun auf die Menge A von diskreten Knoten abgebildet werden. Zu diesem Zweck wird für jeden

Knoten $\alpha_{f,h,i}$ ein Einzugsbereich $\Theta_{f,h,i}$ definiert ($\alpha_{f,h,i} \mapsto \Theta_{f,h,i}$), der durch die beiden Bedingungen

$$s_f \in z_{h,f} \tag{3.12}$$

und

$$\frac{t^*_{i-1} + t^*_i}{2} \leq t < \frac{t^*_i + t^*_{i+1}}{2} \tag{3.13}$$

gegeben ist (siehe Abbildung 3.3):

$$\forall \alpha_{f,h,i}: \quad \Theta_{f,h,i} = \left\{ s_f(t) \,\middle|\, s_f \in z_{h,f},\ \frac{t^*_{i-1} + t^*_i}{2} \leq t < \frac{t^*_i + t^*_{i+1}}{2} \right\}. \tag{3.14}$$

Dies entspricht im Wesentlichen der Zonendefinition gemäß (3.4), erweitert um die Dimension Zeit. Sobald die beiden Bedingungen (3.12) und (3.13) für einen beliebigen Punkt auf einer Trajektorie entlang des Weges $s_f(t)$ erfüllt sind, ist der zugehörige Knoten zu sperren bzw. als *besetzt* zu kennzeichnen. Dies wird mit der Abbildung $\omega : A \to \{0, 1\}$ erreicht, die folgendermaßen definiert ist:

$$\alpha \in A \mapsto \omega(\alpha) = \begin{cases} 0, & \text{wenn Knoten frei,} \\ 1, & \text{wenn Knoten besetzt.} \end{cases} \tag{3.15}$$

Auf besetzte Knoten kann bei der Trajektorienplanung für neu in die Kreuzung einfahrende Fahrzeuge nicht mehr zugegriffen werden, da es sonst zu Kollisionen kommen kann. Weiterhin müssen für jeden Zeitpunkt t^*_i sämtliche Fälle berücksichtigt werden, die durch die Beziehung (3.6) und ihre Implikationen beschrieben werden. Existieren also Schnittbeziehungen für den betrachteten Knoten α_a der Fahrspur a mit einem Knoten α_b einer Fahrspur b, so ist für die einzuplanende Trajektorie nicht nur die Operation

$$\omega(\alpha_{a,h_a,i}) = 1 \tag{3.16}$$

auszuführen, sondern zusätzlich auch noch die Operation

$$\omega(\alpha_{b,h_b,i}) = 1. \tag{3.17}$$

Ferner darf in dem beschriebenen Fall die Operation (3.16) nur ausgeführt werden, wenn $\omega(\alpha_{b,h_b,i}) = 0$ gilt.

Die Menge $S_{D,f}$ bzw. die Gesamtheit aller Wegstützstellen $s^*_{h,f}$ der Knotenanordnung A_f wird hiermit als ***Kollisionsraster*** der Fahrspur f definiert.

Das beschriebene Vorgehen lässt sich in gleicher Weise für Kreisverkehre und beliebige andere Verkehrsknotenpunkte anwenden.

3.2 Modellierung der Fahrzeuge

Um für die Fahrzeuge im Kreuzungsbereich sinnvolle Trajektorien bzw. Soll-Geschwindigkeitsverläufe planen zu können, müssen ihre unterschiedlichen fahrdynamischen Eigenschaften in ausreichender Weise berücksichtigt werden. Ein PKW kann in der Regel schneller beschleunigen als ein Bus oder LKW; auch werden während Kurvenfahrten in der Regel höhere Querbeschleunigungen auf einen PKW wirken können, bevor dieser ausbricht. Da die Algorithmen für das Kreuzungsmanagement echtzeitfähig sein sollen, muss bei der Berücksichtigung der Fahrzeugdynamik ein adäquater Kompromiss zwischen Genauigkeit und Berechnungsaufwand gefunden werden. Der aus dieser Aufgabenstellung resultierende Lösungsansatz wird in den folgenden Abschnitten skizziert.

Längsdynamik als Basis der Modellierung

Ein bewährter Ansatz für die Modellierung der Fahrzeuglängsdynamik besteht in der Beschreibung durch eine Differentialgleichung 2. Ordnung der folgenden Form:

$$m \cdot \ddot{s} = \sum F = F_u - d \cdot \dot{s}, \tag{3.18}$$

mit
- m : Masse des Fahrzeugs,
- s : zurückgelegter Weg,
- $\dot{s}$: Geschwindigkeit $v = ds/dt$,
- $\ddot{s}$: Beschleunigung $a = dv/dt$,
- d : Widerstandsbeiwert (Luftwiderstand, Rollreibung etc.) und
- F_u : Systemeingang bzw. Eingangskraft, mit max. Bremskraft $F_{B,max} \leq F_u \leq$ max. Antriebskraft $F_{A,max}$.

Überführt man Gleichung (3.18) in die Zustandsraumdarstellung, so erhält man das Gleichungssystem

$$\begin{bmatrix} \dot{s} \\ \ddot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -d/m \end{bmatrix} \cdot \begin{bmatrix} s \\ \dot{s} \end{bmatrix} + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} \cdot F_u. \tag{3.19}$$

Die Tragfähigkeit dieses Ansatzes wird durch Abbildung 3.4 illustriert, in der

beispielhaft ein Beschleunigungsdiagramm eines PKW[3] wiedergegeben wird. Befindet sich das Fahrzeug zu Beginn der Betrachtung im Stillstand, so erhält

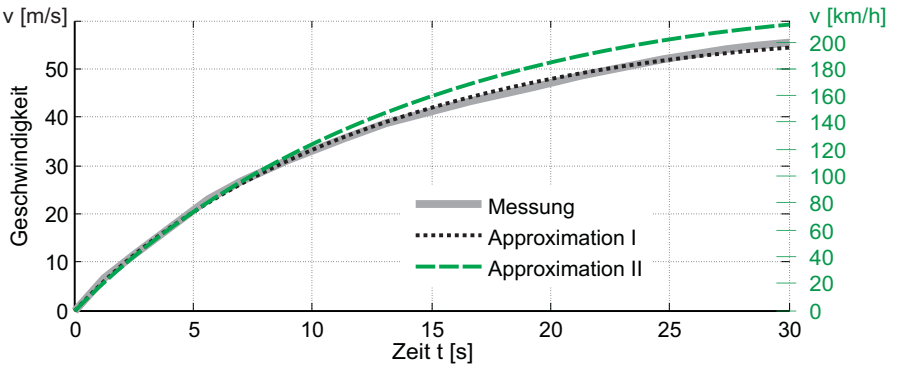

Abbildung 3.4: Beschleunigungsdiagramme im Vergleich

man nach Lösung des Differentialgleichungssystems (3.19) den Ausdruck

$$v(t) = \frac{1}{d}\left(1 - e^{-\frac{d}{m}t}\right) F_u \tag{3.20}$$

für die Geschwindigkeit im Zeitbereich. Das Beschleunigungsdiagramm gibt den schnellstmöglichen Geschwindigkeitsaufbau in Abhängigkeit von der Zeit wieder. Dies entspricht einer Beschleunigung mit maximaler Antriebskraft ($F_u = F_{A,max}$), die idealisiert als konstant über den zulässigen Drehzahlbereich des Motors angenommen wird. Mit der Beziehung (3.20) und entsprechenden Messwerten können im Rahmen einer Parameteridentifikation die Werte für die Parameter bzw. Variablen d und F_u bestimmt werden. Die Masse m wird als bekannt angenommen. Das Ergebnis des Identifikationsprozesses ist in Abbildung 3.4 als **Approximation I** dargestellt. Man kann erkennen, dass selbst bei hohen Geschwindigkeiten (bis etwa 200 km/h) mit dieser Approximation eine hohe Genauigkeit erzielt werden kann.

Gleichung (3.20) weist PT_1-Verhalten auf. Die Parameter eines PT_1-Glieds können auch anhand eines Punktes auf der Kurve der Sprungantwort und der Lage der waagerechten Asymptote bestimmt werden. Die Lage der waagerechten Asymptote entspricht in diesem Fall der maximal möglichen Geschwindigkeit v_{max}. Der Punkt auf dem Kurvenverlauf kann prinzipiell ein

[3]Sämtliche Daten des Beispielfahrzeugs AUDI A4 2.0 TFSI stammen von www.autozeitung.de bzw. aus dem Magazin AUTOZEITUNG, Ausgabe 24/04, Bauer Vertriebs KG, Hamburg.

beliebiger sein, für die meisten Fahrzeuge ist jedoch die Dauer t_{100} der Beschleunigung von 0 auf $v_{100} = 100$ km/h bekannt. Für das Beispielfahrzeug wurden $t_{100} = 7,4$ s und $v_{max} = 241$ km/h angegeben. Auf Basis dieser Angaben kann die Beziehung

$$v(t) = v_{\max} \left(1 - \left(\frac{v_{\max} - v_{100}}{v_{\max}} \right)^{\frac{t}{t_{100}}} \right) \tag{3.21}$$

abgeleitet werden, mit der sich schließlich die **Approximation II** in Abbildung 3.4 ergibt. Sind für ein Fahrzeug v_{max} und t_{100} gegeben, so können mit den folgenden Formeln die Werte der Parameter d und $F_u = F_{A,max}$ berechnet werden:

$$d = -\ln \left(\frac{v_{\max} - v_{100}}{v_{\max}} \right) \frac{m}{t_{100}}, \tag{3.22}$$

$$F_{A,max} = d \cdot v_{\max}. \tag{3.23}$$

Man kommt schnell zu einer höheren Genauigkeit für einen vorgegebenen Geschwindigkeitsbereich, indem man einfach v_{max} anpasst. Auf diese Art kann man sich die Parameteridentifikation mit einem Optimierungsverfahren sparen und erhält trotzdem eine in der Regel völlig ausreichende Genauigkeit.

Im Kreuzungsmanagement treten Geschwindigkeiten auf, die deutlich unterhalb des Bereiches von 100 km/h liegen. Aus diesem Grund erscheint die Approximation der Fahrzeuglängsdynamik durch das Gleichungssystem (3.19) als völlig ausreichend.

Der Fall der Verzögerung eines Fahrzeugs wird in ähnlicher Weise behandelt wie zuvor beschrieben. Prinzipiell kann, unabhängig vom Fahr- bzw. Systemzustand, von einer konstanten Verzögerung ausgegangen werden. Dadurch vereinfacht sich die Modellierung signifikant und die erzielte Genauigkeit ist immer noch völlig ausreichend.

Übergang in den zeitdiskreten Bereich

Nachdem nun ein zeitkontinuierliches Modell für die Fahrzeuge entwickelt wurde, sind die zwei folgenden Aufgaben zu bearbeiten: Das lineare, zeitinvariante Differentialgleichungssystem (3.19) muss gelöst werden. Weiterhin muss das Fahrzeugmodell in den zeitdiskreten Bereich übertragen werden, damit die Optimierungsaufgabe als Netzwerkflussproblem formuliert und

letztlich auch gelöst werden kann. Beide Aufgaben lassen sich in einem Arbeitsgang erledigen, was zunächst im Allgemeinen erläutert und dann auf den speziellen Fall des Gleichungssystems (3.19) übertragen wird.

Lineare, zeitinvariante Differentialgleichungssysteme sind häufig in der Zustandsraumdarstellung gegeben. Das Gleichungssystem (3.19) ist ein Beispiel dafür. Allgemein wird für diese Art der Beschreibung die Form

$$\dot{\underline{x}} = \underline{A} \cdot \underline{x} + \underline{B} \cdot \underline{u} \tag{3.24}$$

verwendet, mit
$\underline{x}$: Zustandsvektor,
$\underline{u}$: Eingangsvektor,
$\underline{A}$: Systemmatrix,
$\underline{B}$: Eingangsmatrix.

Die Lösung für ein solches System im Zeitbereich lautet gemäß [16]

$$\underline{x}(t) = \int_{t_0}^{t} e^{\underline{A}\cdot(t-\tau)} \cdot \underline{B} \cdot \underline{u}(\tau)\, d\tau \; + \; e^{\underline{A}\cdot(t-t_0)} \cdot \underline{x}(t_0). \tag{3.25}$$

Betrachtet man nur den Zustandsübergang zwischen zwei diskreten Zeitpunkten t_k und $t_{k+1} = t_k + T$ und nimmt an, dass der Eingangsvektor $\underline{u}(\tau)$ zwischen diesen beiden Zeitpunkten konstant gehalten wird ($\underline{u}(\tau) = \underline{u}(t_k),\ t_k < \tau < t_{k+1}$), so kann die Gleichung (3.25) umgeformt werden in

$$\underline{x}(t_{k+1}) = \int_{0}^{T} e^{\underline{A}\cdot(T-\tau)} \cdot \underline{B}\, d\tau \cdot \underline{u}(t_k) + \; e^{\underline{A}\cdot T} \cdot \underline{x}(t_k). \tag{3.26}$$

Im vorliegenden Beispiel kann die Matrizen-e-Funktion durch die Potenzreihenentwicklung für die e-Funktion analytisch bestimmt werden. Für das System gemäß dem Gleichungssystem (3.19) führt dies zum folgenden Ergebnis:

$$e^{\underline{A}z} = \begin{bmatrix} 1 & \frac{m}{d}\left(1 - e^{-\frac{d}{m}z}\right) \\ 0 & e^{-\frac{d}{m}z} \end{bmatrix}. \tag{3.27}$$

Aus (3.26) und (3.27) ergibt sich dann nach Lösung des Integrals von (3.26) und in Anlehnung an [14] der Ausdruck

$$\underline{x}_{k+1} = \underline{\Phi}(T) \cdot \underline{x}_k \; + \; \underline{H}(T) \cdot \underline{u}_k, \tag{3.28}$$

mit $\underline{x}_k = \underline{x}(t_k)$, $\underline{x}_{k+1} = \underline{x}(t_{k+1})$, $\underline{u}_k = \underline{u}(t_k)$,

$$\underline{\Phi}(T) = \begin{bmatrix} 1 & \frac{m}{d}\left(1 - e^{-\frac{d}{m}T}\right) \\ 0 & e^{-\frac{d}{m}T} \end{bmatrix} \tag{3.29}$$

und

$$\underline{H}(T) = \begin{bmatrix} \frac{T}{d} - \frac{m}{d^2} \cdot \left(1 - e^{-\frac{d}{m}T}\right) \\ \frac{1}{d} \cdot \left(1 - e^{-\frac{d}{m}T}\right) \end{bmatrix}. \tag{3.30}$$

Mit den Bezeichnungen aus (3.19) ergibt sich die äquivalente Darstellung

$$\begin{bmatrix} s_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \frac{m}{d} \cdot \left(1 - e^{-\frac{d}{m}T}\right) \\ 0 & e^{-\frac{d}{m}T} \end{bmatrix} \cdot \begin{bmatrix} s_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{T}{d} - \frac{m}{d^2} \cdot \left(1 - e^{-\frac{d}{m}T}\right) \\ \frac{1}{d} \cdot \left(1 - e^{-\frac{d}{m}T}\right) \end{bmatrix} \cdot F_{u,k}. \tag{3.31}$$

Bei der Kraft $F_{u,k}$ handelt es sich um die Eingangs- bzw. Steuergröße des Systems, die gemäß der oben getroffenen Annahme im Intervall $[t_k,\ t_{k+1})$ einen konstanten Wert annimmt. Je nach Größe und Vorzeichen dieser Kraft legt das Fahrzeug im Intervall bzw. innerhalb der Zeit T den Weg $\Delta s = s_{k+1} - s_k$ zurück und ändert seine Geschwindigkeit von v_k auf v_{k+1}. Die Basis für die Berücksichtigung der Längsdynamik bilden die beiden folgenden Gleichungen (3.32) und (3.33), die sich unmittelbar aus der Darstellung (3.31) ergeben:

$$\Delta s = \frac{m}{d}\left(1 - e^{-\frac{d}{m}T}\right) \cdot v_k + \left(\frac{T}{d} - \frac{m}{d^2}\left(1 - e^{-\frac{d}{m}T}\right)\right) \cdot F_{u,k}, \tag{3.32}$$

$$v_{k+1} = e^{-\frac{d}{m}T} \cdot v_k + \frac{1}{d}\left(1 - e^{-\frac{d}{m}T}\right) \cdot F_{u,k}. \tag{3.33}$$

Vergleicht man allgemein die zeitkontinuierliche Darstellung eines dynamischen Systems in Form eines Differentialgleichungssystems gemäß (3.24) mit der zeitdiskreten Darstellung eines diskretisierten Differentialgleichungssystems bzw. einem Differenzengleichungssystem gemäß (3.28) bei konstanter Eingangsgröße, beispielsweise bei einer Sprunganregung, so liefert das Differenzengleichungssystem an den diskreten Stützstellen bzw. zu den Zeitpunkten t_k für jeden Zustand exakt denselben Wert, den auch das kontinuierliche

System liefern würde. Es spielt dabei keine Rolle, ob die Intervalllänge T stets äquidistant ist oder variiert. Im Rahmen der Diskretisierung sollte T klein genug gewählt werden, so dass die realen Brems- und Beschleunigungsvorgänge noch hinreichend gut angenähert werden.

Berücksichtigung der Querdynamik als Modellerweiterung

Bei der Betrachtung der Querdynamik liegt das primäre Interesse auf dem Verlauf der Quer- bzw. Zentripetalbeschleunigung a_Q während Kurvenfahrten. Die Zentripetalbeschleunigung darf je Fahrsituation einen bestimmten Grenzwert nicht überschreiten, damit die Fahrsicherheit bzw. die Fahrstabilität nicht gefährdet wird. Des Weiteren wird sie auch als Maßzahl für das Komfortempfinden der Fahrzeuginsassen angesehen[4]. Daher muss das Fahrzeugmodell eine ausreichend genaue Approximation dieser Kenngröße zur Verfügung stellen. Auf welche Art und Weise dies innerhalb des Kreuzungsmanagements durchgeführt wird, wird im Folgenden erläutert.

Auf ein Fahrzeug, das sich auf einer Kurvenfahrt befindet, wirkt die Zentripetalkraft F_Z, die durch die Beziehung

$$F_Z = \frac{m \cdot v^2}{r_\kappa} = \kappa \cdot m \cdot v^2 \tag{3.34}$$

gegeben ist. Bei r_κ handelt es sich um den Radius der Bahnkurve, auf dem sich das Fahrzeug bewegt. Mit κ wird die Krümmung der Bahnkurve bezeichnet. Es gilt $r_\kappa = 1/\kappa$. Je nach Verlauf der Bahnkurve kann sich die Krümmung ändern. Die Krümmung einer beliebigen Bahnkurve in der xy-Ebene wird mit der folgenden Formel berechnet:

$$\kappa = \frac{x'(s)y''(s) - x''(s)y'(s)}{(x'(s)^2 + x'(s)^2)^{\frac{3}{2}}}, \quad \text{mit} \quad x'(s) = \frac{d}{ds}x(s), ... \tag{3.35}$$

Auch hier steht s wieder für den zurückgelegten Weg auf der Bahnkurve und wird als Parameter der Bahnkurve bezeichnet. In Abhängigkeit von s ist ein beliebiger Punkt auf der Bahnkurve exakt durch seine Koordinaten $x(s)$ und $y(s)$ bestimmt. Derartige Bahnkurven lassen sich gut durch bikubische parametrische Splines beschreiben, die auch zu diesem Zweck im Kreuzungssimulator implementiert sind. Die innerhalb des Simulators implementierte

[4] Bei der Deutschen Bahn gilt beispielsweise ein Regelwert für die Querbeschleunigung a_Q von $0,65$ bis $0,85\ m/s^2$.

Geometrie beschränkt sich zunächst auf Bahnkurven im Sinne von Kreissegmenten, so dass während eines Abbiegevorgangs mit konstanten Krümmungen κ gearbeitet werden kann.

Aus (3.34) erhält man ferner die Quer- bzw. Zentripetalbeschleunigung

$$a_Q(t) = \frac{F_Z}{m} = \frac{v(t)^2}{r_\kappa(t)} = v(t)^2 \cdot \kappa(t) \tag{3.36}$$

bzw.

$$a_{Q,k} = \frac{v_k^2}{r_{\kappa,k}} = v_k^2 \cdot \kappa_k \tag{3.37}$$

für den diskreten Zeitpunkt t_k. Damit sind nun sämtliche Bestandteile des Fahrzeugmodells eingeführt und abgeleitet. Die Struktur des resultierenden kontinuierlichen Modells wird durch das Blockbild gemäß Abbildung 3.5 zusammengefasst.

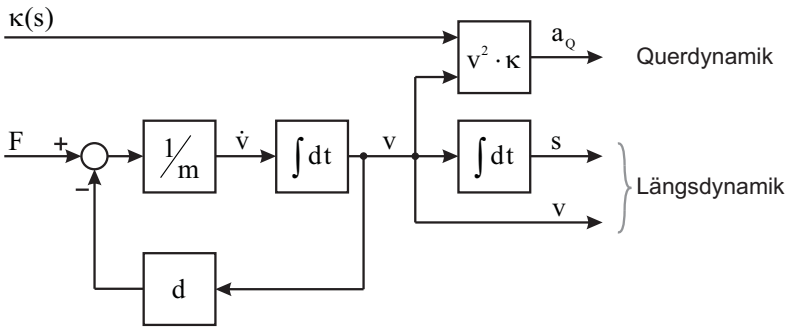

Abbildung 3.5: Blockbild des verwendeten Fahrzeugmodells

3.3 Bildung des Netzwerks

Das Fahrzeugmodell gemäß dem Differenzengleichungssystem (3.31) bestimmt in Abhängigkeit der intervallweise konstanten Eingangskraft $F_{u,k-1}$ den zurückgelegten Weg s_k und die Geschwindigkeit v_k zum diskreten Zeitpunkt t_k.

Für die Gesamtheit aller betrachteten Zeitpunkte t_k $(k = 0, ..., n_t)$ ergibt sich dann die Folge

$$\tilde{P}_k = \{\tilde{p}_k | \tilde{p}_k := (s_k, t_k, v_k),\ \tilde{p}_k \in \mathbb{R}^3_{\geq 0}\}, \quad k = 0, ..., n_t, \tag{3.38}$$

aus diskreten Punkten[5] $\tilde{p}_k$ im dreidimensionalen Raum $B \in \mathbb{R}^3_{\geq 0}$, der durch die Dimensionen Weg, Zeit und Geschwindigkeit aufgespannt wird. Diskrete Punkte in diesem Raum werden als *Knoten* bezeichnet. Zulässige Übergänge gemäß (3.31) vom Zustand $[s_k, v_k]^T$ im Zeitpunkt t_k in den Zustand $[s_{k+1}, v_{k+1}]^T$ zum Zeitpunkt t_{k+1} werden als *Kanten* bezeichnet. Jeder Kante wird ein *Gütewert* c_k für die Bewertung des zugehörigen Zustandsübergangs zugeordnet.

3.3.1 Die potenziellen Knoten des Netzwerks

Die Punktfolge (3.38), welche die Trajektorie des Fahrzeugs repräsentiert, wird für ein Fahrzeug auf Fahrspur f nun derart gestaltet, dass sie konform ist mit der Knotenanordnung A_f und auf diese abgebildet werden kann. Zu diesem Zweck wird im Folgenden der Raum B bzw. vielmehr seine diskretisierte Repräsentation B_D genauer definiert:

Zunächst wird die Dimension Geschwindigkeit diskretisiert, indem auf Basis der Diskretisierungsschrittweite $\Delta v^* \in \mathbb{R}_{>0}$ die Menge

$$V_D = \{v_j^* | v_j^* := j \cdot \Delta v^*\}, \quad j = 0, ..., n_{v^*}, \tag{3.39}$$

diskreter Geschwindigkeiten (Index j) gebildet wird.

Im nächsten Schritt wird gemäß der Definition

$$S_{D,B_f} : S_{D,B_f} \subseteq S_{D,f} \tag{3.40}$$

die Knotenmenge

$$B_{D,f} = \{\beta_{h,i,j} | \beta_{h,i,j} := (s_h^*, t_i^*, v_j^*),\ s_h^* \in S_{D,B_f},\ t_i^* \in T_D,\ v_j^* \in V_D\}, \tag{3.41}$$

mit $f \in F$, definiert. Diese Menge entspricht einer Diskretisierung von B, die konform ist zu A_f. Jeder Knoten von $B_{D,f}$ kann durch Projektion bzw. durch einfaches Streichen seiner Geschwindigkeitskoordinate auf einen Knoten der

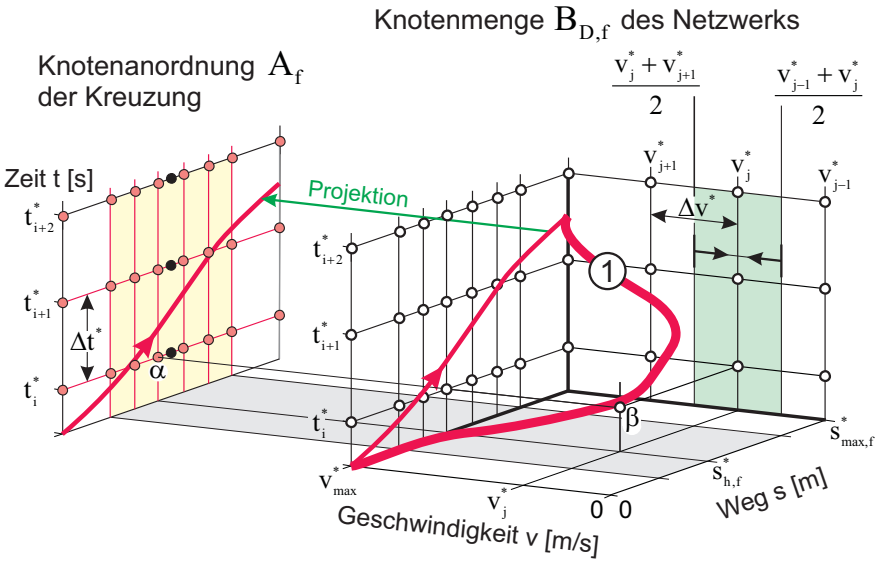

Abbildung 3.6: Zusammenhang zwischen A_f und $B_{D,f}$

Menge A_f abgebildet werden ($\beta_{h,i,j} \in B_{D,f} \mapsto \alpha_{f,h,i} \in A_f$). Die Abbildung 3.6 veranschaulicht diesen Zusammenhang.

Die Menge $B_{D,f}$ kann auch durch das Kreuzprodukt

$$B_{D,f} = S_{D,B_f} \times T_D \times V_D \tag{3.42}$$

ausgedrückt werden. Die Elemente von $B_{D,f}$ entsprechen den Knoten des Netzwerks, mit dem die optimale Trajektorie eines Fahrzeugs auf Fahrspur f berechnet wird. Sie bilden einen Quader mit *maximal* $(n_{s^*,f} + 1) \cdot (n_{t^*} + 1) \cdot (n_{v^*} + 1)$ diskreten Punkten bzw. Knoten.

Den dreidimensionalen Raum B, der durch die Knotenmenege $B_{D,f}$ repräsentiert wird, kann man sich auch als Aneinanderreihung von zweidimensionalen Unterräumen $B_{VT,h}$ mit den Dimensionen Geschwindigkeit (v) und Zeit (t) vorstellen, wobei jeder diskreten Wegstützstelle $s_{h,f}^* \in S_{D,f}$ ein solcher Unterraum zugeordnet werden kann, aber nicht muss ($S_{D,B_f} \subseteq S_{D,f}$). Wichtig ist lediglich die Projektionseigenschaft von $B_{D,f}$: Jeder Knoten von $B_{D,f}$ muss

[5]Diese Punkte sind nicht zu verwechseln mit den Positionen $p_f(s_f)$ gemäß (3.1), die ein Fahrzeug auf Fahrspur f einnehmen kann.

auf einen Knoten von A_f projiziert werden können. In den folgenden Ausführungen (vgl. Abschnitt 3.4) wird noch deutlich werden, dass es sinnvoll ist, nicht an jeder Wegstützstelle einen Unterraum $B_{VT,h}$ zu definieren.

Die Gesamtheit aller Wegstützstellen $s_k \in S_{D,B_f}$ der Knotenanordnung $B_{D,f}$ wird hiermit als ***Optimierungsraster*** der Fahrspur f definiert. Wird nicht für jede Wegstützstelle $s^*_{h,f} \in S_{D,f}$ ein Unterraum $B_{VT,h}$ definiert, so bildet die Gesamtheit der Stützstellen des Optimierungsrasters eine *echte Teilmenge* der Gesamtheit aller Stützstellen des zugehörigen Kollisionsrasters ($S_{D,B_f} \subset S_{D,f}$).

3.3.2 Die Kanten des Netzwerks

Mit dem Differenzengleichungssystem (3.31) ist nun eine Trajektorie zu planen, die durch die Folge (3.38) diskreter Punkte repräsentiert wird und die Bedingung

$$\forall \tilde{p}_k \in \tilde{P}_k : s_k \in S_{D,B_f} \wedge t_k \in T_D \wedge v_k \in V_D \quad \text{bzw.} \tag{3.43}$$

$$\forall \tilde{p}_k \in \tilde{P}_k : \tilde{p}_k \in B_{D,f}$$

erfüllt.

Das Gleichungssystem (3.31) besteht aus zwei Gleichungen mit den zwei Unbekannten s_{k+1} und v_{k+1}, die sich unmittelbar nach Vorgabe von $T = t_{k+1} - t_k$ und $F_{u,k}$ ergeben. Im Falle des Kreuzungsmanagements ist diese Konstellation allerdings ungünstig, da Wegabhängigkeiten berücksichtigt werden müssen, die unmittelbar aus der Kreuzungsgeometrie resultieren. So muss beispielsweise bei abbiegenden Fahrzeugen sichergestellt werden, dass an der Wegstützstelle, an der die Einfahrtsgerade in eine Bahnkurve übergeht, eine vorgegebene Geschwindigkeit nicht überschritten wird. Es wäre also wünschenswert, wenn man anstelle von t_{k+1} ein s_{k+1} vorgeben könnte. Leider lässt sich das Gleichungssystem (3.31) nicht explizit nach t_{k+1} bzw. T und $F_{u,k}$ auflösen. Die Formulierung des Fahrzeugmodells in Abhängigkeit des Weges s führt auf ein nichtlineares Differentialgleichungssystem und ist somit auch nicht zielführend. Aus diesem Grund wird die folgende Lösung gewählt:

Im Vorfeld wird das Gleichungssystem (3.31) nach v_{k+1} und $F_{u,k}$ aufgelöst. Dies führt auf die beiden Gleichungen

$$v_{k+1} = \frac{v_k \cdot d \cdot T \cdot e^{-\frac{d}{m}T} \; + \; (\Delta s \cdot d - m \cdot v_k)\left(1 - e^{-\frac{d}{m}T}\right)}{d \cdot T - m\left(1 - e^{-\frac{d}{m}T}\right)} \tag{3.44}$$

und

$$F_{u,k} = \frac{d\left(\Delta s \cdot d - m \cdot v_k\left(1 - e^{-\frac{d}{m}T}\right)\right)}{d \cdot T - m\left(1 - e^{-\frac{d}{m}T}\right)}. \tag{3.45}$$

Sei nun für den Punkt $\tilde{p}_k(s_k, t_k, v_k)$ die Bedingung $\tilde{p}_k \in B_{D,f}$ bereits erfüllt, gibt man weiterhin Δs durch $s_{k+1} \in S_{D,B_f}$ und T durch $t_{k+1} \in T_D$ vor, so erhält man gemäß (3.44) und (3.45) die Größen v_{k+1} und $F_{u,k}$. Es müssen nun lediglich noch die Bedingungen $v_{k+1} \in V_D$ und $F_{B,max} \leq F_{u,k} \leq F_{A,max}$ erfüllt werden.

Was die erste Bedingung betrifft, so wird in der Regel $v_{k+1} \notin V_D$ der Fall sein. Aus diesem Grund wird für jede diskrete Geschwindigkeitsstützstelle v_j^*, analog zum Vorgehen bei (3.4) bzw. (3.13), ein Einzugsbereich festgelegt (siehe Abbildung 3.6). Liegt der Wert v_{k+1} also innerhalb des Intervalls

$$\left\{ v | \frac{v_{j-1}^* + v_j^*}{2} \leq v < \frac{v_j^* + v_{j+1}^*}{2} \right\}, \tag{3.46}$$

so wird als Geschwindigkeitskoordinate des Punktes $\tilde{p}_{k+1}$ der Wert v_j^* anstelle von v_{k+1} festgehalten. Weiterhin wird für jeden Punkt $\tilde{p}_{k+1}$ ein Korrekturwert

$$\Delta v_{k+1} = v_{k+1} - v_j^* \tag{3.47}$$

für die Geschwindigkeitskoordinate abgespeichert. Anhand des Korrekturwertes Δv_{k+1} kann die tatsächliche Geschwindigkeit v_{k+1} gemäß (3.44) rekonstruiert werden und somit auch die exakte Trajektorie, die man erhalten würde, wenn man während der Zeit $t_k \leq t < t_{k+1}$ den Systemeingang auf den nach obigem Vorgehen berechneten Wert $F_{u,k}$ setzte.

Die Bedingung $F_{B,max} \leq F_{u,k} \leq F_{A,max}$ lässt sich durch die Bedingung

$$t_{k+1_{min}} = t_k + T_{min} \leq t_{k+1} \leq t_{k+1_{max}} = t_k + T_{max}, \tag{3.48}$$

$$\text{mit} \quad t_{k+1} \in T_D,$$

ersetzen, denn der Zeitpunkt t_{k+1}, in dem der Weg s_{k+1} zurückgelegt worden ist, hängt unmittelbar von der Eingangskraft $F_{u,k}$ ab. Wird das Fahrzeug mit der maximal möglichen Antriebskraft $F_{u,k} = F_{A,max}$ beschleunigt, so erreicht es die Position s_{k+1} zum frühestmöglichen Zeitpunkt $t_{k+1_{min}}$ nach $T_{min} = t_{k+1_{min}} - t_k$ Sekunden. Analog dazu lässt sich im Falle der maximal zulässigen Verzögerung auch ein spätester Zeitpunkt $t_{k+1_{max}} = t_k + T_{max}$ bestimmen. Nun lässt sich aber, wie bereits weiter oben erwähnt, das Gleichungssystem (3.31) nicht explizit nach t_{k+1} bzw. T und $F_{u,k}$ auflösen. Aus diesem Grund wird ein numerisches Iterationsverfahren, das NEWTON-Verfahren für einfache Nullstellen, für die Berechnung von T_{min} und T_{max} verwendet. Da im Vorfeld bereits eine grobe Abschätzung der gesuchten Werte durch einfachste Berechnungen möglich ist, kann auch ein sehr günstiger Startwert für das Iterationsverfahren festgelegt werden. Dies führt in seiner Konsequenz dazu, dass bereits nach 1 bis 2 Iterationsschritten eine Lösung mit ausreichender Genauigkeit vorliegt. Der Rechenaufwand bleibt in einem vertretbaren Bereich und gefährdet die geforderte Echtzeitfähigkeit nicht.

Um die Menge

$$E_k(\tilde{p}_k) = \{e_{k,1}, e_{k,2}, ...\} \tag{3.49}$$

der prinzipiell möglichen Kanten zu bestimmen, die in einem Knoten bzw. Punkt $\tilde{p}_k \in B_{D,f}$ beginnen und in $\tilde{p}_{k+1} \in B_{D,f}$ mit vorgegebenem $s_{k+1} \in S_{D,B_f}$ enden (vgl. Abbildung 3.8), werden zunächst also $t_{k+1_{min}}$ und $t_{k+1_{max}}$ numerisch berechnet. Dann werden für alle Paarungen s_{k+1} und t_{k+1}, die der Bedingung (3.48) genügen, die zugehörigen Werte von v_{k+1} und $F_{u,k}$ berechnet.

Nach diesem Vorgehen erhält man die Trajektoriensegmente, die in Abbildung 3.7 (Projektion auf die Weg-Zeit-Ebene) dargestellt sind. Jedes dieser Trajektoriensegmente entspricht einer prinzipiell möglichen Kante $e_k \in E_k$ im Netzwerk. Die Abbildungen 3.7 und 3.8 sind deshalb äquivalente Darstellungen des gleichen Sachverhalts und können somit synonym verwendet werden. Jedem Trajektoriensegment bzw. jeder Kante e_k sind ein Startknoten $\tilde{p}_k$, ein Zielknoten $\tilde{p}_{k+1}$, eine Kraft $F_{u,k}$, ein Korrekturwert gemäß (3.47) für die Geschwindigkeitskomponente des Zielknotens und ein Gütewert c_k zugeordnet:

$$e_k := (\tilde{p}_k, \tilde{p}_{k+1}, F_{u,k}, \Delta v_{k+1}, c_k). \tag{3.50}$$

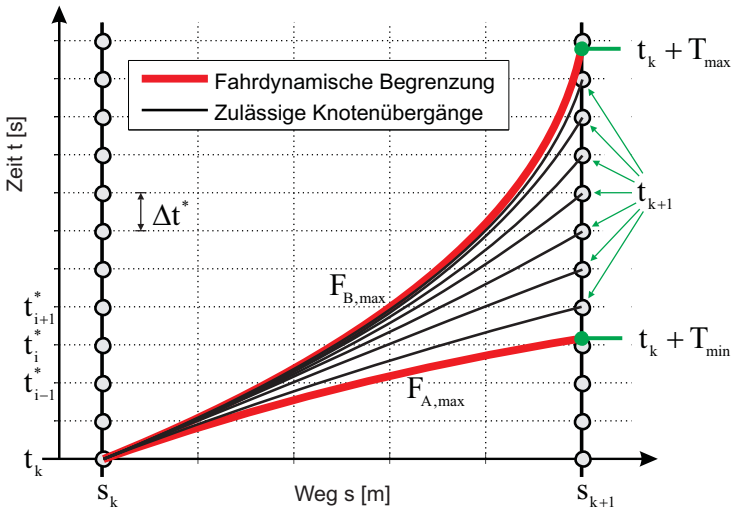

Abbildung 3.7: Trajektoriensegmente

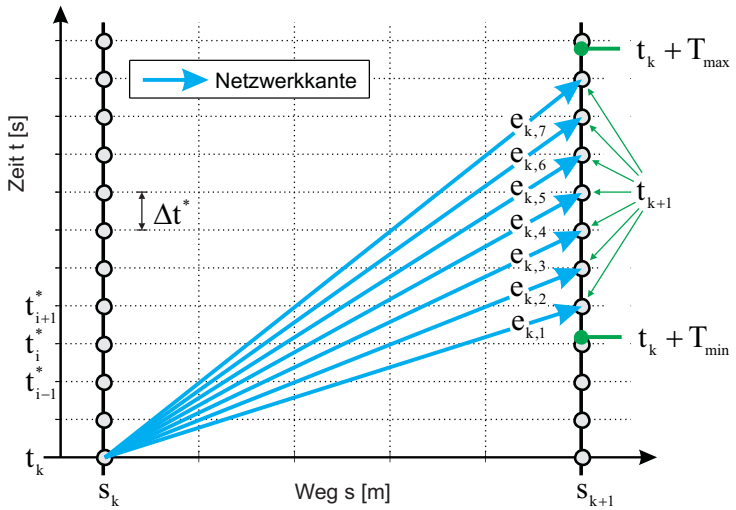

Abbildung 3.8: Kanten des Netzwerks

Mit dem Gütewert c_k wird der Knotenübergang bzw. der Übergang vom Zustand $[s_k, v_k]^T$ im Zeitpunkt t_k in den Zustand $[s_{k+1}, v_{k+1}]^T$ zum Zeitpunkt t_{k+1} bewertet. Die Bestimmung von c_k wird in Kapitel 5 detailliert behandelt.

Die Menge aller prinzipiell möglichen Kanten im Netzwerk wird durch die Menge

$$E_f = \bigcup_{k=0}^{n_{s^*,f}-1} E_k \left(\tilde{p}_k \left(s^*_{k,f}, t_k \in T_D, v_k \in V_D \right) \right) \tag{3.51}$$

ausgedrückt. Infolge der Berücksichtigung von Nebenbedingungen bzw. Restriktionen, die im Wesentlichen aus den fahrdynamischen Eigenschaften der einzelnen Fahrzeuge resultieren, wird diese Menge jedoch drastisch reduziert.

3.4 Berücksichtigung der Fahrzeuggeometrien

Bisher wurden die Fahrzeuge als sich auf einer Trajektorie bewegende Massepunkte betrachtet. Mit der Einführung der Zonen $z_{h,f}$ bzw. Einzugsbereiche $\Theta_{f,h,i}$ in Abschnitt 3.1 wurde jedoch bereits die Basis für die Berücksichtigung der geometrischen Abmessungen der Fahrzeuge gelegt. Im Folgenden wird zunächst erläutert, wie die Breite der Fahrzeuge im Lösungsalgorithmus berücksichtigt wird. Dies wiederum bildet die Grundlage für die Lage der diskreten Wegstützstellen $s^*_{h,f}$. Darauf aufbauend, wird dann die Berücksichtigung der Fahrzeuglänge im Hinblick auf die Dimension Zeit beschrieben.

Fahrzeugausdehnung in Querrichtung

Die Bewegung der Fahrzeuge wird als Bewegung in einem *Fahrschlauch* aufgefasst, dessen Breite der Breite eines Fahrstreifens entspricht. Die Begriffe Fahrschlauch und Fahrspur werden im Folgenden synonym verwendet. Im Simulator wurde eine Fahrspurbreite von 3 m implementiert. Auf diesen Wert ist somit auch die Breite der Fahrzeuge beschränkt. Exakt in der Mitte einer Fahrspur verläuft die Trajektorie eines sich auf der jeweiligen Fahrspur bewegenden Fahrzeugs (vgl. Abbildung 3.9). Schneiden die Fahrschläuche der zwei Routen $a \in F$ und $b \in F$ einander, mit $a \neq b$, so bilden ihre jeweiligen Begrenzungen insgesamt 4 Schnittpunkte, welche die 4 Ecken der gemeinsamen Fläche (Schnittfläche) beider Fahrschläuche bilden. In diese Schnittfläche, wie übrigens in jeden anderen Bereich einer Zone $z_{h,f}$ auch, darf zu jedem

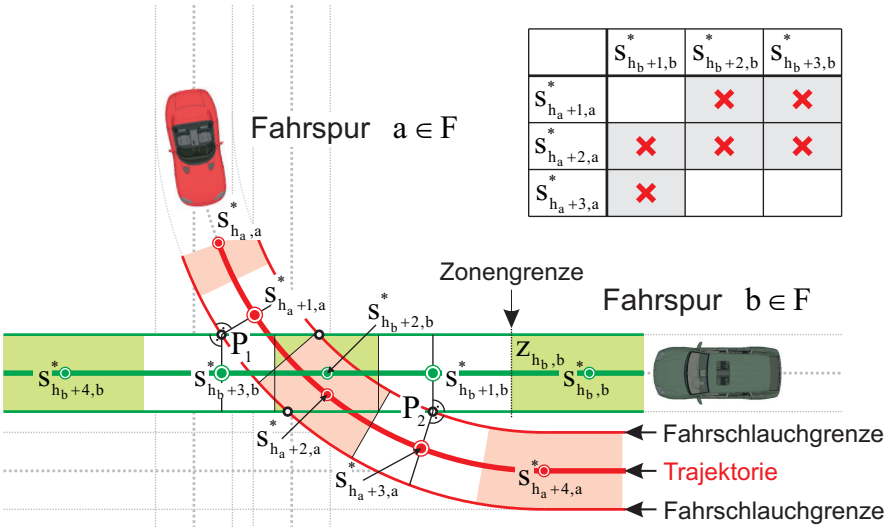

Abbildung 3.9: Berücksichtigung der Fahrzeugbreite

Zeitpunkt immer nur ein Fahrzeug hineinragen. Nur so können Kollisionen definitiv ausgeschlossen werden.

Nachdem die 4 Schnittpunkte der Fahrschläuche bestimmt wurden, werden nun die beiden Punkte ausgewählt, deren orthogonale Projektionen auf die Trajektorien beider Routen außerhalb der Schnittfläche liegen. Im Beispiel gemäß Abbildung 3.9 sind dies die Punkte P_1 und P_2. Ein Sonderfall liegt vor, wenn die Fahrschläuche einander orthogonal schneiden. In diesem Fall liegen die Projektionen jeweils genau auf der Begrenzung der Schnittfläche. An die Positionen dieser Projektionen werden nun Wegstützstellen $s^*_{h,f}$ der Routenprofile gelegt. Im Beispiel sind dies die Stützstellen $s^*_{h_a+1,a}$ und $s^*_{h_a+3,a}$ für die Route a und $s^*_{h_b+1,b}$ und $s^*_{h_b+3,b}$ für die Route b. Ebenfalls denkbar – und im Hinblick auf den Durchsatz wahrscheinlich auch sinnvoller – wäre es, die Wegstützstellen so zu legen, dass genau eine Zonengrenze duch die orthogonalen Projektionen verläuft. In der aktuellen Implementierung des Simulators ist dies jedoch noch nicht der Fall.

Je nachdem, wie fein eine Route diskretisiert werden soll, müssen nun zwischen diesen Stützstellen weitere Stützstellen platziert werden. Bei der Simulation eines Kreisverkehrs kann der Stützstellenabstand über die Benutzerschnittstelle konfiguriert werden. Bei der Modellierung einer Kreuzung müssen erheblich mehr und auch komplexere Kollisionsbeziehungen betrachtet

werden, die sich mit Änderung des Stützstellenabstands ebenfalls ändern. Ein Automatismus kann natürlich auch hier implementiert werden; dies wäre allerdings erheblich aufwändiger als im Falle eines Kreisverkehrs. Aus diesem Grund wurden für eine spezielle Kreuzungsgeometrie mit gegebenen Stützstellenabständen die Zonen und Kollisionsbeziehungen berechnet und im Simulator *hardcodiert*. Das in Abbildung 3.9 dargestellte Beispiel ist an die Diskretisierung der im Simulator verwendeten Kreuzungsgeometrie angelehnt: Exakt auf halbem Wege zwischen $s^*_{h_a+1,a}$ und $s^*_{h_a+3,a}$ wurde eine weitere Wegstützstelle $s^*_{h_a+2,a}$ platziert. Gleiches gilt entsprechend für die Route b.

Nachdem auf die beschriebene Art die jeweils relevanten Wegstützstellen $s^*_{h,f}$ festgelegt wurden, werden nun gemäß der Beziehung (3.4) die zu den Stützstellen gehörenden Zonen $z_{h,f}$ definiert. Im Anschluss sind die Kollisionsbeziehungen der Zonen im Schnittbereich zu analysieren. Für jede Zone der Route a muss also überprüft werden, mit welchen Zonen der Route b Schnittflächen existieren – und umgekehrt. Die Ergebnisse dieses Vorgehens für das Beispiel in Abbildung 3.9 sind in der dort ebenfalls dargestellten Tabelle zusammengefasst. Wenn sich also ein Fahrzeug auf der Route a bewegt und die Zone $z_{h_a+1,a}$ der Stützstelle $s^*_{h_a+1,a}$ belegt, so müssen aus Sicherheitsgründen ebenfalls die Zonen um $s^*_{h_b+2,b}$ und $s^*_{h_b+3,b}$ der Route b gesperrt werden.

Fahrzeugausdehnung in Längsrichtung unter Berücksichtigung der Dimension Zeit

Es wird angenommen, dass sich ein Fahrzeug mit beliebiger Länge L auf einer beliebigen Route $f \in F$ bewegt und dabei die Position seines geometrischen Mittelpunktes durch das Differenzengleichungssystem (3.31) determiniert wird. Die in Abbildung 3.7 dargestellten Trajektoriensegmente beschreiben dann also die Bewegung des geometrischen Mittelpunktes eines Fahrzeug in Folge einer jeweils vorgegebenen Eingangskraft $F_{u,k}$. Diese Trajektoriensegmente muss man nun auf die Knotenanordnung A_f projizieren, um dort zu prüfen, ob die für die Realisierung des jeweiligen Trajektoriensegments erforderlichen Knoten $\alpha_{f,h,i}$ bzw. Einzugsbereiche $\Theta_{f,h,i}$ nicht bereits durch andere Fahrzeuge belegt sind. Nur im Fall der Nichtbelegung sämtlicher Einzugsbereiche darf ein Trajektoriensegment bzw. eine Netzwerkkante e_k realisiert werden.

Um das genaue Vorgehen im Rahmen dieser Prüfung zu veranschaulichen, wurde aus der Abbildung 3.7 ein Trajektoriensegment herausgegriffen und

in der Abbildung 3.10 detailliert betrachtet. Das dargestellte Segment re-

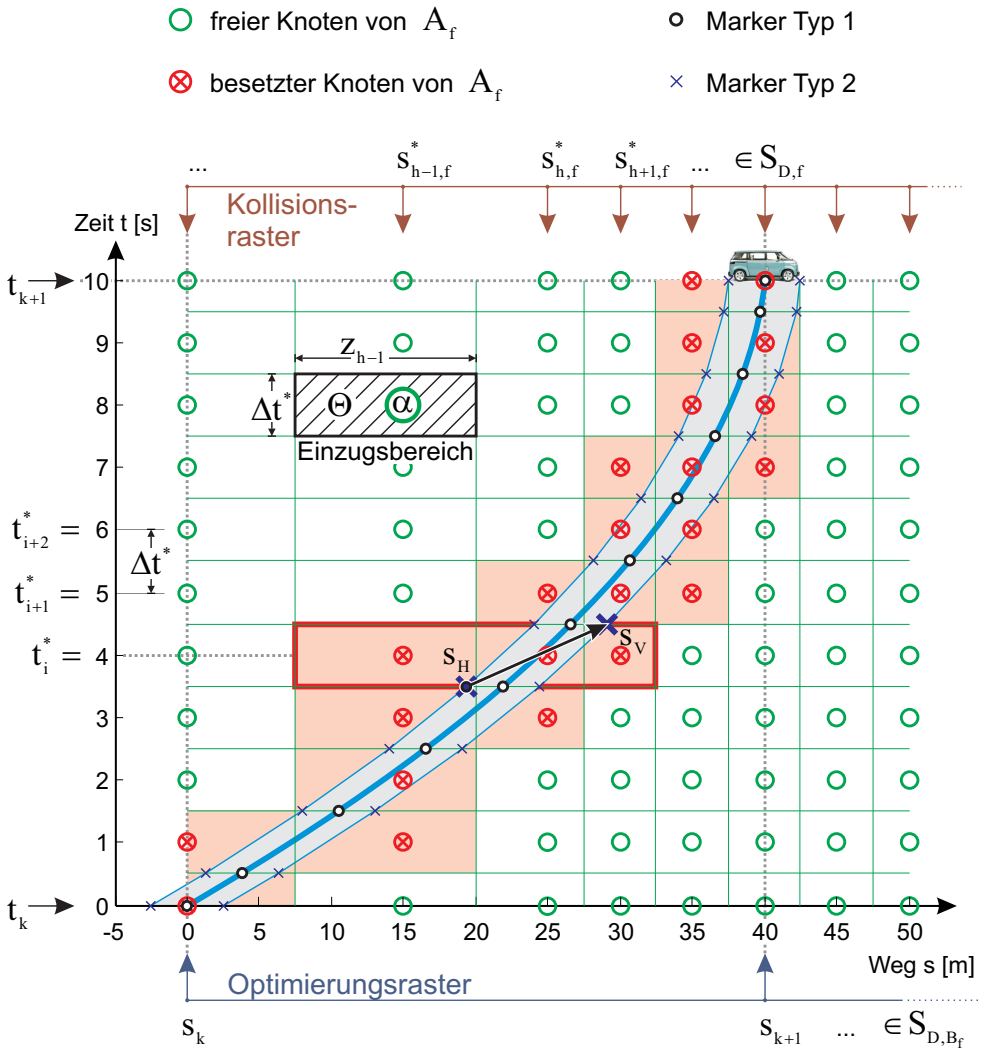

Abbildung 3.10: Systematik der Knotensperrung

präsentiert die zeitabhängige Position des geometrischen Mittelpunktes eines Fahrzeugs, das während der Zeit $T = t_{k+1} - t_k = 10\ s$ mit der konstanten Eingangskraft $F_{u,k} < 0$ verzögert wird und dabei den Weg $\Delta s = s_{k+1} - s_k = 40$ m zurücklegt. Nun werden für die Zeitpunkte $t_i^* + \frac{1}{2}\Delta t^*$, welche die zeitli-

chen Grenzen der Einzugsbereiche beschreiben, die zugehörigen Positionen des geometrischen Mittelpunktes berechnet (**Marker Typ 1** in Abbildung 3.10). Um die jeweiligen Positionen des Fahrzeughecks zu bekommen, muss man von den Positionen des Mittelpunktes lediglich die halbe Fahrzeuglänge ($L/2$) subtrahieren. Im Falle der Fahrzeugfront muss entsprechend addiert werden (**Marker Typ 2**).

Betrachtet man nun die Knoten $\alpha \in A_f$ eines beliebigen Zeitpunktes t_i^*, etwa des Zeitpunktes $t_i^* = 4\ s$, so befinden sich das Fahrzeugheck bezüglich der zugehörigen Einzugsbereiche Θ an der Positon s_H und die Fahrzeugfront an der Position s_V. Es müssen dann weiterhin sämtliche Knoten α dieses Zeitpunktes geprüft werden, deren größere örtliche Begrenzung ihres jeweiligen Einzugsbereichs Θ größer ist als s_H und deren kleinere örtliche Begrenzung gleichzeitig kleiner ist als s_V. Im betrachteten Beispiel sind dies die Knoten

$$\left\{ \alpha_{f,h,i} | t_i^* = 4,\ \frac{s_{h,f}^* + s_{h+1,f}^*}{2} > s_H,\ \frac{s_{h-1,f}^* + s_{h,f}^*}{2} \leq s_V \right\}, \tag{3.52}$$

$$\text{mit} \quad h = 1, 2, ...$$

Im Falle des Trajektoriensegments in Abbildung 3.10 wurde dieses Vorgehen für sämtliche Zeitpunkte

$$t_k \leq t_i^* \in T_D \leq t_{k+1} \tag{3.53}$$

durchgeführt und die betroffenen Knoten als *besetzt* markiert ($\omega(\alpha) := 1$). Die zu dem dargestellten Trajektoriensegment gehörende Netzwerkkante ist somit nur zulässig, wenn keiner dieser Knoten bereits zuvor durch ein anderes Fahrzeug belegt wurde. Soll ein Trajektoriensegment für ein Fahrzeug realisiert bzw. eingeplant werden, so muss man sämtliche durch oben beschriebenes Vorgehen betroffenen Knoten als besetzt bzw. belegt markieren, um sie dem Zugriff anderer Fahrzeuge zu entziehen und dadurch Kollisionen auszuschließen.

Man kann in Abbildung 3.10 anhand der gesperrten Knoten mit ihren dunkler markierten Einzugsbereichen auch gut erkennen, dass der Abstand von aufeinader folgenden Fahrzeugen, und damit auch der Durchsatz, stark von der Diskretisierung abhängt. Eine zu grobe Diskretisierung führt zu großen Fahrzeugabständen, die wiederum den Durchsatz verringern. Weiterhin sieht man, dass das Verhältnis von zeitlicher Diskretisierung und Wegdiskretisierung einen signifikanten Einfluss hat und somit sinnvoll gewählt werden sollte.

Eine zu feine Diskretisierung führt allerdings zu einem erhöhten Rechenaufwand, der ebenfalls den Durchsatz einschränken kann. Es ist also für das jeweilige Zielsystem ein optimaler Kompromiss zwischen geringem Rechenaufwand und feiner Diskretisierung anzustreben, der den Durchsatz maximiert.

Abbildung 3.10 verdeutlicht noch einen weiteren Zusammenhang: Der Fahrer eines Fahrzeugs ändert nicht mehrmals innerhalb kürzester Zeit die Gaspedalstellung oder wechselt sogar zwischen Gas- und Bremspedal hin und her. Genauso wenig ist es sinnvoll, die Eingangskraft $F_{u,k}$ für das Fahrzeug mehrmals innerhalb kürzester Zeit festzulegen. Aus diesem Grund wurde bei der Generierung des Trajektoriensegments in Abbildung 3.10 die Eingangskraft an der Wegstützstelle s_k festgelegt und bis zum Erreichen der Wegstützstelle s_{k+1} konstant gehalten. Um einen hohen Durchsatz am jeweiligen Verkehrsknotenpunkt sicherzustellen, müssen aber im Rahmen der Kollisionsvermeidung weitere Wegstützstellen berücksichtigt werden. Im dargestellten Beispiel sind dies die Stützstellen zwischen s_k und s_{k+1}. Diesen Stützstellen $s^*_{h,f} \in S_{D,f}$ des Kollisionsrasters werden im Netzwerk bzw. im *Optimierungsraum* $B_{D,f}$ keine Unterräume $B_{VT,h}$ zugeordnet (vgl. Abschnitt 3.3.1). Die Wegstützstellen s_k der Knotenmenge $B_{D,f}$ des Netzwerks bilden als in der Regel echte Teilmenge des Kollisionsrasters das Optimierungsraster, wobei stets $s_k \in S_{D,f}$ gilt.

4 Wunschfahrprofile als Ziele der Optimierung

Als Wunschfahrprofil wird eine Trajektorie bezeichnet, der ein Fahrzeug folgen würde, sofern es auf keine anderen Verkehrsteilnehmer Rücksicht nehmen muss. Diese Trajektorie kann das Ergebnis einer oder mehrerer Zielgrößenvorgaben sein, die vom jeweiligen Fahrzeug bzw. deren Insassen gemacht wurden.

Einerseits ist das Wunschfahrprofil damit selbst das Ergebnis einer Optimierung (vgl. Abschnitt 1.2.4), andererseits dient es aber auch als essentielle Grundlage für die Trajektorienoptimierung mit den in Kapitel 2 vorgestellten Verfahren aus dem Bereich der kombinatorischen Optimierung, indem es die Grundlage für die Berechnung der Gütewerte c_k von Trajektoriensegmenten bzw. Kanten e_k im Netzwerk bzw. Optimierungsraum $B_{D,f}$ bildet. Auf diese Systematik wird in Kapitel 5 noch ausführlich eingegangen.

In [22] wurde eine Methode für die Generierung von Wunschfahrprofilen vorgestellt, die auf dem Einsatz von Verfahren der nichtlinearen beschränkten Optimierung basiert. Das Problem wurde als Optimierungsproblem mit mehreren Zielgrößen und mehreren Parametern beschrieben, für dessen Lösung Abstiegsverfahren wie das Gradienten- oder das Quasi-Newton-Verfahren eingesetzt wurden. Als Zielgrößen für die Optimierung wurden Durchfahrtszeit, Kraftschlussbeanspruchung, Quer- und Längsbeschleunigung sowie Energieverbrauch festgelegt. Vor Beginn der Optimierung mussten die relevanten Zielgrößen ausgewählt und Unter- bzw. Obergrenzen für ihre jeweiligen Werte definiert werden. Optimiert wurde die Lage von Punkten einer Geschwindigkeits-Weg-Kurve ($v(s)$-Kurve), die entsteht, wenn man einen kubischen Spline durch diese Punkte legt. Die Weg-Koordinaten (Stützstellen) waren dabei fest vorgegeben, und die jeweils zugehörigen Geschwindigkeitskoordinaten (Stützwerte) bildeten die Parameter der Optimierung. Die Werte dieser Parameter definierten ein Fahrprofil, mit dem die komplette Kreuzungsdurchfahrt des

Fahrzeugs simuliert wurde. Während der Simulation wurden Zielgrößenwerte berechnet und am Ende der Simulation an das Optimierungsverfahren übergeben. Der Optimierungsprozess wurde beendet, sobald ein Parametersatz bestimmt wurde, bei dem sämtliche Werte der relevanten Zielgrößen innerhalb der jeweils gewählten Unter- und Obergrenzen optimal ausbalanciert waren. Das oben beschriebene Verfahren ist zwar prinzipiell einsetzbar, im Rahmen von Echtzeit-Anwendungen allerdings mit folgenden Nachteilen behaftet:

- Pro Optimierungsschritt müssen mehrere Simulationen der geplanten Kreuzungsdurchfahrt durchgeführt werden. Bei p Parametern sind mindestens p Simulationen je Optimierungsschritt erforderlich, bei einigen Verfahren sogar $2 \cdot p$ oder mehr. Diese Tatsache schlägt sich in erheblichem Rechenaufwand bzw. in erheblicher Berechnungsdauer nieder.

- A priori kann keine Aussage über die Anzahl der erforderlichen Optimierungsschritte getroffen werden. Je nach Beschaffenheit der Zielfunktionen kann nicht einmal festgestellt werden, ob das Verfahren überhaupt gegen das globale Optimum konvergiert, da der Optimierungsprozess prinzipiell auch in einem lokalen Minimum abgebrochen werden kann.

Aus diesen Gründen wird im Folgenden eine Methodik vorgestellt, bei der die gewünschten Werte der wesentlichen Zielgrößen direkt (oder indirekt) vorgegeben werden. Auf Basis dieser Vorgaben wird dann mit quadratischen Splines bei deterministischem und die Echtzeit-Ansprüche erfüllendem Zeitaufwand das Wunschfahrprofil bestimmt.

4.1 Ausgangssituation

Bei der Generierung von Fahrprofilen muss berücksichtigt werden, dass während Abbiegevorgängen und Kreisfahrten oder allgemein auf Bahnkurven bestimmte Geschwindigkeitsgrenzen nicht überschritten werden dürfen, da die Fahrsicherheit sonst gefährdet ist. Diese Geschwindigkeitsgrenzen hängen vom Ort und nicht von der Zeit ab. Die Kreuzungsüberquerung kann prinzipiell in drei Phasen unterteilt werden (vgl. Abbildung 4.1). Die erste Phase entspricht der Kreuzungseinfahrt. Bei der aktuellen Implementierung des Kreuzungssimulators (R2008b) ist dieser Phase ein Geradenstück

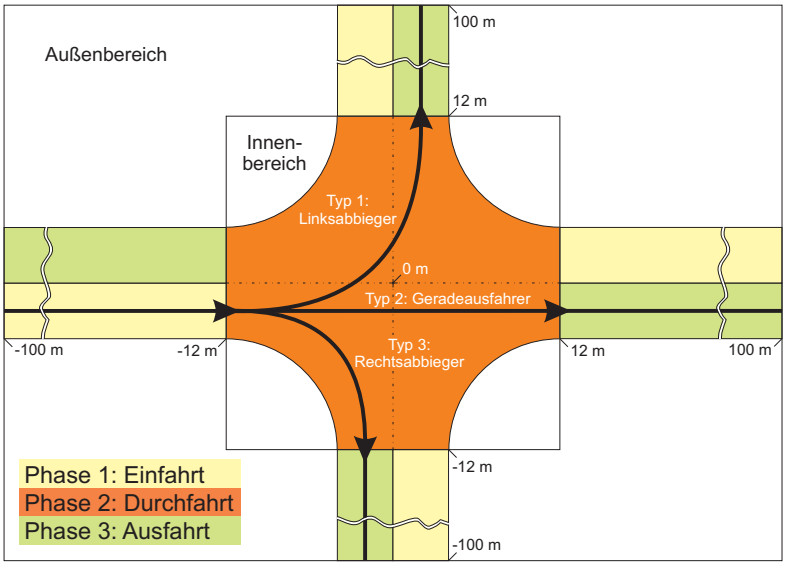

Abbildung 4.1: Phasen beim Passieren einer Kreuzung

mit einer Länge von 88 m zugeordnet. In dieser Phase reduziert das Fahrzeug gegebenenfalls seine Geschwindigkeit, falls in der zweiten Phase, der Durchfahrtsphase, eine Kurvendurchfahrt erfolgt. Bei der zugrunde gelegten Kreuzungsgeometrie wird bei den Abbiegerouten von einer konstanten Krümmung ausgegangen, das heißt die Fahrzeuge bewegen sich auf einem Kreisbogen. Bei realen Kreuzungsgeometrien wird eine Abbiegeroute in der Regel durch eine Klothoide oder einen dreiteiligen Korbbogen beschrieben. Bezüglich der in dieser Arbeit propagierten Lösungsidee für das Kreuzungsmanagement ist diese Tatsache aber ohne Belang. Die Kreuzungsgeometrie des Simulators kann prinzipiell beliebig aufwändig hinsichtlich der Anzahl von Fahrspuren und ihrer Verläufe gestaltet werden, ohne dass sich dies auf den Lösungsalgorithmus auswirkt. Wichtig ist lediglich, dass im Durchfahrtsbereich von den Fahrzeugen bestimmte Geschwindigkeiten, die von der jeweiligen Krümmung des gerade durchfahrenen Kurvensegments abhängen, nicht überschritten werden, um die Fahrsicherheit nicht zu gefährden. Der zurückgelegte Weg während der Durchfahrtsphase hängt von der Fahrtrichtung ab; bei Linksabbiegern beträgt er ca. 21, 2 m, bei Geradeausfahrern 24 m und bei Rechtsabbiegern ca. 16, 5 m. Am Ende der Durchfahrtsphase fährt das Fahrzeug wieder geradeaus und verlässt den Kreuzungsbereich. Dieser Phase ist ebenfalls ein Geradenstück von 88 m Länge zugeordnet. Der oben beschriebe-

ne Sachverhalt fasst im Wesentlichen die Ausgangssituation zusammen, vor deren Hintergrund für unterschiedliche Fahrzeuge Wunschfahrprofile erzeugt werden müssen. Ab der Version R2008b des Simulators ist weiterhin die Simulation eines Kreisverkehrs möglich, wobei der Radius des Innenkreisels frei vorgegeben werden kann. Die Systematik hier ist identisch mit der zuvor für die Kreuzung beschriebenen. Abbildung 4.2 verdeutlicht dies. Der wesentliche

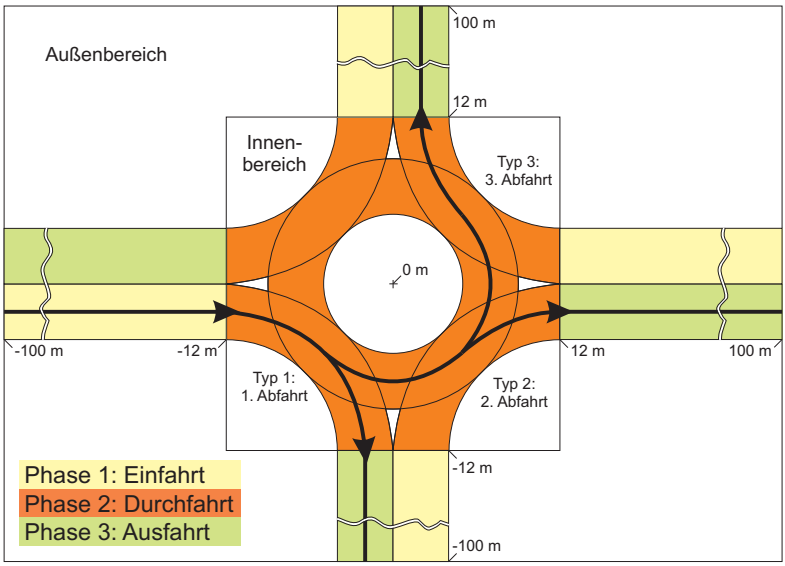

Abbildung 4.2: Phasen beim Passieren eines Kreisverkehrs

Unterschied besteht lediglich darin, dass sich die Längen der zurückgelegten Wege innerhalb der Durchfahrtsphase stärker unterscheiden.

4.2 Vereinfachte Fahrdynamik

Die speziellen fahrdynamischen Eigenschaften der am Kreuzungsmanagement beteiligten Fahrzeuge müssen sowohl bei der Generierung der Wunschfahrprofile als auch bei der Bestimmung der optimalen Trajektorie berücksichtigt werden. Bei der Bestimmung der Wunschfahrprofile werden diese dynamischen Eigenschaften in vereinfachter, aber vollkommen ausreichender Weise berücksichtigt. Dabei wird auf den Ausführungen aus Abschnitt 3.2 aufgebaut.

Längsdynamik

Die wichtigste Kenngröße bei der Betrachtung der Längsdynamik ist die Längsbeschleunigung bzw. ihr zeitlicher Verlauf. Anhand dieser Kenngröße lässt sich direkt das Komfortempfinden der Fahrzeuginsassen beurteilen. Weiterhin können anhand des zeitlichen Verlaufs der Längsbeschleunigung und bei Kenntnis des Streckenverlaufs Aussagen zur Durchfahrtszeit und zum Kraftstoffverbrauch gemacht werden. Generell zeichnen sich Fahrzeuge bei maximal möglicher Beschleunigung durch ein Beschleunigungsverhalten aus, das mit zunehmender Geschwindigkeit abnimmt (vgl. Abschnitt 3.2, Abbildung 3.4). Diese Abnahme wird bei der Bestimmung der Wunschtrajektorie intervallweise als linear angenommen (konstanter *Ruck*), wobei die Genauigkeit im Hinblick auf das tatsächliche Beschleunigungsverhalten bei maximal möglicher Beschleunigung mit kleiner werdender Intervalllänge zunimmt. Der Hauptgrund für diesen Ansatz besteht darin, dass eine lineare Änderung der Beschleunigung von Fahrern als angenehm und eine Veränderung in wahrnehmbaren Stufen als unangenehm empfunden wird. Die zeitliche Ableitung der Beschleunigung wird als Ruck bezeichnet. Die mögliche Beschleunigung und der zugehörige Ruck in Abhängigkeit einer gegebenen Geschwindigkeit und einer gegebenen Antriebskraft werden aus den Daten abgeleitet, die auch die Grundlage der in Abschnitt 3.2 eingeführten Fahrzeugmodellierung bilden. Es ist aber auch denkbar, diese Daten in Kennfeldern zu speichern. Anhand dieser Kennfelder können dann für bestimmte Fahrsituationen zulässige Rücke und Beschleunigungen festgelegt werden. Im Rahmen der im Folgenden vorgestellten Methodik werden für einige ortsfeste Wegstützstellen eines Geschwindigkeitsverlaufs ($v(s)$-Kurve) maximale Beschleunigungen vorgegeben, die vom jeweiligen Fahrzeug im jeweiligen Geschwindigkeitsbereich problemlos realisiert werden können. In gleicher Weise kann dieser Sachverhalt auch auf den Vorgang der Verzögerung (negative Beschleunigung) übertragen werden. Für die Bestimmung von Wunschfahrprofilen ist diese vereinfachte Berücksichtigung der Fahrdynamik völlig ausreichend, was in den folgenden Abschnitten noch deutlich wird.

Querdynamik

Bei der Betrachtung der Querdynamik ist die wichtigste Kenngröße die Quer- bzw. Zentripetalbeschleunigung a_Q. Sie lässt sowohl die direkte Beurteilung

des Komfortempfindens als auch Aussagen zur Fahrstabilität bzw. zur Fahrsicherheit zu.

Erst die Reibungskraft zwischen Reifen und Fahrbahn ermöglicht es dem Fahrzeug, die für die Kurvendurchfahrt erforderliche Zentripetalkraft aufzubringen. Die maximal mögliche Reibungskraft errechnet sich aus dem Produkt von Haftreibungskoeffizient μ_H und Normalkraft F_N, die hier der Gewichtskraft des Fahrzeugs entspricht. Daraus folgt näherungsweise für die maximale Haftreibungskraft

$$F_{R,max} = \mu_H \cdot F_N = \mu_H \cdot m \cdot g. \tag{4.1}$$

Einige Richtwerte für Haftreibungszahlen in Abhängigkeit unterschiedlicher Materialpaarungen sind in der folgenden Tabelle[1] gegeben. Das Fahrzeug

Materialpaarungen	Haftreibungszahl μ_H
Gummireifen auf Asphalt, trocken	≤ 0.9
Gummireifen auf Asphalt, nass	≤ 0.5
Gummireifen auf Beton, trocken	≤ 1.0
Gummireifen auf Beton, nass	≤ 0.6

Tabelle 4.1: Richtwerte für Haftreibungen

droht auszubrechen bzw. ins Schleudern zu geraten, wenn die erforderliche Zentripetalkraft für die Durchfahrt der Kurve nicht mehr durch den Kraftschluss zwischen Reifen und Fahrbahn aufgebracht werden kann. Lässt man in einem ersten Ansatz die Besonderheiten des jeweiligen Fahrzeugs wie Schwerpunktlage und Fahrwerk außer Acht, so wird dies spätestens der Fall sein, wenn die erforderliche Zentripetalkraft die maximale Haftreibungskraft übersteigt bzw. die Kraftschlussbeanspruchung

$$\sigma = \frac{F_Z}{F_{R_{\max}}} = \frac{v^2 \cdot \kappa}{g \cdot \mu_H} = \frac{a_Q}{g \cdot \mu_H} \tag{4.2}$$

den Wert von 1 übersteigt. Möchte man beispielsweise eine Sicherheit von ca. 30 % auf trockener Fahrbahn realisieren, so entspräche das einer maximalen Durchfahrtsgeschwindigkeit von ca. 8 m/s auf der Rechtsabbiegerspur

[1] Werte gemäß `http://de.wikipedia.org/`

(10, 5 m Krümmungsradius) und ca. 9 m/s auf der Linksabbiegerspur (13, 5 m Krümmungsradius). Die zugehörige Zentripetalbeschleunigung liegt in diesem Fall bei ca. 6 m/s^2 bzw. bei ca. 0, 63 g. Wirkt gleichzeitig zur Quer- auch eine Längsbeschleunigung a_L, so ist die Kraftschlussbeanspruchung durch

$$\sigma = \frac{\sqrt{a_Q^2 + a_L^2}}{g \cdot \mu_H} \tag{4.3}$$

gegeben.

4.3 Methodik der Generierung

Die hier vorgestellte Methodik basiert auf der Verwendung von quadratischen Splines für die Definition von Wunschfahrprofilen. Der Grund für die Wahl von quadratischen Splines besteht einerseits darin, dass die damit verbundene Berechnung der Koeffizienten sehr einfach erfolgen kann. Andererseits wird eine lineare Änderung der Beschleunigung vom Menschen als angenehm empfunden. Eine lineare Beschleunigungsänderung bedeutet, dass die Ableitung der Beschleunigungsfunktion – die Ruckfunktion – intervallweise konstant ist. Genau diese Eigenschaft wird durch den Einsatz quadratischer Splines erreicht. Die folgende Tabelle verdeutlicht diesen Zusammenhang. Vor diesem

	Quadratischer Spline	Physikalische Funktion
Geschwindigkeit	$v(t) = k_0 + k_1 t + k_2 t^2$	$v(t) = v_0 + a_0 t + \frac{1}{2} r_0 t^2$
Beschleunigung	$a(t) = k_1 + 2k_2 t$	$a(t) = a_0 + r_0 t$
Ruck	$r(t) = 2k_2 = \text{konst.}$	$r(t) = r_0 = \text{konst.}$

Tabelle 4.2: Physikalische Funktion und quadratischer Spline

Hintergrund bietet es sich an, die Kreuzungsdurchfahrt in mehrere Zeitfenster bzw. Zeitintervalle aufzugliedern, deren Lage und Größe jeweils von den örtlichen Gegebenheiten und den jeweiligen Fahrzeugeigenschaften abhängen. Die Methodik ist sowohl für geradeausfahrende als auch für abbiegende Fahrzeuge geeignet, das heißt das Geschwindigkeitsniveau im Innenbereich

der Kreuzung kann sowohl angehoben als auch gesenkt werden. Letzteres ist insbesondere bei abbiegenden Fahrzeugen erforderlich.

Ein Wunschfahrprofil für Abbieger sieht prinzipiell so aus wie das in Abbildung 4.3 dargestellte Profil. Das Passieren einer Kreuzung oder eines Kreis-

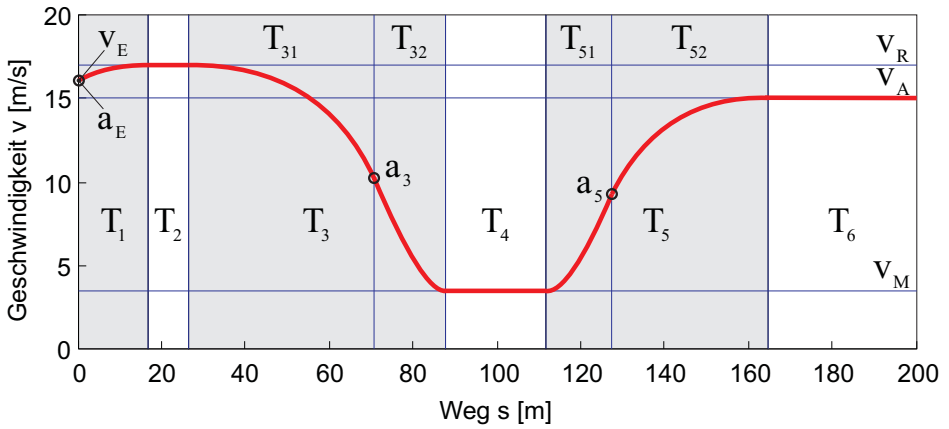

Abbildung 4.3: Prinzipielle Struktur von Wunschfahrprofilen

verkehrs wird in die sechs Zeitintervalle T_1 bis T_6 untergliedert, wobei T_1, T_2 und T_3 die Einfahrt (Phase 1, Abbildung 4.1 bzw. Abbildung 4.2) beschreiben, T_4 die Durchfahrt des Innenbereichs (Phase 2) und letztlich T_5 und T_6 die Ausfahrt (Phase 3). Anhand von definierten Freiheitsgraden bzw. Parametern kann das Wunschfahrprofil für diese Intervalle individuell eingestellt werden. Für die Intervalle T_1 und T_2 ist dies der konstante Ruck r_1, mit dem das Fahrzeug innerhalb der Zeit T_1 auf die sich aus der Vorgabe des Rucks ergebende konstante Geschwindigkeit v_R gebracht wird. Mit dieser Geschwindigkeit wird das Intervall T_2 durchfahren. Die Vorgabe der maximalen Verzögerung bzw. Beschleunigung (a_3 und a_5) legt das Profil für die Intervalle T_3, T_5 und T_6 fest. Der Profilverlauf im Intervall T_4 wird durch die Geschwindigkeitsvorgabe v_M determiniert. Bei geradeaus fahrenden Fahrzeugen erfolgt die Vorgabe von v_M direkt, während im Falle von Fahrzeugen, die sich auf einer Bahnkurve bewegen, die maximale Querbeschleunigung a_Q vorgegeben und anhand der Beziehung (3.36) auf die Geschwindigkeit v_M umgerechnet wird. Das Wunschfahrprofil eines Fahrzeugs wird somit durch Vorgabe der vier Parameter r_1, v_M (bzw. a_Q), a_3 und a_5 exakt festgelegt. Es bleibt noch die Frage, wie die Fahrerpräferenzen am besten in die Generierung der Wunschfahrprofile einfließen können. Zu diesem Zweck wird ein Ansatz

gewählt, der auch in der Realität als sehr praktikabel erscheint: Es wird ein *Fahrerpräferenz-Parameter* W eingeführt, mit $0 \leq W \leq 1$, der vom Fahrer beispielsweise über einen Schieberegler vorgegeben bzw. eingestellt werden kann. Dieser Parameter dient dazu, die zuvor beschriebenen vier Parameter innerhalb ihrer Gültigkeitsbereiche eindeutig festzulegen. Eine ausführliche Beschreibung der Systematik wird im Folgenden gegeben.

Sobald sich ein Fahrzeug in den Einfahrtsbereich bewegt, wird dem Fahrer die Kontrolle über sein Fahrzeug entzogen. Zu diesem Zeitpunkt besitzt das Fahrzeug die Eintrittsgeschwindigkeit v_E und die Eintrittsbeschleunigung a_E. Die Eintrittsbeschleunigung soll nun bei einem vorgegebenen, konstanten Ruck r_1 innerhalb der Zeit T_1 auf Null reduziert werden. Damit sind nun drei Bedingungen gegeben, mit denen sich die drei Koeffizienten eines quadratischen Splines berechnen lassen. Es folgt daraus für die Geschwindigkeit die Beziehung

$$v_1(t_1) = v_E + a_E t_1 - \frac{1}{2}\frac{a_E}{T_1} t_1^2 \quad \text{mit} \quad t_{1,0} \leq t_1 \leq T_1. \tag{4.4}$$

An dieser Stelle sei angemerkt, dass in jedem der Zeitintervalle mit einer *lokalen* Zeitrechnung gerechnet wird. So tritt ein Fahrzeug beispielsweise zum (*lokalen*) Zeitpunkt $t_{3,0} = 0$ in das Intervall T_3 ein und verlässt es wieder nach genau T_3 Sekunden. Diese Systematik mag auf den ersten Blick etwas verwirren, führt aber zu deutlich einfacheren Gleichungen.

Die Beziehung (4.4) lässt sich in eine Darstellung in Abhängigkeit des Rucks r_1 überführen:

$$v_1(t_1) = v_E + a_E t_1 + \frac{1}{2} r_1 t_1^2 \quad \text{mit} \quad r_1 = -\frac{a_E}{T_1}. \tag{4.5}$$

Der Ruck r_1 dient der Parametrierung dieses ersten Segments des Wunschfahrprofils (Zeitintervall T_1). Sein Vorzeichen ist dem von a_E entgegengesetzt. Am Ende des Segments hat das Fahrzeug den Weg

$$s_1 = \frac{a_E \left(\frac{1}{3} a_E^2 - r_1 v_E\right)}{r_1^2} \tag{4.6}$$

zurückgelegt, seine Beschleunigung auf Null reduziert und dabei die Geschwindigkeit

$$v_R = v_E - \frac{1}{2}\frac{a_E^2}{r_1} \tag{4.7}$$

erreicht (Abbildung 4.3). Sie ist die Geschwindigkeit jeweils am Anfang der Zeitintervalle T_2 und T_3. Während des Zeitintervalls T_2 soll das Fahrzeug mit konstanter Geschwindigkeit fahren. Im Zeitintervall T_3 hingegen wird das Fahrzeug mit konstantem Ruck auf die Geschwindigkeit v_M gebracht. Die maximale Beschleunigung (bzw. Verzögerung) a_3 im Intervall T_3 wird genau nach $T_{31} = \frac{1}{2}T_3$ Sekunden erreicht. Nach weiteren $T_{32} = \frac{1}{2}T_3$ Sekunden erreicht die Beschleunigung (bzw. Verzögerung) wieder den Wert Null, exakt am Ende des Außenbereichs bzw. nach s_E Metern und bei der Geschwindigkeit v_M, mit der dann der Innenbereich der Kreuzung bzw. des Kreisverkehrs durchfahren wird. Die Parametrierung des Zeitintervalls T_3 erfolgt mit den Parametern v_M und a_3, wobei die beiden folgenden Restriktionen beachtet werden müssen:

- Um die Fahrstabilität bzw. -sicherheit des Fahrzeugs garantieren zu können, darf die Geschwindigkeit v_M einen Maximalwert $v_{M,max}$ bei abbiegenden Fahrzeugen nicht übersteigen. Weiterhin muss jedes Fahrzeug innerhalb eines gegebenen Zeitfensters die Kreuzung passiert haben, was wiederum zur Folge hat, dass ein Minimalwert $v_{M,min}$ nicht unterschritten werden darf. Zwischen diesen beiden Grenzwerten kann der Parameter v_M frei eingestellt werden. Geht der Parameterwert in Richtung $v_{M,max}$, so wird die Zielgröße Durchfahrtszeit verbessert, während die Zielgröße Komfort durch eine Erhöhung der Zentripetalbeschleunigung verschlechtert wird – und umgekehrt. Bei geradeaus fahrenden Fahrzeugen entfällt diese durch die Querdynamik bedingte Restriktion, so dass prinzipiell auch der Fall $v_{M,max} > v_R$ zulässig ist.
- Der Wert a_3 der Beschleunigung (Abbildung 4.3) muss so gewählt werden, dass einerseits die zulässige maximale Verzögerungsfähigkeit $a_{B,max}$ (Fall $v_M < v_R$) bzw. die zulässige maximale Beschleunigungsfähigkeit $a_{A,max}$ (Fall $v_M > v_R$) des jeweiligen Fahrzeugs nicht überschritten wird. Andererseits darf die Summe der in den Zeitintervallen T_1 bis T_3 zurückgelegten Wege die Länge der Kreuzungseinfahrt s_E (im Beispiel 88 m) nicht überschreiten. Bei dem aus der letzten Forderung resultierenden Grenzwert $a_{3,Gr}$ wird die Länge des Zeitintervalls T_2 zu Null, und das Fahrzeug erreicht exakt am Ende des Einfahrtsbereiches die Geschwindigkeit v_M. Im Falle von $v_M < v_R$ ist $a_{3,Gr}$ die kleinste zulässige Verzögerung, im Falle von $v_M > v_R$ die kleinste zulässige Beschleunigung. Geht der Parameterwert a_3 im Falle von $v_M < v_R$ in Richtung $a_{B,max}$, so wird die Zielgröße Durchfahrtszeit verbessert, da länger mit der relativ hohen Geschwindigkeit v_R gefahren werden kann ($T_2 > 0$). Das Fahrzeug bremst erst später, dafür aber stärker. Stärkeres Brem-

sen bzw. eine höhere Verzögerung hingegen verschlechtert die Zielgröße Komfort. Der hier beschriebene Sachverhalt gilt analog auch für das Zeitintervall T_5.

Für das konkrete Vorgehen folgt daraus, dass zuerst die Grenzwerte $v_{M,min}$ und $v_{M,max}$ bekannt sein müssen. Im Anschluss werden dann anhand der gewählten Geschwindigkeit v_M die Grenzwerte $a_{3,min}$ und $a_{3,max}$ bestimmt sowie a_3 gewählt. Der Grenzwert $v_{M,min}$ bzw. die Mindestgeschwindigkeit sämtlicher Fahrzeuge während der Durchfahrt des jeweiligen Verkehrsknotenpunktes wird entsprechend der Simulatorkonfiguration einmal für alle Fahrzeuge festgelegt. Der Grenzwert $v_{M,max}$ wird bei abbiegenden Fahrzeugen im Wesentlichen durch die Oberflächenbeschaffenheit der Straße und durch fahrzeugindividuelle Parameter wie zum Beispiel Schwerpunktlage und Fahrwerkskinematik bestimmt. Im ersten Schritt wird nun v_M aufgrund der jeweiligen Zielgrößenpräferenz festgelegt (Abschnitt 4.2). Möchte man beispielsweise auf einer Abbiegespur mit der Krümmung κ die Zentripetalbeschleunigung a_Q realisieren, so muss gemäß Formel (3.36)

$$v_M = \sqrt{\frac{a_Q}{\kappa}} \tag{4.8}$$

gewählt werden. Der Grenzwert $a_{3,max}$,

$$\begin{aligned} a_{3,max} > 0 \quad &\Longrightarrow \quad a_{3,max} := a_{A,max}, \\ a_{3,max} < 0 \quad &\Longrightarrow \quad a_{3,max} := a_{B,max}, \end{aligned}$$

ist durch die Fahrzeugeigenschaften (Motorleistung, Bremssystem etc.) und die daraus resultierenden maximalen Beschleunigungs- und Verzögerungsfähigkeiten des jeweiligen Fahrzeugs im Geschwindigkeitsbereich bis ca. $15-20$ m/s festgelegt, während der Grenzwert $a_{3,Gr}$ nach Wahl von v_M gemäß (4.12) berechnet wird. Sind beide Größen bekannt, so kann die Einfahrt in den Verkehrsknotenpunkt durch direkte bzw. indirekte Vorgabe der Zielgrößenausprägungen mittels der Parameter r_1, v_M (bzw. a_Q bei abbiegenden Fahrzeugen) und a_3 festgelegt werden.

Das Geschwindigkeitsprofil im Zeitintervall T_3 wird mit zwei quadratischen Splines bestimmt. Dazu müssen sechs Bedingungen angeben werden, um die insgesamt sechs Koeffizienten der beiden Splines zu bestimmen. Im Übergangspunkt bei $\frac{1}{2}T_3$ müssen beide Splines den gleichen Funktionswert und die gleiche Steigung besitzen (2 Bedingungen); der erste Spline startet mit dem Funktionswert v_R und hat die Steigung Null (2 Bedingungen), und der

zweite Spline endet mit dem Funktionswert v_M und soll an dieser Stelle ebenfalls die Steigung Null besitzen (2 Bedingungen). Damit ergeben sich für den ersten Spline die Gleichung

$$v_{31}(t_{31}) \;=\; v_R + \frac{2\,(v_M - v_R)}{T_3^2} t_{31}^2, \quad t_{31,0} \le t_{31} \le T_{31}, \tag{4.9}$$

und für den zweiten die Gleichung

$$v_{32}(t_{32}) \;=\; \frac{v_R + v_M}{2} - \frac{2\,(v_R - v_M)}{T_3} t_{32} + \frac{2\,(v_R - v_M)}{T_3^2} t_{32}^2, \tag{4.10}$$

$$t_{32,0} \le t_{32} \le T_{32}.$$

Aus der weiteren Forderung, dass nach $T_{31} = \frac{1}{2}T_3$ Sekunden die Steigung genau a_3 betragen soll, kann man die folgende Definition für T_3 ableiten:

$$T_3 = \frac{2\,(v_M - v_R)}{a_3}. \tag{4.11}$$

Letztlich muss aber noch eine weitere Forderung erfüllt sein. Diese besteht darin, dass die Länge s_E der Kreuzungseinfahrt exakt zurückgelegt wurde, wenn das Zeitfenster T_4 erreicht wird. Wenn das Fahrzeug mit minimal zulässiger Geschwindigkeitsänderung $a_{3,min}$ fährt, dann schrumpft das Zeitfenster T_2 auf Null zusammen. Mit den Integralen der Gleichungen (4.9) und (4.10) kann man dann die folgende Formel für den Grenzwert $a_{3,Gr}$ bestimmen:

$$a_{3,Gr} = \frac{v_M^2 - v_R^2}{s_E - s_1}. \tag{4.12}$$

Prinzipiell müssen an dieser Stelle die folgenden 3 Fälle unterschieden werden:

1. Fall: $v_M > v_R \implies a_{3,Gr} > 0 \implies a_{3,Gr} \le a_3 \le a_{A,max}$.
2. Fall: $v_M < v_R \implies a_{3,Gr} < 0 \implies a_{B,max} \le a_3 \le a_{3,Gr}$.
3. Fall: $v_M = v_R \implies a_3 = 0$.

Eine stärkere Geschwindigkeitsänderung $|a_{3,Gr}| < |a_3|$ führt dazu, dass aus $s_1 + s_3 = s_E$ die Ungleichung $s_1 + s_3 < s_E$ wird. In diesem Fall gilt $T_2 > 0$. Da innerhalb des Zeitfensters T_2 mit der konstanten Geschwindigkeit v_R gefahren werden soll, ergibt sich T_2 zu

$$T_2 = \frac{(s_E - s_1 - s_3)}{v_R}. \tag{4.13}$$

Das Wegstück s_3 wird mit der Beziehung

$$s_3 = \frac{v_M^2 - v_R^2}{a_3} \tag{4.14}$$

berechnet. Zu Beginn und während des Zeitfensters T_4 besitzt das Fahrzeug die Geschwindigkeit v_M. Mit dieser Geschwindigkeit wird der Innenbereich der Kreuzung durchfahren. Die dafür erforderliche Zeit beträgt

$$T_4 = \frac{s_M}{v_M} \tag{4.15}$$

Sekunden, wobei s_M der Länge der Bahnkurve im Innenbereich entspricht.

Die Ausfahrt aus der Kreuzung erfolgt analog zu dem beschriebenen Vorgehen für die Einfahrt. Der Unterschied besteht lediglich in der Vereinfachung, dass sämtliche Fahrzeuge, welche die Kreuzung verlassen, eine identische Geschwindigkeit v_A (15 m/s) und Beschleunigung Null haben sollen. Dies liegt daran, dass am Ende der Kreuzungsausfahrt die Kontrolle über die Fahrzeuge wieder an die Fahrer übergeben wird. Würde beispielsweise auf der selben Ausfahrt einem stark verzögernden Fahrzeug mit geringer Geschwindigkeit ein beschleunigendes Fahrzeug mit hoher Geschwindigkeit folgen, so könnte der Fahrer im Zeitpunkt der Übergabe nicht mehr rechtzeitig reagieren, um ein Auffahren zu verhindern. Durch die Strategie, sämtliche Fahrzeuge mit identischer Geschwindigkeit und Beschleunigung Null aus der Kreuzung zu schleusen bzw. zu entlassen, wird diese Gefahr weitestgehend entschärft.

Für die Bestimmung des Ausfahrtssegments der Trajektorie sind also die Geschwindigkeiten v_M und v_A sowie die Länge der Ausfahrt s_A ($s_A = s_E$ beim derzeitigen Simulator) gegeben. Der Verlauf der Geschwindigkeit im Zeitintervall T_5 wird wieder mit zwei quadratischen Splines beschrieben, die, auf das Intervall T_5 bezogen, ihre größte Steigung nach $T_{51} = \frac{1}{2}T_5$ Sekunden haben. Diese Steigung entspricht der vorgegebenen Beschleunigung (bzw. Verzögerung) a_5. Auch hier muss a_5 wieder innerhalb der Grenzen $a_{5,Gr}$ und $a_{5,max}$ gewählt werden. Bei $a_{5,Gr}$ wird nach s_A Metern exakt die vorgegebene Ausfahrtsgeschwindigkeit v_A erreicht. Für das Zeitfenster T_6 gilt in diesem Fall $T_6 = 0$. $a_{5,Gr}$ ist durch die Formel

$$a_{5,Gr} = \frac{v_A^2 - v_M^2}{s_A} \tag{4.16}$$

gegeben. Sobald $|a_{5,Gr}| < |a_5| \leq |a_{5,max}|$ gilt, gilt ebenfalls $T_6 > 0$, da $s_5 < s_A$. Der zurückgelegte Weg s_5 innerhalb des Zeitfensters beträgt

$$s_5 = \frac{v_A^2 - v_M^2}{a_5} \tag{4.17}$$

Meter. Der Geschwindigkeitsverlauf im Zeitfester T_5 ist in Anlehnung an das für die Einfahrt beschriebene Vorgehen durch die Gleichungen

$$v_{51}(t_{51}) \;=\; v_M + \frac{2\,(v_A - v_M)}{T_5^2} t_{51}^2, \quad t_{51,0} \leq t_{51} \leq T_{51}, \tag{4.18}$$

und

$$v_{52}(t_{52}) \;=\; \frac{v_M + v_A}{2} - \frac{2\,(v_M - v_A)}{T_5} t_{52} + \frac{2\,(v_M - v_A)}{T_5^2} t_{52}^2, \tag{4.19}$$

$$t_{52,0} \leq t_{52} \leq T_{52},$$

gegeben, mit

$$T_5 = \frac{2\,(v_A - v_M)}{a_5}. \tag{4.20}$$

Für den Fall $T_6 > 0$ gilt ferner

$$T_6 = \frac{(s_A - s_5)}{v_A}. \tag{4.21}$$

Für die Berücksichtigung der Fahrerpräferenzen stehen gemäß obigen Ausführungen die vier Parameter r_1, v_M (bzw. a_Q), a_3 und a_5 zur Verfügung. Weiterhin sind für diese Parameter Unter- und Obergrenzen bekannt, innerhalb derer der jeweilige Parameter variiert werden darf. So kann zum Beispiel der Parameter a_3 innerhalb der Grenzen $a_{3,Gr}$ und $a_{3,max}$ gewählt werden. Über die lineare Transformation

$$a_3(W) = a_{3,Gr} + (a_{3,max} - a_{3,Gr}) \cdot W \tag{4.22}$$

wird nun der Fahrerpräferenz-Parameter W, mit $0 \leq W \leq 1$, auf den Wertebereich des Parameters a_3 abgebildet. Analog zu diesem Beispiel wird auch bei allen anderen Parametern verfahren. Auf die beschriebene Art wird die Parametrierung des Wunschfahrprofils für den Fahrer auf die Festlegung eines einzigen Parameters reduziert.

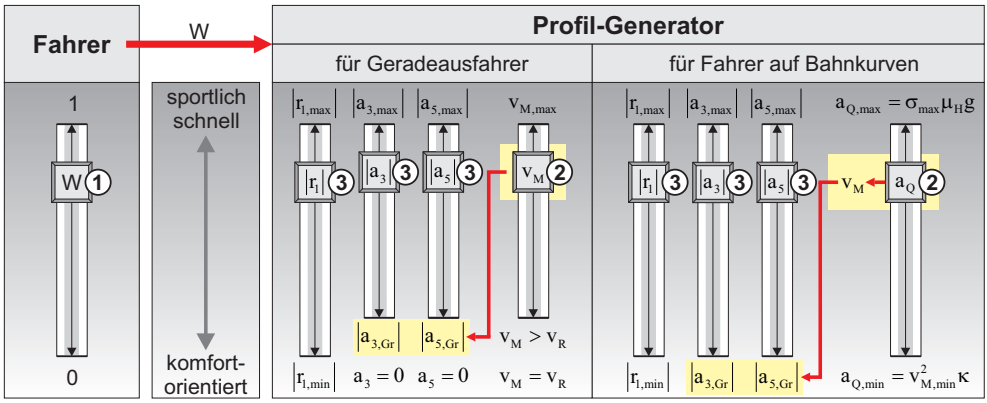

Abbildung 4.4: Adaption der Profilgenerierung an den Fahrer

Die bisherigen Ausführungen werden in Abbildung 4.4 zusammengefasst: Im ersten Schritt (siehe ①) gibt der Fahrer den Parameter W vor, der seine Präferenz bezüglich der Überquerung des Verkehrsknotenpunktes zwischen komfortorientiert bzw. energiesparend (0) und sportlich/schnell (1) einordnet. Dieser Wert wird an den Profilgenerator übergeben. Dort wird dann im zweiten Schritt (siehe ②) die Durchfahrtsgeschwindigkeit v_M für den Innenbereich gemäß dem zuvor beschriebenen Beispiel durch eine lineare Abbildung von W auf den Gültigkeitsbereich von v_M festgelegt. Nach der Bestimmung von v_M werden die Grenzwerte für die Beschleunigungen bzw. Verzögerungen a_3 und a_5 berechnet. Im dritten und letzten Schritt (siehe ③) werden r_1, a_3 und a_5 ebenfalls auf Basis einer Transformation von W berechnet. Somit sind nun sämtliche Parameter des Wunschfahrprofils festgelegt.

Abbildung 4.5 zeigt zum Vergleich zwei unterschiedliche Wunschfahrprofile, die nach der oben beschriebenen und implementierten Systematik erzeugt wurden.

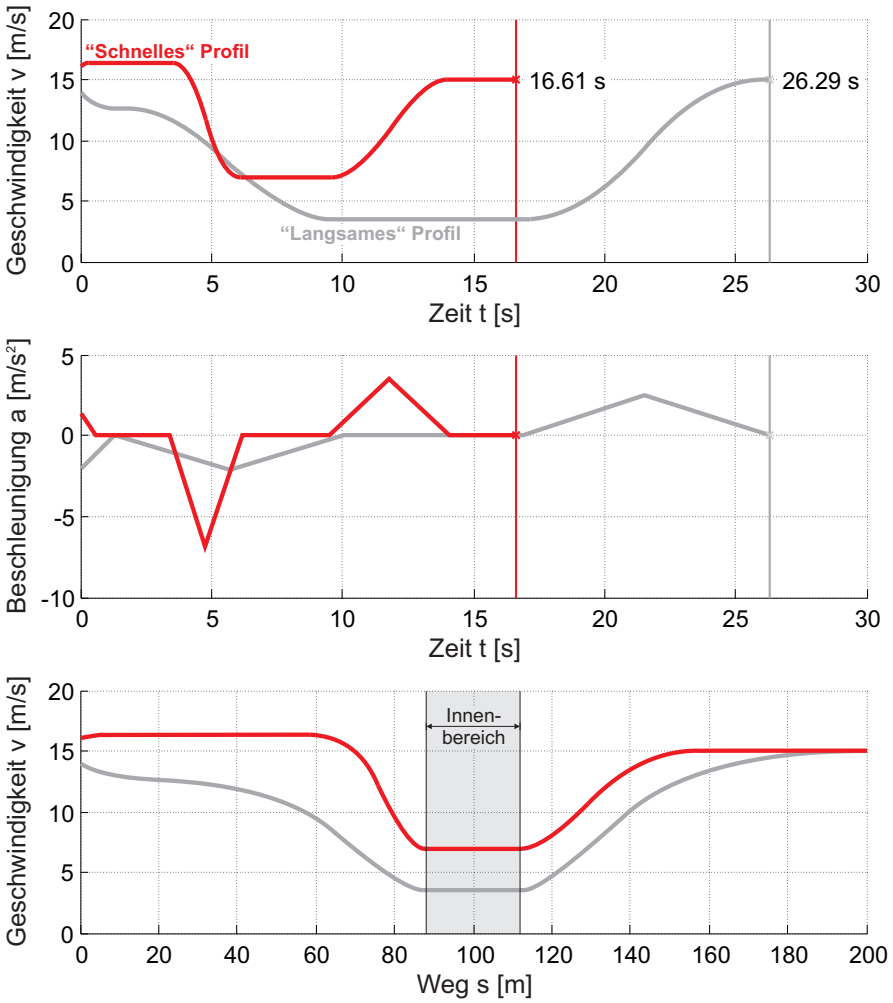

Abbildung 4.5: Vergleich unterschiedlicher Wunschfahrprofile

5 Strukturierung und Lösung des Netzwerkflussproblems

In Kapitel 3 wurde gezeigt, wie man die Vorgänge im Kreuzungsbereich modellieren kann, um sie als Netzwerkflussproblem darzustellen. Die Knoten des Netzwerks sind durch die Knotenmenge $B_{D,f}$ gegeben. Diese Menge spannt einen Raum mit den Dimensionen Weg (s), Geschwindigkeit (v) und Zeit (t) auf. Die Trajektorie eines Fahrzeugs wird vorwärtsschreitend entlang der Wegstützstellen s_k dieses Raumes (Optimierungsraster) geplant, wobei sich eine Trajektorie dann aus mehreren Segmenten zusammensetzt. Ein Trajektoriensegment beschreibt den Übergang vom Zustand $[s_k, v_k]^T$ im Zeitpunkt t_k in den Zustand $[s_{k+1}, v_{k+1}]^T$ zum Zeitpunkt t_{k+1}. Trajektoriensegmente sind jedoch nur zulässig, wenn ihre Projektion auf die Knotenmenge A_f (Kollisionsraster) nicht mit als *besetzt* markierten Knoten zusammenfällt (Abschnitt 3.3).

Die Wunschtrajektorie bzw. das Wunschfahrprofil, das gemäß der in Kapitel 4 beschriebenen Systematik gebildet wird, kann man sich prinzipiell als im Raum $B_{D,f}$ liegend vorstellen. Es spiegelt die Fahrerpräferenzen bezüglich Komfortempfinden, Kraftstoffverbrauch, Sicherheit und Zeitbedarf wider. Das Wunschfahrprofil kann nur realisiert werden, wenn keine Kollisionen mit anderen Verkehrsteilnehmern zu befürchten sind. Dies ist jedoch in der Regel nicht der Fall. Daher muss ein Fahrprofil gefunden werden, das dem Wunschfahrprofil möglichst nahe kommt, jedoch nicht mit Fahrprofilen anderer Verkehrsteilnehmer *kollidiert*. Dieses Fahrprofil ist dann das für das jeweilige Fahrzeug optimale zulässige Fahrprofil. In der Terminologie der Graphentheorie ausgedrückt, resultiert daraus die folgende Aufgabe: Für jedes Fahrzeug ist ein optimaler *Fluss* von der Einfahrt bis zur Ausfahrt des jeweiligen Verkehrsknotenpunktes (Kreuzung, Kreisverkehr etc.) zu bestimmen, der zu keinen Kollisionen mit anderen Fahrzeugen führt.

Die prinzipielle Vorgehensweise für die Bestimmung der optimalen zulässigen Trajektorie sieht so aus, dass vorwärtsschreitend, entlang der Wegstütz-

stellen von $B_{D,f}$, alternative Trajektoriensegmente entwickelt und anhand des zugehörigen Trajektoriensegments des Wunschfahrprofils bewertet werden. In die Bewertung eines Segments fließen die Bewertungen der diesem Segment unmittelbar vorausgehenden Segmente mit ein (Additivitätseigenschaft, Abschnitt 2.3.1). Durch dieses Vorgehen erhält man einen Graphen in der Form eines Wurzelbaumes (Abschnitt 2.2), dessen Wurzel der Knoten mit den Koordinaten $s_{k=0} = 0$ $v_{k=0} = v_e$ und $t_{k=0} = 0$ bildet, also der Knoten, der den Eintritt des Fahrzeugs in den Verkehrsknotenpunkt repräsentiert. Im Anschluss wird von den Knoten, deren Wegkoordinate $s_{max,f}$ entspricht, derjenige mit der besten Bewertung gewählt. Der Weg durch den Graphen, von seiner Wurzel bis zu diesem Knoten, repräsentiert die optimale Trajektorie.

5.1 Die Bewertung von Trajektoriensegmenten

Prinzipiell werden Trajektoriensegmente gemäß ihrer Abweichung vom Wunschfahrprofil bewertet. Die Trajektorie eines Fahrzeugs ist gegeben durch die Folge P_k diskreter Punkte in $B_{D,f}$ (vgl. (3.38) in Abschnitt 3.3), wobei die Wegkoordinaten s_k a priori festgelegt sind. Diese Punktfolge P_k unterteilt die Trajektorie in Segmente. Nach dem gleichen Prinzip wird nun aus dem Wunschfahrprofil die Punktfolge

$$\tilde{P}_{W,k} = \{\tilde{p}_{W,k} | \tilde{p}_{W,k} := (s_k, t_{W,k}, v_{W,k}),\ \tilde{p}_{W,k} \in \mathbb{R}^3_{\geq 0}\}, \quad k = 0, ..., n_t, \quad (5.1)$$

gebildet. Gemäß dem Wunschfahrprofil würde das Fahrzeug also die Wegstützstelle s_k zum Zeitpunkt $t_{W,k}$ erreichen. Seine Geschwindigkeit würde zu diesem Zeitpunkt $v_{W,k}$ betragen.

Weicht nun ein Trajektoriensegment der durch $\tilde{P}_k$ gegebenen Trajektorie von dem entsprechenden Segment der durch $\tilde{P}_{W,k}$ repräsentierten Wunschtrajektorie ab, so gibt es prinzipiell eine Vielzahl von Möglichkeiten, wie diese Abweichung bewertet werden kann. Eine Möglichkeit könnte beispielsweise in der folgenden Kosten- bzw. Gütefunktion bestehen:

$$c_k = (t_{W,k+1} - t_{k+1})^2 \quad (5.2)$$

Durch die Quadrierung werden einerseits Vorzeichen eliminiert, und andererseits werden größere Abweichungen progressiv steigend stärker gewichtet. Die Anwendung einer solchen Gütefunktion würde dazu führen, dass im Rahmen

der Optimierung versucht würde, möglichst geringe Abweichungen zu den jeweiligen Zeitpunkten $t_{W,k}$ des Wunschfahrprofils zu realisieren. Müsste also ein Fahrzeug wegen zuvor den Innenbereich des Verkehrsknotenpunktes durchfahrender Fahrzeuge abbremsen, so würde im Folgenden versucht, den aus dem Bremsen resultierenden Zeitverlust wieder zu kompensieren. Wenn möglich, würde das Fahrzeug also in folgenden Trajektoriensegmenten stärker beschleunigen, als im Wunschfahrprofil vorgesehen. Dies ist jedoch bei Fahrern mit komfortorientierten Wunschfahrprofilen unerwünscht. Bei dieser Art von Wunschfahrprofilen wäre es am besten, die Verzögerung in Kauf zu nehmen und dann zu versuchen, für den Rest des noch zurückzulegenden Weges möglichst die entsprechenden Segmente des Wunschfahrprofils zu realisieren. Eine Gütefunktion, die dieser Forderung schon sehr nahe kommt, wäre

$$c_k = \left|\bar{v}_{W,k} - \bar{v}_k\right|, \tag{5.3}$$

$$\text{mit} \quad \bar{v}_{W,k} = \frac{s_{k+1} - s_k}{t_{W,k+1} - t_{W,k}} \quad \text{und} \quad \bar{v}_k = \frac{s_{k+1} - s_k}{t_{k+1} - t_k}.$$

Mit dieser Gütefunktion wird zumindest sichergestellt, dass im Zuge des Optimierungsprozesses in den einzelnen Trajektoriensegmenten eine gleiche Durchschnittsgeschwindigkeit wie im zugehörigen Segment des Wunschfahrprofils angestrebt wird. Es ist jedoch auch der Fall denkbar, dass ein Segment des Wunschfahrprofils mit einer geringen Geschwindigkeit startet, dann ein Beschleunigungsvorgang erfolgt und das Segment mit einer hohen Geschwindigkeit endet, während es im Fall des entsprechenden zu bewertenden Trajektoriensegments genau umgekehrt ist. In diesem Fall würde das zu bewertende Segment signifikant vom Wunschfahrprofil abweichen, hätte aber dennoch eine gute Bewertung. Eine mögliche Lösung für dieses Problem besteht darin, die Geschwindigkeiten am Ende der Trajektoriensegmente ebenfalls in die Gütefunktion miteinzubeziehen. Dies führt dann auf die Gütefunktion

$$c_k = k_{\bar{v}} \cdot \left|\bar{v}_{W,k} - \bar{v}_k\right| + k_v \cdot \left|v_{W,k+1} - v_{k+1}\right|. \tag{5.4}$$

Bei $k_{\bar{v}}$ und k_v handelt es sich um Gewichtungsfaktoren für die beiden Abweichungstypen. Es hat sich jedoch in Simulationen gezeigt, dass die besten Ergebnisse mit $k_{\bar{v}} \approx k_v \approx 1$ erzielt werden, so dass man die Faktoren prinzipiell auch wegfallen lassen könnte.

Unter normalen Bedingungen, bei gleichartigem Verkehrsaufkommen für alle Routen, erzielt man mit der Gütefunktion gemäß (5.4) gute Ergebnisse. Dennoch kann auch noch eine erhebliche (situationsabhängige) *Feinabstimmung*

betrieben werden. So hat sich beispielsweise gezeigt, dass Staubildung im Einfahrtsbereich bei starkem einseitigen Verkehrsaufkommen reduziert bzw. aufgelöst werden kann, wenn man die Faktoren $k_{\bar{v}}$ und k_v an den Stützstellen s_k variiert bzw. bei den Stützstellen im Einfahrtsbereich größer wählt als bei denen im Ausfahrtsbereich. Prinzipiell sind hier jedoch noch weitere Untersuchungen erforderlich, um den Algorithmus für eine gegebene Verkehrssituation optimal konfigurieren zu können.

5.2 Formulierung eines heuristischen Ansatzes

Eine eingeplante Trajektorie auf der Fahrspur $a \in F$ führt zu besetzten Knoten innerhalb der Knotenanordnung A_a (vgl. Abbildung 3.6 und Abbildung 3.10). Durch Schnittbeziehungen von a (Abschnitt 3.1 und Abschnitt 3.4) werden dadurch automatisch auch auf anderen Fahrspuren $b \in F \wedge b \neq a$ Knoten gesperrt. Ist eine Schnittbeziehungen derart, dass der Einfahrts- oder Ausfahrtsbereich von a identisch ist mit dem von b, so sieht das Muster gesperrter Knoten in A_b für diesen Bereich genauso aus wie in A_a. Es besitzt die Form eines Schlauches (vgl. Abbildung 3.10). Anders sieht dies aus, wenn sich die Fahrspuren a und b im eigentlichen Sinne des Wortes schneiden (Abbildung 3.9). In diesem Fall wird von A_b nur ein Knoten-Cluster geringer Ausdehnung im Innenbereich der Kreuzung gesperrt.

Vor diesem Hintergrund ist nun für ein Fahrzeug auf Fahrspur $f \in F$ mit einem Wunschfahrprofil, gegeben durch die Punktfolge $\tilde{P}_{W,k}$, ein Fahrprofil zu bestimmen, das dem Wunschfahrprofil möglichst nahe kommt, gleichzeitig aber nur auf freie Knoten von A_f zugreift. Ein möglicher, heuristischer Ansatz für diese Problematik besteht darin, sich an einem Wasserfluss, wie er in Abbildung 5.1 dargestellt ist, zu orientieren. Das Wasser fließt durch die tiefsten Stellen des Flussbettes und wird erst auf ein höheres Niveau ansteigen, wenn die Tiefe abnimmt oder andere Hindernisse, zum Beispiel in Form von Steinen oder Felsen, den Fluss behindern. In diesem Fall wird das Wasser dann solange gestaut, bis ein ausreichend hohes Niveau erreicht ist, damit das jeweilige Hindernis umflossen werden kann.

Überträgt man dieses Beispiel nun auf die Problematik der Trajektorienplanung, so kann man die Wunschtrajektorie prinzipiell mit dem Verlauf der tiefsten Stellen des Flussbettes vergleichen. Die durch andere Fahrzeuge besetzten Knoten von A_f entsprechen den Hindernissen (Steine, Felsen etc.).

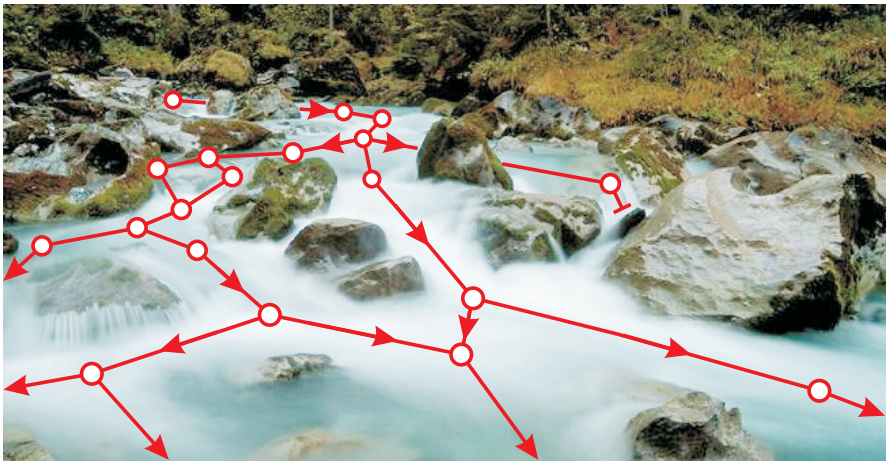

Abbildung 5.1: Vorbild Wasserfluss

Die Trajektorienplanung folgt der Wunschtrajektorie. Ist dies aufgrund von besetzten Knoten (Hindernisse) nicht mehr möglich, so müssen diese Knoten *umflossen* werden.

Anhand dieser Überlegungen kann ein erster Ansatz für die Trajektorienplanung nun folgendermaßen aussehen: Ausgehend von einem gegebenen Startknoten $\tilde{p}_k$, wird die Wunschtrajektorie bis zu der Stelle realisiert, an der auf einen besetzten Knoten $\alpha_{f,h,i}$ bzw. einen besetzten Einzugsbereich $\Theta_{f,h,i}$ des Kollisionsrasters gestoßen wird. In diesem Fall werden, wenn möglich, zwei alternative Trajektoriensegmente bestimmt, die in $\tilde{p}_k$ beginnen und in $\tilde{p}_{k+1,1}$ bzw. $\tilde{p}_{k+1,2}$ enden (① und ② in Abbildung 5.2). Diese beiden Segmente werden so gebildet, dass ihre Abweichung von dem zugehörigen Segment der Wunschtrajektorie minimal ist, jedoch der besetzte Bereich nicht mehr geschnitten wird. Das zum Knoten $\tilde{p}_{k+1,1}$ führende Segment weicht von der Wunschtrajektorie zu einem früheren Zeitpunkt ab, das zum Knoten $\tilde{p}_{k+1,2}$ führende Segment zu einem späteren Zeitpunkt. Von den Knoten $\tilde{p}_{k+1,1}$ und $\tilde{p}_{k+1,2}$ ausgehend, wird nun versucht, den Rest der Wunschtrajektorie zu realisieren. Ein alternatives Trajektoriensegment entspricht dem zugehörigen Segment der Wunschtrajektorie, wenn es in der Darstellung gemäß Abbildung 5.2 parallel zur Wunschtrajektorie verläuft, also lediglich entlang der Zeitachse verschoben ist. Ist es nicht möglich, den Rest des Wunschfahrprofils zu realisieren, so wird die oben beschriebene Prozedur wiederholt. In Abbildung 5.2 ist dies beispielhaft gezeigt. Ergebnis sind die beiden Trajektorien mit

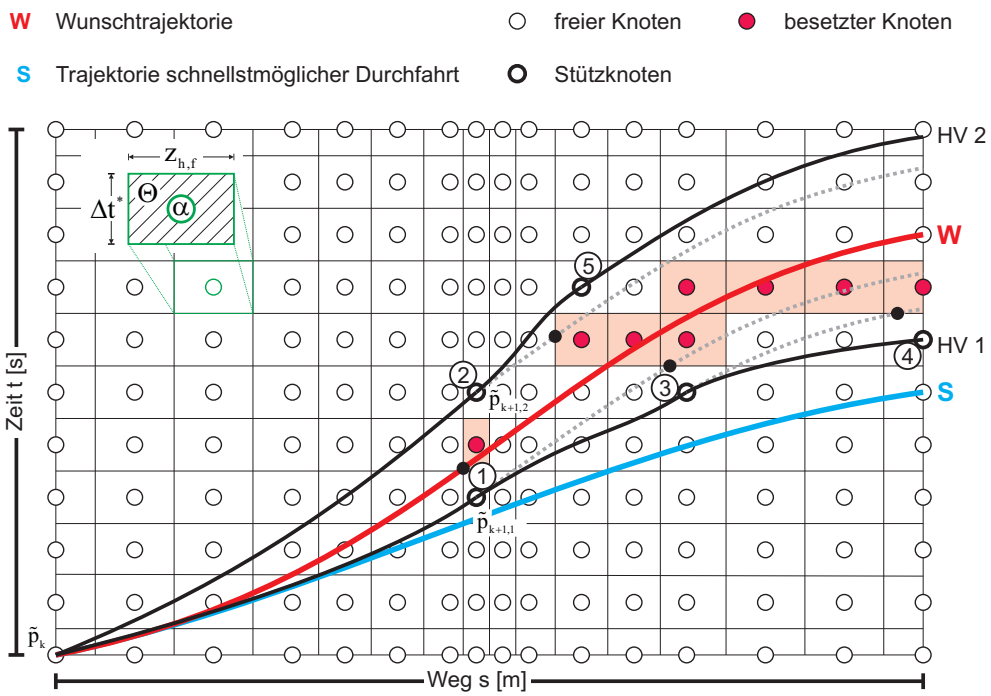

Abbildung 5.2: Beispiel für einen heuristischen Ansatz

den Bezeichnungen **HV 1** und **HV 2**. Die Bewertung kann analog zu den in Abschnitt 5.1 beschriebenen Berechnungen erfolgen. Letztlich ist aus sämtlichen zulässigen Trajektorien **HV 1**, **HV 2**, ... die Trajektorie mit dem besten Gütewert zu wählen.

Der skizzierte heuristische Ansatz ist sehr einfach und bietet noch viel Optimierungspotenzial. Beispielsweise ist es sicherlich sinnvoll, mehr als 2 alternative Trajektoriensegmente zu bilden, falls an einer Stelle der Wunschtrajektorie nicht mehr gefolgt werden kann. Weiterhin ist zu erwarten, dass dieser Ansatz im Hinblick auf die Berechnungsdauer deutlich schneller ist als die Dynamische Programmierung, die im nächsten Abschnitt vorgestellt wird. Allerdings hat diese Schnelligkeit auch einen Preis, der darin besteht, dass das Auffinden des Optimums bzw. der optimalen Trajektorie nicht garantiert werden kann.

Prinzipiell sind neben dem zuvor skizzierten heuristischen Ansatz noch eine Vielzahl anderer Ansätze denkbar. Dazu zählt zum Beispiel auch die Anwen-

dung genetischer Algorithmen. Diese Ansätze zeichnen sich aber dadurch aus, dass sich Aussagen bezüglich der Laufzeit und der Optimalität der gefundenen Lösung in der Regel nicht oder nur sehr schwer treffen lassen.

Im Rahmen des entstandenen Simulators wurde bisher der Ansatz auf Basis der Dynamischen Programmierung implementiert, da zuerst analysiert werden sollte, inwieweit ein mathematisch exaktes Verfahren unter Echtzeitanforderungen verwendet werden kann. Es wird an dieser Stelle ausdrücklich darauf hingewiesen, dass die Dynamische Programmierung hier so verstanden wird, dass sämtliche vorhandenen Entscheidungsmöglichkeiten[1] des mehrstufigen Entscheidungsprozesses systematisch ausgewertet werden. Dadurch wird der gesamte erreichbare Zustandsraum analysiert und das Auffinden des Optimums für die jeweilige Konstellation des Optimierungsproblems garantiert. Erst wenn absehbar ist, dass der mit der Dynamischen Programmierung verbundene Rechenaufwand nicht mehr in Echtzeit abgearbeitet werden kann, ist ein Übergang auf einen rein heuristischen Ansatz sinnvoll. Alternativ dazu kann die Dynamische Programmierung auch mit dem Ziel der Laufzeitverbesserung um Heuristiken erweitert werden. Diese Möglichkeit wird am Ende dieses Kapitels nochmals aufgegriffen.

5.3 Anwendung der Dynamischen Programmierung

Ein heuristischer Ansatz, wie er in Abschnitt 5.2 beschrieben wurde, kann das Auffinden des Optimums nicht garantieren. Dies liegt daran, dass Entscheidungen im Hinblick auf die Entwicklung von alternativen Trajektoriensegmenten getroffen werden, bevor sämtliche Informationen über den Lösungsraum bekannt sind. So können beispielsweise zu Beginn Trajektoriensegmente entwickelt werden, die später in einer *Sackgasse* enden. Bei der Dynamischen Programmierung sieht dies anders aus, denn im Rahmen dieses Verfahrens wird der gesamte Raum $B_{D,f}$ (in Verbindung mit A_f) systematisch analysiert. Erst dann wird von der letzten Wegstützstelle aus die optimale Trajektorie entwickelt. Die Ergebnisse dieses Vorgehens sind schematisch in Abbildung 5.3 dargestellt. Der dunkel hinterlegte Bereich repräsentiert analy-

[1] Das bedeutet, sämtliche Kanten aller erreichbaren Knoten gemäß der Definitionen in Abschnitt 3.3.

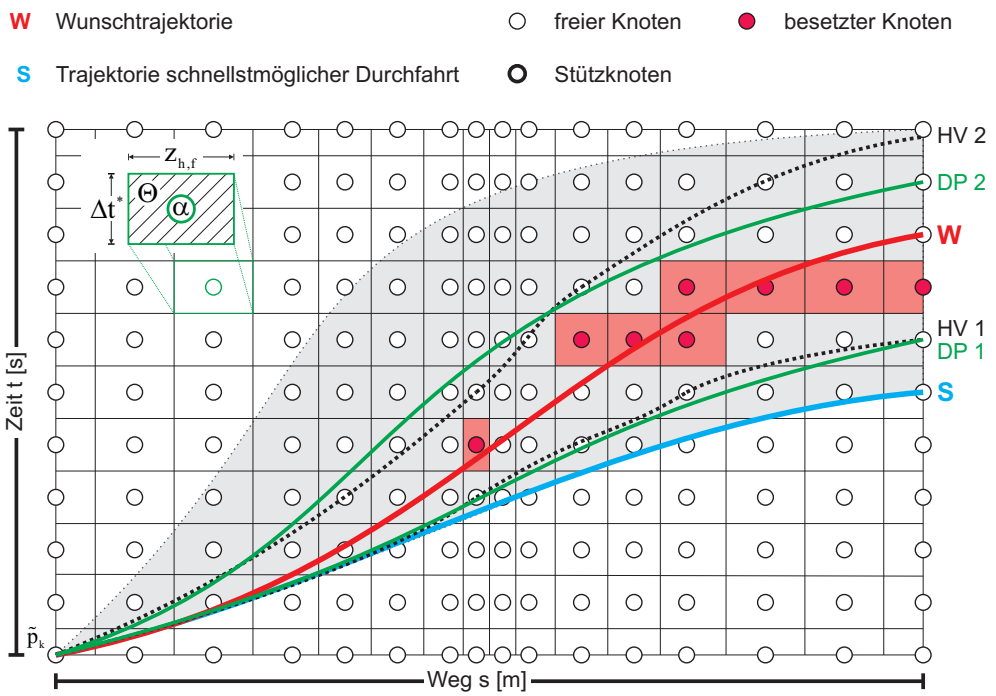

Abbildung 5.3: Beispiel für die Dynamische Programmierung

sierte Knoten- bzw. Zustandsübergänge, die einem Trajektoriensegment entsprechen, mit dem das Fahrzeug innerhalb des zulässigen maximalen Zeitfensters den jeweiligen Verkehrsknotenpunkt überqueren kann. Jeder Knoten $\tilde{p}_{k+1} \in B_{D,f}$ eines potenziellen Trajektoriensegments innerhalb dieses Bereichs verfügt über die Bewertung

$$C(\tilde{p}_{k+1}) = \sum_{\kappa^*=0}^{k} c_{\kappa^*} \, (e_{\kappa^*} \in E_{\kappa^*}), \quad \text{mit} \quad k = 0, ..., n_t - 1, \tag{5.5}$$

sowie über die Information des optimalen Vorgänger-Knotens (Abschnitt 3.3.2). Am Ende der Durchfahrt – an der Stützstelle $s_{k=n_t}$ – wird der Knoten gewählt, der die Bedingung

$$min \, \{C(\tilde{p}_{n_t}) | t_{n_t} \in T_D \wedge v_{n_t} \in V_D)\} \tag{5.6}$$

erfüllt. Von diesem Knoten aus wird über die Vorgänger-Beziehung die optimale Trajektorie rückwärts schreitend entwickelt. Dieses Vorgehen führt dann

zu Trajektorien, die ähnlich aussehen wie die mit **DP 1** und **DP 2** bezeichneten Trajektorien in Abbildung 5.3.

5.3.1 Adaption und Umsetzung im Simulator

Die Dynamische Programmierung entspricht einem mehrstufigen Entscheidungsprozess. Aus diesem Grund sind im Vorfeld Entscheidungsstufen festzulegen. Im Hinblick auf das Kreuzungsmanagement entspricht eine Entscheidungsstufe prinzipiell einer Wegstützstelle $s_k \in S_{D,B_f}$ des Optimierungsrasters, an der für das Fahrzeug eine Antriebs- bzw. Bremskraft festgelegt wird, die bis zum Erreichen der Wegstützstelle $s_{k+1} \in S_{D,B_f}$ konstant gehalten wird.

Das im Simulator implementierte Optimierungsraster besteht aus fixen ($s_{k,fix}$) und variablen ($s_{k,var}$) Wegstützstellen, wobei auch bei den variablen Stützstellen die Restriktion zu beachten ist, dass nur Stützstellen zulässig sind, die auch zum Kollisionsraster gehören ($s_{k,var} \in S_{D,B_f}, S_{D,B_f} \subseteq S_{D,f}$): Gemäß Abschnitt 4.1 wird die Durchfahrt eines Verkehrsknotenpunktes in 3 Phasen unterteilt, die von den Grenzen zwischen Innen- und Außenbereich des jeweiligen Verkehrsknotenpunktes abhängen. An diesen Grenzen sowie am Anfang und am Ende des gesamten Durchfahrtsweges sind fixe Wegstützstellen definiert. Dazwischen können, jeweils an der Position einer Stützstelle des Kollisionsrasters, beliebig viele variable Stützstellen realisiert werden. Experimente haben jedoch gezeigt, dass das Einfügen von 1-2 variablen zwischen den fixen Stützstellen bereits zu sehr guten Ergebnissen führen. Mehr variable Stützstellen erhöhen im Wesentlichen lediglich den Berechnungsaufwand. Sämtliche fixe und variable Stützstellen, mit Ausnahme der letzten Wegstützstelle, bilden damit die Entscheidungsstufen der Dynamischen Programmierung. Auf dieser Basis wird dann der Raum $B_{D,f}$ gebildet, indem für jede Stützstelle s_k des Optimierungsrasters ein Unterraum $B_{VT,k}$ definiert wird (Abschnitt 3.3.1).

Der Eintritt eines Fahrzeugs in einen Verkehrsknotenpunkt ist durch den Knoten $p_{k=0} \in B_{D,f} := (s_{k=0} = 0, t_{k=0} = 0, v_{k=0} = v_e)$ definiert. In diesem Knoten startet der Algorithmus der Dynamischen Programmierung, indem sämtliche Kanten bzw. Trajektoriensegmente bestimmt werden, mit denen das Fahrzeug zu der Wegstützstelle s_{k+1} gelangen kann (Abschnitt 3.3.2), ohne dass dabei auf besetzte Knoten von A_f zugegriffen wird. Für sämtliche Zielknoten auf der Stufe s_{k+1} wird nun gemäß der Gütefunktion (5.4) der

zugehörige Gütewert berechnet und gespeichert, wobei für sämtliche Knoten von $B_{D,f}$ der Wert ∞ vorinitialisiert ist. Dieser vorinitialisierte Wert wird jeweils durch den berechneten Gütewert überschrieben. Weiterhin wird der optimale Vorgänger gespeichert, der in diesem Fall stets in dem Knoten $p_{k=0}$ besteht.

Beim Übergang zur Stufe $s_{k=1}$ muss das im vorherigen Abschnitt beschriebene Verfahren für sämtliche Knoten durchgeführt werden, die von der vorherigen Stufe $s_{k=0}$ aus erreicht wurden. Diese Knoten sind daran zu erkennen, dass ihr Gütewert nicht mehr dem Wert ∞ entspricht. Das Resultat dieses Vorgehens besteht darin, dass auf der Stufe $s_{k=2}$ erstmalig Knoten auftauchen, die mehrere Vorgänger haben können bzw. durch mehrere Knoten auf der Stufe $s_{k=1}$ erreicht werden können. Hier wird nun die zentrale Idee des Bellman-Ford-Algorithmus (Abschnitt 2.2) aufgegriffen und angewendet: Wenn von einem Knoten der Stufe s_k aus ein Knoten auf der Stufe s_{k+1} erreicht werden kann, so wird zuerst geprüft, ob der Gütewert des jeweiligen Knotens auf der Stufe s_{k+1} verbessert werden kann. Wird dieser Knoten erstmalig erreicht, so wird dies immer der Fall sein, denn der initiale Wert ist ∞. Im anderen Fall werden Gütewert und Vorgängerbeziehung nur dann überschrieben, wenn eine Verbesserung des Gütewertes möglich ist (Prinzip der LC-Verfahren). Bevor beide Werte jedoch überschrieben werden, sind die entsprechenden Knoten von A_f (Kollisionsraster) zu prüfen. Ein Überschreiben darf nur stattfinden, wenn das jeweilige Trajektoriensegment nur auf solche Knoten von A_f zugreift, die nicht bereits durch andere Fahrzeuge besetzt sind. Nach dieser Systematik wird nun der gesamte Raum $B_{D,f}$ sukzessive entlang der Wegachse, im Unterraum $B_{VT,0}$ beginnend und im Unterraum B_{VT,n_t} endend, exploriert. Der diesem Vorgehen entsprechende Pseudo-Code hat prinzipiell die Form des Listings 5.1.

Die äußere Schleife (**while** ...) läuft über die Entscheidungsstufen der Dynamischen Programmierung. Auf der letzten Entscheidungsstufe (Stützstelle $n_t - 1$) werden sämtliche Möglichkeiten bzw. Trajektoriensegmente ermittelt, mit denen das jeweilige Fahrzeug den Verkehrsknotenpunkt verlassen kann. Diese Möglichkeiten führen in der Regel zu Bewertungen von Knoten an der Stützstelle $s_{k=n_t}$ bzw. von Knoten im Unterraum B_{VT,n_t}. Sollte an dieser Stelle kein Trajektoriensegment realisiert werden können, so ist die Trajektorienplanung fehlgeschlagen. Es konnte in diesem Fall für das Fahrzeug keine Trajektorie berechnet werden, mit der es den Verkehrsknotenpunkt in dem gegebenen Zeitfenster durchfahren kann. Die Konsequenz daraus ist, dass das Fahrzeug warten und der Algorithmus erneut ausgeführt werden muss. Die

```
100   Startknoten a = p_{k=0};
101   Bewertung C[i] von Knoten i := ∞ für alle i∈B_{D,f};
102   Bewertung C[a] := 0;
103   Routenvorgänger R[i] von Knoten i := ∅ für alle i∈B_{D,f};
104   k := 0;
105
106   while (k < n_t)
107   {
108      for   (alle Knoten i∈B_{VT,k}, mit C[i] < ∞ )
109         for (alle fahrdyn. zulässigen Kanten [i,j], mit j∈B_{VT,k+1})
110         {
111            if (C[i] + c(i,j) < C[j]))
112            {
113               if (keine Kollisionen auf A_f durch Kante [i,j])
114               {
115                  C[j] := C[i] + c(i,j);
116                  R[j] := i;
117               }
118            }
119         }
120      }
121      k := k+1;
122   }
```

Listing 5.1: Pseudo-Code der Dynamischen Programmierung im Kreuzungsmanagement

Trajektorienplanung schlägt immer dann fehl, wenn die maximale Kapazität des jeweiligen Verkehrsknotenpunktes erreicht wird.

Konnte die Trajektorienplanung erfolgreich ausgeführt werden, so befinden sich im Unterraum B_{VT,n_t} Knoten mit einer Bewertung ungleich ∞. Von diesen Knoten wird nun derjenige ausgewählt, der über die beste Bewertung verfügt. Dann wird von diesem Knoten aus über die Vorgängerbeziehung die optimale Trajektorie, vom Ende ausgehend, rückwärts schreitend bis zu ihrem Anfang entwickelt. Letztlich ergeben sich daraus die optimalen Entscheidungen (bzw. die optimale Strategie) für die Entscheidungsstufen s_k der Dynamischen Programmierung mit $k = 0, ..., n_t - 1$. Ein repräsentatives Ergebnis der vorgestellten Systematik wird durch Abbildung 5.4 geliefert. Die Abbildung zeigt die Ergebnisse der Trajektorienplanung mittels der Dynamischen Programmierung für ein Fahrzeug, das eine Kreuzung durchfährt. Das

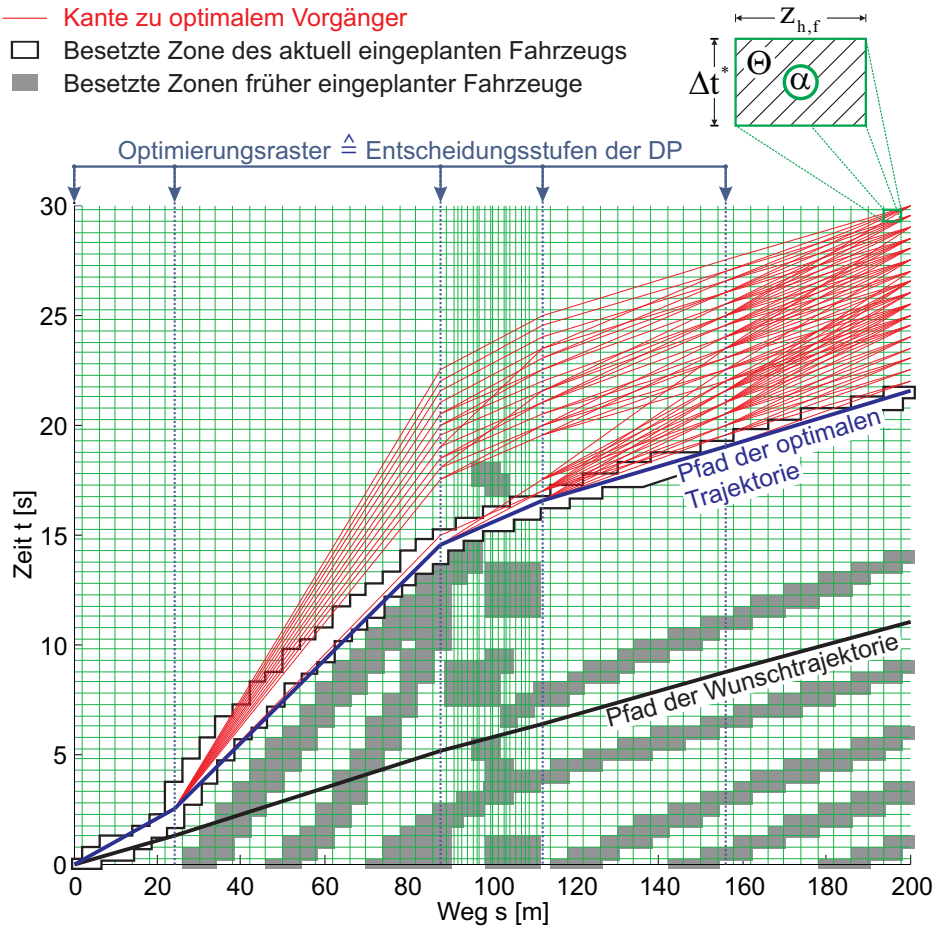

Abbildung 5.4: Beispiel für die Ergebnisse einer Trajektorienberechnung

maximale Zeitfenster für die Durchfahrt entspricht 30 Sekunden, die Diskretisierungsschrittweite der Zeit Δt^* entspricht 0,5 Sekunden. Die Diskretisierung des Weges variiert, wie bereits in Kapitel 3 angedeutet. Der Innenbereich der Kreuzung ist aufgrund der vielen Kollisionsbeziehungen mit anderen Routen deutlich feiner und in nicht äquidistanten Abständen diskretisiert. Im Außenbereich, also auf der Einfahrts- und der Ausfahrtsgerade, wurde eine äquidistante Diskretisierung mit einem Abstand von 4 Metern gewählt[2]. Die

[2]Die Diskretisierung für die Kreuzung ist *hardcodiert* (vgl. Abschnitt 3.4). Der Stützstellenabstand im Ausfahrtsbereich kann mit geringem Aufwand modifiziert werden, im Innenbereich ist dies jedoch nicht

dunklen Bereiche entsprechen Einzugsbereichen $\Theta_{f,h,i}$, die bereits durch andere Fahrzeuge gesperrt wurden. Die dargestellten Kanten entsprechen Entscheidungsmöglichkeiten an den Entscheidungsstufen der Dynamischen Programmierung. Ferner repräsentieren sie in ihrer Gesamtheit sämtliche Trajektorien bzw. Möglichkeiten, mit denen das Fahrzeug den Verkehrsknotenpunkt innerhalb des eingeräumten Zeitfensters durchfahren kann. Der die optimale Trajektorie repräsentierende Pfad[3] ist dicker hervorgehoben und führt zu dem Knoten an der Stützstelle s_{n_t}, der die günstigste Bewertung hat. Man kann gut erkennen, wie durch die gewählte Gütefunktion versucht wird, der Wunschtrajektorie möglichst zu folgen und wie *Unstetigkeiten* im Innenbereich, in Form von besetzten Knoten-Clustern, einfach *umflossen* werden.

Weiterhin sollte klar sein, dass mit einer feineren Diskretisierung bessere Trajektorien und ein höherer Durchsatz bzw. eine Erhöhung der Kapazität des jeweiligen Verkehrsknotenpunktes zu erzielen sind. Eine Verfeinerung der Diskretisierung ist jedoch mit einer Erhöhung des Rechenaufwands verbunden. Die Diskretisierung muss also je nach (Rechner-)Zielsystem so gewählt werden, dass die Berechnungen noch in Echtzeit ausgeführt werden können. Bezüglich der Echtzeitfähigkeit wurde die folgende Anforderung festgelegt: Die Trajektorienplanung mittels der Dynamischen Programmierung für ein Fahrzeug soll in einer Zeit unterhalb von einer Sekunde abgeschlossen sein ($T_{DP} < 1.0$ [s]), und zwar weil zu erwarten ist, dass die Bewegung eines Fahrzeugs für den Zeitraum von einer Sekunde mit hoher Genauigkeit extrapoliert bzw. geschätzt werden kann. Im Zeitpunkt $t_{-1} = x$ [s] wird dann mit der Planung einer Trajektorie begonnen, die nahtlos an den Zustand anknüpft, den das Fahrzeug mit höchster Wahrscheinlichkeit im Zeitpunkt $t_0 = x + 1.0$ [s] erreicht haben wird. Ab dem Zeitpunkt t_0 realisieren die Fahrdynamikregler des Fahrzeugs die berechnete Trajektorie. Erst wenn diese Trajektorie vollständig abgefahren wurde, wird am Ende des Verkehrsknotenpunktes wieder die Kontrolle an den Fahrer übergeben.

5.3.2 Betrachtungen zum Rechenaufwand

Der große Vorteil der Dynamischen Programmierung besteht darin, dass für die gegebene Konstellation das Optimum in jedem Fall gefunden wird, und dies auch innerhalb einer deterministischen Zeitspanne, die weiterhin durch

ohne Weiteres möglich.

[3] Ein *Pfad* ist als Weg durch ein Netzwerk im Sinne einer Folge von Kanten zu verstehen.

die gewählte Diskretisierung stark beeinflusst werden kann. Für Systeme mit geringer Ordnung ist die Methode somit gut geeignet, wenn Echtzeitanforderungen zu erfüllen sind, denn alternative Optimierungsverfahren aus dem Bereich der nichtlinearen Optimierung, wie beispielsweise das Gradientenverfahren, können weder Konvergenz in endlicher Zeit noch (globale) Optimalität der gefundenen Lösung garantieren.

Der Nachteil der Dynamischen Programmierung offenbart sich bekanntermaßen bei höheren Dimensionen, denn diese gehen exponentiell in den Rechenaufwand ein (Abschnitt 2.3.3).

Zur Abschätzung des Rechenaufwands beim Kreuzungsmanagement wird das folgende Beispiel betrachtet:

Eine verwendete Konfiguration berücksichtigt die Geschwindigkeit im Intervall von 0 bis 20 m/s, wobei in Schritten von $\Delta v^* = 0,5$ m/s diskretisiert wird (41 diskrete Werte). Den Fahrzeugen wird für das Durchfahren der gegebenen Kreuzung (Ein- und Ausfahrt jeweils ca. 100 m) ein maximales Zeitfenster von 40 Sekunden eingeräumt. Es wird dabei in Schritten von $\Delta t^* = 0,5$ s diskretisiert (81 diskrete Werte). Die Anzahl der Entscheidungsstufen bzw. Wegstützstellen $s_k \in S_{D,f}$ liegt bei dieser Konfiguration bei 8, es sind jedoch sowohl mehr als auch weniger denkbar. Daraus ergibt sich ein Umfang von $8 \cdot 41 \cdot 81 = 26.568$ diskreten Knoten für den Raum $B_{D,f}$, wobei für jeden dieser Knoten, der durch ein Fahrzeug erreicht werden kann, sämtliche ausgehende Kanten (Abschnitt 3.3.2) analysiert werden müssen. Da jedoch, beginnend mit einem konkreten Anfangsknoten ($p_{k=0}$), der Raum $B_{D,f}$ sukzessive exploriert wird – bestimmte Knoten des Raumes also gar nicht erst erreicht werden können (vgl. Abbildung 5.4) – und weiterhin sinnvolle Begrenzungen dieses Raumes im Vorfeld möglich sind, lässt sich diese Anzahl signifikant reduzieren, teilweise auf ein Drittel oder weniger. Geht man von etwa 10.000 Knoten aus, so müssen für 10.000 Knoten T_{min} und T_{max} bestimmt sowie die zwischen diesen Grenzen liegenden Übergänge analysiert werden. Auf einem 3-GHz-Rechner (Single-Core) werden dafür je Optimierung bzw. Fahrzeug ca. 50 Millisekunden benötigt. Durch Anpassung der Diskretisierung und Variation der Entscheidungsstufen lässt sich dieser Wert stark beeinflussen und prinzipiell auf die Kapazität beliebiger Zielsysteme adaptieren. Eine gröbere Diskretisierung geht zu Lasten der Genauigkeit und reduziert auch den Durchsatz, da sich Fahrzeugabstände vergrößern, ermöglicht andererseits aber, dass Rechnersysteme mit geringer Kapazität die Optimierungsaufgabe noch in Echtzeit lösen können.

Neben der bereits in Abschnitt 3.3 eingeführten Unterscheidung zwischen Kollisionsraster und Optimierungsraster gibt es noch weitere Möglichkeiten, die Laufzeit der Dynamischen Programmierung zu verbessern. Diese werden in den folgenden Abschnitten kurz vorgestellt.

Ausschluss von Netzwerkknoten

Eine Möglichkeit zur Reduktion des Rechenaufwands besteht in der bereits im obigen Beispiel erwähnten Begrenzung des für die Dynamische Programmierung erreichbaren Raumes, also einer Begrenzung von $B_{D,f}$. Aufgrund diverser Rahmenparameter (maximales Zeitfenster für Kreuzungsdurchfahrt, Geschwindigkeitslimits in bestimmten Bereichen etc.) kann im Vorfeld mit geringem Aufwand ein Bereich zulässiger Trajektorien bestimmt werden, mit denen Knoten ausgeschlossen werden, die zu keiner zulässigen Trajektorie führen würden. Abbildung 5.5 zeigt ein Beispiel für den Bereich zulässiger Knoten. Die Optimierung berücksichtigt lediglich Knoten bzw. Knotenübergänge, die in diesem zulässigen Bereich liegen. An dieser Stelle werden letztlich auch die querdynamischen Restriktionen berücksichtigt, die durch bereichsabhängige Geschwindigkeitslimits in die Begrenzung einfließen. So kann eine querdynamische Restriktion zum Beispiel darin bestehen, dass ein abbiegendes Fahrzeug während des eigentlichen Abbiegevorgangs eine bestimmte Geschwindigkeit nicht überschreiten darf, da sonst die Gefahr des Ausbrechens besteht. Daraus folgt, dass an Wegstützstellen s_k vor Beginn des Abbiegevorgangs nicht jedes Geschwindigkeitsniveau zulässig ist. Dazu gehören sämtliche Geschwindigkeiten, von denen aus das Fahrzeug nicht mehr rechtzeitig auf das maximal zulässige Niveau für den Abbiegevorgang abbremsen kann. Man erkennt dies in Abbildung 5.5 an der Wölbung des zulässigen Bereichs.

Wie hoch das Potenzial für die Reduktion der Berechnungen ist bzw. wie viele Knoten von der Berechnung ausgeschlossen werden können, hängt letztlich von mehreren Faktoren ab, wie zum Beispiel von der Eintrittsgeschwindigkeit des Fahrzeugs (v_E niedrig oder hoch?) oder von den Fahrzeugeigenschaften (agiler PKW oder träger LKW?). Die Erfahrungen zeigen aber, dass bei einer leeren Kreuzung bei geradeaus fahrenden Fahrzeugen ca. 50 % der Knoten von $B_{D,f}$ nicht analysiert werden müssen, sei es durch die sukzessive Exploration oder durch direkten Ausschluss. Bei abbiegenden Fahrzeugen kann sogar von ca. 75 % ausgegangen werden. Mit steigendem Verkehrsaufkommen

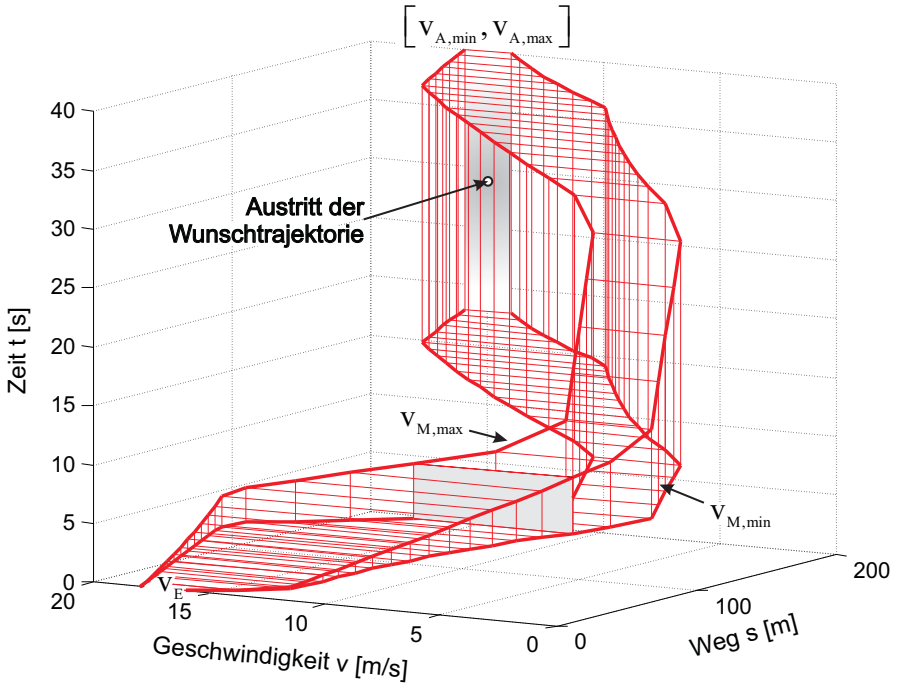

Abbildung 5.5: Begrenzung von $B_{D,f}$ am Beispiel eines abbiegenden Fahrzeugs

findet eine weitere Reduktion der zu analysierenden Knoten statt, da parallel zum Anstieg des Verkehrsaufkommens auch immer mehr Knoten durch andere Fahrzeuge gesperrt werden (vgl. Abbildung 5.4). Die durchschnittliche Laufzeit der Dynamischen Programmierung sinkt daher mit steigendem Verkehrsaufkommen (vgl. auch Abbildung 6.5).

Parallelisierung

In Abschnitt 2.3 wurde gezeigt, dass die Auswertungen von Knoten derselben Entscheidungsstufe prinzipiell unabhängig voneinander sind. Bei dem hier vorgestellten Verfahren ist diese Unabhängigkeit leider nicht mehr vollständig gegeben, da die Kanten nicht a priori feststehen, sondern zur Laufzeit der Dynamischen Programmierung aus den Fahrzeugmodellen gewonnen werden. Dies führt dazu, dass mehrere unterschiedliche Knoten der Entscheidungsstu-

fe s_k auf einen Knoten der Entscheidungsstufe s_{k+1} parallel zugreifen können. Dennoch besteht auch in diesem Fall die Möglichkeit einer Parallelverarbeitung. Ein möglicher Ansatz hierfür wurde in [36] vorgestellt.

In diesem Ansatz werden die zweidimensionalen Unterräume $B_{VT,h}$ (Abschnitt 3.3.1), die zu den jeweiligen Entscheidungstufen s_k gehören, in n Partitionen zerlegt. Jede dieser Partitionen wird von einer Recheneinheit bearbeitet, so dass die Dynamische Programmierung prinzipiell auf n Rechnern parallel durchgeführt wird. Jede dieser n Recheneinheiten berechnet für sämtliche Knoten ihrer zugeordneten Partition die optimalen Trajektoriensegmente zu den Knoten der nächsten Entscheidungsstufe s_{k+1}.

Das Problem besteht nun darin, dass auf einen Knoten der Entscheidungsstufe s_{k+1} von mehreren Knoten der Entscheidungsstufe s_k zugegriffen werden kann, die aber wiederum zu unterschiedlichen Partitionen bzw. Recheneinheiten gehören können. Diesem Problem wurde begegnet, indem zunächst jede Recheneinheit mit einer Kopie des zugehörigen Unterraumes $B_{VT,h}$ der Entscheidungsstufe s_{k+1} arbeitet. Nachdem sämtliche Recheneinheiten ihre Partitionen vollständig bearbeitet haben, müssen die Kopien der Unterräume synchronisiert werden. Dies nimmt Rechenzeit in Anspruch. Weiterhin führt die Verwendung von Kopien der Unterräume zu zusätzlichen Berechnungen gegenüber der nichtparallelisierten Variante: Wurde bei der nichtparallelisierten Variante für einen Knoten der Stufe s_{k+1} bereits frühzeitig der optimale Vorgänger gefunden bzw. wurde dieser Knoten mit einer günstigen Bewertung versehen, so werden Überprüfungen bzw. Zugriffe durch andere Knoten frühzeitig abgebrochen.

Es hat sich jedoch gezeigt, dass sich diese nachteiligen Eigenschaften durch geschickte Partitionierung und sequenzielle Synchronisation während der parallelen Verarbeitung der Partitionen deutlich reduzieren lassen. Es konnte gezeigt werden, dass durch eine Parallelisierung eine signifikante Laufzeitverbesserung erreicht werden kann. Als weiteres Hemmnis wurden Kommunikationszeiten identifiziert, sofern die parallel arbeitenden Recheneinheiten über ein Netzwerk (TCP/IP) kommunizieren. Prinzipiell bietet die Möglichkeit der Parallelverabreitung noch hohes Potenzial für eine Laufzeitverbesserung. Vor diesem Hintergrung erscheint es vielversprechend, das Potenzial von Multi-Prozessor-Graphikkarten auszuloten. Die Prozessoren von Graphikkarten können sehr gut für parallelisierbare Berechnungen genutzt werden. Unter der Bezeichnung *General Purpose Computation on Graphics-Processing Units (GPGPU)* entstehen derzeit immer komfortablere Software-

Bibliotheken für die parallele Nutzung von Recheneinheiten innerhalb eigener Anwendungen.

Einsatz von Heuristiken

In Abschnitt 5.2 wurde ein heuristischer Ansatz für die Planung alternativer Trajektoriensegmente vorgestellt. Dabei sollte klar geworden sein, dass es sich bei heuristischen Ansätzen um keine mathematisch exakten Verfahren handelt, sondern vielmehr um *sinnvolle* Annahmen, die für eine schnelle Berechnung einer zulässigen, aber in der Regel nicht optimalen Lösung herangezogen werden können. Heuristiken können zu einem ausgewogenen Kompromiss der beiden in den meisten Fällen gegensätzlichen Zielgrößen *Güte* einer Lösung und *Berechnungsaufwand* dieser Lösung führen.

Anders als in Abschnitt 5.2 werden Heuristiken im Folgenden nicht als eigenständige Ansätze aufgefasst, sondern als sinnvolle Modifikationen für mathematisch exakte Verfahren, wie zum Beispiel die Dynamische Programmierung.

Führt man sich die Systematik der Trajektorienplanung gemäß Kapitel 3 vor Augen, so stellt man fest, dass sich der Raum alternativer Trajektoriensegmente mit zunehmender Wegstützstelle bzw. Entscheidungsstufe s_k immer weiter aufweitet. Folglich finden die meisten Berechnungen im Ausfahrtsbereich des jeweiligen Verkehrsknotenpunktes statt (vgl. Abbildung 5.4). Im Einfahrtsbereich müssen nur sehr wenige Berechnungen durchgeführt werden, während im Durchfahrtsbereich bereits ein starker Anstieg zu beobachten ist. Der Durchfahrtsbereich muss als kritisch angesehen werden, da hier das größte Kollisionspotenzial besteht. Im Ausfahrtsbereich hingegen können nur noch Kollisionen mit vorausfahrenden Fahrzeugen auftreten. Um diesem Kollisionspotenzial zu begegnen, sind in der Regel beim Übergang vom Durchfahrts- in den Ausfahrtsbereich keine extremen Fahrmanöver erforderlich. Diese Beobachtung führt auf folgende beiden Heuristiken, die unabhängig voneinander ab der letzten Entscheidungsstufe s_k des Durchfahrtsbereichs für alle weiteren Entscheidungsstufen anzuwenden sind:

1. Es werden von allen erreichten Knoten $\tilde{p}_k \in B_{D,f}$ nur noch die n Knoten mit der besten Bewertung weiterentwickelt.

2. Für sämtliche Knoten $\tilde{p}_k$ werden nicht sämtliche Kanten $e_k \in E_k$ weiterentwickelt, sondern nur eine bestimmte Anzahl. Die Auswahl der weiterzuentwickelnden Kanten sollte sich dabei an der Kante des Wunschgeschwindigkeitsprofils für die jeweilige Entscheidungsstufe orientieren.

In [36] wurde die letzte dieser beiden Möglichkeiten genauer untersucht. Als Ergebnis konnte gezeigt werden, dass sich dadurch der Berechnungsaufwand um mehr als 50 % reduzieren lässt, während im Mittel nur eine minimale Abweichung von der optimalen Trajektorie hingenommen werden musste.

6 Ergebnisse und Ausblick

Mit dem vorgestellten Verfahren wurden bisher Kreuzungen (Abbildung 6.1) und unterschiedlich große Kreisverkehre (Abbildung 6.2) mit jeweils einer

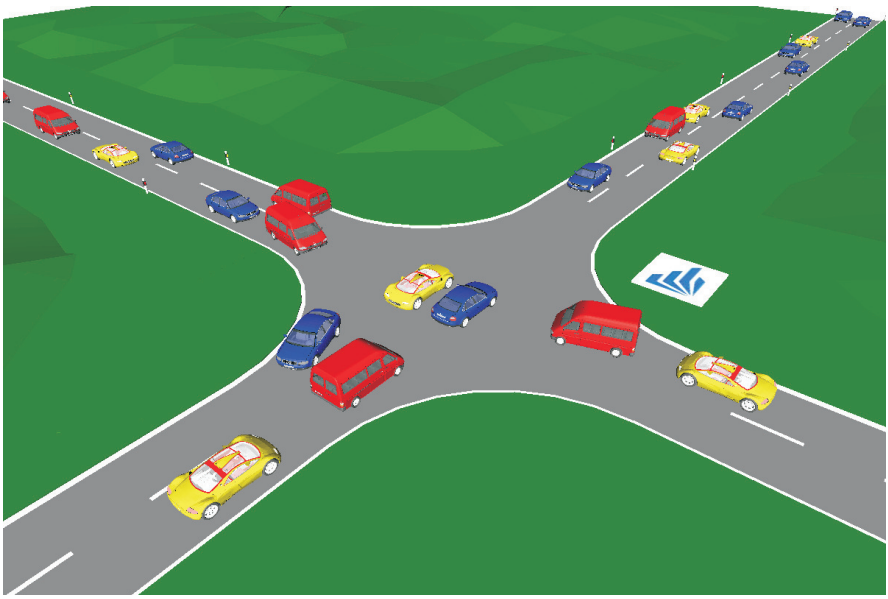

Abbildung 6.1: Simulation einer ampelfreien Kreuzung

Ein- bzw. Ausfahrtsspur je Richtung untersucht. Für die maximale Kapazität einer solchen Kreuzung bei Nutzung herkömmlicher Lichtsignalanlagen kann ein Richtwert von maximal ca. 1.800 Fahrzeuge/Std. angenommen werden; den Durchsatz reduzierende Faktoren wie ein LKW-Anteil oder bevorrechtigte Fußgänger sind bei diesem Wert nicht berücksichtigt [13].

In Simulationen mit dem in dieser Arbeit vorgestellten Verfahren wurden mit einem Zufallsgenerator (Gleichverteilung) Fahrzeuge (PKW) für alle 12

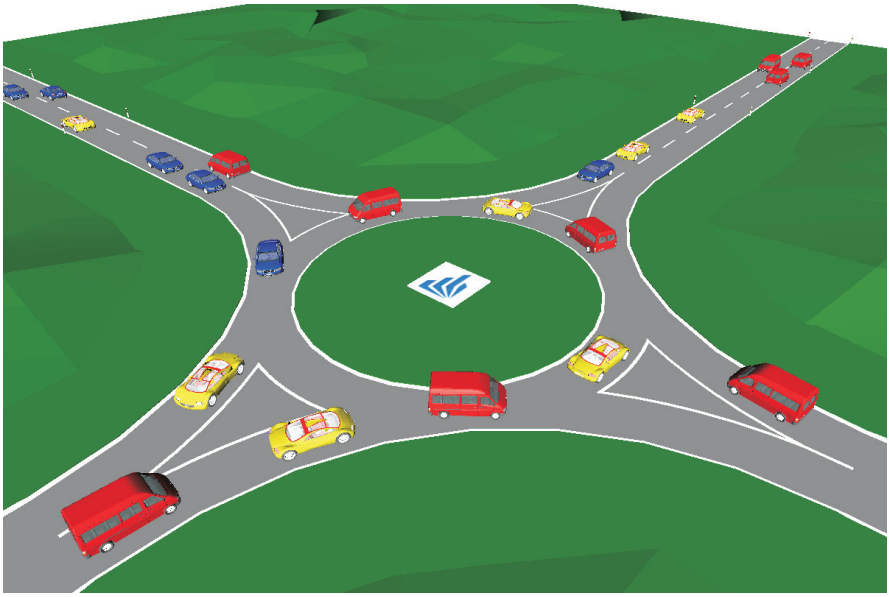

Abbildung 6.2: Simulation eines Kreisverkehrs (Durchmesser: 12 m)

Durchfahrtsmöglichkeiten bzw. *Routen* erzeugt. Die fahrdynamisch relevanten Eigenschaften der Fahrzeuge (maximale Antriebs- und Bremskraft, Gewicht, Luftwiderstand etc.), ihre Eintrittsgeschwindigkeiten sowie ihre Wunschfahrprofile wurden ebenfalls mit einem Zufallsgenerator bestimmt.

In den folgenden Abschnitten werden die Simulationsergebnisse[1] vorgestellt. Sie geben Aufschluss über die Leistungsfähigkeit des Verfahrens.

6.1 Simulationsergebnisse

Die im Folgenden dargestellten Simulationsergebnisse wurden mit einem 3,4-GHz-Rechner (Single-Core) erzielt und sind in unterschiedliche *Simulationsläufe* unterteilt. Jeder Simulationslauf entspricht einer bestimmten Konfiguration des Simulators, mit der jeweils eine bestimmte Eigenschaft des Verfahrens untersucht wurde. Weiterhin besteht jeder Simulationslauf aus $n = 16$

[1] Der Simulator, mit dem die Ergebnisse generiert wurden, kann über das Internetportal `http://www.kreuzungsmanagement.de/` bezogen werden. (Unter Umständen ist eine Simulation nicht möglich, wenn die Firewall *Zonealarm* auf dem Simulationsrechner installiert ist.)

Einzelsimulationen, die sich bezüglich ihrer Konfiguration lediglich durch das jeweilige Verkehrsaufkommen unterscheiden. Ein beim Simulator eingestelltes Verkehrsaufkommen von zum Beispiel 8.000 Fahrzeugen/Std. bedeutet, dass durch den Zufallsgenerator im Durchschnitt 8.000 Fahrzeuge/Std. erzeugt werden, mit ebenfalls zufällig erzeugten Eigenschaften. Für alle diese Fahrzeuge müssen Trajektorien berechnet werden. Konnten bei dieser Einstellung im Rahmen einer einstündigen Simulation lediglich 7.500 Fahrzeuge den Verkehrsknotenpunkt passieren, so bedeutet dies prinzipiell, dass etwa 500 Fahrzeuge inzwischen an den 4 Einfahrten des Verkehrsknotenpunktes warten, also ca. 125 Fahrzeuge an jeder Einfahrt. Die Kapazität des Verkehrsknotenpunktes bzw. der Durchsatz beträgt bei der entsprechenden Konfiguration des Simulators also 7.500 Fahrzeuge/Std. Bezüglich der zuvor genannten Simulationsläufe gilt ferner das Folgende: Für die i-te Simulation ($i = 1, ..., n$) eines Simulationslaufes beträgt das Verkehrsaufkommen $500 \cdot i$ Fahrzeuge/Std. Jeder Einzelsimulation liegt eine Simulationsdauer von 10 Minuten zugrunde. Weiterhin wurde während sämtlicher Simulationen bezüglich der Dimension Geschwindigkeit eine Diskretisierungsschrittweite von $\Delta v^* = 0,5$ m/s vorgegeben.

6.1.1 Variation der Fahrzeugabstände und Einfahrtsgeschwindigkeiten

Allgemeine Aussagen zu Kreuzung und Kreisverkehr

Zunächst wird gezeigt, wie sich die Durchschnittsgeschwindigkeit $\bar{v}$ während der Überquerung des jeweiligen Verkehrsknotenpunktes, gemittelt über alle Fahrzeuge, und der (maximale) Durchsatz an diesem Verkehrsknotenpunkt verhalten, wenn der Abstand sowie die Eintrittsgeschwindigkeit v_E der einfahrenden Fahrzeuge variiert werden. In Abbildung 6.3 sind die Ergebnisse von insgesamt 5 Simulationsläufen für die Kreuzung dargestellt. Mit dem Faktor **sp** wurde der minimale Abstand[2] zum vorausfahrenden Fahrzeug für neu in den Verkehrsknotenpunkt einfahrende Fahrzeuge festgelegt. Der Abstand ergibt sich dabei aus der Multiplikation der Geschwindigkeit [m/s] des vorausfahrenden Fahrzeugs mit dem Faktor **sp**. Der in Fahrschulen propagierten

[2]Dieser Abstand darf bei hohem Verkehrsaufkommen nicht unterschritten werden; mit abnehmendem Verkehrsaufkommen steigen die Abstände entsprechend der Einfahrtszeitpunkte der Fahrzeuge an.

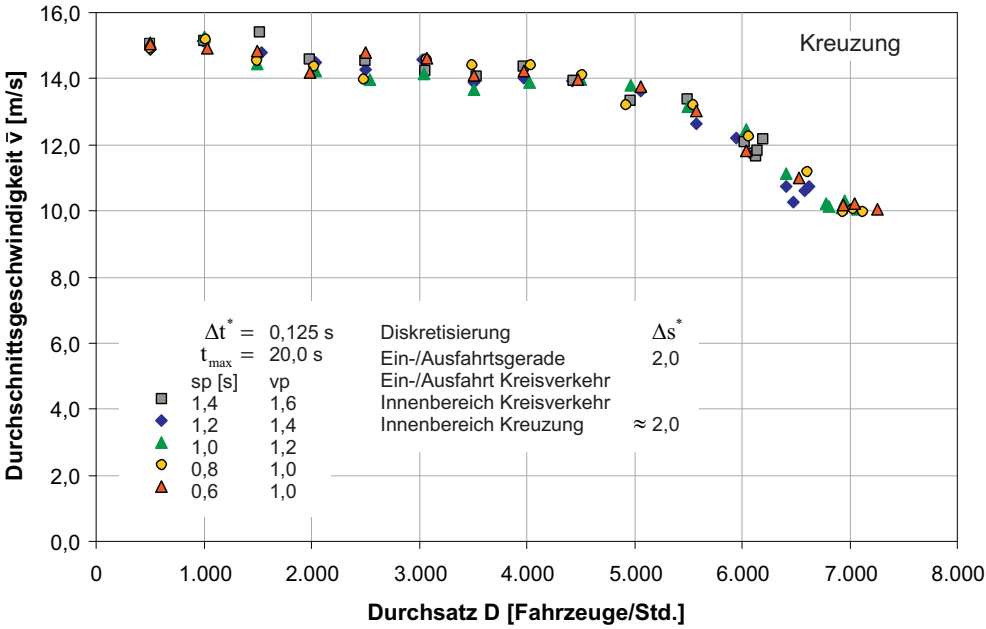

Abbildung 6.3: Durchschnittsgeschwindigkeit und max. Kapazität (Kreuzung)

Faustformel *"Halbe Tachoanzeige gleich Sicherheitsabstand in Metern"* für den Sicherheitsabstand entspricht ein Wert von **sp** = **1,8**:

v [km/h] $\rightarrow v \cdot 1000/3600$ [m/s] $\cdot$ 1,8 [s] $= \frac{1}{2} \cdot v$ [m] = Abstand [m]

Die Einfahrtsgeschwindigkeiten von neu in den Verkehrsknotenpunkt einfahrenden Fahrzeugen werden mit einem Zufallsgenerator (Gleichverteilung) erzeugt, wobei die Intervallgrenzen der möglichen Werte variieren: Sofern es vorausfahrende Fahrzeuge im Einfahrtsbereich gibt, so wird der Faktor **vp** mit der Geschwindigkeit des vorausfahrenden Fahrzeugs v_{vF} multipliziert. Die Obergrenze des Intervalls ergibt sich in diesem Fall dann gemäß $min(max(v_j^* \in V_D)$, **vp** $\cdot\ v_{vF})$. Für die Untergrenze gilt $max(v_{M,min}, 0.9 \cdot v_{vF})$. Bewegt sich das vorausfahrende Fahrzeug mit 10 m/s , so fährt ein folgendes Fahrzeug bei einem Faktor von **1,6** maximal mit einer Geschwindigkeit von 16 m/s in die Kreuzung oder den Kreisverkehr ein. Bei einem Faktor von **1,0** hat das neu einfahrende Fahrzeug maximal die Geschwindigkeit des ihm vorausfahrenden Fahrzeugs.

Die Abbildung 6.3 zeigt, dass der prinzipielle Verlauf der Durchschnittsgeschwindigkeit unabhängig ist vom Abstand zum vorausfahrenden Fahrzeug und dem jeweiligen Niveau der Eintrittsgeschwindigkeit. Es kann jedoch ebenfalls abgelesen werden, dass sich die Kapazität bzw. der maximale Durchsatz an der betrachteten Kreuzung steigern lässt, wenn ein geringer Fahrzeugabstand, zusammen mit einem möglichst gleichbleibenden Geschwindigkeitsniveau, eingestellt wird. Auf diese Weise konnte in den dargestellten Simulationsergebnissen eine Kapazitätssteigerung von ca. 6.000 auf ca. 7.000 Fahrzeuge/Std. erreicht werden.

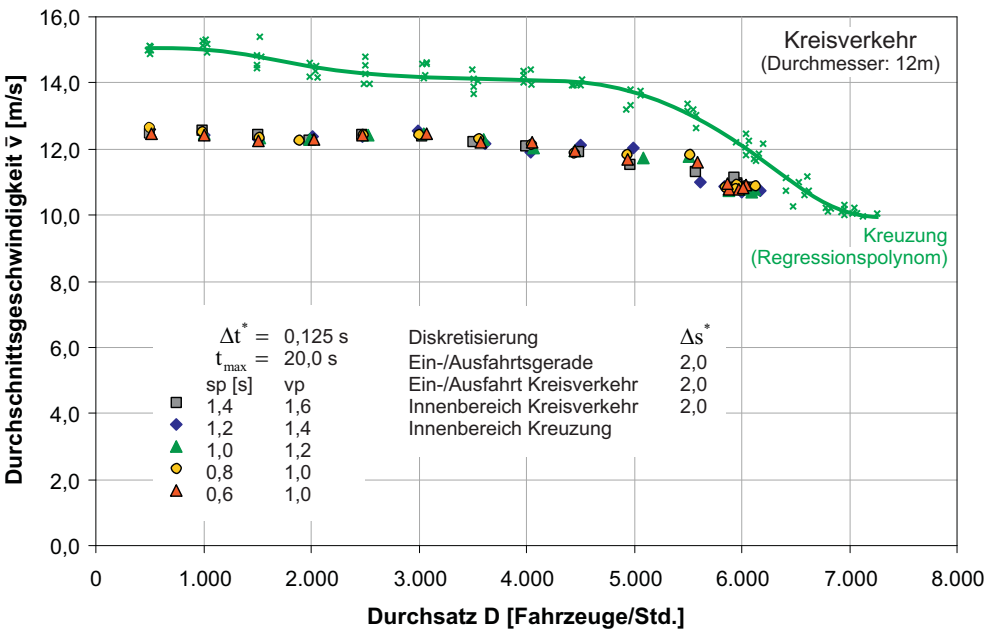

Abbildung 6.4: Durchschnittsgeschwindigkeit und max. Kapazität (Kreisverkehr mit 12 m Durchmesser)

In Abbildung 6.4 sind die Ergebnisse für den Kreisverkehr dargestellt. Auch hier kann man erkennen, dass der prinzipielle Verlauf der Durchschnittsgeschwindigkeiten unabhängig zu sein scheint von den Abständen und den Eintrittsgeschwindigkeiten der Fahrzeuge. Weiterhin fällt auf, dass die Durchschnittsgeschwindigkeiten und auch die maximale Kapazität unterhalb der jeweiligen Werte der Kreuzung liegen[3]. Dies erscheint jedoch plausibel: Wäh-

[3]Zum Vergleich sind in Abbildung 6.4 die Ergebnisse gemäß Abbildung 6.3 für eine Kreuzung eingefügt.

rend es sich bei ca. einem Drittel (bei Gleichverteilung) aller Fahrzeuge, die eine Kreuzung passieren, um Geradeausfahrer handelt, die ihre Geschwindigkeit nicht reduzieren müssen, prinzipiell während der Durchfahrt des Innenbereichs sogar beschleunigen können, müssen in einem Kreisverkehr sämtliche Fahrzeuge ihre Geschwindigkeit während der Durchfahrt des Kreisels deutlich verringern. Zusätzlich befinden sich Fahrzeuge beim Durchfahren eines Kreisverkehrs häufig auch deutlich länger auf einer Bahnkurve, bei relativ niedrigem Geschwindigkeitsniveau, als Fahrzeuge während des Durchfahrens einer Kreuzung. Insbesondere Linksabbieger müssen den Kreisel fast völlig durchfahren. All dies resultiert dann in einer deutlich geringeren Durchschnittsgeschwindigkeit für das Durchfahren eines Verkehrsknotenpunktes vom Typ Kreisverkehr. Im Gegensatz zur Kreuzung scheinen in dem untersuchten Kreisverkehr keine Kapazitätssteigerungen durch eine Verringerung der Fahrzeugabstände mehr möglich. Dies deutet auf einen Engpass im Kreisel hin. Während bei einer Kreuzung durch Verringerung der Fahrzeugabstände mehr Fahrzeuge auf den jeweiligen Einfahrtsgeraden Platz finden und so zu einer Durchsatzanhebung führen, hat dies beim untersuchten Kreisverkehr keine nennenswerte Auswirkung, da das zusätzliche Fahrzeugaufkommen nicht über den Kreisel abfließen kann. Auch dies erscheint plausibel, da sich Fahrzeuge, insbesondere Linksabbieger, die den Kreisel fast völlig durchfahren müssen, länger im Kreisel bzw. im Innenbereich des Verkehrsknotenpunktes befinden und diesen blockieren, als es bei einer Kreuzung der Fall ist.

Zusammenhang zwischen Rechenzeit und Kapazitätsgrenze

Abbildung 6.5 zeigt die Laufzeiten T_{DP} der Dynamischen Programmierung am Beispiel der Kreuzung, während Abbildung 6.6 die Häufigkeiten der jeweiligen Simulationsläufe darstellt, mit denen die Dynamische Programmierung abbricht und für jeweilige Fahrzeug erneut aufgerufen werden muss. Betrachtet man beide Abbildungen zusammen, so fällt auf, dass bis zu einem Verkehrsaufkommen von ca. 6.000 Fahrzeugen/Std. die Dauer der Dynamischen Programmierung bei der zugrundeliegenden Simulationskonfiguration relativ konstant zwischen 500 und 600 Millisekunden liegt. Bis zu diesem Verkehrsaufkommen können auch die Fahrprofile für sämtliche in die Kreuzung einfahrenden Fahrzeuge im ersten Versuch erfolgreich berechnet werden. Wird die Grenze D_{Gr} von ca. 6.000 Fahrzeugen/Std. jedoch überschritten, so

Ferner wurde für diese Ergebnisse ein Regressionspolynom 6. Ordnung bestimmt.

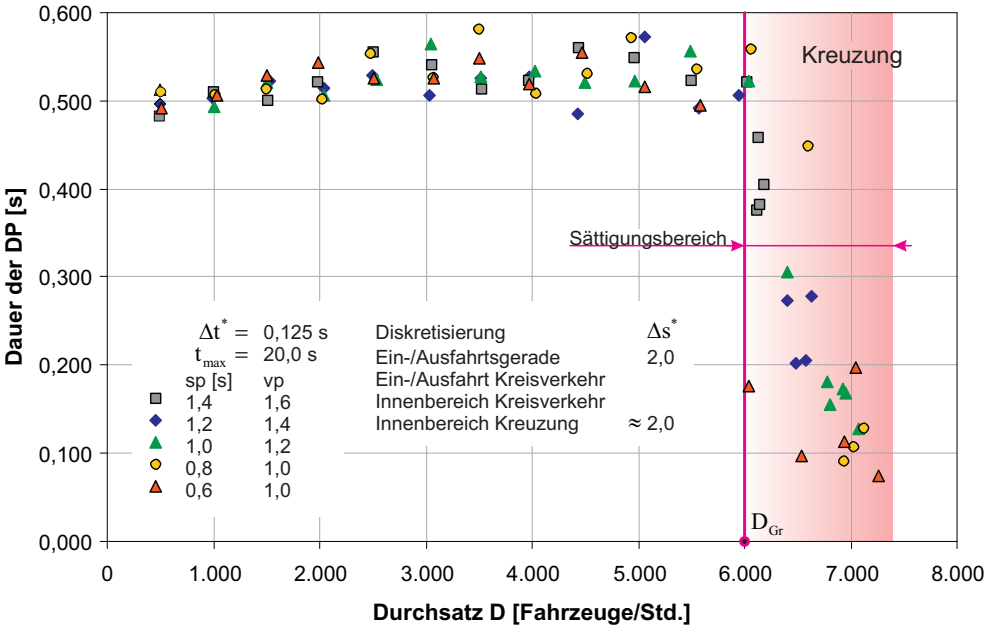

Abbildung 6.5: Durchschnittliche Dauer T_{DP} einer Optimierung (Kreuzung)

ist zu beobachten, dass die Anzahl der abgebrochenen Optimierungen rapide ansteigt, während sich die durchschnittliche Dauer eines Optimierungslaufes signifikant verkürzt. Der Grund dafür ist, dass je nach Simulationskonfiguration (Diskretisierungsschrittweite, Zeitfenster etc.) der Optimierungsraum $B_{D,f}$ mit bereits eingeplanten Trajektorien gefüllt bzw. der Großteil seiner Knoten als *besetzt* markiert sind. Die zugehörige Grenze D_{Gr} beim Fahrzeugaufkommen entspricht dem Beginn des Sättigungsbereiches. Sie deutet darauf hin, dass die maximale Kapazität des jeweiligen Verkehrsknotenpunktes bei der gegebenen Simulationskonfiguration fast erreicht ist. Für neu einzuplanende Fahrzeuge ergeben sich im nahezu gefüllten Optimierungsraum nur noch wenige Möglichkeiten für alternative Trajektoriensegmente. Diese wenigen Möglichkeiten können relativ schnell berechnet werden, was zu einer signifikanten Verkürzung der Laufzeit der Dynamischen Programmierung führt. Gleichzeitig steigt die Zahl der nicht erfolgreich durchgeführten Optimierungen an. Mit Beginn des Sättigungsbereiches besteht die Möglichkeit, dass Fahrzeuge kurzzeitig warten müssen, da nicht umgehend ein Fahrprofil errechnet werden kann. Errechnete Fahrprofile weichen ab dieser Grenze unter Umständen erheblich von dem Wunschfahrprofil ab. Mit zunehmen-

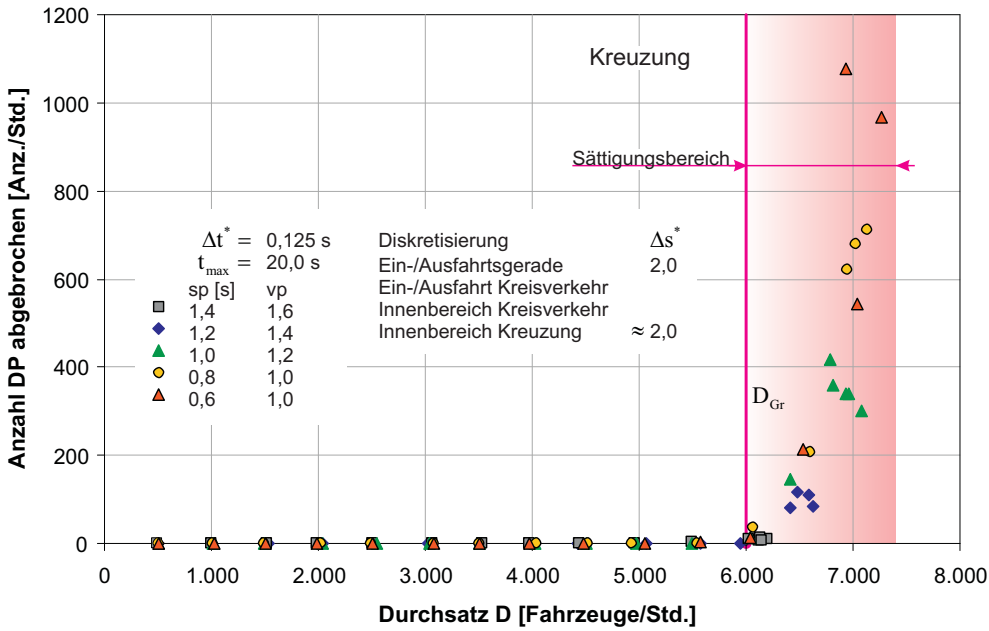

Abbildung 6.6: Anzahl der nicht erfolgreichen Optimierungen (Kreuzung)

der Eindringung in den Sättigungsbereich steigt also die Wahrscheinlichkeit für Wartezeiten der Verkehrsteilnehmer, genauso wie die Abweichung vom Wunschfahrprofil.

Bei Kreisverkehren wird dieselbe Beobachtung gemacht. Der einzige Unterschied besteht darin, dass bei einem Kreisverkehr bei gleicher bzw. ähnlicher Wahl der Diskretisierung der Beginn des Sättigungsbereiches bereits bei einem geringeren Verkehrsaufkommen (ca. 5.500 Fahrzeuge/Std.) erreicht wird. Dies liegt daran, dass die maximale Kapazität eines Kreisverkehrs in der Regel[4] geringer ist als bei einer Kreuzung, wie schon weiter oben angedeutet.

6.1.2 Variation der Wegdiskretisierung am Beispiel des Kreisverkehrs

In Abbildung 6.7 sind für den Kreisverkehr die Auswirkungen einer Änderung der Wegdiskretisierung Δs^* dargestellt. Als Ergebnis kann prinzipiell

[4]Hier spielt natürlich auch der Durchmesser des Kreisverkehrs eine Rolle (vgl. Abschnitt 6.1.4).

festgehalten werden, dass sich der Durchsatz durch eine Verfeinerung der Diskretisierungsschrittweite erwartungsgemäß signifikant anheben lässt. Dem

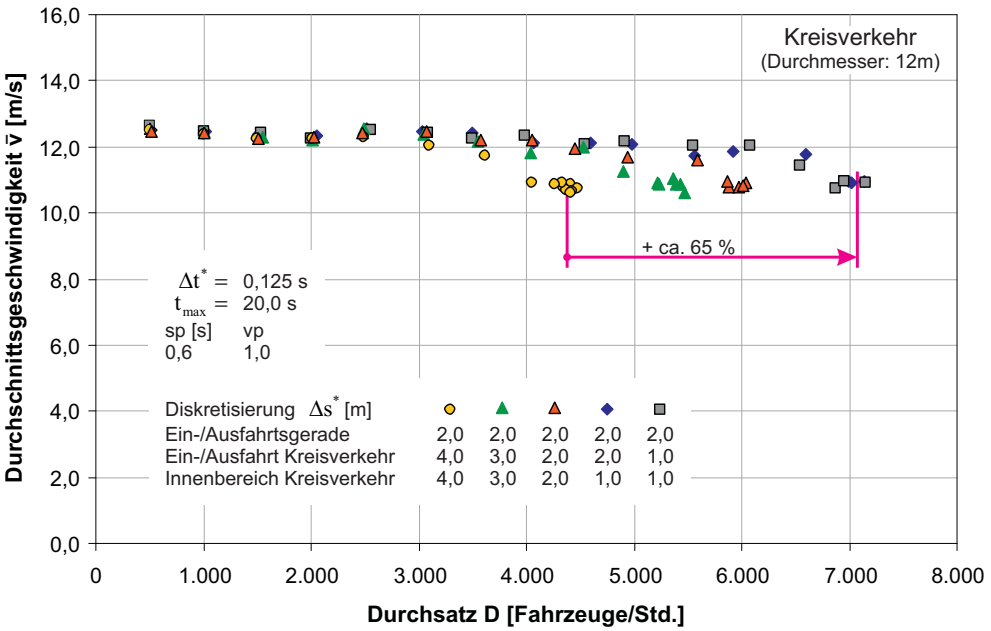

Abbildung 6.7: Durchschnittsgeschwindigkeit bei Variation von Δs^* (Kreisverkehr mit 12 m Durchmesser)

Experiment liegen insgesamt 5 Simulationsläufe zugrunde, wobei sich die letzten beiden Läufe lediglich bezüglich der Wegdiskretisierung der Ein- bzw. der Ausfahrt des Kreisels unterscheiden. Da diese beiden Läufe zu nahezu identischen Ergebnissen führen, liegt der Schluss nahe, dass der Engpass eines Kreisverkehrs primär durch die Diskretisierungsschrittweite des Innenkreisels bestimmt ist.

Aus den Ergebnisdaten der einzelnen Simulationsläufe können Mittelwerte für die Laufzeit einer Optimierung ($\bar{T}_{DP}$ [s]) sowie für die maximale Kapazität ($\bar{K}_S$ [Fahrzeuge/Std.]) bei der jeweiligen Simulationskonfiguration errechnet werden. Diese Mittelwerte sind für einen Simulationslauf mit $i = 1, ..., n$ Simulationen gemäß den Beziehungen

$$\bar{T}_{DP} = \frac{1}{|I|} \sum_{i \in I} T_{DP}(i), \quad \text{mit} \quad I = \{i \mid D(i) < D_{Gr}\}, \tag{6.1}$$

und

$$\bar{K}_S = \frac{1}{|I|} \sum_{i \in I} D(i), \quad \text{mit} \quad I = \{i \mid D(i) \geq D_{Gr}\}, \tag{6.2}$$

definiert, wobei $|I|$ für die Anzahl der summierten Werte steht.

Bei der Mittelwertbildung gemäß (6.1) werden nur Laufzeiten berücksichtigt, die zu Simulationen gehören, die unterhalb des Sättigungsbereichs liegen (vgl. Abbildungen 6.5 und 6.6). Bei (6.2) werden hingegen nur Durchsätze von Simulationen berücksichtigt, die innerhalb des Sättigungsbereiches liegen, da hier die maximale Kapazität bzw. der maximale Durchsatz von Interesse ist.

Anhand dieser Mittelwerte wird nun versucht, einen funktionalen Zusammenhang zwischen der Diskretisierungsschrittweite Δs^*, als unabhängige Variable, und den abhängigen Variablen $\bar{T}_{DP}(\Delta s^*)$ und $\bar{K}_S(\Delta s^*)$ herzustellen. Dazu wird die Ausgleichsrechnung verwendet.

In Abbildung 6.8 sind die entsprechenden Ergebnisse dargestellt. Es hat sich

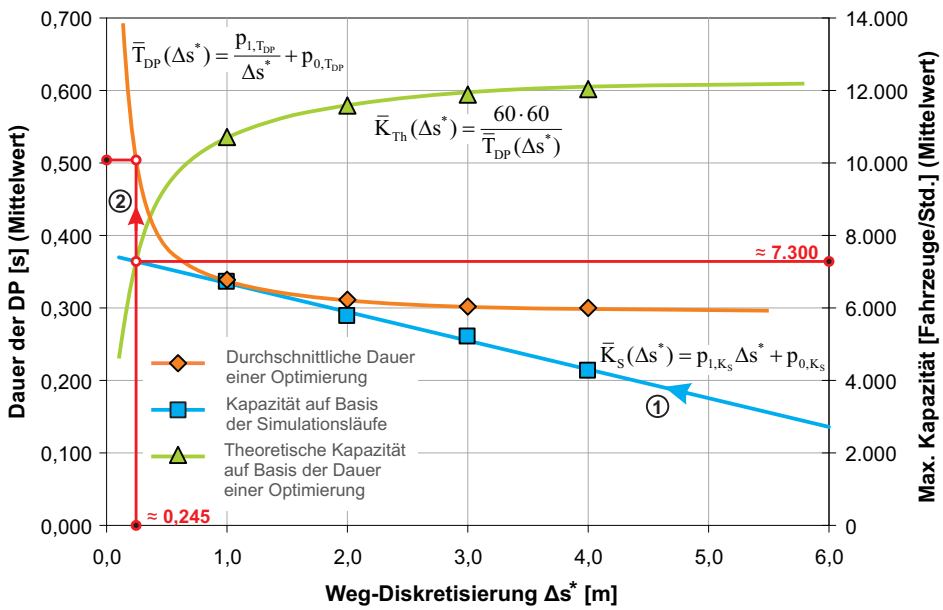

Abbildung 6.8: Durchsatzmaximierung durch Variation von Δs^* (Kreisverkehr mit 12 m Durchmesser)

gezeigt, dass die durchschnittliche Dauer $\bar{T}_{DP}(\Delta s^*)$ einer Optimierung am besten mit einer Funktion der Form

$$\bar{T}_{DP}(\Delta s^*) = \frac{p_{1,\bar{T}_{DP}}}{\Delta s^*} + p_{0,\bar{T}_{DP}} \tag{6.3}$$

beschrieben werden kann. Auf dieser Basis kann dann eine *theoretische* maximale Kapazität $\bar{K}_{Th}$ [Fahrzeuge/Std.] gemäß

$$\bar{K}_{Th}(\Delta s^*) = \frac{60 \cdot 60}{\bar{T}_{DP}(\Delta s^*)} \tag{6.4}$$

berechnet werden, die lediglich auf der durchschnittlichen Dauer einer Optimierung beruht und somit ein Maß für die Kapazität der für die Optimierung verwendeten Recheneinheit darstellt. Dem gegenüber steht die mittlere maximale Kapazität

$$\bar{K}_S(\Delta s^*) = p_{1,\bar{K}_S} \Delta s^* + p_{0,\bar{K}_S}, \tag{6.5}$$

die hier durch eine Geradengleichung approximiert wird. Sie steht für die Kapazität des Verkehrsknotenpunktes bei der jeweiligen Simulatorkonfiguration.

Soll nun für eine gegebene Simulatorkonfiguration und eine gegebene Recheneinheit lediglich durch Variation von Δs^* der Rechenaufwand derart skaliert werden, dass bei voll ausgelasteter Recheneinheit der Durchsatz maximiert wird, so ist folgendermaßen vorzugehen:

Auf Basis von Simulationsläufen mit unterschiedlicher Diskretisierungsschrittweite Δs^* sind für die Größen $\bar{T}_{DP}(\Delta s^*)$, $\bar{K}_{Th}(\Delta s^*)$ und $\bar{K}_S(\Delta s^*)$ Ausgleichs- bzw. Regressionskurven zu bilden. Dann ist der Schnittpunkt der Funktionen von $\bar{K}_{Th}(\Delta s^*)$ und $\bar{K}_S(\Delta s^*)$ zu ermitteln. Er gibt unmittelbar an, bei welcher Wegdiskretisierung die verwendete Recheneinheit voll ausgelastet ist.

6.1.3 Variation der Zeitdiskretisierung

Variation der Zeitdiskretisierung beim Kreisverkehr

Die Abbildung 6.9 zeigt die Auswirkung von Änderungen der Zeitdiskretisiereung Δt^* für einen Kreisverkehr. Auch in diesem Fall kann beobachtet

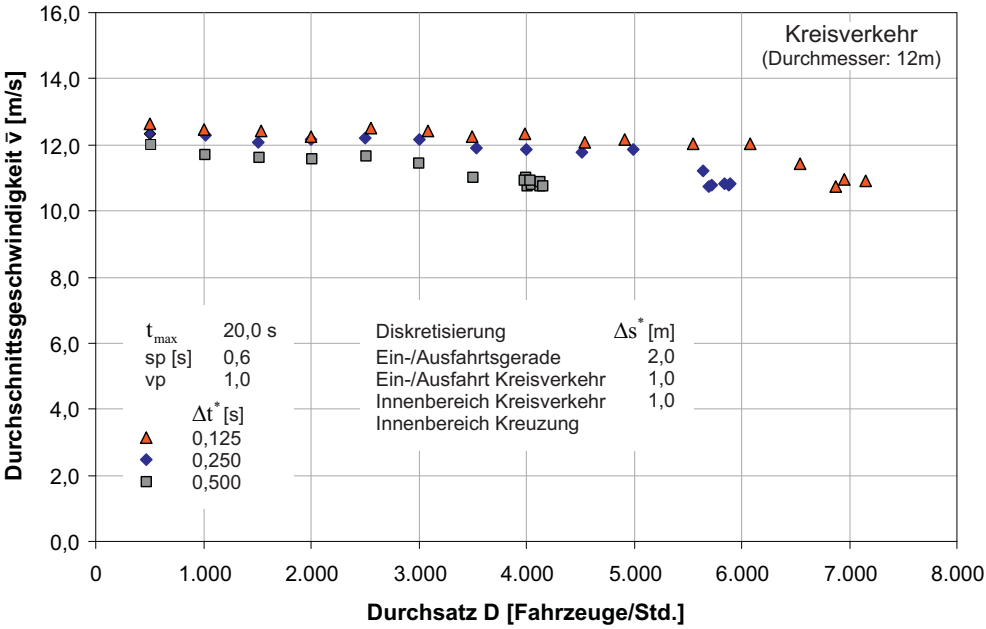

Abbildung 6.9: Durchschnittsgeschwindigkeit bei Variation von Δt^* (Kreisverkehr mit 12 m Durchmesser)

werden, dass sich der Durchsatz durch eine Verfeinerung der Diskretisierungsschrittweite signifikant anheben lässt.

Bildet man nun gemäß den Gleichungen (6.1) und (6.2) auch hier innerhalb der Simulationsläufe die Mittelwerte für die Laufzeit einer Optimierung $\bar{T}_{DP}(\Delta t^*)$ sowie für die maximale Kapazität $\bar{K}_S(\Delta t^*)$ gemäß Simulatorkonfiguration, so führt die anschließende Suche nach geeigneten Regressionskurven auf die Potenzfunktion

$$\bar{T}_{DP}(\Delta t^*) = p_{0,\bar{T}_{DP}}(\Delta t^*)^{p_{1,\bar{T}_{DP}}} \tag{6.6}$$

und die Geradengleichung

$$\bar{K}_S(\Delta t^*) = p_{1,\bar{K}_S}\Delta t^* + p_{0,\bar{K}_S}. \tag{6.7}$$

Für $\bar{K}_{Th}$ folgt, analog zum Beispiel der Wegdiskretisierung, die Beziehung

$$\bar{K}_{Th}(\Delta t^*) = \frac{60 \cdot 60}{\bar{T}_{DP}(\Delta t^*)}. \tag{6.8}$$

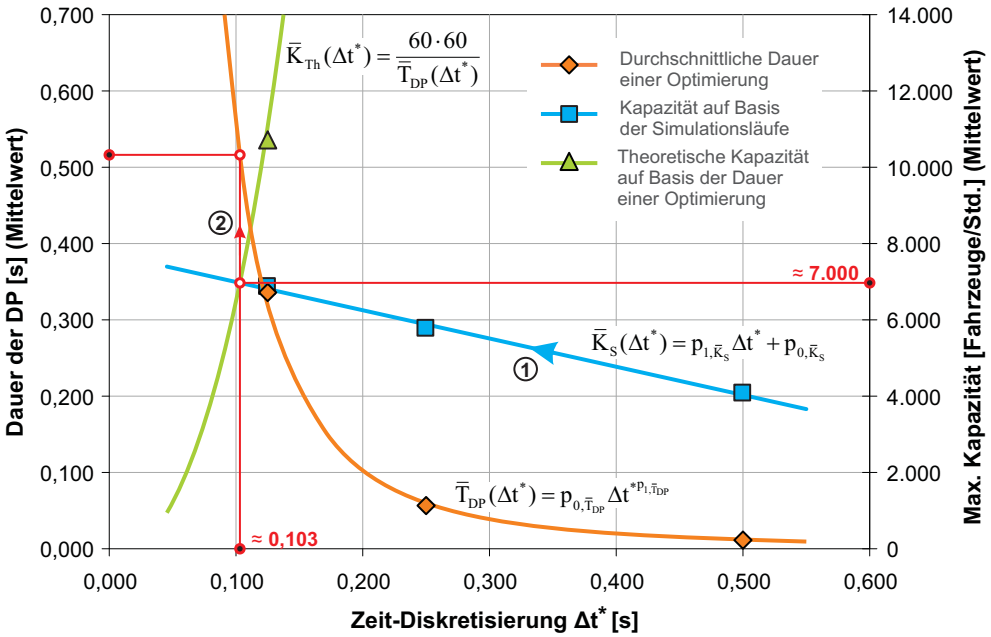

Abbildung 6.10: Durchsatzmaximierung durch Variation von Δt^* (Kreisverkehr mit 12 m Durchmesser)

Diese Ergebnisse sind graphisch in Abbildung 6.10 dargestellt.

Es fällt auf, dass sich durch eine Änderung der Zeitdiskretisierung der Rechenaufwand erheblich reduzieren lässt. So kann im dargestellten Fall bei einer mittleren Laufzeit $\bar{T}_{DP}$ der Dynamischen Programmierung von ca. 10 Millisekunden ($\Delta t^* = 0,5$ Sekunden) immer noch ein Durchsatz von ca. 4.000 Fahrzeuge/Std. realisiert werden, also mehr als doppelt so viel wie im Falle einer Kreuzung mit herkömmlicher Lichtsignalanlage. Bei diesem Durchsatz bleiben im Mittel jedoch ca. 0,9 Sekunden Zeit für die Optimierung eines Fahrzeugs. Unterstellt man nun einen linearen Zusammenhang zwischen Prozessortaktung und Dauer der Optimierung, so führt dies zu dem Ergebnis, dass eine zentrale Recheneinheit mit ca. 40 MHz ausreichend ist, um einen Durchsatz von ca. 4.000 Fahrzeuge/Std. zu gewährleisten.

Als wesentliche *Stellschrauben* für die Skalierung der Optimierungsdauer dienen die Diskretisierungsschrittweiten Δt^* und Δs^*. Die Schrittweite Δv^* für die Dimension Geschwindigkeit wurde während sämtlicher Simulationsläufe bei 0,5 Meter/Sekunde konstant gehalten. Dies hat sich als völlig ausreichend

erwiesen, da dieser Dimension gegenüber den Dimensionen Zeit und Weg nur eine untergeordnete Bedeutung bezüglich des erzielbaren Durchsatzes an einem Verkehrsknotenpunkt zukommt. Bei weniger leistungsstarken Recheneinheiten kann diese Schrittweite auch noch problemlos vergrößert werden.

Die Gleichungen (6.3) und (6.6) geben Hinweise, wie Δt^* oder Δs^* zu wählen sind, wenn die jeweils andere Größe konstant gehalten wird. Die optimale Wahl von Δt^* oder Δs^* für einen maximalen Durchsatz lässt sich auf diesem Wege allerdings nicht finden. Abhilfe schafft hier die Möglichkeit, durch die gemäß (6.1) und (6.2) ermittelten Punkte eine *Regressionsfläche* zu legen. Sofern die Fläche durch eine Linearkombination beschrieben oder ihre Beschreibung auf eine Linearkombination transformiert werden kann, so lassen sich die gesuchten Parameter leicht anhand der Pseudoinversen [26] ermitteln: Für den zuvor betrachteten Fall des Kreisverkehrs wird eine Regressionsfläche der Form

$$\bar{T}_{DP}(\Delta t^*, \Delta s^*) = p_0 \cdot \Delta t^{*p_1} \cdot \Delta s^{*p_2} \tag{6.9}$$

definiert. Die Parameter p_0, p_1 und p_2 der Fläche $\bar{T}_{DP}(\Delta t^*, \Delta s^*)$ lassen sich nach einer Transformation von (6.9) derart bestimmen, dass der Abstand der Punkte $\bar{T}_{DP,i}(\Delta t_i^*, \Delta s_i^*)$, mit $i = 1, ..., n$, zu dieser Fläche nach der Methode der kleinsten Quadrate minimiert wird. Das Ergebnis dieses Vorgehens zeigt Abbildung 6.11. Jeder Punkt auf der dargestellten Fläche entspricht einer Kombination von Δt^* und Δs^* und liefert einen Schätzwert für die zugehörige mittlere Dauer einer Optimierung $\bar{T}_{DP}(\Delta t^*, \Delta s^*)$. Für sämtliche Kombinationen ist nun unter Berücksichtigung der Kapazitätsbeschränkungen von Recheneinheit und Simulationskonfiguration ein Maximum zu finden.

Analog zu den Beziehungen (6.4) und (6.8) kann die *theoretische* Kapazität bzw. die Kapazität der Recheneinheit als

$$\bar{K}_{Th}(\Delta t^*, \Delta s^*) = \frac{60 \cdot 60}{\bar{T}_{DP}(\Delta t^*, \Delta s^*)} = \frac{60 \cdot 60}{p_0 \cdot \Delta t^{*p_1} \cdot \Delta s^{*p_2}} \tag{6.10}$$

definiert werden. Die Kapazitätsbeschränkungen $\bar{K}_S(\Delta t^*, \Delta s^*)$ gemäß Simulation des jeweiligen Verkehrsknotenpunktes werden mit einer Regressionsebene der Form

$$\bar{K}_S(\Delta t^*, \Delta s^*) = p_3 \Delta t^* + p_4 \Delta s^* + p_5 \tag{6.11}$$

angenähert. Die Parameter p_3, p_4 und p_5 können hier wieder leicht anhand der Pseudoinversen bestimmt werden. Die Ergebnisse sind in Abbildung 6.12 dargestellt. Man erkennt, dass sich die Flächen $\bar{K}_{Th}(\Delta t^*, \Delta s^*)$ und $\bar{K}_S(\Delta t^*, \Delta s^*)$

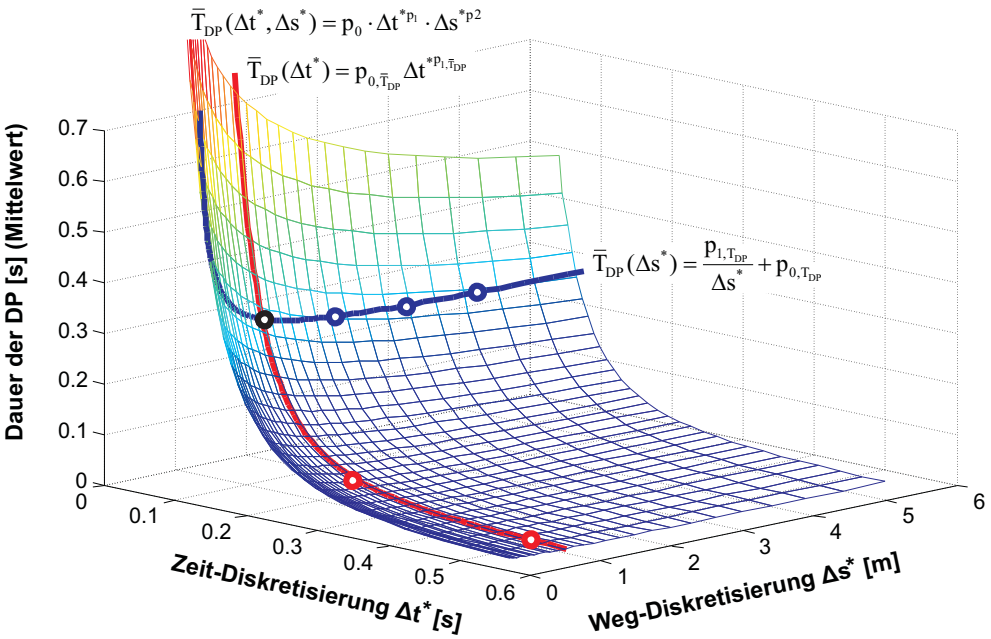

Abbildung 6.11: Regressionsfläche für $\bar{T}_{DP}(\Delta t^*, \Delta s^*)$ (Kreisverkehr mit 12 m Durchmesser)

schneiden. Sämtliche Punkte, die auf der Schnittlinie beider Flächen liegen, entsprechen Kombinationen von Δt^* und Δs^*, bei denen die verwendete Recheneinheit voll ausgelastet ist. Auf dieser Schnittlinie ist nun derjenige Punkt zu identifizieren, der dem maximalen Durchsatz entspricht.

Dazu wird $\bar{K}_{Th}(\Delta t^*, \Delta s^*) = \bar{K}_S(\Delta t^*, \Delta s^*)$ gesetzt und die Funktion

$$g(\Delta t^*, \Delta s^*) = \frac{60 \cdot 60}{p_0 \cdot \Delta t^{*p_1} \cdot \Delta s^{*p_2}} - p_3 \Delta t^* - p_4 \Delta s^* - p_5 \tag{6.12}$$

gebildet. Mit dem Newton-Verfahren können dann für vorgegebene Δt^* die zugehörigen Nullstellen von Δs^* berechnet werden. Für die daraus resultierenden Wertepaare $(\Delta t^*, \Delta s^*)$ sind dann die zugehörigen Durchsatzwerte zu berechnen und der maximale Wert zu identifizieren. Im beschriebenen Fall führt dies auf die Diskretisierungsschrittweiten $\Delta t^* \approx 0{,}130$ Sekunden und $\Delta s^* \approx 0{,}120$ Meter. Dieser Kombination entspricht ein maximaler Durchsatz gemäß (6.2) von ≈ 7.400 Fahrzeugen/Std.

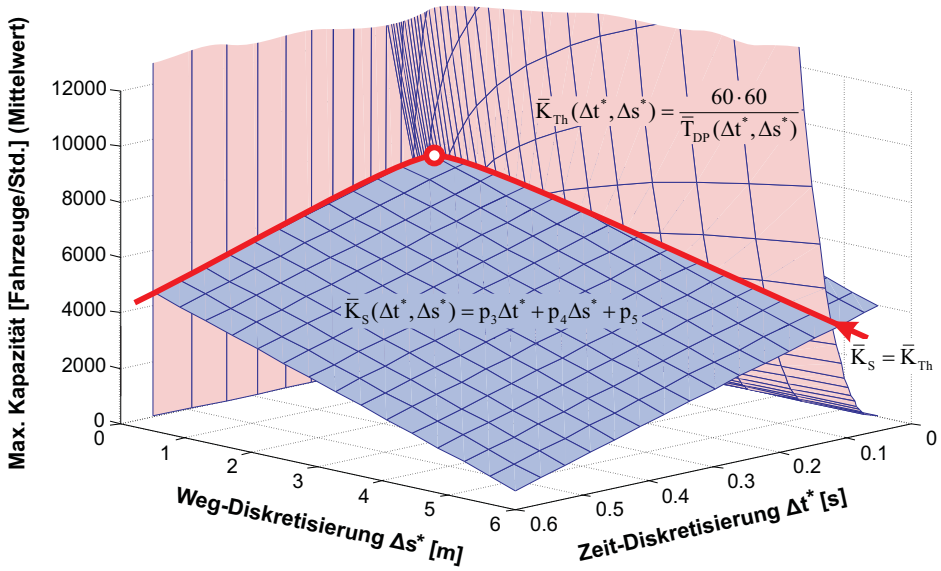

Abbildung 6.12: Skalierung der Diskretisierung auf max. Kapazität (Kreisverkehr mit 12 m Durchmesser)

Variation der Zeitdiskretisierung bei der Kreuzung

Variiert man die Zeitdiskretisierung Δt^* bei der Kreuzung, so führt dies auf prinzipiell identische Ergebnisse wie beim Kreisverkehr. Dies gilt sowohl für den Verlauf von $\bar{T}_{DP}(\Delta t^*)$ als auch den von $\bar{K}_S(\Delta t^*)$. Abbildung 6.13 zeigt die Auswirkungen bezüglich Durchschnittsgeschwindigkeit und maximalem Durchsatz. Beide Größen befinden sich jedoch bei ähnlicher Diskretisierung auf einem höheren Niveau als beim Kreisverkehr. Das annähernd gleich hohe Durchsatzniveau beim Kreisverkehr (Abbildung 6.9) resultiert ausschließlich daraus, dass im Vergleich zur Kreuzung mit einer etwa halb so großen Wegdiskretisierung Δs^* im Engpassbereich bzw. in der Mitte des Verkehrsknotenpunktes gearbeitet wurde. Bei der Kreuzungsgeometrie ist dies aufgrund der *hardcodierten* Wegstützstellen im Innenbereich zur Zeit noch nicht möglich. Dies ist auch der Grund dafür, weshalb zuvor die optimale Abstimmung von Δt^* und Δs^* am Beispiel des Kreisverkehrs gezeigt wurde.

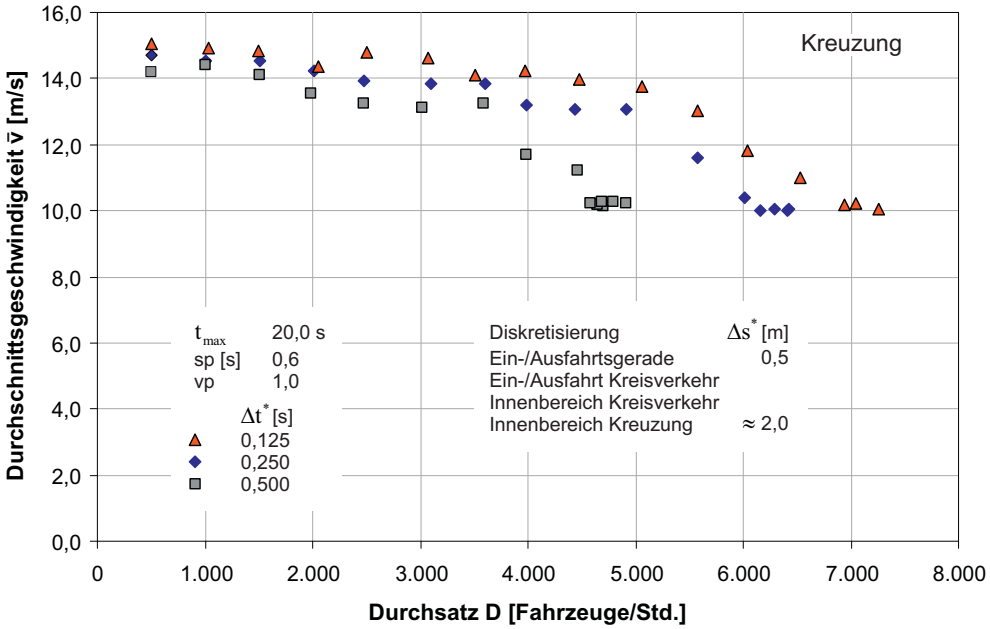

Abbildung 6.13: Durchschnittsgeschwindigkeit und maximale Kapazität (Kreuzung)

6.1.4 Einfluss des Durchmessers eines Kreisverkehrs

Wie Abbildung 6.14 zeigt, können sowohl die Durchschnittsgeschwindigkeit als auch der Durchsatz eines Kreisverkehrs durch den Durchmesser des Innenkreisels beeinflusst werden. Je größer der Durchmesser, desto höhere Geschwindigkeiten sind auf dem Innenkreisel möglich. Weiterhin finden auch mehr Fahrzeuge im Innenbereich des Verkehrsknotenpunktes Platz, so dass dem prinzipiellen Engpasscharakter des Innenkreisels eines Kreisverkehrs auf diese Weise gut entgegengewirkt werden kann. Andererseits haben die Fahrzeuge aber bei einem größeren Durchmesser auch einen längeren Weg zurückzulegen, was zu längeren Laufzeiten der Dynamischen Programmierung führt und in der Regel den maximal möglichen Durchsatz wieder reduziert. In realen Szenarien kommt hinzu, dass aufgrund des begrenzten Platzes nicht beliebig große Kreisverkehre realisiert werden können.

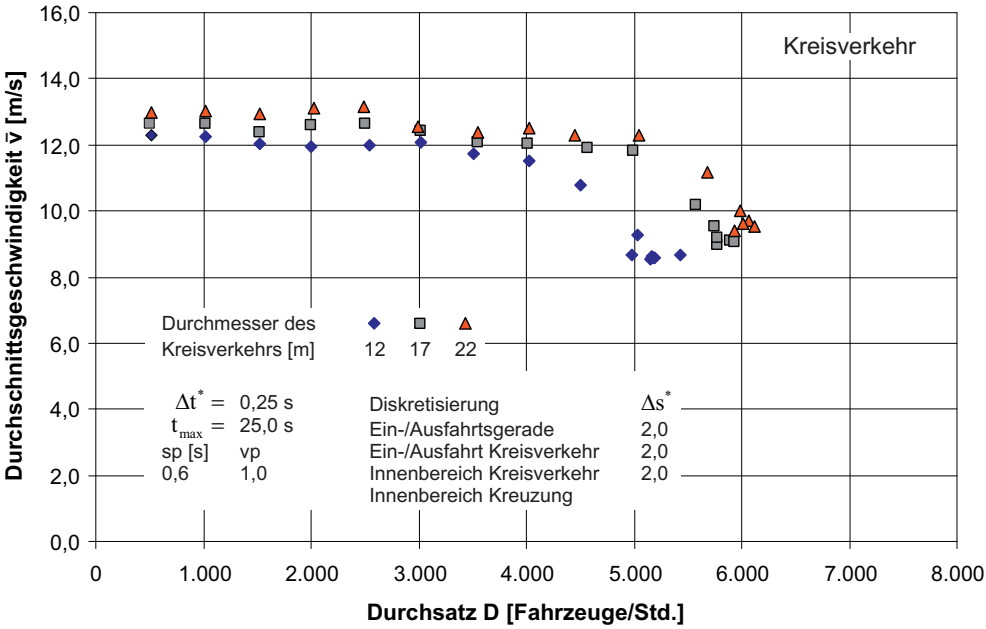

Abbildung 6.14: Durchschnittsgeschwindigkeit und maximale Kapazität in Abhängigkeit der Kreisverkehrgröße

6.2 Zusammenfassung und Ausblick

6.2.1 Zusammenfassung

Die wesentlichen Stärken des vorgestellten Verfahrens sind seine Leistungsfähigkeit und Flexibilität. Die auf systematischer Diskretisierung und kombinatorischer Optimierung basierende Trajektorienplanung ist echtzeitfähig bzw. kann auf die Kapazitäten unterschiedlicher Zielsysteme (im Sinne von Recheneinheiten) skaliert werden. Die Skalierung erfolgt im Wesentlichen über die Wahl der Diskretisierungsschrittweiten. Große Schrittweiten reduzieren zwar die Leistungsfähigkeit, im konkreten Anwendungsfall die maximale Kapazität eines Verkehrsknotenpunktes, ermöglichen dafür aber noch Recheneinheiten mit geringer Leistungsfähigkeit die Abarbeitung der Optimierungsaufgabe in Echtzeit. Es kann für die jeweils gegebene Konstellation der Optimierungsaufgabe garantiert werden, dass das Optimum innerhalb einer deterministischen Zeitspanne gefunden wird.

Die Flexibilität findet ihren Ausdruck darin, dass das Verfahren prinzipiell auf beliebige Verkehrsknotenpunkte angewendet werden kann, unabhängig von den Verläufen und der Anzahl der Fahrspuren. Auch können die unterschiedlichsten Fahrzeugtypen, unabhängig von ihren geometrischen Abmessungen oder von ihren fahrdynamischen Eigenschaften, berücksichtigt werden. Theoretisch könnten sogar Fahrspuren des Gegenverkehrs für eine Durchsatzerhöhung genutzt werden. Weiterhin können über die Wunschtrajektorie beliebige Zielgrößen wie beispielsweise Gesamtdurchsatz, Durchfahrtszeit, Komfort oder Energieverbrauch in die Optimierung eingebunden werden.

6.2.2 Ausblick auf zukünftige Arbeiten

Die in dieser Arbeit vorgestellten Methoden und Verfahren können als Basis für unterschiedlichste weitere Arbeiten dienen. Diese Arbeiten können im Bereich des Kreuzungsmanagements, aber durchaus auch in anderen Bereichen, angesiedelt sein, in denen ähnliche Optimierungsaufgaben unter Echtzeitanforderungen zu lösen sind. Im Folgenden werden einige mögliche weitere Arbeiten kurz skizziert.

Weiterführende Arbeiten im Rahmen des Kreuzungsmanagements

Eine besondere Herausforderung besteht in der Realisierung eines Demonstrators. Als Basis können herkömmliche Modellfahrzeuge in einem geeigneten Maßstab benutzt werden, die um entsprechende Sensorik, Aktorik und um eine leistungsfähige Recheneinheit in der Größenordnung eines Pocket-PCs erweitert werden.

Innerhalb dieser Arbeit wurden die theoretischen Grundlagen und ein tragbares Konzept für die Trajektorienplanung unter Echtzeitanforderungen vorgestellt. Man muss sich allerdings darüber im Klaren sein, dass die Ergebnisse der Simulationen auf idealen Bedingungen basieren: Die Positionen sämtlicher Fahrzeuge sind stets beliebig genau bekannt, zwischen den Fahrzeugen ist stets eine uneingeschränkte, verzögerungsfreie Kommunikation möglich, und die Fahrzeuge folgen immer exakt der vorgegebenen Trajektorie. In der Realität sind diese idealen Rahmenbedingungen allerdings nicht gegeben.

Es muss ein speziell auf den konkreten Anwendungsfall zugeschnittenes Verfahren entwickelt werden, das eine echtzeitfähige und ausreichend genaue Positionsbestimmung der Fahrzeuge ermöglicht. Dies kann beispielsweise auf der Auswertung von elektromagnetischen Wellen (Infrarot, Funk etc.) oder von Videoinformationen (Bilderkennung) basieren. Erste Untersuchungen und Konzepte liegen hier bereits vor. So wurde bereits im Jahr 2007 im Rahmen eines Projektseminars ein Demonstrator aufgebaut, in dem 3 *Leuchtfeuer* integriert waren. Bei diesen Leuchtfeuern handelte es sich um LEDs, deren Standorte fix und den Empfängern bzw. Fahrzeugen bekannt waren. Sie waren in einem Dreieck angeordnet und sendeten Lichtimpulse (Wellenlänge: 530 nm) mit unterschiedlicher Frequenz. Ein Fahrzeug wurde mit einer rotierenden Fotodiode ausgestattet und war in der Lage, jedes einzelne Leuchtfeuer exakt anhand der Frequenz des jeweils detektierten Lichtsignals zu identifizieren. Weiterhin wurden die Relativwinkel zwischen zwei unterschiedlichen Impulsfolgen bzw. zwischen zwei benachbarten Leuchtfeuern gemessen. Damit waren ausreichend Informationen vorhanden, um den Standort eines Fahrzeugs bestimmen zu können.

Die zweite große Aufgabe besteht in dem Aufbau einer geeigneten Funkkommunikation, wobei der informationstechnische Umgang mit deren Eigenheiten in adäquater Form zu spezifizieren ist. Anstatt mit verzögerungsfreier Kommunikation muss man bei der Kommunikation via Funk mit variablen Latenzzeiten umgehen. Zeitkritische Daten kommen also erst mit einer variablen Verzögerung beim jeweiligen Empfänger an. Weiterhin muss der Fall berücksichtigt werden, dass Teile des Kommunikationsnetzes ausfallen. All dies muss bei der Regelung der Fahrzeuge berücksichtigt werden. Vergleichbare Problemstellungen rücken momentan stärker in den Fokus des wissenschaftlichen Interesses und resultieren in entsprechenden Forschungsaktivitäten wie zum Beispiel im Schwerpunktprogramm *"Regelungstheorie digital vernetzter dynamischer Systeme"*[5] (SPP 1305) der Deutschen Forschungsgemeinschaft (DFG).

Aber auch im Bereich der theoretischen Grundlagen und Basiskonzepte des vorgestellten Verfahrens sind noch weitere Arbeiten denkbar. Hier steht die Laufzeitoptimierung der Dynamischen Programmierung im Vordergrund. Vorarbeiten im Bereich der Parallelisierung und beim Einsatz geeigneter Heuristiken sind bereits erfolgt [36]. Sehr interessant wäre auch eine Analyse, inwieweit sich das Laufzeitverhalten ändert, wenn ein Algorithmus aus dem

[5] Weitere Informationen sind unter `http://spp-1305.atp.rub.de/` abrufbar.

Bereich der LS-Verfahren für die Trajektorienberechnung verwendet wird. Prinzipiell scheint in diesem Bereich noch ein hohes Optimierungspotenzial zu bestehen.

Berechnung von Ausweichtrajektorien

Fahrerassistenzsysteme der Zukunft werden zunehmend autonome Fahrfunktionen integrieren. Dazu gehört beispielsweise auch das autonome Ausweichen vor plötzlich auftretenden Hindernissen (Fahrzeuge, Fußgänger, Wild etc.), wenn diese vom Fahrer nicht oder zu spät erkannt werden. Zu diesem Zweck muss von dem Fahrzeug innerhalb weniger Millisekunden eine Ausweichtrajektorie berechnet werden. Für diese Berechnung erscheint die vorgestellte Methode der Dynamischen Programmierung als sehr gut geeignet.

Trajektorienplanung im dreidimensionalen Raum

Die Trajektorienplanung beim Kreuzungsmanagement beschränkt sich auf eine Ebene. Es sind aber durchaus auch Anwendungen denkbar, bei denen eine Trajektorienplanung im dreidimensionalen Raum stattfinden muss. Ein Beispiel dafür sind Roboter, deren Arbeitsräume einander überschneiden und für die optimale und kollisionsfreie Trajektorien zu planen sind. Ein weiteres Szenario kann darin bestehen, für einen Roboter in einer unbekannten Umwelt, die über eine adäquate Umfeldsensorik erst nach und nach wahrgenommen wird, Trajektorien in Echtzeit zu planen, bspw. um Hindernisse zu überwinden. Das in dieser Arbeit vorgeschlagene Verfahren muss in den zuvor skizzierten Fällen um eine Dimension erweitert werden. Dabei ist jedoch mit einem überproportionalen Zuwachs bei der Dauer der Dynamischen Programmierung zu rechnen. Je nachdem, welche Echtzeitanforderungen zu erfüllen sind, ist dann zu untersuchen, inwieweit die Dynamische Programmierung angewendet werden kann bzw. inwieweit sich ihre Laufzeit auf ein tolerierbares Niveau reduzieren lässt. Es erscheint jedoch als sehr wahrscheinlich, dass die Dynamische Programmierung auch bei der Erweiterung des Optimierungsraumes um eine weitere Dimension immer noch sehr gut eingesetzt werden kann. Falls dies nicht der Fall sein sollte, so kann natürlich auch versucht werden, einen anderern Algorithmus für die Lösung des gemäß der Kapitel 3 und 4 modellierten Problems zu verwenden, bspw. eine Heuristik oder ein

LS-Verfahren. In jedem Fall sollte jedoch eine leistungsfähige Adaption bzw. Modifikation des vorgestellten Verfahrens möglich sein.

Literaturverzeichnis

[1] Bellman, Richard: *Dynamische Programmierung und selbstanpassende Regelprozesse*. Oldenbourg-Verlag, München/Wien, 1967

[2] Braess, Dietrich: Die Bestimmung kürzester Pfade in Graphen und passende Datenstrukturen. In: *Computing*, Springer-Verlag, Wien, 1971

[3] Bruns, Torsten ; Münch, Eckehard: Intersection Management as Self-Organisation of Mechatronic Systems. In: *New Trends in Parallel & Distributed Computing, 6th Int. Heinz Nixdorf Symposium*. Paderborn, 2006

[4] Bruns, Torsten ; Trächtler, Ansgar: Autonomes Kreuzungsmanagement – Trajektorienplanung mittels Dynamischer Programmierung. In: *4. Fachtagung "Steuerung und Regelung von Fahrzeugen und Motoren – AUTOREG 2008"*. Baden-Baden, 2008

[5] Bruns, Torsten ; Trächtler, Ansgar: Kreuzungsmanagement: Trajektorienplanung mittels Dynamischer Programmierung. In: *at – Automatisierungstechnik* 57 (2009), S. 253–261

[6] Dantzig, George B.: On the shortest route through a network. In: *Management Science* 6 (1960), S. 187–190

[7] Deppe, Markus ; Oberschelp, Oliver ; Münch, Eckehard: Echtzeit-Parameter-Optimierung und Überwachung in mechatronischen Systemen. In: *5. Magdeburger Maschinenbautage – Entwicklungsmethoden und Entwicklungsprozesse im Maschinenbau*, Logos-Verlag, Berlin, 2001, S. 355–364

[8] Dijkstra, Edsger W.: A note on two problems in connection with graphs. In: *Numerische Mathematik* 1 (1959), S. 269–271

[9] Domschke, Wolfgang: *Logistik: Transport*. Oldenbourg-Verlag, München, 1995

[10] Domschke, Wolfgang ; Drexl, Andreas: *Einführung in Operations Research*. Springer-Verlag, Berlin/Heidelberg, 2005

[11] Dresner, Kurt ; Stone, Peter: Multiagent Traffic Management: A Reservation-Based Intersection Control Mechanism. In: *The Third International Joint Conference On Autonomous Agents and Multiagent Systems (AAMAS 04)*. New York City, USA, 2004

[12] Dresner, Kurt ; Stone, Peter: A Multiagent Approach to Autonomous Intersection Management. In: *Journal of Artificial Intelligence Research (JAIR)* Bd. 31, 2008, S. 591–656

[13] FGSV: *Handbuch für die Bemessung von Straßenverkehrsanlagen (HBS) der Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV)*. FGSV-Verlag, Köln, 2005

[14] Föllinger, Otto: *Lineare Abtastsysteme*. Oldenbourg-Verlag, München/Wien, 1990

[15] Föllinger, Otto: *Optimale Regelung und Steuerung*. Oldenbourg-Verlag, München/Wien, 1994

[16] Föllinger, Otto: *Regelungstechnik – Einführung in die Methoden und ihre Anwendungen*. Hüthig-Verlag, Heidelberg, 1994

[17] Ford, Lester R. ; Fulkerson, Delbert R.: *Flows in Networks*. Princeton University Press, Princeton, 1962

[18] Görz, Günther ; Rollinger, Claus-Rainer ; Schneeberger, Josef: *Handbuch der Künstlichen Intelligenz*. Oldenbourg-Verlag, München/Wien, 2003

[19] Lückel, Joachim ; Hestermeyer, Thorsten ; Liu-Henke, Xiaobo: Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '01)*. Como, Italy, 2001

[20] Moore, Edward: The shortest path through a maze. In: *Int. Symposium on Theory of Switching, Part II*. Harvard University Press, Cambridge, MA, USA, 1959, S. 285–292

[21] Naumann, Rolf ; Rasche, Rainer: Intersection Collision Avoidance by Means of Decentralized Security and Communication Management of Autonomous Vehicles. In: *30th ISATA - ATT/IST Conference*. Florence, Italy, 1997

[22] Neuendorf, Norbert ; Bruns, Torsten: The Multi-Objective Optimiser MOPO in the Decentralised, Autonomous Intersection Management of Vehicles. In: *12th Mediterranean Conference on Control and Automation (Med 2004)*. Kusadasi, Turkey, 2004

[23] Neuendorf, Norbert ; Bruns, Torsten: The Vehicle Platoon Controller in the Decentralised, Autonomous Intersection Management of Vehicles. In: *IEEE International Conference on Mechatronics 2004 (ICM 2004)*. Istanbul, Turkey, 2004

[24] Neuendorf, Norbert ; Deppe, Markus: Vernetzung mechatronischer Systeme am Beispiel des dezentralen Kreuzungsmanagements für Kfz. In: *5. VDI-Mechatroniktagung 2003 – Innovative Produktentwicklung*. Fulda, 2003

[25] Nägler, Günter ; Walther, Hansjoachim: *Graphen, Algorithmen, Programme*. Springer-Verlag, Berlin/Heidelberg, 1987

[26] Papula, Lothar: *Mathematik für Ingenieure und Naturwissenschaftler 3: Vektoranalysis, Wahrscheinlichkeitsrechnung, Mathematische Statistik, Fehler- und Ausgleichsrechnung*. Bd. 3. Vieweg-Verlag, Braunschweig/Wiesbaden, 2001

[27] Rasche, Rainer: *Kreuzungsmanagement – informationstechnische Vernetzung autonomer Fahrzeuge als Beispiel für Selbstoptimierung im Maschinenbau*. VDI-Verlag, Düsseldorf, 2005

[28] Rasche, Rainer ; Lückel, Joachim ; Naumann, Rolf: Entwurf vernetzter mechatronischer Systeme am Beispiel des dezentralen Kreuzungsmanagements unter Verwendung von Workflowprozessen. In: *33. Regelungstechnisches Kolloquium*. Boppard, 1999

[29] Rasche, Rainer ; Lückel, Joachim ; Naumann, Rolf: Systematic Design of Crosslinked Mechatronic Systems, Exemplified by a Decentralized Intersection Management. In: *European Control Conference*. Karlsruhe, 1999

[30] Rasche, Rainer ; Naumann, Rolf ; Tacken, Jürgen: Managing Autonomous Vehicles at Intersections. In: *IEEE Intelligent Systems & Their Applications* 13 (1998), Nr. 3, S. 82–86

[31] Rasche, Rainer ; Naumann, Rolf ; Tacken, Jürgen ; Tahedl, Carsten: Validation and Simulation of Decentralized Intersection Collision Avoidance Algorithm. In: *IEEE Conference on Intelligent Transportation Systems (ITSC 1997)*. Boston, MA, USA, 1997

[32] Russel, Stuart ; Norvig, Peter: *Künstliche Intelligenz – Ein moderner Ansatz*. Pearson Education Deutschland, München, 2004

[33] Schepperle, Heiko ; Böhm, Klemens: Agent-Based Traffic Control Using Auctions. In: *Cooperative Information Agents XI: 11th International Workshop, CIA 2007, Delft, The Netherlands, Proceedings: No. 11 (Lecture Notes in Computer Science)*, Springer-Verlag, Berlin/Heidelberg, 2007

[34] Schneider, Gerhard ; Mikolcic, Hrvatin: *Einführung in die Methode der Dynamischen Programmierung*. Oldenbourg-Verlag, München/Wien, 1972

[35] Sedgewick, Robert: *Algorithmen in C*. Pearson-Studium, München, 2005

[36] Sommer, Philip: *Laufzeitoptimierung und Parallelisierung der Dynamischen Programmierung für den Anwendungsfall Trajektorienplanung im Kreuzungsmanagement*. Bachelorarbeit, Lehrstuhl für Regelungstechnik und Mechatronik (RtM), Universität Paderborn, 2009

[37] Tanenbaum, Andrew S.: *Moderne Betriebssysteme*. Hanser-Verlag, München, 1995

[38] Thiele, Rüdiger: *Leonhard Euler*. B. G. Teubner Verlagsgesellschaft, Leipzig, 1982

[39] Vasirani, Matteo ; Ossowski, Sascha: Exploring the Potential of Multiagent Learning for Autonomous Intersection Management. In: *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*. Estoril, Portugal, 2008

[40] VDI (Hrsg.): *VDI Richtlinie 2206 - Entwicklungsmethodik für mechatronische Systeme*. Beuth-Verlag, Berlin, 2004

[41] Zou, Xi ; Levinson, David: Vehicle Based Intersection Management with Intelligent Agents. In: *ITS America Annual Meeting Proceedings*. Minneapolis, MN, USA, 2003

Das Heinz Nixdorf Institut – Interdisziplinäres Forschungszentrum für Informatik und Technik

Das Heinz Nixdorf Institut ist ein Forschungszentrum der Universität Paderborn. Es entstand 1987 aus der Initiative und mit Förderung von Heinz Nixdorf. Damit wollte er Ingenieurwissenschaften und Informatik zusammenzuführen, um wesentliche Impulse für neue Produkte und Dienstleistungen zu erzeugen. Dies schließt auch die Wechselwirkungen mit dem gesellschaftlichen Umfeld ein.

Die Forschungsarbeit orientiert sich an dem Programm „Dynamik, Mobilität, Vernetzung: Eine neue Schule des Entwurfs der technischen Systeme von morgen“. In der Lehre engagiert sich das Heinz Nixdorf Institut in Studiengängen der Informatik, der Ingenieurwissenschaften und der Wirtschaftswissenschaften.

Heute wirken am Heinz Nixdorf Institut sieben Professoren mit insgesamt 200 Mitarbeiterinnen und Mitarbeitern. Etwa ein Viertel der Forschungsprojekte der Universität Paderborn entfallen auf das Heinz Nixdorf Institut und pro Jahr promovieren hier etwa 30 Nachwuchswissenschaftlerinnen und Nachwuchswissenschaftler.

Heinz Nixdorf Institute – Interdisciplinary Research Centre for Computer Science and Technology

The Heinz Nixdorf Institute is a research centre within the University of Paderborn. It was founded in 1987 initiated and supported by Heinz Nixdorf. By doing so he wanted to create a symbiosis of computer science and engineering in order to provide critical impetus for new products and services. This includes interactions with the social environment.

Our research is aligned with the program “Dynamics, Mobility, Integration: Enroute to the technical systems of tomorrow.” In training and education the Heinz Nixdorf Institute is involved in many programs of study at the University of Paderborn. The superior goal in education and training is to communicate competencies that are critical in tomorrows economy.

Today seven Professors and 200 researchers work at the Heinz Nixdorf Institute. The Heinz Nixdorf Institute accounts for approximately a quarter of the research projects of the University of Paderborn and per year approximately 30 young researchers receive a doctorate.