TUHH

Technische Universität Hamburg-Harburg

Diplomarbeit

Entwurf und Realisierung eines autonomen Multiagentensystems zur Transportsteuerung

eingereicht von:

Maurice Rosenfeld

Betreuer:

Prof. Dr. Ralf Möller

Prof. Dr. Volker Turau

Dipl.-Inform. Rainer Marrone

Technische Universität Hamburg-Harburg

Institut für Softwaresysteme

Harburger Schloßstraße 20

21079 Hamburg

Deutschland

Juli 2008

Selbstständigkeitserklärung

Hiermit erkläre ich, Maurice Rosenfeld (Matrikelnummer 18542), dass ich die von mir am heutigen Tage am Institut für Softwaresysteme der Technischen Universität Hamburg-Harburg eingereichte Diplomarbeit zum Thema

„Entwurf und Realisierung eines autonomen Multiagentensystems zur Transportsteuerung“

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet, sowie Zitate kenntlich gemacht habe.

Hamburg, den 31. Juli 2008

Unterschrift: _____

Inhaltsverzeichnis

1 Einleitung

Die moderne Transportlogistik verzeichnet eine stetig ansteigende Komplexität. In nahezu allen Branchen sind die Kosten der wichtigste Faktor für Entscheidungen. Ein besonders aus der Automobil- und Luftfahrtindustrie bekanntes Beispiel ist die Reduzierung der Lagerhaltungskosten auf ein Minimum durch eine Just-in-time-Lieferung der benötigten Produkte und Materialien. Der Einsatz dieser Form der Zulieferung wurde in den letzten Jahren stark forciert und ist heutzutage nicht mehr wegzudenken. Grundlage für das Funktionieren solcher Zustellmodelle sind eine vielseitige Transportlogistik und flexible Transportketten. Sie müssen eine Reihe von Anforderungen erfüllen, zu denen unter anderem das Reagieren auf Verkehrsbehinderungen und Produktionsengpässe sowie die Vermeidung von Verlust der Ware durch Systemausfälle zählen. Solche Voraussetzungen können jedoch durch zentrale Steuerungsstrukturen nur sehr schwer erfüllt werden. Ein Ansatz, gleichermaßen der steigenden Komplexität und der geforderten Flexibilität Rechnung zu tragen ist das Ausstatten einzelner Objekte der Transportkette mit Intelligenz und Kommunikationsfähigkeiten. Dieses Vorgehen hat zur Folge, dass den Objekten ein gewisses Maß an Autonomie zugesprochen werden muss. Aus softwaretechnischer Sicht eignet sich daher der Einsatz von Agenten, da sie diese Eigenschaften verkörpern.

Basierend auf diesem Ansatz ergibt sich die Zielsetzung dieser Arbeit. Zwei Objekte der Transportkette, die Frachtgüter und die Transportmittel, werden als Softwareagenten in einem Multiagentensystem zur Transportsteuerung modelliert. In ihm sollen Frachtgüter autonom an ihren Zielort transportiert werden. Dazu haben sie die Möglichkeit, ein geeignetes Transportmittel zu suchen und eigenständig die Transportkonditionen auszuhandeln. Mit in diesem Prozess sind unsicheres Wissen über die Transportmittel und die Konditionen zu berücksichtigen. Darüber hinaus sind im Frachtgutagenten Konzepte zur Überwachung der vereinbarten Konditionen zu integrieren. Ebenso soll der Aspekt von verteiltem Wissen zwischen kooperierenden Transportmittelagenten berücksichtigt werden. Diese Agenten sollen in der Lage sein, fehlendes Wissen des anderen Agenten zu erkennen um so eine kontextbedingte Wissenserweiterung zu unterstützen. Die Transportmittelagenten haben die Aufgabe, das gesamte Frachtgut autonom

zusammenzustellen und die Transportwege unter Berücksichtigung von kooperierenden Transportmittelagenten und unsicherem Wissen bezüglich der Verkehrssituation zu planen und durchzuführen.

Diese Arbeit gliedert sich in sechs weitere Kapitel. Das zweite Kapitel dient als Einführung zum Thema Softwareagenten. Darin wird erläutert, was ein Softwareagent ist und wann er als intelligent angesehen werden kann. Anschließend wird dargestellt, welche Betrachtungsweise die agentenorientierte Programmierung auf Softwaresysteme ermöglicht. Im dritten Kapitel wird beschrieben, aus welcher Intention die in [RZ94] gegebenen Grundlagen für Verhandlungen zwischen Agenten aufgestellt werden. Darüber hinaus werden drei Domänen vorgestellt, in denen die Vereinbarungen zwischen Agenten untersucht werden. Kapitel vier geht auf das Tourenplanungsproblem ein. Zuerst wird dargelegt, wie dieses Problem mathematisch beschrieben werden kann. Anschließend werden verschiedene Varianten aufgezeigt und es wird erläutert, welche Lösungsmethoden es für diese Klasse von Problemen gibt. Das fünfte Kapitel beschreibt ausführlich das dieser Arbeit zu Grunde liegende Modell. Die beteiligten Agenten werden vorgestellt und es wird aufgezeigt, unter Zuhilfenahme welcher Ansätze sie agieren. Neben den auf Kooperationen beruhenden Interaktionen wird ebenfalls geschildert, wie die Agenten mit Wissen umgehen und welche Möglichkeiten sie haben, dieses in Entscheidungsfindungen mit einzubeziehen. Im sechsten Kapitel wird die Auswahl einer passenden Agentenplattform für die Realisierung des Fallbeispiels dargelegt. Neben einer Beschreibung der Architektur wird erklärt, wie die für die Realisierung benötigten Kernfunktionen des Agentensystems eingebunden werden können. Danach erfolgt die detaillierte Beschreibung der implementierten Agenten. Hierbei wird ebenso erwähnt, wie die im Modell gefundenen Ansätze umgesetzt wurden. Der letzte Abschnitt dieses Kapitels schildert die Durchführung der Tests. Die Ergebnisse dieser Arbeit werden abschließend im siebten Kapitel zusammengefasst und es wird ein Ausblick auf weitere Arbeiten gegeben.

2 Softwareagenten

Dieses Kapitel erläutert die wichtigsten Begriffe in Bezug auf Softwareagenten. Zunächst wird auf die Fragestellung eingegangen, was genau ein Softwareagent ist. Anschließend werden die Voraussetzungen geklärt, unter denen ein Agent zusätzlich als intelligent angesehen werden kann. Danach folgt die Erläuterung, was ein Multiagentensystem ist. Zum Abschluss werden die Merkmale der agentenorientierten Softwareentwicklung in Anlehnung an [WJ05] beschrieben.

Agenten

Möchte man den Begriff des Agenten klären, so wäre es wünschenswert, wenn man auf eine einheitliche Definition zurückgreifen könnte, welche alle ihn beschreibenden Attribute enthält. Leider gibt es solch eine Definition in diesem Fall nicht. Stattdessen wird man, je mehr Bücher, Artikel und sonstige Quellen man zu diesem Thema liest, immer auf verschiedene Definitionen stoßen. Zwei Definitionen, die häufiger angegeben werden, sind die beiden folgenden von Russell und Norvig beziehungsweise von Wooldridge:

> „An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.“ [RN02]

> „An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.“ [WJ95]

Letztere Definition wird unter anderem in [Wei99] und [Woo01] noch einmal aufgegriffen und dient als Basis für die dortigen Einleitungen zum Thema Agenten. Wooldridge selbst macht noch einige Anmerkungen zu seiner oben angegebenen Definition. So wird dort nicht von intelligenten Agenten sondern nur von Agenten im allgemeinen Sinn

gesprochen. Zudem wird der Begriff der Umwelt in der sich ein Agent bewegt so generell wie möglich gehalten, da hier diverse Möglichkeiten vorstellbar sind.

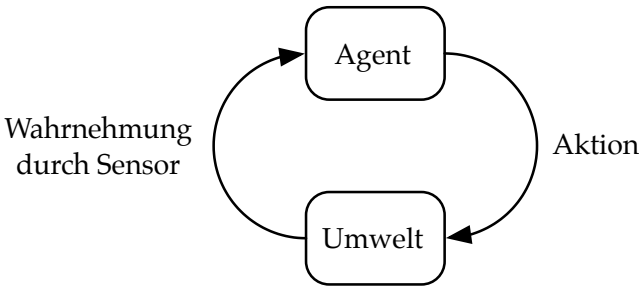

Abbildung 2.1: Interaktion eines Agenten mit seiner Umwelt

Abbildung 2.1 zeigt die abstrakte Darstellung eines Agenten in Bezug auf seine Umwelt aus [Wei99]. Diese deckt sich ziemlich genau mit der von Russell und Norvig gegebenen Definition. Ein Agent empfängt Daten über seine Umwelt mittels eines Sensors. Diese werden verarbeitet und veranlassen den Agenten zum Ausführen einer Aktion, welche die Umwelt beeinflusst. Dieser Prozess ist im Allgemeinen als nicht terminierend anzusehen.

Intelligente Agenten

Ein intelligenter Agent ist nach [Woo01] jener der in der Lage ist, flexible Aktionen auszuführen um seine vorgegebenen Ziele zu erreichen. Flexibel bedeutet in diesem Zusammenhang, dass ein Agent Aktionen durchführen kann, die aus den folgenden Eigenschaften hervorgehen:

- **Autonomie** Der Agent handelt selbstständig. Er löst seine Aufgaben ohne dass ein Mensch oder System eingreifen müssen.
- **Reaktivität** Ein Agent nimmt seine Umwelt wahr und reagiert auf Veränderungen die in ihr stattfinden, um seine Ziele zu erfüllen.
- **Pro-aktives Verhalten** Intelligente Agenten sind zu zielorientiertem Verhalten fähig indem sie selbst die Inititiative ergreifen um ihre Ziele zu verfolgen.
- **Soziale Fähigkeiten** Ein intelligenter Agent kann mit anderen Agenten kommunizieren um seine Ziele zu verfolgen.

- **Lernfähigkeit/Anpassungsfähigkeit** Der Agent kann aus zurückliegenden Entscheidungen oder Beobachtungen lernen und sich gegebenenfalls anpassen.

Wird im Folgenden von einem Agenten gesprochen, so ist damit ein intelligenter Agent gemeint.

Multiagentensysteme

Agieren mehrere intelligente Agenten zusammen in einer Umgebung, so spricht man von einem Multiagentensystem. Dazu muss eine solche Umgebung besondere Voraussetzungen erfüllen. Diese werden von Huhns und Stephens in [Wei99] durch die folgenden drei Aussagen näher charakterisiert.

1. Eine Multi-Agenten Umgebung bietet eine Infrastruktur an, welche die Kommunikation und Interaktionsprotokolle spezifiziert.
2. Multi-Agenten Umgebungen sind typischerweise offen und haben keinen zentralen Designer.
3. Multi-Agenten Umgebungen enthalten autonome und verteilte Agenten, welche eigennützig oder kooperativ handeln.

Merkmale agentenorientierter Softwareentwicklung

In der Vergangenheit kam es in der Softwareentwicklung durch die softwaretechnische Umsetzung von grundlegenden Systembetrachtungsweisen immer wieder zu Fortschritten. Zu diesen Betrachtungsweisen gehören unter anderem die Patternorientierung, die Komponentenorientierung und nicht zuletzt die Objektorientierung. Mit der Agentenorientierung steht Softwareentwicklern nun eine neue Systembetrachtungsweise zur Verfügung. Sie ermöglicht es, Softwaresysteme als menschliche Organisation aufzufassen und bietet somit den Vorteil, dass sich Entwickler vieler organisationstheoretischer Konzepte und Techniken bedienen können. Diese Sicht auf Software ist sehr intuitiv, da sie Begrifflichkeiten unseres Alltags verwendet. Ein Softwaresystem kann so als Organisation angesehen werden, in der Softwareeinheiten (Agenten) unter Beachtung von Regeln und Gesetzen bestimmte Aufgaben erfüllen und dazu beispielsweise autonom verhandeln, Teams zur gemeinsamen Aufgabenbewältigung bilden oder definierte Rollen übernehmen.

Die Komplexität von Software wird Untersuchungen zufolge auch in Zukunft stark zunehmen [Bun03]. Um mit dieser Komplexität umzugehen, haben sich in der Softwareentwicklung hauptsächlich vier Techniken hervorgehoben [BME+07; McC97]. Die erste dieser Techniken ist die Dekomposition der Software. Darunter ist die Zerlegung in übersichtlichere Teile zu verstehen, welche möglichst unabhängig voneinander zu entwickeln sind. Eine zweite Technik ist die Abstrahierung, welche die unwichtigen Aspekte eines Modells ausblendet und sich nur mit den wesentlichen Aspekten befasst. Die Strukturierung ist als dritte Technik zu nennen. Sie sorgt für eine Bestimmung der Beziehungen und Wechselwirkungen in einem Softwaresystem. Die letzte Technik ist die Wiederverwendung, welche den Einsatz von bereits erarbeiteten Ergebnissen in zukünftigen Projekten vorsieht.
Mit Hilfe der agentenorientierten Programmierung werden alle genannten Techniken unterstützt. So ist die Zerlegung eines Softwaresystems in atomare Bestandteile, also in Agenten, möglich. Ebenso ist eine Abstrahierung auf Wissens- sowie Sozialitätsebene möglich. Die Beziehungen der Agenten untereinander und somit zwischen den einzelnen Bestandteilen des Softwaresystems lassen sich aus den Interaktionen ableiten, welche zur Ausführung der jeweiligen Aufgaben erforderlich sind. Zu den wiederverwendbaren Artefakten bei der agentenorientierten Softwareentwicklung können zum einen ganze Agenten gehören, die bestimmte Aufgaben ausführen, zum anderen aber auch einzelne Bestandteile wie beispielsweise Wissensbasen. Ebenso ist es möglich, dass auch Kommunikationskomponenten wie Interaktionsprotokolle wiederverwendet werden.

Das bedeutendste Merkmal der Agentenorientierung stellt aus softwaretechnischer Sicht die Autonomie dar. Im Laufe der Weiterentwicklung von Softwaresystemen weisen die elementaren Softwareeinheiten einen zunehmenden Grad an Kapselung auf. Ein Vertreter dieser Einheiten sind Objekte in objektorientierten Programmiersprachen. Sie haben die Kontrolle über den Zugriff auf ihre Methoden und Attribute, so das diese nicht von anderen Objekten ausgelesen oder verändert werden können, sollte das besitzende Objekt dieses nicht vorsehen. Durch die Notwendigkeit, dass Methoden für andere Objekte zugänglich gemacht werden müssen um ein System oder Programm aufzubauen, verliert das Objekt die Kontrolle über sein Verhalten. Durch die Veröffentlichung von Methoden hat es sich dazu verpflichtet, die angebotenen Methoden immer dann auszuführen, wenn sie von anderen Objekten aufgerufen werden. Zusammenfassend kapseln Objekte somit ihre Identität, ihren Zustand und ihr passives Verhalten, indem sie Kontrolle über den Ablauf nach einer externen Aktivierung behalten. Agenten hingegen kapseln zusätzlich ihr aktives Verhalten und sind so in der Lage, selbst zu entscheiden wie, wann und mit wem sie interagieren. Fordert ein Agent einen anderen dazu auf, eine bestimmte Aktion auszuführen, so kann dieser die Aufforderung sogar ablehnen. Ein oft in diesem

Zusammenhang zitierter Spruch, der diese Möglichkeit beschreibt, lautet:

> „Objects do it for free, agents do it for money“

und stammt ebenfalls von Wooldridge.

Abschließend ist zu konstatieren, dass Agentenorientierung nicht den Anspruch erhebt, andere Systembetrachtungsweisen zu ersetzen. Durch die Integration von Kernaspekten anderer Ansätze stellt sie vielmehr eine sinnvolle Ergänzung zu ihnen dar.

3 Grundlagen für Vereinbarungen zwischen Agenten

Finden Verhandlungen zwischen Menschen statt, so treten häufig voneinander differierende Ziele auf. Dieser Tatsache kann im Verlauf der Verhandlungen durch die unterschiedlichsten Reaktionen Rechnung getragen werden. Um eine Einigung zu erzielen werden beispielsweise Versprechen gemacht oder Kompromisse eingegangen. Diese Einigungen treten im Alltag sehr häufig auf und werden manchmal noch nicht einmal mehr als solche wahrgenommen, da sie unterschwellig ablaufen. Treten jedoch Maschinen in Verhandlungen um Einigungen zu erzielen, so erfolgt dieses meistens in abgeschlossenen Systemen. Die für solche Verhandlungen aufgestellten Regeln sind im Allgemeinen denen einer Verhandlung zwischen Menschen ähnlich, jedoch sehr einfach gehalten. Aufgrund dieser Einschränkung kann es zu unerwünschten Resultaten kommen. Heutzutage treffen Maschinen jedoch immer mehr Entscheidungen in einer zunehmend autonomen Weise. Viele dieser Entscheidungen werden zwischen Maschinen untereinander getroffen. Sind die dafür vorgesehenen Protokolle einfach gehalten und nicht in der Lage, kooperative Möglichkeiten auszuschöpfen, arbeiten die Maschinen unwirtschaftlich.

Um Maschinen, und damit auch Agenten, flexible Interaktionen und konstruktive Kooperationen zu ermöglichen, haben Rosenschein und Zlotkin Ansätze aus der Spieltheorie auf Kooperationen in Multiagentensystemen (MAS) übertragen. In [ZR96b] beschreiben die beiden Autoren, dass in ihrer Vorstellung die Vertreter verschiedener, an der Entwicklung interessierter Unternehmen, Gremien bilden, in denen sie Verhandlungsmechanismen für die Interaktion der zu entwickelnden Agenten festlegen. Diese Mechanismen sollen auf bestimmten Standards beruhen, die gewisse Attribute in sich vereinigen. Welche Attribute dabei von den beiden Autoren vorgeschlagen werden, ist in Abschnitt 3.1 aufgezeigt. Ziel ihrer Arbeit ist es, den Entwicklern von Agenten geeignete Mechanismen für von ihnen definierte Verhandlungsdomänen zur Verfügung zu stellen. Dabei unterscheiden sie insgesamt drei Domänen, um speziellere Aussagen über Agenteninteraktionen treffen zu können. Sie differenzieren zwischen einer aufgabenorientierten Domäne (*engl. task*

oriented domain), die in Kapitel 3.2 beschrieben wird, einer zustandsorientierten Domäne (*engl. state oriented domain*), welche in Kapitel 3.3 erläutert wird und einer wertorientierten Domäne (*engl. worth oriented domain*), die in Kapitel 3.4 dargelegt wird. Zu allen drei Typen werden zudem eine formale Definition sowie ein Beispiel gegeben.

3.1 Standards für Verhandlungsmechanismen

Zlotkin und Rosenschein beschreiben fünf Attribute für Standards, auf die sich die Entwickler einigen sollten. Diese lauten:

- **Effizienz** Die gefundenen Verhandlungsmechanismen sollten pareto-optimale Ergebnisse hervorbringen. Pareto-optimal heißt, dass die Ergebnisse nicht in dem Sinne verändert werden können, dass ein einzelner Agent einen größeren Nutzen erhält, ohne dass gleichzeitig ein anderer Agent schlechter gestellt wird.
- **Stabilität** Der Mechanismus gibt jedem an der Verhandlung teilnehmenden Agenten einen Anreiz, sich in einer bestimmten Weise zu verhalten.
- **Einfachheit** Ein Standard sollte geringe Rechen- und Kommunikationskosten aufweisen, beziehungsweise keinen großen Overhead produzieren.
- **Verteiltheit** Die Verhandlungsmechanismen sollen nicht von einem zentralen Entscheidungsträger abhängig sein. So soll unter anderem verhindert werden, dass es zu einem Leistungsengpass kommt oder die Verhandlungen scheitern, sofern dieser Agent ausfällt.
- **Symmetrie** Symmetrie bedeutet, dass alle Agenten vom Mechanismus gleich behandelt werden.

Dabei kann es durchaus vorkommen, dass sich nicht in allen Punkten eine Einigung erzielen lässt und man zwischen einzelnen Eigenschaften abwägen muss beziehungsweise anwendungsspezifische Prioritäten setzt.

3.2 Aufgabenorientierte Domänen

Aufgabenorientierte Domänen sind solche, in denen die Aktivitäten von Agenten mittels einer Menge von Aufgaben, welche ausgeführt werden müssen, definiert werden können.

Dabei beeinflussen sich die Agenten nicht gegenseitig und alle zur Ausführung benötigten Ressourcen sind den Agenten zugänglich. Zugleich können in dieser Domäne Absprachen getroffen werden, welche eine Neuverteilung der Aufgaben nach sich ziehen. Diese Absprachen sind jedoch nur dann gültig, wenn alle daran beteiligten Agenten davon profitieren. Verhandlungen haben also immer die Aufgabe, beidseitig profitable Neuverteilungen von Aufgaben zu finden. Schlüsselelement dieser Domänen ist der Begriff der Aufgabe, den Zlotkin und Rosenschein als nicht weiter teilbaren, domänenspezifischen Job, welcher durchgeführt und erledigt werden muss, sehen.

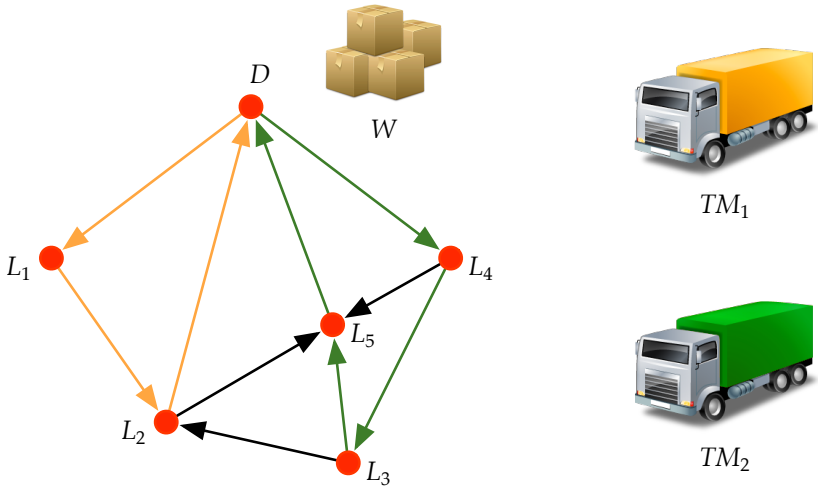

Abbildung 3.1: Beispiel einer aufgabenorientierten Domäne

Ein Beispiel für eine aufgabenorientierte Domäne ist die Auslieferungs-Domäne. In dieser Domäne geht es darum, dass Agenten Waren an Lagerhäuser ausliefern müssen. Diese werden durch Knoten V in einem gerichteten Graphen $G = G(V, E)$ repräsentiert und haben keine Beschränkung in Bezug auf die Menge an maximal aufzunehmender Ware. Alle Agenten starten von einem zentralen Depot, in dem sie Container kostenlos austauschen können, bevor sie mit der Auslieferung beginnen. Die spezifische Aufgabenmenge entspricht in diesem Fall allen Knoten des Graphen. Liegt ein Knoten auf der Route eines Agenten heißt das, dass er dort mindestens einen Container abliefern muss. Als beispielhafte Kostenfunktion einer Teilmenge an Adressen $X \subseteq V$ wird die Länge des minimalen Pfades angenommen, welcher im Verteilungspunkt anfängt und alle Knoten aus X beinhaltet. Dieses Beispiel ist in Abbildung 3.1 exemplarisch dargestellt. Gegeben sind die beiden Transportmittelagenten TM_1 und TM_2 sowie die auszuliefernden Frachtgüter F, welche sich im Depot D befinden. Als Startpunkt für TM_1 und TM_2 wird ebenfalls D angenommen. Die zu beliefernden Lagerhäuser werden durch $V = \{L_1, \ldots, L_5\}$ repräsentiert.

Sind die den beiden Transportmittelagenten zugeordneten Routen $R_1 = \{L_1, L_2\}$ und $R_2 = \{L_3, L_4, L_5\}$, so würde TM_1 die Strecke $D \rightarrow L_1 \rightarrow L_2 \rightarrow D$ und TM_2 die Strecke $D \rightarrow L_4 \rightarrow L_3 \rightarrow L_5 \rightarrow D$ fahren.

Die formale Definition einer aufgabenorientierten Domäne lautet wie folgt:

Definition 1 *Eine aufgabenorientierte Domäne ist ein Tupel $< \mathcal{T}, \mathcal{A}, c >$, wobei*

1. *$\mathcal{T}$ die Menge aller möglichen Tasks ist;*
2. *$\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ eine geordnete Liste aller Agenten darstellt;*
3. *c eine monotone Funktion $c : 2^{\mathcal{T}} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ ist. $2^{\mathcal{T}}$ stellt alle finiten Untermengen von $\mathcal{T}$ dar. Für jede finite Menge von Tasks $\mathcal{X} \subseteq \mathcal{T}$ stellt $c(\mathcal{X})$ die Kosten dar, wenn alle Tasks in $\mathcal{X}$ von einem Agenten ausgeführt werden. c ist monoton. Das heißt, für jegliche zwei finiten Untermengen gelten $\mathcal{X} \subseteq \mathcal{Y} \subseteq \mathcal{T}$ und $c(\mathcal{X}) \leq c(\mathcal{Y})$;*
4. *$c(\emptyset) = 0$.*

3.3 Zustandsorientierte Domänen

Zustandsorientierte Domänen stellen eine Obermenge von aufgabenorientierten Domänen dar und unterscheiden sich maßgeblich durch zwei Eigenschaften von ihnen. Zum einen können Seiteneffekte auftreten und zum anderen haben die Agenten nicht mehr einzelne Aufgaben auszuführen sondern Pläne, welche eine Reihe von atomaren Operationen enthalten. Diese Pläne dienen dazu, ihre Umwelt, in der sich die Agenten bewegen, von einem Anfangszustand in den nächsten zu bringen. Dieses geschieht so lange, bis ein gewünschter Endzustand erreicht ist. Führt ein Agent in einer aufgabenorientierten Domäne seine eigene Menge von Aktionen aus, so hat dieses keinen Einfluss auf die anderen Agenten und hindert sie nicht daran, ihre eigenen Ziele zu erreichen. In einer zustandsorientierten Domäne kann es jedoch vorkommen, dass ein Agent zufällig einen Zustand herstellt, welcher dem gewünschten Endzustand eines anderen Agenten entspricht. Ebenso ist es allerdings möglich, dass zwei herzustellende Endzustände einen Konflikt auslösen. Tritt solch ein Fall auf, muss über den gemeinsam zu realisierenden Endzustand der Umwelt verhandelt werden.

Ein Beispiel für eine zustandsorientierte Domäne ist die Auslieferungs-Domäne mit beschränkten Kapazitäten. Sie stellt eine Erweiterung des im vorherigen Abschnitt 3.2

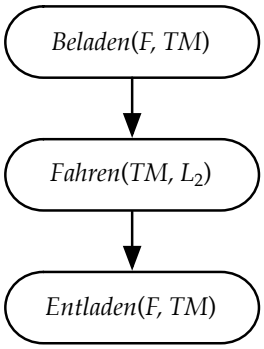

Abbildung 3.2: Beispiel eines Plans bestehend aus atomaren Operationen

beschriebenen Szenarios dar. Dabei sei die Kapazität der Transportmittel auf zwei Frachtgüter und die Kapazität der Lagerhäuser auf zehn Frachtgüter limitiert. Die Aufgabe der Agenten besteht in diesem Fall darin, Frachtgüter von einem Lagerhaus zum nächsten zu transportieren. Um diese Lieferungen durchzuführen, haben sie die Möglichkeit, Transportmittel an jedem beliebigen Lagerhaus zu mieten. Eine Limitierung bezüglich der verfügbaren Transportmittel, welche an jedem Lagerhaus zur Verfügung stehen, besteht dabei nicht. Die Kostenfunktion ist durch die Länge des minimalen Pfades, ausgehend von dem Lagerhaus an dem das Frachtgut abgeholt werden soll, hin zu dem Lagerhaus, an dem sie abgeliefert werden soll, gegeben.
Atomare Operationen für dieses Beispiel sind:

- $Beladen(F, TM)$: Das Frachtgut F wird auf das Transportmittel TM geladen.
- $Entladen(F, TM)$: Das Frachtgut F wird vom Transportmittel TM abgeladen.
- $Fahren(TM, L)$: Das Transportmittel TM fährt zum Lagerhaus L.

Diese Operationen befähigen die Agenten, die von ihnen verlangten Aufgaben durch das Aufstellen von Plänen auszuführen. Hat ein Agent die Aufgabe ein Frachtgut F vom Lagerhaus L_1 zum Lagerhaus L_2 zu bringen, so kann er an L_1 ein Transportmittel anmieten und dann den in Abbildung 3.2 dargestellten Plan durchführen.

Eine Konfliktsituation kann in diesem Beispiel entstehen wenn zwei Transportmittel die Aufgabe haben, jeweils eine Einheit Frachtgut in einem Lagerhaus abzuliefern, in dem bereits neun von zehn Frachtgüter eingelagert sind. Liefert nun einer der beiden Agenten sein Frachtgut ab, so hat diese Aktion den Seiteneffekt, dass das Lagerhaus voll ist und der zweite Agent nicht mehr in der Lage ist, seine Aufgabe zu erfüllen.

Die formale Definition einer zustandsorientierten Domäne ist folgende:

Definition 2 *Eine zustandsorientierte Domäne ist ein Tupel* $< \mathcal{S}, \mathcal{A}, \mathcal{J}, c >$*, wobei*

1. $\mathcal{S}$ *die Menge aller möglichen Umweltzustände ist;*
2. $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ *eine geordnete Liste aller Agenten darstellt;*
3. $\mathcal{J}$ *die Menge aller möglichen Verbundpläne (engl. joint plans) ist. Ein Verbundplan (zum Beispiel ein n-Agenten Plan)* $J \in \mathcal{J}$ *bewegt dabei die Umwelt, in der sich die Agenten bewegen, von einem Zustand* $\mathcal{S}$ *in den nächsten. Die Aktionen, die Agent k dabei übernimmt werden als k's Rolle in J bezeichnet und mit* J_k *abgekürzt. J kann zudem als* $J = (J_1, J_2, \ldots, J_n)$ *geschrieben werden.*
4. *c eine Funktion* $c : \mathcal{J} \rightarrow (\mathbb{R}^+)^n$ *ist. Für jeden Verbundplan J aus* $\mathcal{J}$ *ist* $c(J)$ *ein Vektor mit n positiven, reellen Zahlen, welche die Kosten jeder Agentenrolle in dem Verbundplan darstellen.* $c(J)_i$ *ist das i-te Element des Kostenvektors, das heißt es stellt die Kosten der i-ten Rolle aus J dar. Hat ein Agent keine Rolle in J, so sind seine Kosten* 0*.*

3.4 Wertorientierte Domänen

Wertorientierte Domänen stellen wiederum eine Obermenge von zustandsorientierten Domänen dar. In solchen Domänen ordnen die Agenten jedem potenziellen Zustand der Umwelt einen bestimmten Wert zu. Der Wertebegriff wird in [ZR96a] als die maximal zu erwartenden Kosten, die ein Agent zu tragen bereit ist, um ein bestimmtes Ziel zu erreichen, definiert. In zustandsorientierten Domänen hingegen werden nur den unmittelbaren End- beziehungsweise Zielzuständen Werte zugeordnet; alle anderen Zustände haben den Wert 0. Eine weitere Eigenschaft, welche die beiden zuvor genannten Domänen nicht aufweisen, ist die Möglichkeit der Agenten, Kompromisse bezüglich des Erreichungsgrades ihrer Ziele einzugehen.
Die formale Definition einer wertorientierten Domäne gleicht der einer zustandsorientierten Domäne. Die für diesen Typ von Domäne benötigte Wertefunktion ist durch

$$W_i(f) = \begin{cases} w_i & \text{wenn } f \models g_i \\ 0 & \text{sonst.} \end{cases}$$

gegeben. $W_i(f)$ beschreibt dabei, wie viel des Ziels von Agent A_i bereits erreicht wurde, oder anders ausgedrückt, wie nahe Zustand f an dem Abschluss von A_is Gesamtziel ist.

$f \models {g^k}_i$ bedeutet, dass ein Teilziel ${g^k}_i$ in Zustand f erreicht wurde. W_i kann dabei jede Funktion sein, die einen Wertebereich von $\mathbb{R}$ hat.

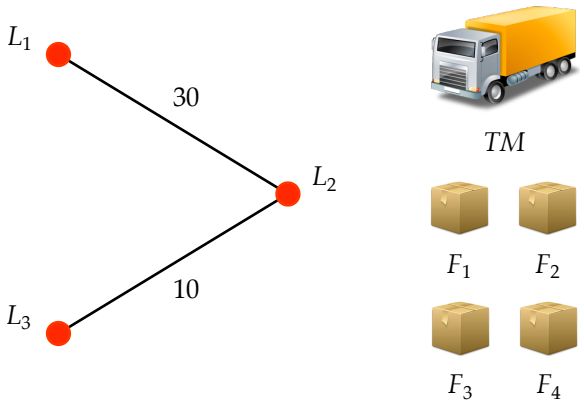

Abbildung 3.3: Beispiel für eine wertorientierte Domäne

Ein Beispiel für eine wertorientierte Domäne zeigt Abbildung 3.3. Es basiert auf dem Beispiel der in Abschnitt 3.3 beschriebenen Auslieferungs-Domäne und geht ebenfalls von der Annahme aus, dass Agenten die Aufgabe haben, Frachtgüter zu transportieren. Dazu haben sie wiederum die Möglichkeit, Transportmittel an den Lagerhäusern zu mieten. Um Pläne für das Erreichen ihrer Ziele aufzustellen, haben die Agenten die Operationen $Beladen(F, TM)$, $Entladen(F, TM)$ sowie $Fahren(TM, L)$ zur Verfügung. Das Beladen und Entladen erzeugt Kosten in Höhe von 1, während das Fahren die Kosten des zurückgelegten Weges verursachen ($L_1 \rightarrow L_2 : 30$ und $L_2 \rightarrow L_3 : 10$). Während die Transportmittel eine Kapazität von vier Frachtgütern haben sind die beiden Lagerhäuser L_2 und L_3 bereits so gefüllt, dass nur noch je zwei Frachtgüter eingelagert werden können. Kooperationen zwischen den verschiedenen Agenten sind möglich, indem sich mehrere von ihnen ein Transportmittel zur Auslieferung der Waren teilen. Gegeben seien zwei Agenten A_1 und A_2 sowie die Frachtgüter $F = \{F_1, F_2, F_3, F_4\}$. Die Aufgabe von Agent A_1 besteht in $F_1, F_2 : L_1 \rightarrow L_3$, die von Agent A_2 in $F_3 : L_1 \rightarrow L_2$ und $F_4 : L_1 \rightarrow L_3$. Kann ein Agent seine Aufgabe komplett erfüllen, so ordnet er diesem Zustand einen Wert von $w = 50$ zu. Der Nutzen (*engl. utility*), den ein Agent von der Durchführung eines Plans hat, kann in diesem Beispiel als Differenz zwischen dem Wert des erreichten Zustands und den dafür aufzubringenden Kosten angegeben werden. Würde nur Agent A_1 existieren, so könnte er seine Aufgabe erfüllen, indem er am Lagerhaus L_1 ein Transportmittel TM mietet und den Plan aus Abbildung 3.4(a) ausführt. Dieser Plan würde zu einem Nutzen von

$$Utility_{A_1} = 50 - (1 + 1 + 40 + 1 + 1) = 6$$

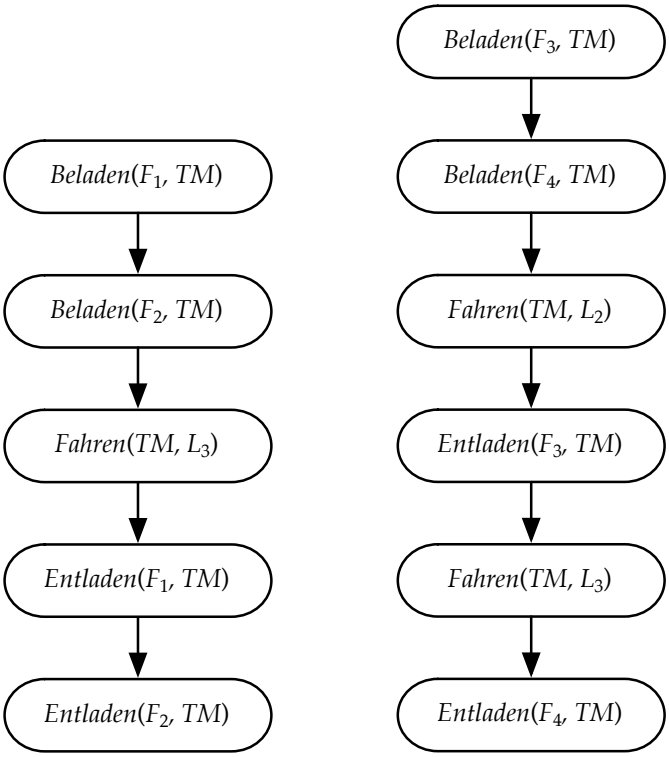

Abbildung 3.4: Ein-Agenten-Pläne: (a) Plan für Agent A_1 bei alleiniger Existenz; (b) Plan für Agent A_2 bei alleiniger Existenz

führen, da im Lagerhaus L_3 noch Platz ist, um genau zwei Frachtgüter aufzunehmen. Somit könnte A_1 seine Aufgabe erfüllen und dafür den Wert 50 annehmen. Auch Agent A_2 könnte, würde er alleine existieren, seine Aufgabe erfolgreich durchführen, indem er den Plan aus Abbildung 3.4(b) befolgt. Sein Nutzen wäre, genau wie für Agent A_1 auch, 6.

Betrachtet man das Beispiel unter Berücksichtigung der Koexistenz beider Agenten und ihrer Aufgaben, so ist festzustellen, dass ein Konflikt entsteht. Insgesamt wollen A_1 und A_2 drei Frachtgüter im Lagerhaus L_3 abladen. Da dieses jedoch nur eine Restkapazität von zwei Frachtgütern hat, kann einer der beiden Agenten seine Aufgabe nicht vollständig erfüllen. Trotzdem können sie von einer Kooperation profitieren. Dazu wird nur ein Transportmittel an L_1 gemietet und alle vier Frachtgüter aufgeladen. Das Anmieten erfolgt dabei mit der gleichen Wahrscheinlichkeit von Agent A_1 oder von Agent A_2. Das Transportmittel würde zu L_2 fahren und dort F_3 entladen. Ebenso würden F_1 oder F_4 (beide mit einer Wahrscheinlichkeit von $0,5$) an L_3 entladen werden. Danach fährt das

Transportmittel weiter zu L_3 und lädt dort die restlichen beiden Frachtgüter ab. Jeder der Agenten erreicht sein Ziel mit einer Wahrscheinlichkeit von $0,5$ und die Kosten für den gesamten Transport werden zu gleichen Teilen auf beide Agenten aufgeteilt. Der daraus resultierende Nutzen beider Agenten liegt bei

$$Utility_{A_1,A_2} = 0,5 \cdot (50 - 12) + 0,5 \cdot 50 - 0,5 \cdot 40 - 0,5 \cdot 8) = 20.$$

Dadurch kann eine Kooperation in einer wertorientierten Domäne auch dann sinnvoll sein, wenn Agenten kooperieren, obwohl nicht beide ihre Ziele erreichen können.

4 Tourenplanungsproblem

Das Tourenplanungsproblem (*engl. vehicle routing problem*), im Folgenden allgemein als Routingproblem bezeichnet, stellt eine Erweiterung des bekannten Problems des Handlungsreisenden um mehrere Reisende dar und steht für eine ganze Klasse von Problemen. Allen gemein ist die Aufgabenstellung, eine Menge von Routen für eine Flotte von an einem oder mehreren Depots stationierten Fahrzeugen für mehrere, geografisch verteilte Städte oder Kunden zu finden. Diese Stationen mit bekannten Anforderungen sollten zudem kostenminimierend beliefert werden, wobei der Kostenbegriff als problemspezifisch anzusehen ist. Eine gute Einführung in die Thematik der Tourenplanungsprobleme gibt [AD07].

In Kapitel 4.1 wird das Tourenplanungsproblem mathematisch beschrieben. Auf die verschiedenen Varianten der Routingproblems wird in Kapitel 4.2 eingegangen. Abschließend wird in Kapitel 4.3 die für diese Arbeit verwendete Lösungsmethode allgemein beschrieben.

4.1 Formulierung als mathematisches Problem

Um das Routingproblem mittels mathematischer Verfahren behandeln zu können, muss zuerst die Abbildung auf ein Modell erfolgen. In diesem Fall kann das Problem mittels eines bewerteten schlichten Graphen, wie in Abbildung 4.1 dargestellt, repräsentiert werden.

Ein bewerteter schlichter Graph G ist gegeben durch ein Tripel (V, E, f), wobei $V = \{v_0, v_1, \ldots, v_n\}$ eine Menge von Knoten, $E = \{e_{ij} | i \neq j \wedge i, j = 0, \ldots, n\}$ eine Menge von Kanten und $f\left(e_{ij}\right)$ die Bewertung oder Länge der Kante e_{ij} darstellt. Für die Länge einer Kante gilt, dass $f\left(e_{ij}\right) > 0$ für alle $e_{ij} \in E$ ist. Darüber hinaus dürfen keine Schlingen oder Mehrfachkanten beziehungsweise Mehrfachbögen vorkommen. Für jeden bewerteten schlichten Graphen kann zudem eine Entfernungs- oder Distanzmatrix

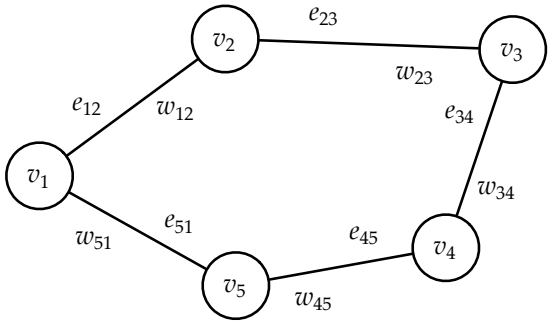

Abbildung 4.1: Repräsentation des Routingproblems als Graph

$\boldsymbol{D} = \left(d_{\mu\nu}\right)_{\mu,\nu=1,\ldots,n+1}$ aufgestellt werden. Für $\boldsymbol{D}$ gilt: $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ und $\boldsymbol{D} = \boldsymbol{D}^{\mathrm{T}}$. Eine Kante $e_{ij} = \left(v_i, v_j\right)$ stellt die Verbindung zweier Knoten dar und ist durch $d_{ij} = \delta\left(e_{ij}\right)$ gewichtet. In Bezug auf das Routingproblem stellen die Knoten anzufahrende Städte oder Kunden dar, während die Kanten die Wege zwischen den Stationen angeben und die Distanz d_{ij} die Weglänge beschreibt. Dabei muss die Weglänge nicht unbedingt der geografischen Entfernung entsprechen, sondern kann zum Beispiel auch als Reisezeit zwischen den Stationen angesehen werden. Durch die Forderung einer symmetrischen Distanzmatrix ist die Entfernung zweier Stationen nicht richtungsabhängig.

Ist eine Knotenmenge $V = \{v_0, v_1, \ldots, v_{m+1}\}$ gegeben, so werden v_0 und v_{m+1} als das Depot angesehen, von dem aus ein Lieferfahrzeug, welches die Stationen $V' = V\backslash\{v_0\}$ anzufahren hat, startet und zu dem es nach Auslieferung aller Frachtgüter auch wieder zurückkehrt. Diese Stationen können auch als Route R_i für das ihnen zugeordnete Fahrzeug i interpretiert werden, wobei $1 \leq i \leq s$ gilt und s somit die Fahrzeuganzahl beziehungsweise die Flottengröße angibt. Jedem Knoten $v_i \in V'$ sind zudem noch eine Größenangabe q_i sowie eine Servicezeit t_i zugeordnet. Dabei gibt q_i die Menge an aufzuladender oder abzuladender Ware in v_i an und t_i stellt die Zeit dar, welche für das Auf- oder Abladen benötigt wird. Die Kosten für eine Route $R_i = \{v_0, v_1, \ldots, v_{m+1}\}$ mit $v_i \in V$ sind gegeben durch $C\left(R_i\right) = \sum_{i=0}^{m} c_{i,i+1} + \sum_{i=1}^{m} t_i$.

Das Routingproblem hat somit die Aufgabenstellung, s Routen mit minimalen Kosten zu ermitteln, die alle in einem Depot starten und auch wieder enden. Dabei muss jeder Knoten in V' genau einmal von einem Fahrzeug besucht werden.

4.2 Varianten des Tourenplanungsproblems

Kapazitiertes Tourenplanungsproblem

Das kapazitierte Tourenplanungsproblem (*engl. capacitated vehicle routing problem*) unterscheidet sich vom allgemeinen Routingproblem in der Hinsicht, dass eine bestimmte Anzahl an Lieferfahrzeugen als feste Flottengröße gegeben ist, von denen alle eine einheitliche Ladekapazität besitzen. Die im Vorwege bekannten Kundenanforderungen für eine bestimmte Ware werden dabei von einem Depot aus bedient. Ziel ist es, wie beim ursprünglichen Routingproblem auch, dieses zu minimalen Kosten zu realisieren. Eine Lösung des Problems ist genau dann gültig, wenn die Gesamtmenge an Waren einer Route nicht die Kapazität des bedienenden Fahrzeugs überschreitet.

Die formale Formulierung des Problems lautet: Sei K die Kapazität des Lieferfahrzeugs. Die Menge der auszuliefernder Ware auf einer Route R_i mit m Stationen darf K nicht übersteigen: $\sum_{i=1}^{m} q_i \leq K$.

Tourenplanungsproblem mit Zeitfenstern

Das Tourenplanungsproblem mit Zeitfenstern (*engl. vehicle routing problem with time windows*) unterscheidet sich vom Ursprungsproblem durch die Zuordnung eines Zeitfensters $[b_v, e_v]$ zu jedem Kunden, innerhalb dessen er bedient werden muss. Dazu wird die Zielsetzung verfolgt, die Fahrzeugflottengröße sowie die Summe aller Fahrzeiten zu minimieren. Darüber hinaus soll auch die möglicherweise auftretende Wartezeit, welche entsteht wenn ein Fahrzeug vor Beginn eines Zeitfensters bei einem Kunden eintrifft, so gering wie möglich sein. Für eine mögliche Lösung des Problems gilt, dass sie ungültig wird, sobald ein Kunde nach Ablauf seines Zeitfensters beliefert wird. Entschärft man das Tourenplanungsproblem etwas, so kann eine Lösung auch bei einer Lieferung nach Ablauf des Zeitfensters als gültig angesehen werden, allerdings wird dadurch die Kostenfunktion durch eine Strafe erhöht.

Die formale Definition der Anforderungen lautet: Sei s_v der Beginn der Zustellung bei Kunde v. Damit eine Route $R_i = \{v_0, v_1, \ldots, v_{m+1}\}$ gültig bleibt, müssen $b_{v_i} \leq s_{v_i} \leq e_{v_i}$ sowie $s_{v_i} + t_{v_i} + c_{v_m,0} \leq e_{v_0}$ mit $i = 1, \ldots, m$ gelten. Solange $s_{v_i} < b_{v_i}$ gilt, entsteht dem Fahrzeug eine Wartezeit w_{v_i}. Die Kosten für eine Route R_i sind folglich gegeben durch: $C(R_i) = \sum_{i=0}^{m} c_{i,i+1} + \sum_{i=1}^{m} t_{v_i} + \sum_{i=0}^{m} w_{v_i}$.

Offenes Tourenplanungsproblem

Beim offenen Tourenplanungsproblem (*engl. open vehicle routing problem*) handelt es sich um eine Variante des Routingproblems, bei der die zu planenden Routen nicht im Ausgangsdepot enden müssen, sondern offen bleiben können.

Die formale Beschreibung lautet: Sei $R_i = \{v_0, v_1, \ldots, v_{m+1}\}$ eine Route, dann ist v_0 das Startdepot des Fahrzeugs i und $v_0 \neq v_{m+1}$ darf gelten.

4.3 Lösungsmethoden für das Tourenplanungsproblem

Das Routing, welches in dieser Arbeit verwendet wird, setzt sich aus den drei genannten Variationen des Problems zusammen und berücksichtigt die in ihnen genannten Bedingungen bezüglich der Kapazität, der Zeitfenster und der Offenheit der Route. Das damit verbundene Routingproblem ist NP-schwer (*engl. nondeterministic polynomial-time hard*). Die Größe des Suchraums, welcher alle möglichen Routen enthält, hängt überexponentiell von der Anzahl der Stationen ab, die angefahren werden können. Für solche Probleme gibt es drei Klassen von Lösungsmethoden; die exakten Methoden, die Heuristiken und die Metaheuristiken. Die exakten Methoden, wie beispielsweise das Branch-and-Bound (*dtsch. Verzweigung und Schranke*) basieren auf dem Vorgehen, alle möglichen Lösungen auszuprobieren und die kostenminimalste als Ergebnis zurück zu liefern. Diese Verfahren sind allerdings sehr unpraktikabel, da der Suchraum mit der Anzahl an Stationen sehr schnell anwächst. So ist die Anzahl an möglichen Routen bei der einfachsten, der symmetrischen Variante des Problems des Handlungsreisenden gegeben durch $\frac{(n-1)!}{2}$. Bei einer Menge von 10 Stationen wären somit 181 440 Routen möglich, bei 14 Stationen jedoch schon 43 589 145 600 Routen.

Da man auf der Suche nach Verfahren ist, die in überschaubarer Zeit brauchbare Ergebnisse liefern, wurden in der Vergangenheit vornehmlich Heuristiken und Metaheuristiken erforscht und zur Lösung solcher Probleme verwendet. Heuristiken sind Näherungsverfahren, welche einen begrenzten Teil des Suchraums betrachten und versuchen, in diesem eine möglichst gute Lösung zu finden. Eine Qualitätsaussage bezüglich der gefundenen Lösung kann bei diesen Verfahren jedoch in der Regel nicht getroffen werden. Zudem sind sie meistens nur auf ein bestimmtes Optimierungsproblem anwendbar. Zu ihnen gehört unter anderem der Clarke-Wright Algorithmus [CW64], welcher bereits 1964 veröffentlicht wurde.

Im Gegensatz zu den Heuristiken definieren Metaheuristiken eine abstrakte Folge von Schritten, die auf beliebige Problemstellungen angewandt werden können. Jeder dieser Schritte muss jedoch problemspezifisch implementiert werden, so dass hierbei häufig wiederum Heuristiken zum Einsatz kommen. Das Finden einer optimalen Lösung kann auch bei der Verwendung von Metaheuristiken nicht garantiert werden. Ebenso ist eine Qualitätsaussage hinsichtlichen der gefundenen Lösung nicht möglich, da die Optimallösung in der Regel nicht bekannt ist. Zu den Metaheuristiken gehören unter anderem das simulierte Abkühlen, die Tabu-Suche, der Bergsteigeralgorithmus sowie die evolutionären und genetischen Algorithmen.

Die in dieser Arbeit verwendete Lösungsmethode basiert auf einem genetischen Algorithmus, dessen allgemeiner Ablauf im Folgenden näher beschrieben wird.

4.3.1 Genetische Algorithmen

Das für die Realisierung verwendete Framework JOpt.SDK der Firma DNA evolutions benutzt zur Lösungsfindung einen genetischen Algorithmus. Solche Algorithmen sind Problemlösungssysteme welche Schlüsselelemente evolutionärer Prozesse verwenden. Ihre Entwicklung geht zurück bis in die 1970 er Jahre, als Holland in [Hol75] erste Ansätze präsentierte. Grundsätzlich geht es bei genetischen Algorithmen um den Wettbewerb zwischen sich im Laufe der Zeit entwickelnden Lösungskandidaten in einer Population. Dabei evaluiert eine Fitnessfunktion jede Lösung und entscheidet somit darüber, ob sie an der Generierung von Folgelösungen teilnimmt. Die neue Population wird dann durch Operationen wie der Rekombination und der Mutation erzeugt und nimmt sich damit ein Vorbild an der Natur.

Generell bieten sich genetische Algorithmen zum Lösen vieler komplexer Aufgaben an. So gibt es keine grundsätzlichen Einschränkungen in Bezug auf die zu optimierende Funktion, wie zum Beispiel eine vorausgesetzte Stetigkeit. Besonders gut lassen sich solche Algorithmen bei besonders großen Suchräumen einsetzen, in denen eine Optimumsuche über alle Lösungen nicht mehr möglich ist. Dabei wird immer versucht ein globales Maximum zu finden, wobei dessen tatsächliches Aufspüren nicht garantiert werden kann.

Begrifflich orientieren sich die genetischen Algorithmen ebenfalls an ihrem natürlichen Vorbild. So besteht eine Population aus einer festen Anzahl an Lösungen für ein Problem, wobei jede einzelne von ihnen Individuum genannt wird. Jedes Individuum steht dabei

für einen Punkt im Suchraum und enthält alle Parameter für eine potentielle Lösung in kodierter Form. Diese Form hat meistens die Gestalt einer aus Genen bestehenden Zeichenkette fixer Länge.

Aufbau eines Genetischen Algorithmus

Der generelle Aufbau eines genetischen Algorithmus wird anhand des Pseudocodes verdeutlicht.

Algorithmus 1 Pseudocode eines genetischen Algorithmus

```
t ← 0;
initialize (P(t));
evaluate (P(t));
while (not termination-condition) do
  t ← t + 1;
  P_s(t) ← select (P(t − 1));
  P_r(t) ← recombine (P_s(t));
  P(t) ← mutate (P_r(t));
  evaluate (P(t));
end while
```

Dabei wird zuerst eine Ausgangspopulation erzeugt und anschließend bewertet. Solange die Abbruchbedingung nicht erfüllt ist, wird mittels Selektion eine neue Population $P_s(t) = \{S_1^t, \ldots, S_N^t\}$ gebildet. Die so erzeugten Individuen S_i^t werden dann durch Rekombination und Mutation zu einer neuen Population an Lösungen, welche als Eltern für die nächste Generation fungieren. Dieser Vorgang wird so lange wiederholt bis eine gute Lösung gefunden wurde.

Die einzelnen Schritte werden im Folgenden genauer beschrieben.

- **Initialisierung** In diesem Schritt wird eine zufällige, jedoch zulässige Ausgangspopulation erzeugt.
- **Bewertung** Die Bewertung einer generierten Population erfolgt mittels dreier Funktionen. Dazu gehören die schon erwähnte Fitnessfunktion, eine Bewertungsfunktion und eine Zielfunktion. Das Ergebnis der Fitnessfunktion ist dabei die Wahrscheinlichkeit, mit der ein Individuum an der Generierung weiterer Nachkommen beteiligt ist. Die Fitnessfunktion für ein Individuum $f(S_i)$ hat folgende

Eigenschaften: $f(S_i)$ lässt sich für alle möglichen Individuen berechnen, $f(S_i) \geq 0$ und aus $f(S_i) > f(S_j)$ folgt, dass Individuum S_i besser ist als Individuum S_j.

Die Bewertungsfunktion misst die Qualität einer gefundenen Lösung und beschreibt damit, wie nahe ein Individuum dem gesuchten, optimalen Wert kommt. Allgemein ist die Bewertungsfunktion $g(S)$ gegeben durch $g(S) = h(S) + p(S)$, wobei $p(S)$ eine Straffunktion darstellt und $h(S)$ die Zielfunktion ist. Die Straffunktion hat für alle gültigen Individuen den Wert null.

Die Zielfunktion beschreibt das Optimierungsziel des Problems.

- **Selektion** In der Selektionsphase werden die Individuen ausgewählt, die zur Generierung nachfolgender Populationen geeignet sind und damit ihre Elternindividuen darstellen. In den meisten Fällen erfolgt diese Selektion stochastisch, wobei bessere Individuen mit einer größeren Wahrscheinlichkeit ausgewählt werden als schlechtere Individuen.
- **Rekombination** Im Rekombinationsschritt erfolgt die Informationsgewinnung. Hier werden neue Individuen aus den selektierten Eltern erzeugt. Dazu werden Informationen zwischen zwei potentiellen Lösungen ausgetauscht. Dieser Prozess wird entweder auf alle Individuen oder den größten Teil davon angewendet. Die Paare, unter denen die Rekombination stattfindet, werden wieder zufällig ausgewählt.
- **Mutation** Nach der Rekombination stellt der Schritt der Mutation den zweitwichtigsten Bestandteil zur Informations(wieder-)gewinnung dar. Sie wird jedoch nur sehr selten ausgeführt, da sonst der Suchvorgang nach einer optimalen Lösung komplett zufällig wäre. Es existieren verschiedene Varianten der Mutation. So wird zum Beispiel bei der Flipmutation nach dem Zufallsprinzip ein bestimmtes Gen verändert, wohingegen bei einer Swapmutation der Inhalt zweier zufällig ausgewählten Gene ausgetauscht wird.
- **Abbruchkriterium** Durch das Abbruchkriterium wird angegeben, wann der Algorithmus terminieren soll. Häufige Kriterien sind dabei das Überschreiten einer maximalen Iterationszahl oder die Stagnation des Suchprozesses. Die absolute Terminierung stellt dabei die einfachste Variante dar und stoppt die Ausführung sobald eine bestimmte Anzahl an Generationen generiert wurde. Im Gegensatz dazu lässt die relative Terminierung den Algorithmus stoppen wenn eine vorgegebene Anzahl an Generationen keine Verbesserung des Ergebnisses mehr erzielt hat.

5 Modell des Fallbeispiels

Dieses Kapitel beschreibt das der Arbeit zugrunde liegenden Modell. Dabei wird ausführlich auf die Agenten eingegangen, die für das Transportszenario benötigt werden. Neben den Funktionsweisen und den zu erfüllenden Aufgaben werden außerdem die zwischen den Agenten stattfindenden Interaktionen dargestellt. Darüber hinaus wird auf die in der Realität vorzufindenden Verhältnisse Bezug genommen. In diesem Zusammenhang werden besonders die Vereinfachungen des Modells diskutiert.

5.1 Modell des Frachtgutagenten

Das Frachtgut soll autonom von einer Produktionsstätte zu einem Zielort transportiert werden. Um dieses Ziel zu erreichen, werden beim Erstellen einer neuen Ware zwei Agenten initiiert, zum einen der Transportagent und zum anderen der Frachtgutagent. Ersterer ist für die Verhandlungen mit dem Transportmittel verantwortlich, während Letzterer die Überwachung der Ware übernimmt. In Abschnitt 5.1.1 wird zunächst auf den Transportagenten eingegangen. Dieser hat die Suche von geeigneten Transportmitteln sowie die darauf folgende Vergabe des Transportauftrags zur Aufgabe. Zum Abschluss wird in Abschnitt 5.1.2 die Funktionsweise der Überwachungsagenten beschrieben.

5.1.1 Transportagent

Suche eines geeigneten Transportmittels

Für die Suche nach einem geeigneten Transportmittel sind in diesem Modell zwei verschiedene Möglichkeiten vorgesehen, die als komplementär anzusehen sind. Die erste Möglichkeit besteht im Zurückgreifen auf bereits angeeignetes Wissen. Dazu müsste dem Frachtgut initial eine Menge von möglichen Transportmitteln zur Verfügung stehen,

die in der Vergangenheit zum Beispiel Kühlgut transportiert haben und mit denen der Produzent, auf den in diesem Modell nicht genauer eingegangen wird, indirekt über das Frachtgut zusammengearbeitet hat. Diese Möglichkeit ist, besonders beim ersten Start des gesamten Systems, nicht immer gegeben. Aus diesem Grund gibt es eine zweite Möglichkeit, ein Transportmittel zu finden. Sie besteht darin, in einem Verzeichnis, ähnlich den Gelben Seiten, nach Transportmitteln zu suchen. Das Verzeichnis hat dabei den sehr einfachen, in Tabelle 5.1 dargestellten Aufbau. Zum einen ist der Transportmittelname beziehungsweise ein Identifikator aufgeführt, der es dem Frachtgut ermöglicht, mit dem Transportmittel Kontakt aufzunehmen. Zum anderen ist der dem Transportmittel zugehörige Typ aufgelistet. Aufgrund dieser Information wird das Frachtgut in die Lage versetzt, geeignete Transportmittel herauszufiltern. So macht es, ein hinreichend großes System vorausgesetzt, in Hinblick auf die später zu führenden Verhandlungen allerdings keinen Sinn, alle möglichen Transportmittel herauszusuchen. Die Suche sollte sich auf eine festgelegte Menge beschränken.

Transportmittelname	**Transportmitteltyp**
Transportmittel 1	Kühltransporter
Transportmittel 2	Kühltransporter
Transportmittel 3	Stückguttransporter
⋮	⋮

Tabelle 5.1: Aufbau des Transportmittelverzeichnisses

Ist die Suche erfolgreich verlaufen und dem Frachtgut stehen geeignete, mögliche Transportmittel zur Verfügung, kann die Verhandlung über die Konditionen des Transports beginnen.

Vergabe eines Transportauftrags

In dieser Arbeit soll die Vergabe des Transportauftrags auf der Grundlage einer Auktion erfolgen. Dazu werden im Folgenden die zur Zeit gängigsten Auktionsformen vorgestellt:

- **Englische Auktion** Einer der am meisten verwendeten Auktionstypen ist die Englische Auktion (*engl. first-price open-cry auction*). Sie ist dadurch charakterisiert,

dass die Gebote öffentlich sind und innerhalb eines vorgegebenen Zeitfensters abgegeben werden müssen. Der Bieter, welcher am Ende der Auktion das höchste Gebot abgegeben hat, gewinnt die Auktion und erhält das Auktionsgut.

- **Holländische Auktion** Der Name dieser Auktionsform (*engl. dutch auction*) stammt von den in Holland üblichen Blumenauktionen. Dieser Typ zeichnet sich durch ein kontinuierliches Reduzieren des Auktionspreises, ausgehend von einem relativ hohen Ausgangspreises aus. Dieser wird durch den Auktionator so lange gesenkt, bis einer der Teilnehmer den ausgerufenen Preis akzeptiert. Der Auktionator ist so in der Lage, die Schnelligkeit, mit der das Auktionsgut versteigert wird, vorzugeben.
- **Verdeckte Auktion** Eine verdeckte Auktion (*engl. first-price sealed-bid auction*) zeichnet sich durch die geheime Abgabe der Gebote aus. Typischerweise wird bei diesen Auktionen genau ein Gebot abgegeben. Nach Ablauf der vorgegebenen Zeit werden die Gebote ausgewertet und der Höchstbietende gewinnt die Auktion. Durch diesen Mechanismus haben die Teilnehmer keine Möglichkeit, auf die Gebote von anderen Teilnehmern zu reagieren. Zudem ist die Anzahl der Teilnehmer unbekannt.
- **Vickery Auktion** Dieser Typ von Auktion wurde nach ihrem Erfinder William S. Vickery benannt ist auch als *second-price sealed-bid auction* bekannt. Er ist dadurch gekennzeichnet, dass der bestbietende Teilnehmer die zu versteigernde Ware zum Preis des zweitbesten Gebots bekommt. Pro Teilnehmer wird nur ein einziges Gebot akzeptiert, welches darüber hinaus geheim abgegeben wird.
- **Reverse Auktion** Diese Auktionsform ist speziell auf die Versteigerung von Dienstleistungen und Gütern ausgelegt. Sie zeichnet sich dadurch aus, dass der Käufer der Dienstleistung oder des Auktionsguts der Initiator der Auktion ist und die Teilnehmer die Verkäufer sind. Innerhalb des Zeitfensters der Auktion geben die Verkäufer Gebote ab, die sich jeweils unterbieten. Diese Gebote stellen die Konditionen dar, zu denen sie den ausgeschriebenen Auftrag ausführen würden. Gewinner der Auktion ist derjenige, der das niedrigste Gebot abgegeben hat.

Um reale Auktionen theoretisch beschreiben zu können bedarf es vereinfachter Modelle, die beispielsweise irrationale Entscheidungen von menschlichen Bietern ausblenden. Dennoch können diese Modelle als Entscheidungshilfe sowohl für Menschen als auch für Softwareagenten dienen. In dieser Arbeit wird das Modell der Präferenzunsicherheit (*engl. independent private value model*) zum Vergleich der oben aufgeführten Auktionstypen verwendet. In ihm wird davon ausgegangen, dass jeder Bieter vollständige Informationen über das zu ersteigernde Auktionsgut besitzt. Diese Informationen dienen dem Bieter zum Ableiten einer Bewertung (Wertschätzung), welche gleich dem tatsächlichen Wert des Auktionsguts für ihn ist. Diese Bewertung erfolgt unabhängig von den Wertschätzungen

des Auktionsguts durch andere Bieter. Der Bieter würde seine Bewertung auch dann nicht ändern, wenn er die Wertschätzungen für das Auktionsgut seiner Mitbieter kennen würde. Sollte dieses jedoch der Fall sein, so könnten sie seine Bietstrategie beeinflussen.

Zur weiteren Untersuchung der Auktionstypen dient das Erlös-Äquivalenz-Theorem, welches ebenfalls von William S. Vickery aufgestellt wurde. Es trifft eine Aussage bezüglich der Gewinnerwartung für den Verkäufer. So bringt jeder Auktionsmechanismus, in dem der Höchstbietende das zu versteigernde Auktionsgut erhält, dem Verkäufer durchschnittlich den gleichen Gewinn, sofern von dem oben beschriebenen Modell der Präferenzunsicherheit ausgegangen wird. Zusätzlich müssen sich die Bieter risikoneutral verhalten und ihre Gebote unabhängig voneinander aus einem bestimmten Bereich der Wahrscheinlichkeitsverteilung abgeben. Ebenfalls in die Betrachtungen fließt der Begriff einer dominanten Strategie mit ein. Eine Strategie wird genau dann als dominant bezeichnet, wenn sie dem Bieter den für ihn höchstmöglichen Ertrag garantiert. Der Ertrag ist als Differenz zwischen der Wertschätzung für das Auktionsgut durch einen Bieter und dem Preis, welchen er beim Gewinn der Auktion bezahlen muss, definiert. Existiert für einen Auktionstyp eine dominante Strategie hat das den Vorteil, dass die Entwickler von Agenten keine Anstrengungen mehr unternehmen müssen, andere Strategien für ihre Agenten zu entwickeln, da als Ergebnis bereits der höchstmögliche Ertrag gewährleistet wird.

Aufgrund dieser Annahmen kann gezeigt werden, dass die verdeckte Auktion zu der Holländischen Auktion äquivalent ist. Beide Auktionstypen weisen jedoch keine dominante Strategie auf, da der Gewinner einer dieser Auktionen immer den Betrag zahlen muss der seiner Bewertung entspricht. Somit ergibt sich für den Bieter in beiden Fällen ein Ertrag von 0. Unter Verwendung des Präferenzunsicherheitsmodells sind auch Englische Auktionen und Vickery Auktionen äquivalent. So wird jeweils das Gebot des am zweithöchsten Bietenden bezahlt, beziehungsweise in einer Englischen Auktion ein Gebotsschritt höher. Für beide Auktionstypen existiert außerdem eine dominante Strategie. Für eine Vickery Auktion besteht sie darin, ein Gebot in Höhe der wahren Wertschätzung für das Auktionsgut abzugeben. Bei Englischen Auktionen liegt ein neues Gebot jeweils geringfügig über dem letzten Gebot. Wird die eigene Wertschätzung für die Ware erreicht, werden keine Gebote mehr abgeben.

Zieht man jedoch komplexere Auktionsmodelle für die Betrachtung heran, so existiert die Äquivalenz zwischen einer Englischen Auktion und einer Vickery Auktion nicht mehr. Durch die Offenheit der Englischen Auktion lassen sich Informationen über das Bietverhalten anderer Teilnehmer sammeln, wodurch die Bietstrategie beeinflusst werden

kann.

Für das Modell dieser Arbeit wird eine Vickery Auktion unter der Sichtweise einer reversen Auktion welche für die Versteigerung von Dienstleistungen geeignet ist verwendet. Der Gewinner ist somit nicht der Höchstbietende, sondern derjenige, der das niedrigste Gebot abgibt. Ausgezahlt bekommt er für die Ausführung der Dienstleistung jedoch einen Betrag in Höhe des zweitniedrigsten Gebots.

Damit dieser Auktionsmechanismus für das Transportszenario funktionieren kann, müssen noch einige Bedingungen erfüllt sein. Zum einen betrifft es die Ausschreibung beziehungsweise die Artikelbeschreibung des Auftrags. In dieser müssen alle notwendigen Angaben, die für einen erfolgreichen Transport erfüllt sein müssen, angegeben werden. Hierfür ist es notwendig, dass dem Frachtgut und damit auch dem Transportagenten initial faktisches Wissen vorliegt, welches den Bietern zugänglich gemacht wird. Dazu gehören in erster Linie der momentane Standort des Frachtguts und der Ort, an dem das Frachtgut abgeliefert werden soll. Des Weiteren sollen Zeitfenster berücksichtigt werden. So kann das Frachtgut nur zu bestimmten Zeiten abgeholt und ausgeliefert werden. Zeitfenster werden zum Beispiel dann benötigt, wenn der Produzent einer Ware feste Öffnungszeiten hat oder die Ware just-in-time bei einem Kunden abgeliefert werden muss, damit der Produktions- oder Weiterverarbeitungszyklus nicht gefährdet ist. Darüber hinaus wird noch der maximale Preis, den das Frachtgut bereit ist für den Transport zu bezahlen, ausgeschrieben. Die letzte Angabe die gemacht werden muss ist die Anzahl an Transporteinheiten. Diese sind von Ladung zu Ladung unterschiedlich. So wären mögliche Angaben 10000 l Milch oder 6 t tiefgefrorener Fisch. Eine Angabe des Typs der Fracht ist an dieser Stelle nicht mehr notwendig, da durch die Suche des Transportagenten nur in Frage kommende Transportmittel ausgewählt wurden.

Unter Zuhilfenahme dieser gegebenen Informationen haben die Transportmittel nun die Möglichkeit, detaillierte Gebote abzugeben. Diese umfassen dabei nicht nur den Gebotspreis, sondern zugleich noch alle anderen Angaben die für den späteren Transport notwendig sind. So werden die Orte, an denen das Frachtgut abgeholt beziehungsweise ausgeliefert werden soll, noch einmal bestätigt und ihnen zudem eine feste Zeit zugeordnet, die innerhalb der ausgeschriebenen Zeitfenster liegen muss. Diese Informationen werden von dem Transportagenten gesammelt und können nach Beendigung der Auktion abgerufen werden. Wie bereits beschrieben ist das Ende der Auktion durch Ablaufen einer bestimmten Frist, welche durch den Transportagenten vorgegeben wird, erreicht. Es werden somit auch nur die Gebote berücksichtigt, die vor dem Erreichen dieser Frist an den Transportagenten übermittelt wurden.

Nach Ablauf der Auktion werden die Gebote der Transportmittel seitens des Transportagenten überprüft. Zwar sollten nur gültige Gebote abgegeben werden, aber geschieht dieses aus irgendwelchen Gründen nicht, so wird der Fehler an dieser Stelle aufgedeckt. Als erstes wird dabei das niedrigste Gebot überprüft. Ergibt die Validierung, dass es sich um ein gültiges Gebot handelt, werden alle anderen Gebote verworfen. Sollte jedoch ein Fehler aufgetreten sein, so muss der Transportagent entweder die Liste der Gebote sukzessive durchgehen und das erste gültige Gebot annehmen oder den Transportauftrag neu ausschreiben. Dieses würde dann unter der Annahme geschehen, dass nicht noch einmal die selben Transportmittel an der Auktion teilnehmen. Dieser Ansatz ist dahingehend als legitim anzusehen, dass nur eine bestimmte Anzahl an Transportmitteln aus dem Verzeichnis herausgesucht wird und dieser Prozess zufallsgesteuert ist.

Ist es zu einer erfolgreichen Vergabe des Transportauftrags gekommen, bestehen die beiden letzten Schritte darin, dem generellen Überwachungsagenten die Daten des Gebots mitzuteilen und den Geldtransfer an das Transportmittel einzuleiten. Die Überwachung wird dann, wie unter 5.1.2 beschrieben, eingeleitet.

Solch eine Auktion stellt einen sehr simplen Verhandlungsmechanismus dar und kann deswegen auf die von Zlotkin und Rosenschein gegebenen Attribute für Standards aus Abschnitt 3.1 hin überprüft werden.

Der erste zu überprüfende Punkt ist die Einfachheit des Mechanismus. Aufgrund der Tatsache, dass die Agenten nur ein Gebot abgeben in dem alle wichtigen Daten enthalten sind, ist dieses Kriterium als erfüllt anzusehen. Ebenfalls ersichtlich ist die Effizienz dieses Verfahrens. Der Transportagent kennt alle eingegangenen Gebote und wählt dann das für ihn beste aus. Dadurch wird auf jeden Fall ein pareto-optimales Ergebnis erzielt. Die Symmetrieeigenschaft ist auch erfüllt, da alle bietenden Agenten gleichbehandelt werden. Sie können ihr Gebot in gleicher Weise abgeben und auch bei der Auswertung gibt es keine Vorzüge hinsichtlich des Gewinners der Auktion. Einen zentralen Entscheidungsträger gibt es in dem Sinne auch nicht, da jedes Frachtgut über seinen Transportagenten selber die Auktion leitet und die Kommunikation regelt. Durch diese Tatsache ist auch die Verteiltheit gegeben. Der wohl wichtigste Punkt ist die Stabilität. Dazu muss untersucht werden, ob die Agenten einen Anreiz haben, von ihrer Strategie abzuweichen und ob sie sich in einer gewünschten Weise verhalten. Gewünscht ist in diesem Fall, dass sie ein reelles Minimalgebot abgeben, also weder zu wenig noch zu viel bieten.

Durch das in Abbildung 5.1 dargestellte, vereinfachte Beispiel soll die Existenz einer dominanten Strategie und damit auch die Stabilität des Mechanismus gezeigt werden.

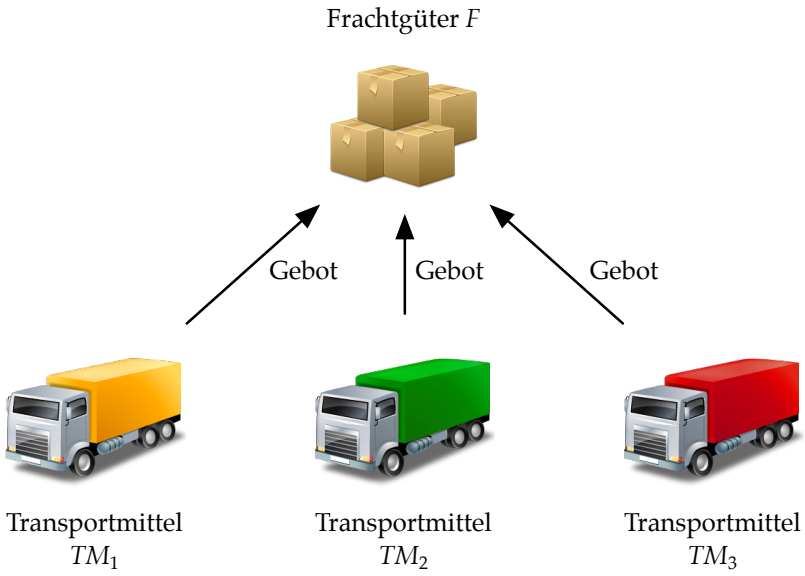

Abbildung 5.1: Vickery Auktion

Gegeben sind drei Transportmittelagenten TM_1, TM_2 und TM_3 sowie die Frachtgüter F, dessen Transportauftrag zur Versteigerung steht. Dazu werden die in Tabelle 5.2 gezeigten Fälle I - III untersucht.

Fall	Gebot			Resultat	Gewinn/Verlust
	TM_1	TM_2	TM_3		
I	200	400	300	TM_1 gewinnt die Auktion	+100 €
II	200	400	150	TM_3 gewinnt die Auktion	−150 €
III	350	400	300	TM_1 verliert die Auktion	−300 €

Tabelle 5.2: Ergebnisse verschiedener Bietstrategien bei einer reversen Vickery Auktion

Im ersten Fall geben die drei Transportmittelagenten ihre Gebote nach der eigenen Wertschätzung für die Frachtgüter ab. Ihr Gebot entspricht somit den ihnen real entstehenden Kosten. Das Resultat wäre der Zuschlag für das Transportmittel TM_1, da dieses das geringste Gebot in Höhe von 200 € abgegeben hat. Der Gewinn den es dabei erzielen würde beliefe sich auf 100 €. Dieser Betrag ergibt sich aus der Differenz des eigenen Gebots und dem des Bieters, der das zweitniedrigste Gebot abgegeben hat. In Fall II ändert TM_3 seine Strategie und versucht durch die Abgabe von niedrigeren Geboten, mehr Auktionen

zu gewinnen. Ein Gebot in Höhe von 150 € führt zwar zum Gewinn der Auktion, hat aber für TM_3 den Nachteil, dass die ihm durch den Transport entstehenden Kosten nicht gedeckt werden. Durch diesen Umstand macht TM_3 einen Verlust von 150 €. Im letzten dargestellten Fall ändert das Transportmittel TM_1 seine Strategie mit dem Ziel, seinen Erlös durch die Abgabe eines höheren Gebots zu steigern. Das hier angegebene Gebot in Höhe von 350 € hat jedoch für TM_1 die Konsequenz, dass es nicht mehr das niedrigste Gebot abgegeben hat und dadurch die Auktion nicht gewinnt. Somit entsteht ihm ein Verlust von 300 €. Durch die Tatsache, dass keine der beiden zuletzt genannten Fälle einen höheren Nutzen für das Transportmittel hätte, wird es bei der Strategie bleiben, immer ein reelles Minimalgebot abzugeben. Dadurch ist auch die Stabilität des Verfahrens gezeigt und alle von Zlotkin und Rosenschein geforderten Attribute sind erfüllt. Der Vickery-Mechanismus ist damit eine für die Vergabe von Transportaufträgen adäquate Lösung.

Das hier beschriebene Beispiel stellt eine Vereinfachung der in der Wirklichkeit auftretenden Verhältnisse dar, zeigt jedoch in Grundzügen, welche Situationen entstehen können und welche Strategie für die Bieter am meisten Erfolg verspricht. In der Realität hingegen müssen bestimmte Risikofaktoren und eventuell unvollständige Informationen in einer Strategie berücksichtigt werden. Zu diesen Faktoren können unter anderem die Entwicklung der Kraftstoffpreise, welche sich auf die Transportkosten auswirken, gehören. Ein Beispiel für unvollständige Informationen in Bezug auf das Auktionsgut ist der Wert, den es für Mitbieter hat. So kann sich die Wertschätzung des Bieters in Abhängigkeit von den Werten der anderen Bieter ändern, wenn eventuelle Wiederverkaufsmöglichkeiten existieren.

5.1.2 Überwachungsagent

Den zweiten Bestandteil des Frachtguts soll ein Überwachungsagent darstellen. Ihm wird die Aufgabe zuteil, die mit dem Transportmittel verhandelten Konditionen zu kontrollieren und gleichzeitig ladungsspezifische Parameter zu beobachten. Aufgrund der Vielzahl an unterschiedlichen Frachtgütern und der daraus resultierenden, sehr großen Anzahl an unterschiedlichen, zu überwachenden Parametern, sieht das Modell an dieser Stelle eine Disposition in zwei verschiedene Arten von Überwachungsagenten vor. Zum einen gibt es einen generalisierten Überwachungsagenten, welcher jene Funktionen bereitstellt, die auf alle Frachtgüter in gleicher Weise angewendet werden können. Im Gegensatz dazu gibt es die spezialisierten Überwachungsagenten, die auf bestimmte Arten von Frachtgütern

exakt zugeschnitten sind und frachtgutspezifische Kontrollen durchführen können. Diese beiden Agententypen werden nun vorgestellt und sind in Abbildung 5.2 exemplarisch dargestellt.

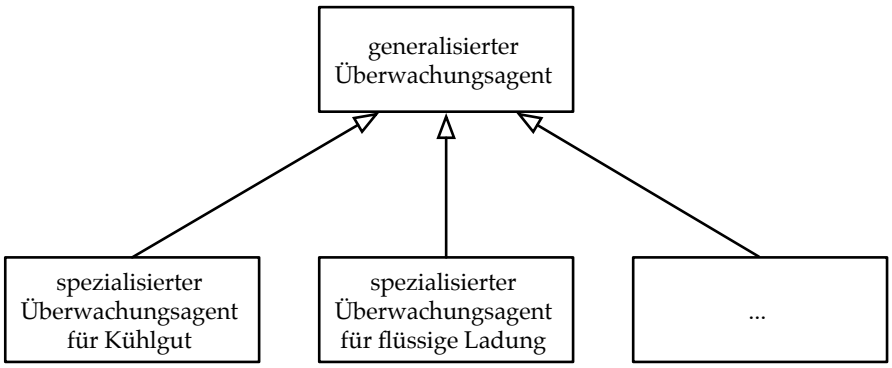

Abbildung 5.2: Spezialisierte Überwachungsagenten

Generalisierter Überwachungsagent

In dem hier zu behandelnden Szenario gibt es für jedes Frachtgut eine Aufgabe, die in gleicher Weise durchgeführt werden muss, die Einhaltung der mit dem Transportmittel verhandelten Konditionen. Da der ausgehandelte Preis, den das Frachtgut für den Transport an seinen Bestimmungsort zu bezahlen hat nach der Verhandlung feststeht und nicht mehr geändert werden darf, bleiben als zu beobachtende Punkte die fristgerechte Abholung und Auslieferung des Frachtguts übrig.
Sollte das Transportmittel, aus welchen Gründen auch immer, nicht rechtzeitig zum Abholen des Frachtguts erscheinen, so muss der Überwachungsagent entsprechende Maßnahmen einleiten. In einem ersten Schritt würde er Kontakt mit dem Transportmittel aufnehmen um herauszufinden, ob dieses verspätet eintrifft oder ein Transport inzwischen, zum Beispiel aufgrund eines Unfalls, unmöglich geworden ist. Trifft der erste Fall zu, so muss das Transportmittel natürlich weiterhin gewährleisten, dass eine rechtzeitige Auslieferung möglich ist. Sollte jedoch kein Transport mehr möglich sein, so müsste der Überwachungsagent dem Frachtgutagenten mitteilen, dass eine neue Ausschreibung des Transports notwendig ist.

Die zweite wichtige Aufgabe eines generalisierten Überwachungsagenten besteht darin, mit einem dem Frachtgut zugeordneten, spezialisierten Überwachungsagenten zusammen zu arbeiten. Hauptaugenmerk dieser Zusammenarbeit ist das Übermitteln von

frachtgutspezifischen Fehlermeldungen. Jene werden von diesem Überwachungsagenten verarbeitet und bei Bedarf an den, für die zu ergreifende Maßnahme zuständigen, Agenten weitergeleitet. So kann dies, stellt man sich ein Kühlgut vor, zum Beispiel eine Warnmeldung über eine zu hohe Temperatur des Frachtguts sein. Eine mögliche Reaktion wäre dann die Kommunikation mit dem Transportagenten, welcher mit dem Transportmittel über eine Lösung des Problems verhandelt. In diesem Fall das Anfahren eines nahegelegenen Kühlhauses in dem die Fracht zwischengelagert werden kann, bis das Problem behoben ist oder die Verhandlung mit einem anderen Transportmittel, welches die Ladung übernehmen kann.

Spezialisierter Überwachungsagent

Der spezialisierte Überwachungsagent ist in der Lage, frachtgutcharakteristische Parameter zu überwachen. Dazu muss eine Kommunikation mit den beobachtenden Sensoren bestehen, beziehungsweise müssen die gelieferten Messdaten verarbeitet werden können. Um das Beispiel des Kühlguts aufzugreifen, muss es dem Agenten möglich sein, Temperaturdaten auszuwerten und bei einem auftretenden Messwert außerhalb der Toleranzgrenzen zu reagieren. Dieses geschieht, wie bereits im obigen Abschnitt beschrieben, indem er eine Fehlermeldung an den generalisierten Überwachungsagenten sendet, der diese dann auf geeignete Weise weiterverarbeitet.

5.2 Modell des Transportmittelagenten

Den Teil mit der höchsten Anforderung stellen in dieser Arbeit die Transportmittel dar. Ihre Arbeit lässt sich in zwei Kategorien einteilen. Zum einen müssen sie für den Transport der Frachtgüter sorgen und zum anderen sollen sie in der Lage sein, untereinander zu kooperieren. In diesem Abschnitt steht das Ersteigern eines Frachtguts sowie die Kooperation zwischen den einzelnen Transportmitteln im Vordergrund.

5.2.1 Ersteigerung von Frachtgütern

Die Ersteigerung eines Frachtguts sei hier noch einmal aus der Sicht des Transportmittels dargestellt. Damit ein Transportmittel überhaupt an einer Auktion teilnehmen kann, muss

es sich zuerst in das Transportmittelverzeichnis, welches in 5.1 gezeigt ist, eintragen. Dadurch hat es die Möglichkeit, von einem Frachtgut eine Ausschreibung für die Vergabe des Transportauftrags zu bekommen. Geht solch eine Ausschreibung ein, hat das Transportmittel zwei Möglichkeiten darauf zu reagieren. Die erste Möglichkeit besteht darin, dass ein Mitbieten zur Zeit nicht möglich, beziehungsweise vom Transportmittel zu diesem Zeitpunkt nicht gewollt ist. Welche Restriktionen hier gelten, wird im Abschnitt 5.2.3 erläutert, da es sich um die gleichen wie bei einer Kooperation unter Transportmitteln handelt. Möchte oder kann das Transportmittel nicht teilnehmen wird eine Mitteilung an das Frachtgut gesendet, in der es über diesen Zustand benachrichtigt wird. Sollte ein Mitbieten jedoch möglich und gewünscht sein, evaluiert das Transportmittel die Kosten für den Transport um ein Gebot abgeben zu können. Dazu werden die bereits geladenen Frachtgüter zusammen mit dem zur Auktion stehenden Frachtgut und der aktuellen Position an den Routingalgorithmus übergeben. Dieser versucht nun, in Abhängigkeit vom momentanen Standort, eine möglichst optimale Wegstrecke zu ermitteln. Ist eine solche gefunden, werden unter anderem Gesamtlänge l_{route} [km] sowie die dafür benötigte Zeit t_{route} [h] zurückgeliefert. Mit Hilfe dieser Daten können die Kosten für die neue Route wie folgt berechnet werden:

$$c_{new} = \left(\left(l_{route} \cdot G_{consumption} \cdot G_{price}\right) + \left(t_{route} \cdot W\right)\right).$$

$G_{consumption}$, G_{price} und W sind Konstanten und stehen für den Kraftstoffverbrauch $\left[\frac{\text{l}}{\text{km}}\right]$, den Kraftstoffpreis $\left[\frac{€}{\text{l}}\right]$ und einen betriebsstundenabhängigen Faktor $\left[\frac{€}{\text{h}}\right]$. Da nur der Preis als Gebot abgegeben werden soll, der dem Transportmittel auch tatsächlich für das Hinzufügen des Frachtguts entsteht, muss eine eventuell schon vorhandene Ladung in die Kostenfunktion mit einbezogen werden und der Gebotspreis ergibt sich zu

$$c_{bid} = c_{new} - c_{actual}.$$

Die Kosten c_{actual} sind dabei genau diejenigen, die das Transportmittel momentan, also vor Beendigung der Auktion, hat. Ist das Transportmittel zur Zeit unbeladen, so sind die Kosten für $c_{actual} = 0$ und $c_{bid} = c_{new}$. Gewinnt das Transportmittel mit diesem Gebot die Auktion, so erhält es, wie in 5.1.1 beschrieben, den nächsthöheren Gebotspreis ausgezahlt. Dieser stellt somit den Gewinn beziehungsweise den Nutzen für das Transportmittel dar. Durch das Bieten des reellen Minimalpreises ist außerdem gewährleistet, dass das Transportmittel die Auktion auf keinen Fall mit einem negativen Nutzen beenden kann. Somit ist ein genereller Anreiz gegeben an Auktionen teilzunehmen um den Nutzen, hier den monetären Gewinn, zu steigern.

In der Realität findet man in solchen Szenarien häufig Mediatoren oder Broker, die als Vermittler fungieren, vor. Aufgrund seiner Vereinfachung sind solche Instanzen nicht

vorgesehen, sollen aber an dieser Stelle erwähnt werden. Auch sie können als Agenten modelliert werden und würden eine alternative Möglichkeit darstellen, wie Transportagenten passende Transportmittelagenten finden. Der Brokeragent würde über eine umfassende Wissensbasis bezüglich der im System befindlichen Transportmittelagenten und deren momentanen Zuständen besitzen. Diese versetzt ihn in die Lage, Transportagenten besonders geeignete Transportmittel zu vermitteln. Diese Eignung kann sich auf verschiedene Aspekte beziehen und wäre problemspezifisch realisierbar. So könnten unter anderem den Transport betreffende Prioritäten berücksichtigt werden, indem beispielsweise eine besonders schnelle Auslieferung oder ein niedriger Preis mit den Transportmitteln ausgehandelt wird. Der Brokeragent würde für diese Vermittlungen entlohnt werden und könnte seinen Nutzen steigern, indem er zum Beispiel nachweislich besonders gute Angebote und Konditionen aushandelt. Durch das Hinzufügen eines solchen Agenten würde sich an der Funktionsweise des Systems jedoch nicht viel ändern, da diese zusätzliche Vermittlungseinheit eine Ergänzung darstellt. Der zu erwartende Vorteil durch die Integration solcher Agenten wäre die höhere Güte und die größere Flexibilität der Vermittlungen zwischen Transportagenten und Transportmittelagenten. Der Kommunikationsaufwand für die reine Vermittlung würde nicht ansteigen, jedoch wäre bei einer Implementierung darauf zu achten, wie der Brokeragent beispielsweise eine aktuelle Wissensbasis beibehält. Zudem müssten komplexere Auswahlmechanismen und Verhandlungsprotokolle eingeführt werden, um die oben beschriebenen Vorteile nutzen zu können.

5.2.2 Kooperation zwischen Transportmitteln

Eine weitere Möglichkeit für Transportmittel an Frachtgüter und die damit verbundenen Transportaufträge zu gelangen, besteht in der Kooperation mit anderen Transportmitteln. Die Motivation, an solchen Kooperationen teilzunehmen ist auch in diesem Fall gegeben. Die Argumentation beruht hierbei, ähnlich wie in 5.2.1, auf dem Bestreben der Agenten, ihren eigenen Nutzen zu erhöhen, indem sie den Gewinn steigern. Da das Resultat einer erfolgreichen Kooperation in einer aufgabenorientierten Domäne immer nur eine Verbesserung des Nutzens sein kann, ist ein entsprechender Anreiz für die Transportmittel gegeben.

Eine Kooperation basiert grundsätzlich auf Verhandlungen, deren Gegenstand in diesem Szenario die den Transportmitteln zugewiesenen Frachtgüter darstellen. Jede Verhandlung besteht dabei aus vier Komponenten: der Verhandlungsmenge, einem Protokoll, einer Strategie und Regeln. Diese Bestandteile werden im Folgenden näher erläutert.

Vorbetrachtungen

Voraussetzung jeglicher Verhandlung ist das Zusammentreffen zweier Transportmittel. Dieses wird in [ZR96b] definiert als:

Definition 3 *Ein Zusammentreffen innerhalb einer TOD* $< \mathcal{T}, \mathcal{A}, c >$ *ist eine geordnete Liste* $(T_1, T_2, \ldots, T_n)$*, so dass für alle* $k \in \{1, \ldots, n\}$ *gilt,* T_k *ist eine finite Menge von Aufgaben aus* $\mathcal{T}$ *die* A_k *abarbeiten muss.* T_k *kann auch als Ziel von* A_k *bezeichnet werden.*

Die Aufgaben T_k die jedes Transportmittel TM_k zu erfüllen hat sind die Transportaufträge aller Frachtgüter, welche es geladen hat. Ziel eines Zusammentreffens zweier Transportmittel ist es, Frachtgüter untereinander auszutauschen um den jeweils eigenen Nutzen zu erhöhen. Diese Umverteilung von Aufgaben wird beschrieben durch:

Definition 4 *Gegeben sei ein Zusammentreffen* (T_1, T_2) *innerhalb einer TOD* $< \mathcal{T}, \{A_1, A_2\}, c >$ *bestehend aus zwei Agenten. Ein Abkommen* δ *ist eine Umverteilung von Aufgaben unter Agenten. Es handelt sich um eine geordnete Liste* (D_1, D_2)*, so dass* $D_1, D_2 \subseteq \mathcal{T}$ *und* $D_1 \cup D_2 = T_1 \cup T_2$ *gelten. Jeder Agent* A_k *verpflichtet sich, alle Aufgaben aus* D_k *auszuführen.*

Um den Begriff des Nutzens einer solchen Umverteilung definieren zu können ist zunächst der Kostenbegriff zu klären. Für ihn gilt:

Definition 5 *Gegeben sei ein Abkommen* δ*. Die Kosten für solch ein Abkommen für jeden Agenten* A_k *sind definiert als* $Cost_k(D_1, D_2) = c(D_k)$ *und werden künftig als* $cost_k(\delta)$ *bezeichnet.*

Die Kosten sind also genau jene, die einem Transportmittel entstehen, wenn es Waren von anderen Transportmitteln übernimmt oder an andere Transportmittel abgibt. Der Nutzen eines Abkommens ist somit gegeben als Differenz der momentanen Kosten eines Transportmittels $c(T_k)$ und den durch das Abkommen entstehenden Kosten $cost_k(\delta)$.

Definition 6 *Gegeben sei ein Zusammentreffen* (T_1, T_2) *innerhalb einer TOD* $< \mathcal{T}, \{A_1, A_2\}, c >$ *bestehend aus zwei Agenten.*

1. *Der Nutzen eines Abkommens* δ *für jeden Agenten* A_k *ist definiert als* $Utility_k(\delta) \equiv c(T_k) - cost_k(\delta)$.

2. *Ein Abkommen $\Theta \equiv (T_1, T_2)$ heißt Konflikt-Abkommen.*

Erzielen die beiden Transportmittel keine Einigung, so liegt ein Konflikt vor und es kommt keine Umverteilung von Frachtgütern zustande. In solch einem Fall gilt für den Nutzen $Utility_k(\Theta) = c(T_k) - c(T_k) = 0$.

Aussagen über die Güte von Abkommen können durch eine Ordnungsfunktion getroffen werden. Die Grundlage für eine Ordnung von Abkommen bildet die Dominanz.

Definition 7 *Gegeben sind zwei Vektoren $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ und $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$. Gelten $\forall k\,(\alpha_k \geq \beta_k)$ und $\exists l\,(\alpha_l > \beta_l)$ wird β von α dominiert und man schreibt $\alpha \succ \beta$. Eine schwache Dominanz $\alpha \succeq \beta$ von α über β ist gegeben, wenn nur $\forall k\,(\alpha_k \geq \beta_k)$ gilt.*

Übertragen auf Abkommen heißt das:

Definition 8 *Gegeben sind die beiden Abkommen δ und δ'.*

- *Eine Dominanz von δ über δ' wird geschrieben als $\delta \succ \delta'$ und ist gegeben wenn $(Utility_1(\delta), Utility_2(\delta)) \succ (Utility_1(\delta'), Utility_2(\delta'))$ gilt.*
- *Eine schwache Dominanz $\delta \succeq \delta'$ ist durch $(Utility_1(\delta), Utility_2(\delta)) \succeq (Utility_1(\delta'), Utility_2(\delta'))$ gegeben.*
- *Die beiden Abkommen sind genau dann gleich $(\delta \equiv \delta')$, wenn $\forall k\,(Utility_k(\delta) = Utility_k(\delta'))$.*

Verhandlungskonstellationen

Abkommen können nicht zwischen zwei beliebigen Transportmitteln geschlossen werden. So müssen sie hinsichtlich der Anzahl der ihnen zugeordneten Transportaufträge bestimmte Voraussetzungen erfüllen. Unter welchen Konstellationen Verhandlungen durchgeführt und zum Abschluss gebracht werden dürfen, ist in Tabelle 5.3 aufgezeigt.

Unter der Voraussetzung, dass beide Transportmittel über keine Frachtgüter verfügen, kann generell keine Kooperation stattfinden; alle anderen Konstellationen sind jedoch gültig. Warum auch zwei Transportmittel, von denen eines der beiden genau ein Frachtgut geladen hat und gleichzeitig das zweite Transportmittel keine Fracht hat, kooperieren

		$\lvert T_1 \rvert$		
		0	1	> 1
$\lvert T_2 \rvert$	0	–	×	×
	1	×	×	×
	> 1	×	×	×

Tabelle 5.3: Kooperationsmöglichkeiten zwischen Transportmitteln

können, wird durch Abbildung 5.3 verdeutlicht. Gegeben sind die beiden Transportmittel TM_1 und TM_2 sowie die Stationen S_1,S_2 und das Frachtgut $F_{S_2S_3}$, welches von S_2 nach S_3 transportiert werden muss. Angenommen wird, dass TM_1 das Frachtgut $F_{S_2S_3}$ ersteigert, weil TM_2 nicht an der Auktion teilgenommen hat. Dieses ist eine Voraussetzung, da die Kostenfunktion für die Abgabe eines Gebots bei allen Transportmitteln gleich ist und TM_2 somit ein niedrigeres Gebot als TM_1 abgegeben hätte.

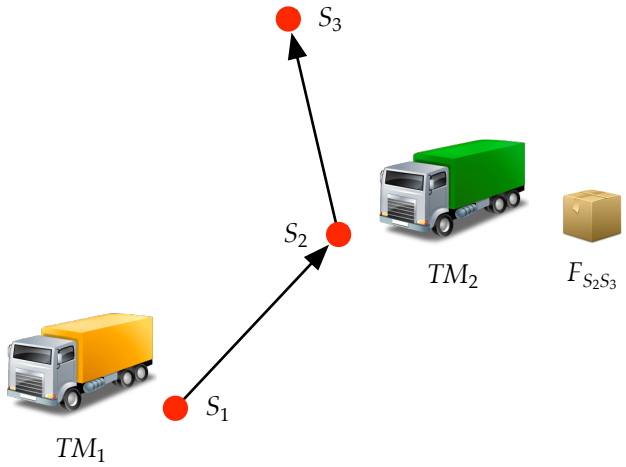

Abbildung 5.3: Verhandlungsbeispiel bei einseitiger Frachtgutverteilung

Damit ergibt sich oben beschriebene Situation, dass nur einer der beiden möglichen Kooperationspartner, hier TM_1, einen Frachtgutauftrag hat.
Eine Verhandlung zwischen diesen beiden Transportmitteln ist sinnvoll, da die Abgabe des Auftrags an TM_2 den Nutzen für TM_1 erhöhen würde. Das Frachtgut wurde unter der Prämisse erworben, dass die Strecke von S_1 über S_2 nach S_3 gefahren werden muss. Diese Tatsache wurde daher im Gebot berücksichtigt. Auf der anderen Seite steigt auch der Nutzen für TM_2, da das Warten auf die Ersteigerung eines Auftrags den Nutzen 0 hat

und die Auslieferung in jedem Fall einen positiven Nutzen bringen würde. So könnten Agenten selbst unter dieser Konstellation erfolgreich Verhandlungen führen.

Verhandlungsmenge

Da die Menge aller möglichen Abkommen zwischen zwei Transportmitteln sehr groß werden kann, muss das Ziel gegeben sein, diese sinnvoll zu reduzieren. Dieses wird dadurch erreicht, dass in die endgültige Verhandlungsmenge, die den beiden Transportmitteln zur Verfügung steht, nur faire Abkommen eingehen. Dabei unterscheidet man zwei Mengen von Abkommen, welche diese Voraussetzung erfüllen. Zum einen sind es die pareto-optimalen Abkommen und zum anderen die individual rationalen Abkommen. Erstere sind in dem Sinne fair, als dass sie nur solche Abkommen beinhalten, die nicht zu Gunsten eines Transportmittels verbessert werden können, wenn das andere Transportmittel dabei benachteiligt wird. Ein Abkommen heißt per Definition genau dann pareto-optimal, wenn kein anderes Abkommen δ' existiert, so dass $\delta' \succ \delta$ gilt. Ebenso fair ist die Menge der individual rationalen Abkommen. Sie gewährleistet, dass ein Transportmittel durch die Teilnahme an einem Abkommen keinen negativen Nutzen haben kann. Formal ausgedrückt sind genau die Abkommen individual rational für die $\delta \succeq \Theta$ gilt. Die Menge dieser beiden Abkommen bildet somit die Verhandlungsmenge.

Anhand des in Abbildung 5.4 dargestellten Beispiels soll diese Reduzierung der Verhandlungsmenge verdeutlicht werden. Die Ausgangssituation ist gegeben durch die beiden Transportmittel TM_1 und TM_2 sowie den ihn zugeordneten Frachtgüter und den Stationen D, S_1 und S_2. Dabei ist D das Depot, von welchem aus die beiden Transportmittel ihre Auslieferungstour starten und S_1 sowie S_2 stellen die anzufahrenden Stationen dar. TM_1 hat die beiden Frachtgüter F_{DS_1} und F_{DS_2} geladen, welche nach S_1 respektive S_2 geliefert werden müssen. TM_2 hingegen hat nur ein Frachtgut geladen, welches nach S_1 geliefert werden muss. Die Kostenfunktion sei hier die Entfernung zwischen den einzelnen Stationen. So betragen die Kosten für das Fahren der Strecken $D \rightarrow S_1$ und $D \rightarrow S_2$ jeweils 1. Daraus ergeben sich die aktuellen Kosten für TM_1 zu 3 und die für TM_2 zu 1, da TM_1 an beiden Stationen Frachtgut abliefern muss, wohingegen TM_1 nur Frachtgut nach S_1 liefern muss und beide Transportmittel von D aus starten. Alle möglichen Abkommen, die TM_1 und TM_2 schließen können, sind in der erste Spalte von Tabelle 5.4 aufgeführt. T_1 sind dabei die Aufgaben, welche dem ersten Transportmittel zugeordnet sind und T_2 diejenigen, die das zweite Transportmittel zum Ziel hat. Der aus diesen Abkommen resultierende Nutzen für das jeweilige Transportmittel ist in der zweiten Spalte gezeigt. Dieser wurde, wie in Definition 6 beschrieben, als Differenz der aktuellen Kosten und den

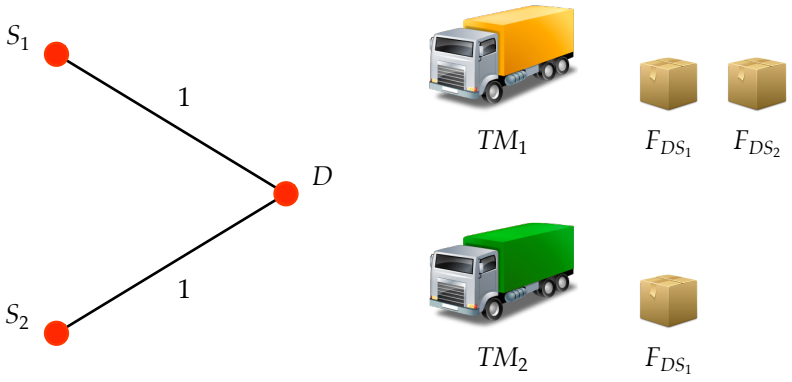

Abbildung 5.4: Beispiel Verhandlungsmenge

Kosten für das Abkommen berechnet. Die Verhandlungsmenge ist die Schnittmenge der individual rationalen Abkommen und der pareto-optimalen Abkommen. Die Transportmittel können nun über die Abkommen 1, 2 und 4 verhandeln. Die dazu verwendeten Verhandlungsprotokolle werden im folgenden Abschnitt erläutert.

Verhandlungsprotokolle und Strategien

Steht die Verhandlungsmenge fest, muss ein geeignetes Verhandlungsprotokoll ausgewählt werden, welches den Ablauf der Verhandlung vorgibt. Alle Protokolle sollten dabei zur Klasse der Produkt-Maximierenden-Mechanismen gehören, das heißt dass am Ende einer Verhandlung ein Abkommen ausgewählt wird, welches das Produkt ihrer Nutzen maximiert. Der bekannteste Ansatz aus der Spieletheorie ist die Nash-Verhandlungslösung. Sie ist definiert als

$$\arg \max_{\substack{Utility_1, \\ Utility_2}} (Utility_1 - \theta_1)\,(Utility_2 - \theta_2) \tag{5.1}$$

und wurde in den 1950 er Jahren von Nash in [Nas50] und [Nas53] vorgestellt.
$\Theta = (\theta_1, \theta_2)$ stellt dabei das Konflikt-Abkommen dar, wodurch in diesem Modell beiden Agenten die Kosten 0 hätten und sich die Verhandlungslösung als maximales Nash-Produkt

$$\arg \max_{\substack{Utility_1, \\ Utility_2}} (Utility_1 \cdot Utility_2) \tag{5.2}$$

ergibt. Nash konnte zeigen, dass die Nash-Verhandlungslösung unter anderem individual rational als auch pareto-optimal ist. Im Folgenden werden zwei Protokolle erläutert, welche sich die Ergebnisse von Nash zu Nutze machen.

Nr.	Abkommen $(\{T_1\}, \{T_2\})$	Nutzen $(Utility_1, Utility_2)$	Individual rationale Abkommen	Pareto-optimale Abkommen	Verhandlungs-menge
1	$(\{S_1\}, \{S_2\})$	$(0, 2)$	$\times$	$\times$	$\times$
2	$(\{S_2\}, \{S_1\})$	$(0, 2)$	$\times$	$\times$	$\times$
3	$(\{S_1, S_2\}, \{\})$	$(-2, 3)$	$-$	$\times$	$-$
4	$(\{\}, \{S_1, S_2\})$	$(1, 0)$	$\times$	$\times$	$\times$
5	$(\{S_1\}, \{S_1, S_2\})$	$(0, 0)$	$\times$	$-$	$-$
6	$(\{S_2\}, \{S_1, S_2\})$	$(0, 0)$	$\times$	$-$	$-$
7	$(\{S_1, S_2\}, \{S_1\})$	$(-2, 2)$	$-$	$-$	$-$
8	$(\{S_1, S_2\}, \{S_2\})$	$(-2, 2)$	$-$	$-$	$-$

Tabelle 5.4: Reduzierung der Verhandlungsmenge

Monotones Zugeständnis-Protokoll Das monotone Zugeständnis-Protokoll ist ein rundenbasiertes Protokoll. In der ersten Runde geben beide Agenten (hier die beiden kooperierenden Transportmittel) zeitgleich einen Vorschlag ab, der ein Abkommen aus der Verhandlungsmenge beinhaltet. Tritt der Fall ein, dass ein Transportmittel den Vorschlag des anderen Transportmittels als mindestens genau so gut wie seinen eigenen ansieht, so ist eine Einigung erreicht. Sollte der Fall nicht eingetreten sein, so beginnt die nächste Runde der Verhandlung, in der die beiden Agenten wiederum zeitgleiche Gebote abgeben. Dabei ist zu beachten, dass nun keine Gebote mehr abgegeben werden dürfen, die schlechter sind als das Gebot der vorherigen Runde. Macht kein Agent im Laufe der Verhandlung ein Zugeständnis, so ist das Ende erreicht und die Verhandlung endet mit einem Konflikt-Abkommen, in dem keine Umverteilung der zu verhandelnden Frachtgüter zustande kommt.

Eine mögliche Verhandlungsstrategie für zwei Agenten, die das monotone Zugeständnis-Protokoll benutzen, ist die Zeuthen Strategie. Sie basiert auf folgenden drei Thesen:

- Das erste Angebot jedes Agenten sollte das für ihn am meisten bevorzugte Abkommen sein.
- In jeder Runde der Verhandlung sollte der Agent Zugeständnisse machen, der am wenigsten Interesse an einem Konflikt hat.
- Sollte ein Agent ein Zugeständnis machen muss dieses genau so groß sein, dass es die Risikobalance umkehrt.

Zusätzlich definiert diese Strategie ein Maß für die Risikobereitschaft der teilnehmenden Agenten, einen Konflikt als Verhandlungsergebnis zu akzeptieren. Angenommen wird zudem, dass ein Agent eine höhere Bereitschaft aufweist einen Konflikt zu riskieren, wenn die Differenz der Nutzen zwischen dem aktuellen Angebot und einem Konflikt klein ist. Das Risiko ist definiert als:

$$Risk_i^t = \begin{cases} 1 & \text{wenn } Utility_i\left(\delta_i^t\right) = 0 \\ \frac{Utility_i\left(\delta_i^t\right) - Utility_i\left(\delta_j^t\right)}{Utility_i\left(\delta_i^t\right)} & \text{sonst.} \end{cases}$$

Ein-Schritt-Protokoll Eine Alternative zum monotonen Zugeständnis-Protokoll stellt das Ein-Schritt-Protokoll dar. Es sieht vor, dass beide Agenten nur einen Vorschlag abgeben, wobei derjenige akzeptiert wird, welcher das höhere Produkt der Nutzen beider Agenten hat. Sollten beide Vorschläge gleichwertig sein, entscheidet das Los darüber,

welches Abkommen angenommen wird. Die Strategie der Transportmittel besteht darin, aus der Menge der Abkommen mit maximalem Produkt der Nutzen jenes auszuwählen, welches am besten für sie selbst ist.

5.2.3 Teilnahme an Auktionen und Verhandlungen

Theoretisch können die Transportmittel zu jeder Zeit an Kooperationen teilnehmen, die sie entweder selbst initiiert haben oder die von anderen Transportmitteln ausgeschrieben wurden. Hinzu kommen noch die Auktionen der Frachtgüter, an denen sie ebenfalls zu jedem Zeitpunkt partizipieren könnten. In der Praxis muss es jedoch eine Reglementierung für die Teilnahme an Auktionen und Verhandlungen geben. Ein Grund hierfür ist das Kommunikationsaufkommen. Dieses würde viel zu hoch werden, wenn alle Agenten ständig versuchen, mit anderen Agenten Kooperationen einzugehen. Zudem wäre bei zu vielen Verhandlungen die Effizienz in Frage gestellt. Nicht jede Verhandlung wird darin enden, dass sich eine Erhöhung des Nutzens für die verhandelnden Agenten einstellt. Die Wahrscheinlichkeit, das der Nutzen durch weitere Verhandlungen noch gesteigert werden kann, wird nach jeder erfolgreichen Auktion geringer. Diese Tatsache hängt zum einen mit der restlichen Fahrstrecke zusammen. Je mehr Verhandlungen geführt werden, desto kürzer wird die noch zu fahrende Strecke. Bei einer positiven Einigung muss jedes Mal eine Fahrt zu einem Depot, welches nicht immer direkt auf der noch verbleibenden Route liegt, mit einbezogen werden. Zum anderen kann der Nutzen nicht beliebig gesteigert werden, wodurch er sich durch jede Verhandlung dem maximal zu erreichenden Nutzen nähert. Aus diesen Gründen ist in diesem Modell vorgesehen, dass Transportmittelagenten genau dann an Ausschreibungen von Transportagenten teilnehmen dürfen, wenn sie gerade keine Transportaufträge für Frachtgüter haben. Kooperationen dürfen, sollte ein Transportauftrag zugewiesen sein, genau einmal als Initiator gestartet werden. Eine Teilnahme an Kooperationen, die von anderen Transportmittelagenten ausgeschrieben wurde, ist hingegen immer möglich. Sind gerade keine Aufträge zugewiesen, kann somit ohne Beschränkungen an Auktionen und Kooperationen teilgenommen werden.

Eine zusätzliche Vorgabe ist in diesem Modell gegeben. Sollte bei einem Transportmittel eine Kapazitätsauslastung von 75% oder mehr vorliegen, nimmt es ebenfalls nicht an Auktionen oder Verhandlungen teil. In dieser Situation wird ebenfalls von einer geringen Erhöhung des Nutzens durch Auktionen und Verhandlungen ausgegangen. Ob dieser Wert realistisch ist, müsste jedoch in der Praxis überprüft und gegebenenfalls angepasst werden. Er dient in diesem Modell als Beispielwert.

5.2.4 Subklassifikation aufgabenorientierter Domänen

In [RZ94] und [ZR96b] wird neben der Aufteilung in verschiedene Domänen eine weitere Klassifikation für aufgabenorientierte Domänen, basierend auf den Eigenschaften der Kostenfunktion, angegeben. So wird zwischen subadditiven, konkaven und modularen Attributen, welche einer Domäne zugeschrieben werden können, unterschieden. Eine subadditive Domäne ist demnach genau dann gegeben, wenn durch die Zusammenführung von zwei Taskmengen die Kosten im Vergleich zur separaten Ausführung der beiden Taskmengen gesenkt, jedoch niemals erhöht werden können. Formal ausgedrückt bedeutet dies:

Definition 9 *Eine aufgabenorientierte Domäne* $< \mathcal{T}, \mathcal{A}, c >$ *heißt subadditiv, wenn für alle finiten Taskmengen* $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{T}$ *gilt, dass* $c(\mathcal{X} \cup \mathcal{Y}) \leq c(\mathcal{X}) + c(\mathcal{Y})$.

Konkavität bedeutet, dass die Kosten, welche eine beliebige Taskmenge $\mathcal{Z}$ der Taskmenge $\mathcal{Y}$ hinzufügt, nicht größer sein kann als die Kosten, die $\mathcal{Z}$ der Taskmenge $\mathcal{Y}$ hinzufügt. Die Definition ist gegeben durch:

Definition 10 *Eine aufgabenorientierte Domäne* $< \mathcal{T}, \mathcal{A}, c >$ *heißt konkav, wenn für alle finiten Taskmengen* $\mathcal{X} \subseteq \mathcal{Y}, \mathcal{Z} \subseteq \mathcal{T}$ *gilt, dass* $c(\mathcal{Y} \cup \mathcal{Z}) - c(\mathcal{Y}) \leq c(\mathcal{X} \cup \mathcal{Z}) - c(\mathcal{X})$.

Das letzte Attribut ist die Modularität. Soll sie gelten, so sind die Kosten einer Kombination aus zwei Taskmengen genau die Summe aus den jeweils alleinigen Kosten abzüglich der Kosten der Schnittmenge.

Definition 11 *Eine aufgabenorientierte Domäne* $< \mathcal{T}, \mathcal{A}, c >$ *heißt modular, wenn für alle finiten Taskmengen* $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{T}$ *gilt, dass* $c(\mathcal{X} \cup \mathcal{Y}) = c(\mathcal{X}) + c(\mathcal{Y}) - c(\mathcal{X} \cap \mathcal{Y})$.

Für die drei Attribute gilt außerdem, dass alle modularen Domänen auch konkav sind und alle konkaven Domänen zudem subadditiv sind.

Um zu überprüfen, ob das in dieser Arbeit betrachtete Fallbeispiel subadditiv ist, wird zunächst ein vereinfachtes Beispiel, welches in Abbildung 5.5 gezeigt ist, betrachtet. Gegeben sind ein Transportmittel TM und zwei Frachtgüter F_{DS_1} und F_{DS_2}, welche durch TM vom Depot D an die Stationen S_1 und S_2 ausgeliefert werden sollen. Das Streckennetz entspricht hierbei genau dem des Beispiels aus Abschnitt 5.2.2 über die Verhandlungsmenge.

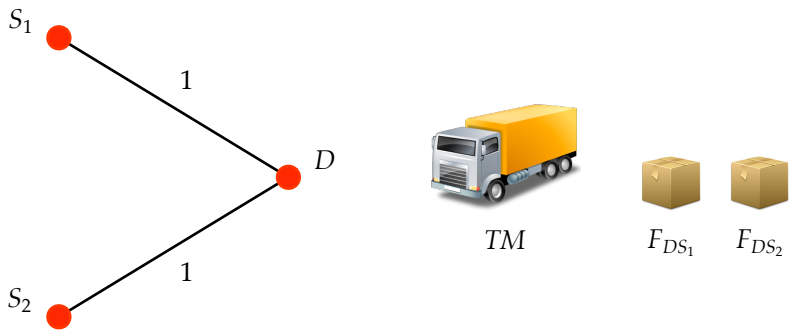

Abbildung 5.5: Beispiel einer nicht subadditiven Domäne

Beinhalte $\mathcal{X}$ in diesem Fall das Frachtgut F_{DS_1} und $\mathcal{Y}$ das Frachtgut F_{DS_2}. Die Kosten für die separate Ausführung der Aufgaben sind dadurch gegeben als

$$c(\mathcal{X}) + c(\mathcal{Y}) = 2.$$

Die Ausführung der Vereinigungsmenge beider Aufgaben ergibt die Kosten

$$c(\mathcal{X} \cup \mathcal{Y}) = 3,$$

womit ist Bedingung für eine subadditive Domäne

$$c(\mathcal{X} \cup \mathcal{Y}) \leq c(\mathcal{X}) + c(\mathcal{Y})$$
$$3 \leq 2$$

nicht gegeben ist.

Da dieses sehr einfache Beispiel einer Transportdomäne nicht subadditiv ist, ist leicht einsehbar, dass auch das in dieser Arbeit betrachtete Fallbeispiel, welches zusätzlich noch kapazitative Beschränkungen der Transportmittel sowie Zeitfenster für die anzufahrenden Stationen beinhaltet, nicht subadditiv sein kann.
Durch diese Tatsache sind auch eine Konkavität sowie eine Modularität der untersuchten Domäne ausgeschlossen.

5.2.5 Unsicheres Wissen bezüglich der Verkehrssituation

Mit in diese Arbeit soll unsicheres Wissen der Transportmittel über die Verkehrssituation einfließen. Dazu werden zwei Punkte untersucht, welche die Verkehrssituation beeinflussen; zum einen wie ein zu erwartendes Verkehrsaufkommen mit in die Routenberechnung eingebunden werden kann und zum anderen wie Transportmittel sich über die Verfügbarkeit von Routen austauschen können.

Verkehrsaufkommen

Die einfachste Abschätzung für die Entfernung zweier Orte ist die Verwendung des euklidischen Abstandes. Für die Berechnung der Distanz werden bei diesem Ansatz nur die Koordinaten der beiden Orte benötigt. Sie können für ein Modell in Form von geographischen oder transformierten Koordinaten vorliegen. Diese Abschätzung ist in den meisten Fällen aber sehr ungenau, da es kaum geradlinige Verbindungen zweier Orte gibt. So beträgt der euklidische Abstand zwischen Hamburg und München ca. 600 km, wohingegen die von einem Routenplaner ausgerechnete, kürzeste Route 721 km lang ist. Sollte trotzdem die Verwendung eines einfachen Modells gewünscht sein, so sollte bei großen Abständen zwischen den Orten an Stelle des euklidischen Abstands die Orthodrome verwendet werden, welche den Abstand auf einer Kugeloberfläche bestimmt und so für die Routenplanung von Flugzeugen besser geeignet ist. Liegen genaue Daten über das Streckennetz vor, wie sie zum Beispiel bei Routenplanern verwendet werden, lassen sich Entfernungen sehr genau abschätzen. Um Aussagen über das Verkehrsaufkommen in das Modell mit einbeziehen und zwischen der kürzesten und der schnellsten Route unterscheiden zu können, muss anstelle der Distanz die Fahrzeit in die Berechnung mit eingehen. Für die Zeit t_{ij}, welche benötigt wird um von einem Ort i an einen Ort j zu gelangen gilt

$$t_{ij} = \left(\frac{s_{ij}}{\overline{v}_{ij}} \right). \tag{5.3}$$

Die Entfernung ist dabei gegeben durch s_{ij} [km], $\overline{v}_{ij}$ $\left[\frac{\text{km}}{\text{h}}\right]$ symbolisiert die Durchschnittsgeschwindigkeit zwischen i und j unter optimalen Verhältnissen. Da diese in der Realität oft nicht zu erreichen ist, besteht die Möglichkeit, statistische Daten über das Verkehrsaufkommen zu ermitteln und die tatsächlich erreichte Durchschnittsgeschwindigkeit zeitabhängig in Formel 5.3 mit einfließen zu lassen. Dadurch ergibt sich die Fahrzeit zu

$$t_{ij} = \left(\frac{s_{ij}}{\overline{v}^t_{ij}} \right). \tag{5.4}$$

Da das Verkehrsaufkommen auf einer Route stark variieren kann, sollte die Fahrzeit teilstreckenweise berechnet werden. Die Kostenfunktion für eine komplette Route R ist dann gegeben durch

$$c(R) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\frac{s_{ij}}{\overline{v}^t_{ij}} \right) \kappa_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} \kappa_{ij} \tag{5.5}$$

mit

$$\kappa_{ij} = \begin{cases} 1 & \text{wenn die Teilstrecke } k_{ij} \in R \\ 0 & \text{sonst.} \end{cases}$$

Verfügbarkeit von Routen

Ebenso wie das Verkehrsaufkommen ist auch die Verfügbarkeit von Teilstrecken bei der Routenplanung zu berücksichtigen. Dazu müssen die Transportmittel in der Lage sein, ihr Wissen an andere Transportmittel weiterzugeben, um sie über etwaige Sperrungen von Straßen oder Teilstrecken zu informieren. Den Designern von Transportmittelagenten bieten sich zwei Möglichkeiten, solch ein System zu realisieren. Entweder erfolgt der Rückgriff auf bereits vorhandene Infrastrukturen, wie sie aktuelle Navigationsgeräte benutzen, oder eine eigene Infrastruktur muss aufgebaut werden. Da ein möglichst offenes Transport-Agentensystem zu bevorzugen ist, ergeben sich aus der Verwendung von bereits vorhandenen Infrastrukturen einige Nachteile. So wären alle Transportmittel verpflichtet, Geräte zu verwenden, die an die zu benutzende Struktur gebunden sind; beispielsweise kämen nur Geräte eines Herstellers in Frage. Möchte man trotzdem solch ein System benutzen, ist die Anwendbarkeit im Zusammenhang mit den Transportmitteln im Vorfeld zu untersuchen. Auch die Kosten für die Anschaffung der Geräte kann als weiterer Nachteil aufgeführt werden.

Da die Möglichkeit der Kommunikation in jedem Multiagentensystem gegeben sein muss, wäre auch die Entwicklung eines eigenen Verkehrsdienstes möglich, welcher dem eines Verkehrsfunks gleicht, jedoch von allen Transportmitteln benutzt und dessen Nachrichten von allen Transportmitteln interpretiert werden können. So sehen die meisten Plattformen bereits einen Verzeichnisdienst vor, bei dem Agenten bestimmte Informationen abonnieren können. Auf Grundlage eines solchen Dienstes würde die Verbreitung von Verkehrsnachrichten funktionieren. Der Ablauf wäre dann wie folgt gegeben:

1. Ein Transportmittel stößt während der Fahrt auf ein Hindernis. Solche Hindernisse können zum Beispiel die Sperrung einer Straße auf Grund eines Unfalls oder die Absage einer Fährüberfahrt auf Grund eines Unwetters sein.
2. Das Transportmittel schickt eine standardisierte, ontologiebasierte Nachricht über die Art und den Umfang der Störung an den Verkehrsdienst.
3. Alle Transportmittel, welche diesen Dienst abonniert haben, bekommen automatisch eine Nachricht, in der sie über das aufgetretene Hindernis informiert werden.
4. Diejenigen Transpormittel, die von der Störung auf ihrer Route betroffen sein können, evaluieren ob eine neue Routenplanung abgebracht ist, oder ob das Hindernis bis zu ihrem Eintreffen an dieser Stelle eventuell schon wieder beseitigt ist.
5. Ist das Hindernis beseitigt oder die Störung entfernt worden, so wird ebenfalls eine Nachricht an den Verkehrsdienst gesendet. Dieses kann entweder durch das Trans-

portmittel erfolgen, welches die Störung gemeldet hat oder durch ein nachfolgendes, welche diese Störung nicht mehr vorfindet.

Die abonnierenden Transportmittel haben so die Möglichkeit, diese Informationen mit in die Kostenfunktion zur Routenberechnung mit einfließen zu lassen.

Ob bereits vorhandene Geräte benutzt oder eigene Dienste von den Designern entworfen werden, hängt sicherlich von der Größe und dem Umfang des zu realisierenden Projekts ab. In diesem Fall wäre die Einbindung solcher Technologien in fortschrittliche Navigationsgeräte, welche in der Lage sind mit Agenten zu kommunizieren, wünschenswert.

5.2.6 Verteiltes Wissen über Verladestationen und Transportwege

Verteiltes Wissen über Verladestationen ist besonders dann zu berücksichtigen, wenn Transportmittel unterschiedlicher Unternehmen kooperieren wollen. So hat jedes Unternehmen seine eigenen, nur ihm bekannten Verladestationen und die Agenten müssen während der Kooperation festlegen, welcher von diesen Umschlagplätzen bei einer erfolgreichen Verhandlung genutzt werden soll. Die Auswahl muss dabei kostenminimierend ausfallen. Damit die Kenntnis von vorhandenen Verladestationen in die Routenplanung mit einfließen kann, müssen zu Beginn einer Kooperation die Wissensbasen der teilnehmenden Agenten abgeglichen und im Bedarfsfall ergänzt werden. Dazu übermitteln die Agenten die Standpunkte der ihnen bekannten Verladestationen samt ihrer geographischen Koordinaten an den Kooperationspartner. Dieser nimmt daraufhin den angesprochenen Abgleich vor und fügt die ihm nicht bekannten Stationen seiner Wissensbasis hinzu.

Ebenso wie es verteiltes Wissen über die Verladestationen gibt, können auch unterschiedliche Wissensbasen bezüglich der nutzbaren Transportwege auftreten. Diese können entstehen, wenn neue Wege erschlossen oder bereits bekannte Strecken nicht mehr nutzbar werden. Beispiele für bisher unbekannte Transportwege wären neu gebaute Straßen oder aber neue Fährverbindungen, welche die Transportmittel benutzen können. Abbildung 5.6 zeigt eine mögliche Konsequenz, die aus diesem verteilten Wissen entstehen kann. Gegeben ist das Transportmittel TM, welches nur die Route r_1 kennt, um von S_1 nach S_2 zu gelangen. Da jedoch auch eine direkte Verbindung r_2 zwischen den beiden Stationen besteht wäre ein Transportmittel welches diesen Weg kennt in der Lage bei einer Auktion um ein Frachtgut das von S_1 nach S_2 transportiert werden soll, einen niedrigeren Preis zu bieten. Da solche Änderungen am Streckennetz in der Regel nicht so häufig auftreten,

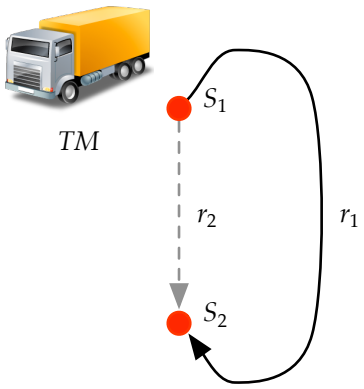

Abbildung 5.6: Beispiel für verteiltes Wissen über einen Transportweg

ist ein wöchentlicher Abgleich mit anderen Agenten ausreichend. Dieser könnte dann zeitgleich mit dem Abgleich der Verladestationen im Vorwege einer Kooperation mit anderen Transportmitteln erfolgen.
Der umgekehrte Fall, dass eine bereits bekannte Strecke nicht mehr nutzbar ist, wird durch den gleichen Mechanismus abgedeckt, der auch für die Verfügbarkeit von Routen verantwortlich ist. So kann der Wegfall einer Strecke als Sperrung mit unendlicher Dauer angesehen werden. Ein endgültiger Wegfall der Route aus dem Streckennetz würde erst dann eintreten, wenn die Streckennetzdaten von einer Instanz außerhalb des hier betrachteten Szenarios aktualisiert werden. Bei der Verwendung von Navigationsgeräten wäre das Erscheinen eines Updates für das Streckennetz solch ein Fall. Eine weitere Möglichkeit bei der Nutzung eines eigenen Systems ist die periodische Aktualisierung der Daten aller im System befindlichen Agenten. Dadurch könnte die Zeit, bis das Transportmittel aus Abbildung 5.6 wieder aktuelle Streckennetzdaten zur Verfügung hat, verkürzt werden, da nicht auf eine ausgeschriebene Kooperation gewartet werden muss.

6 Realisierung des Fallbeispiels

Dieses Kapitel schildert im ersten Teil auf welchen Aspekten die Auswahl einer für die Implementierung der Fallbeispiels geeigneten Agentenplattform erfolgte. In den folgenden Abschnitten werden die Eigenschaften dieser Agentenplattform näher erläutert. Nach einer kurzen Einführung in die Architektur wird auf eines der wichtigsten Merkmale von Agenten eingegangen; dem Modellieren von Aufgaben. Damit ein Agentensystem überhaupt funktionieren kann, muss es den Transport von Nachrichten ermöglichen. Welche Möglichkeiten dieser Zusammenhang bietet, wird in Abschnitt 6.2.3 dargelegt. Aufbauend auf den dort beschriebenen Kommunikationsaspekten wird in 6.2.4 die Verwendung von Ontologien thematisiert. Im zweiten Teil dieses Kapitels wird die implementierte Lösung für das Tourenplanungsproblem diskutiert. In den Abschnitten 6.4 bis 6.5 folgen die Beschreibungen der Implementierungen der am Fallbeispiel beteiligten Agenten. So wird zunächst auf die Frachtgutagenten mit ihren Transport- und Überwachungsagenten eingegangen und danach die Implementierung der Transportmittelagenten erklärt. Abschließend wird beschrieben, welche Daten als Grundlage für die ausgeführten Tests dienten und wie diese durchgeführt wurden.

6.1 Auswahl einer geeigneten Agentenplattform

Die Auswahl einer Plattform zur Implementierung des Fallbeispiels erfolgte auf Grundlage der in [Ros08] zusammengestellten Übersicht aktuell verfügbarer Agentenplattformen; sie ist in Tabelle 6.2 zusammengefasst. Eines der für die Auswahl der Plattform angesetzten Kriterien war die Lizenz, unter der die Plattform vertrieben wird. Während sich fünf der sechs untersuchten Plattformen durch eine Lizenz auszeichneten, welche die freie Benutzung gestattet, fiel Jack durch die relativ hohen Kosten in Höhe von 2 500 € für eine akademische Lizenz aus der engeren Auswahl heraus. Ein weiterer wichtiger Aspekt waren die Dokumentationen. Sie sollten eine schnelle Einarbeitung in den Umgang mit der Plattform ermöglichen. Dabei stellte sich heraus, dass alle Plattformen gut und

zum Teil sehr ausführlich dokumentiert sind. Durch die Forderung, dass die Plattform die Verwendung von Ontologien unterstützen sollte, fiel auch MadKit aus dem Kreis der in Frage kommenden Plattformen heraus. Ebenso sollte die Plattform Anzeichen der Wartung und der Weiterentwicklung aufweisen. Dieses Kriterium erfüllten nur noch zwei der Plattformen, welche noch in Frage gekommen wären: Jade und Cougaar. Die Wahl fiel letzten Endes auf Jade, da hier durch [BCG07] eine sehr gute Einführung zur Verfügung stand, welche viele anwendungsorientierte Beispiele gibt und einen schnellen Einstieg in die Programmierung gestattet.

	Jade	**FIPA-OS**	**Jack**
FIPA-Konformität	ja	ja	ja
Wartung der Plattform	ja	keine Anzeichen erkennbar	ja
Weiterentwicklung der Plattform	ja	keine Anzeichen erkennbar	ja
Verfügbarkeit der Plattform	Download über Homepage	Download über SourceForge	Download über Homepage
letztes Release der Plattform	Version 3.6 vom 05.05.2008	Version 2.2.0 vom 18.03.2003	Version 5.2 vom 08.05.2007
Version für mobile Geräte	ja, Jade Leap und Jade für Android	ja Micro-FIPA-OS	nein, normale Version für alle Geräte geeignet
letztes Release der mobilen Version	25.06.2007	09.05.2001	–
Lizenz	GNU Lesser General Public Licence	Nortel Networks FIPA-OS Public Licence	kommerziell; einmalige Gebühr 2 500 € exkl. Steuern
Programmiersprache	Java	Java	Java
Eignung für bestimme Anwendungsgebiete	nein	nein	zeitkritische Anwendungen, BDI-Systeme
Dokumentation	sehr ausführlich, viele Beispiele	ausführlich	umfangreich, viele Beispiele
Ontologien	werden unterstützt	werden unterstützt	werden nicht unterstützt

Fortsetzung auf der nächsten Seite

	Jade	**FIPA-OS**	**Jack**
Sicherheit	Jade-S (Kommunikation über SSL möglich)	Kommunikation über SSL möglich	keine Sicherheitsvorkehrungen vorgesehen

	agentTool	**Cougaar**	**MadKit**
FIPA-Konformität	ja	ja	nein
Wartung der Plattform	keine Anzeichen erkennbar	ja	ja
Weiterentwicklung der Plattform	keine Anzeichen erkennbar	ja	ja
Verfügbarkeit der Plattform	Download über Homepage	Download über Homepage	Download über Homepage
letztes Release der Plattform	Version 1.8.3 vom 21.05.2002	Version 12.4 vom 24.09.2007	Version 4.1.2 von November 2005
Version für mobile Geräte	nein	ja, Cougaar Micro	nein
letztes Release der mobilen Version	–	01.01.2003	–
Lizenz	nicht bekannt	Cougaar Open Source Licence	Lesser General Public Licence
Programmiersprache	Java	Java	Java
Eignung für bestimme Anwendungsgebiete	nein (uneingeschränkt nur unter Windows einsetzbar)	große Multi-Agenten Systeme	nein
Dokumentation	viele Funktionen undokumentiert	umfangreich, Tutorials und Vorträge verfügbar	ausführlich
Ontologien	Unterstützung ist geplant	werden unterstützt	werden nicht unterstützt
Sicherheit	Kommunikation über SSL möglich	Kommunikation über SSL möglich	Kommunikation über SSL möglich

Tabelle 6.2: Übersicht der in [Ros08] untersuchten Agentenplattformen

6.2 Jade Agentenplattform

Jade steht für „*Java Agent DEvelopment Framework*“und wurde vom Telecom Italia Lab (TILAB) entwickelt. Es ist vollständig in Java implementiert und basiert auf den FIPA Spezifikationen für Agenten. Vertrieben wird es unter der GNU Lesser Public License, wodurch es unter anderem für einen beliebigen Zweck benutzt, vervielfältigt und weitergeben, sowie an eigene Bedürfnisse angepasst und abgeändert werden darf. Es bietet dem Programmierer eine Vielzahl an Kernfunktionalitäten, welche in [BCG07] wie folgt zusammengefasst sind:

- Unterstützung für verteilte Agentensysteme
- Vollständige Einhaltung der FIPA Spezifikationen
- Effizienter Transport asynchroner Nachrichten
- Bereitstellung eines Weiße-, sowie Gelbe-Seiten-Dienstes
- Einfaches und effektives Lebenszyklus-Management für Agenten
- Unterstützung für mobile Agenten
- Möglichkeit des Abonnements von Plattformereignissen für Agenten
- Grafische Tools zum Beseitigen von Fehlern und zur Überwachung
- Unterstützung von Ontologien und Inhaltssprachen
- Bibliothek von Interaktions-Protokollen
- Integration von Web-Technologien wie JSP, Servlets und Applets
- Unterstützung von J2ME
- Starten und Kontrollieren einer Plattform und seiner verteilten Komponenten aus einem externen Programm heraus
- Erweiterbarer Kernel

6.2.1 Architektur der Jade Agentenplattform

Abbildung 6.1 zeigt die Architektur der Jade Agentenplattform. Sie besteht aus einzelnen Containern, welche über ein Netzwerk verteilt sein können. Jeder Agent stellt einen

eigenen Thread dar und existiert in einem Container, welcher durch einen Java Prozess realisiert wird. Dieser stellt die Jade Laufzeit und zusätzlich alle zum Verwalten und Ausführen von Agenten benötigten Services zur Verfügung.

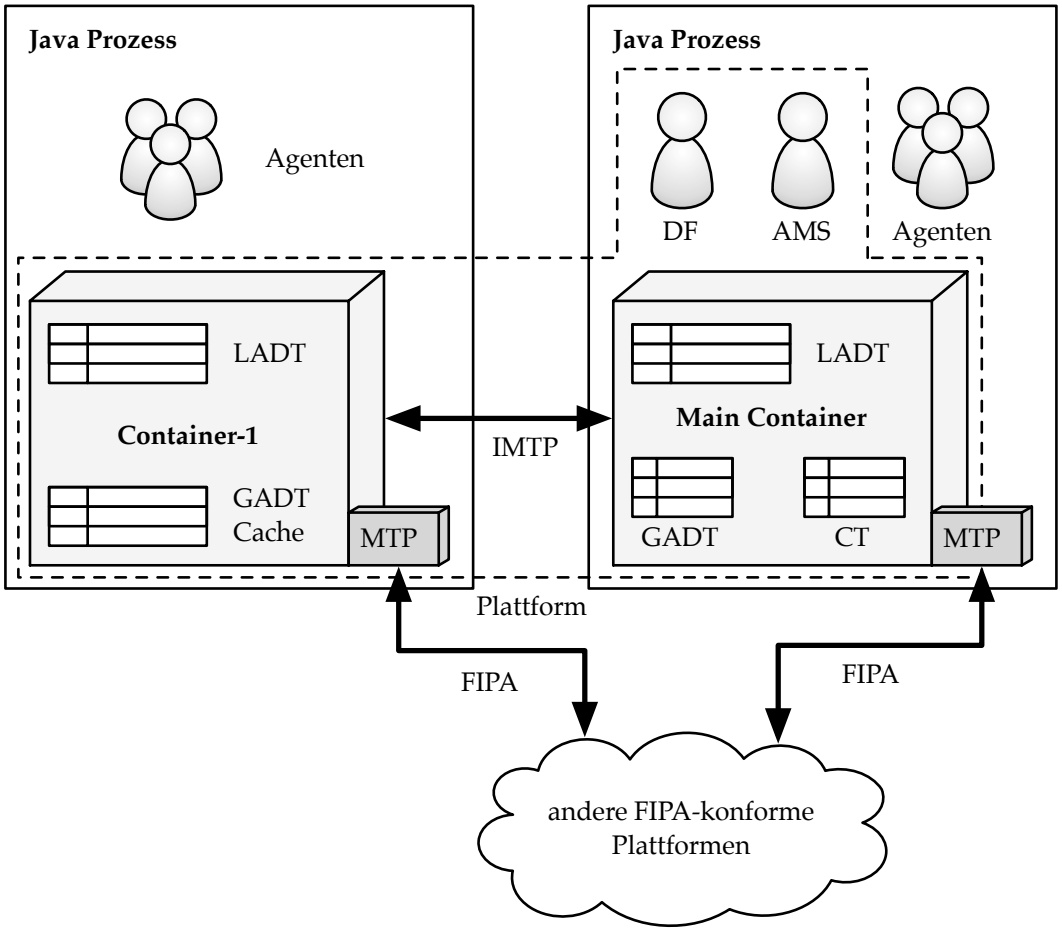

Abbildung 6.1: Hauptbestandteile der Jade Architektur

Eine besondere Rolle spielt der Hauptcontainer (*engl. main container*). Er stellt den Ausgangspunkt der Agentenplattform dar und wird initial geladen. Alle weiteren Container müssen sich mit dem Hauptcontainer verbinden, indem sie sich bei ihm registrieren. Aufgrund der Tatsache, dass dieser den Ausgangspunkt der Plattform darstellt, werden ihm besondere Aufgaben zuteil. So verwaltet er die Container-Tabelle (*engl. container table*) CT, welche die Registrierung für alle Objektreferenzen und Transportadressen der zur Plattform gehörigen Container darstellt. Zusätzlich verwaltet er auch die globale Agentenbeschreibungstabelle (*engl. global agent descriptor table*) GADT. In ihr sind alle Agenten samt aktueller Ortsangabe und momentanem Status eingetragen, die auf der Plattform

laufen. Darüber hinaus beherbergt er das Agent Management System (AMS) sowie den Verzeichnis Vermittler (*engl. directory facilitator*) DF. Beides sind Agenten, welche beim Start des Hauptcontainers automatisch von Jade instantiiert und gestartet werden. Das AMS regelt den Zugang und die Benutzung der Agentenplattform. Zudem stellt es den Weiße-Seiten-Dienst (*engl. white page service*) und den Lebenszyklus-Dienst bereit. Wird ein Agent gestartet, so wird er automatisch beim AMS registriert und erhält eine Agentenkennung (*engl. agent identifier*) AID, welche in ein Verzeichnis eingetragen wird. Der DF stellt den Gelbe-Seiten-Dienst (*engl. yellow page service*) zur Verfügung. Dieser bietet den Agenten die Möglichkeit, eigene Dienste zu veröffentlichen oder nach angebotenen Diensten zu suchen. Ebenso akzeptiert der DF Abonnements von Agenten, die benachrichtigt werden wollen wenn ein bestimmter Dienst angemeldet oder geändert wird. Um zu verhindern, dass der Hauptcontainer zum Flaschenhals des Systems wird, besitzen alle weiteren Container in Jade zwei Tabellen. Zum einen die lokale Agentenbeschreibungstabelle (*engl. local agent descriptor table*) LADT und zum anderen einen Cache der globalen Agentenbeschreibungstabelle. Hat ein Container die Aufgabe, den Empfänger einer Nachricht zu lokalisieren, wird zunächst die LADT durchsucht. Schlägt diese Suche fehl, wird Kontakt mit dem Hauptcontainer hergestellt um die richtige Referenz zu erhalten. Diese wird daraufhin in den Cache geschrieben. Aufgrund der Dynamik des Systems kann es vorkommen, dass Agenten migrieren, terminieren oder neue Agenten geschaffen werden. Dadurch können veraltete Cache-Einträge existieren, welche zu falschen Adressen führen. Tritt dieser Fall ein, so erhält der Container eine Fehlermeldung und muss seinen Cache mit den Werten aus der GADT des Hauptcontainers aktualisieren.

6.2.2 Modellierung von Aufgaben in Jade

Die von Agenten auszuführenden Aufgaben werden in Jade mittels so genannter Verhalten implementiert. Ein Verhalten beschreibt dabei die Reaktion des Agenten auf seine Umwelt. Alle zur Verfügung stehenden Verhaltensklassen sind von der abstrakten Klasse `Jade.core.behaviours.Behaviour` abgeleitet. Sie sind ereignisorientiert und dienen zur Implementierung der Agentenaktivitäten. Die Klasse `Jade.core.Agent`, von der alle selbst implementierten Agentenklassen erben müssen, bietet für die Verhaltenssteuerung die beiden Methoden

- `addBehaviour(Behaviour b)` und
- `removeBehaviour(Behaviour b)`

an. Sie dienen dem Hinzufügen von neuen, beziehungsweise dem Entfernen von nicht mehr benötigten Verhalten. Somit besteht die Möglichkeit, Verhaltensweisen zur Laufzeit zu erzeugen. Ein Scheduler führt alle Verhalten so lange aus, bis die `action()`-Methode beendet ist. Diese abstrakte Methode der `Behaviour`-Klasse muss in den erstellten Unterklassen implementiert werden, um das gewünschte Verhalten zu definieren. Generell erfolgt die Abarbeitung aller Verhalten nach dem folgenden Schema:

- Die `action()`-Methode des Agenten wird durch den Scheduler ausgeführt.
- Ist die Abarbeitung abgeschlossen wird die `done()`-Methode aufgerufen, in der die Erfüllung der Aufgabe überprüft wird.
- Ist die Aufgabe erfüllt, so wird das Verhalten aus der Abarbeitungsschlange des Schedulers gelöscht. Ist dies nicht der Fall, so wird das Verhalten wieder am Ende der Schlange eingefügt.

Die von Jade zur Verhaltenssteuerung angebotenen Klassen sind:

- `Behaviour`: Abstrakte Basisklasse für die Modellierung von Agenten-Tasks und die Verhaltensabarbeitung.
  - `SimpleBehaviour`: Abstrakte Klasse zur Modellierung von einfachen, nicht unterbrechbaren Verhalten.
    - `OneShotBehaviour`: Abstrakte Klasse welche Verhalten modelliert, die nur einmal ausgeführt und nicht unterbrochen werden können.
      - `TickerBehaviour`: Definiert Verhalten, welche in periodischen Abständen wiederholt werden sollen.
    - `CyclicBehaviour`: Abstrakte Klasse, die zyklisch auszuführende Verhalten realisiert.
    - `WakerBehaviour`: Abstrakte Klasse, die ein `OneShotBehaviour` erst nach einer festgelegten Zeit ausführt.
  - `CompositeBehaviour`: Abstrakte Basisklasse für komplexe Verhalten, die sich aus mehreren Aufgaben zusammensetzen können.
    - `ParallelBehaviour`: Modelliert komplexe Verhalten, dessen Subaufgaben parallel ausgeführt werden müssen.
    - `SerialBehaviour`: Abstrakte Basisklasse für alle seriell auszuführenden Verhalten.
      - `SequentialBehaviour`: Modelliert komplexe Verhalten, dessen Subaufgaben sequentiell ausgeführt werden müssen.
      - `FSMBehaviour`: Modelliert komplexe Aufgaben, dessen Subaufgaben

wie ein endlicher Automat ausgeführt werden. Dieses bedeutet, dass die Verhaltensübergänge von bestimmten Bedingungen abhängen.

6.2.3 Kommunikation zwischen Agenten

Die Inter-Agenten-Kommunikation findet in Jade über so genannte ACL-Nachrichten statt. ACL steht für „*Agent Communication Language*“und basiert auf dem in [Fou02a] beschriebenen FIPA Standard. Eine ACL-Nachricht enthält einen Satz von einem oder mehreren Parametern.

Parametername	**Beschreibung**
`performative`	Typ des Sprechakts der ACL Nachricht
`sender`	Absender der Nachricht
`receiver`	Empfänger der Nachricht
`reply-to`	Gibt an, dass alle Antworten auf eine Nachricht an diesen Agenten und nicht an den unter `sender` genannten Agenten geschickt werden sollen
`content`	Der eigentliche Nachrichteninhalt
`language`	Sprache, in der der Inhalt verfasst ist
`encoding`	Kodierung des Inhalts
`ontology`	Ontologie, welche verwendet wird
`protocol`	Interaktionsprotokoll, welches der sendende Agent benutzt
`conversation-id`	Identifikator, welcher eine fortlaufende Kommunikation kennzeichnet
`reply-with`	Ausdruck, welcher vom antwortenden Agenten benutzt wird, um eine Nachricht zu identifizieren
`in-reply-to`	Referenz auf eine frühere Aktion, zu der diese Nachricht eine Antwort darstellt
`reply-by`	Zeit- oder Datumsangabe als Frist für eine Antwort

Tabelle 6.3: Parameter von ACL Nachrichten

Alle zu sendenden Nachrichten müssen jedoch den Parameter `performative` aufweisen, welcher den Typ des kommunikativen Aktes nach [Fou02b] angibt. Zu den am häufigsten verwendeten Typen gehören `request`, `inform`, `refuse` und `not-understood`. Eine Ausnahme bilden die in Jade implementierten Kommunikationsprotokolle. Sie werden durch ihren Namen beziehungsweise ihre Abkürzung im `performative`-Parameter gekennzeichnet. Im Allgemeinen sind jedoch auch die Parameter `sender`, `receiver` und `content` gesetzt. Welche Bestandteile eine ACL-Nachricht noch aufweisen kann, zeigt Tabelle 6.3.

Um die parallele Kommunikation zwischen mehreren Agenten zu ermöglichen, ordnet Jade jeder Nachricht einen global eindeutigen Bezeichner im Parameter `conversion-id` zu. Somit sind die Agenten in der Lage, eingehende Nachrichten einem bestimmten Vorgang zuzuordnen.

Der `language`-Parameter gibt die Darstellung einer ACL-Nachricht während der Übertragung an. Zu den Kodierungsarten gehören unter anderem XML oder der von der FIPA standardisierte SL Codec nach [Fou02d].

6.2.4 Verwendung von Ontologien in Jade

Jade unterstützt die Verwendung von Ontologien um eine Wissensdomäne zu definieren, zu der ein Nachrichteninhalt gehört. Dadurch ist es möglich, zusammengesetzte Datentypen zu erstellen und zwischen den Agenten sprachunabhängig auszutauschen. Eine Ontologie besteht im Wesentlichen aus Prädikaten und Ausdrücken, die sich in Konzepte und weiter in Agentenaktionen untergliedern. Die Beziehungen zwischen den Elementen sind in Abbildung 6.2 dargestellt.

Ein Konzept ist eine strukturierte Einheit (Klasse) die aus Slots (Attribute der Klasse) aufgebaut ist. Solche Slots sind entweder Primitive, die atomare Datentypen repräsentieren oder andere Konzepte. Ein Prädikat hingegen beschreibt logische Aussagen über Gegenstände. Diese Aussagen können entweder wahr oder falsch sein. Agentenaktionen definieren Vorgänge, die von Agenten ausgeführt werden können. Sollen Objekte, welche in einer Ontologie definiert sind, mittels Nachrichten versendet werden, so müssen diese als Agentenaktion angegeben werden.

In Jade werden Ontologien durch die Klasse `Jade.content.onto.Ontology` repräsentiert. Sie stellt Konzepte, Prädikate und Agentenaktionen zur Verfügung, welche als Nach-

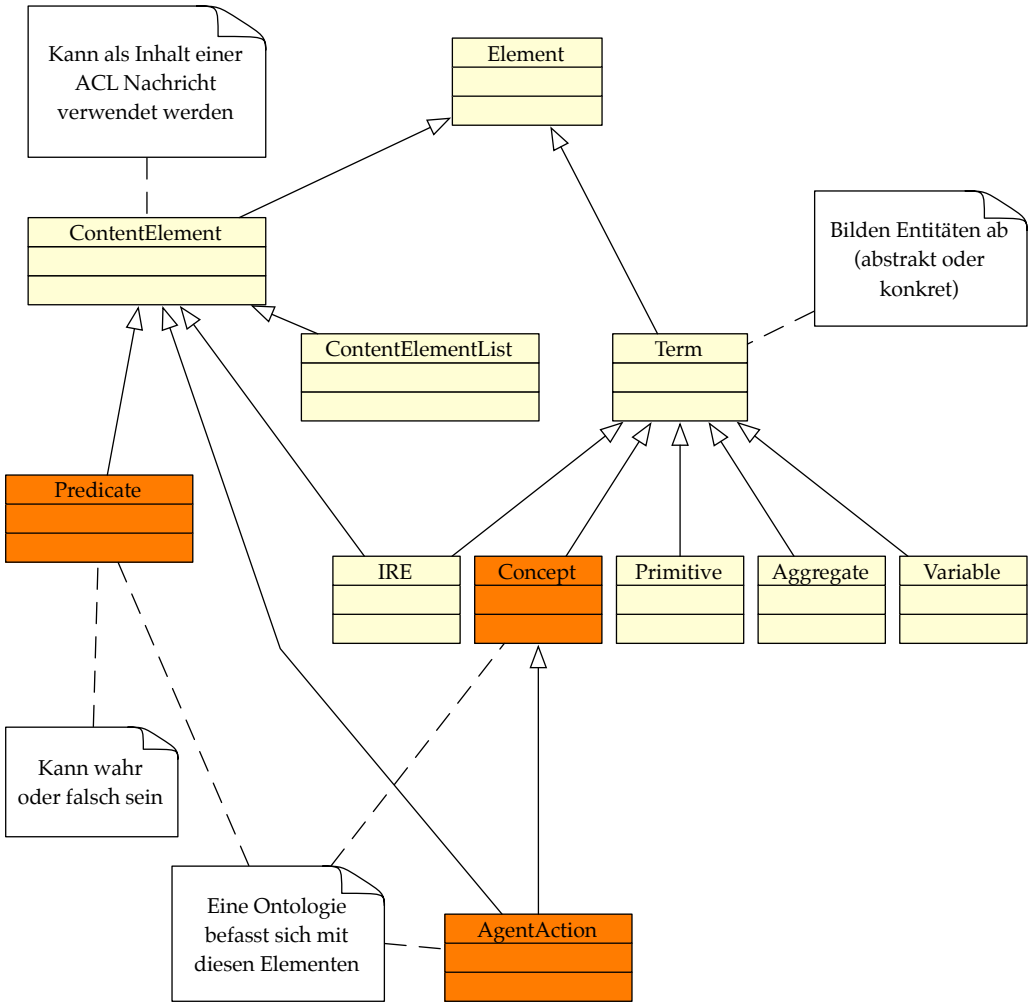

Abbildung 6.2: Content Reference Model in Jade nach [CC04]

richteninhalte benutzt werden können. Für jedes dieser Elemente muss eine Java Klasse existieren. Ist dies der Fall, so können zur Laufzeit Objekte erzeugt werden, die vom Content-Manager mit Hilfe eines Codecs in eine Darstellung überführt werden, welche als Nachrichteninhalt einer ACL-Nachricht verwendet werden können. Dieser Prozess ist in Abbildung 6.3 schematisch dargestellt. Der Content-Manager ist in der Lage, den Nachrichteninhalt empfängerseitig zurück in die Darstellung als Objektstruktur zu überführen, ohne das dazu weiterer Programmieraufwand notwendig ist. Bevor die Nachricht endgültig an den Empfänger gesendet wird, findet noch eine Validierung statt, welche eventuell vorhandene Fehler aufdeckt.

Die Verwendung von Ontologien ist bei Realisierung von Agentensystemen nahezu uner-

lässlich, da so gewährleistet werden kann, dass alle Agenten die gleiche Wissensdomäne als Grundlage für ihre Kommunikation benutzen. In Bezug auf das konkrete Fallbeispiel bedeutet dies unter anderem, dass der Einsatz von Transportmittelagenten unterschiedlicher Hersteller ermöglicht wird und die eindeutige Verständigung mit allen Teilen des Systems gegeben ist.

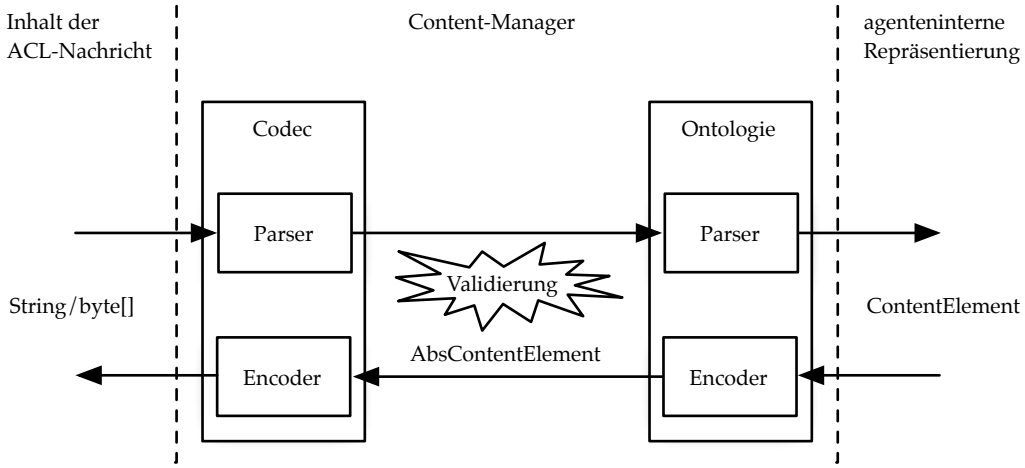

Abbildung 6.3: Konvertierungs-Pipeline in Jade nach [CC04]

6.3 Implementierte Lösung für das Tourenplanungsproblems

In dieser Arbeit wurde zur Lösung des Tourenplanungsproblems die Evaluiertungsversion der kommerziellen Softwarebibliothek JOpt.SDK der Firma DNA evolutions [DNA08] verwendet. Sie ist für Java erhältlich und bietet die Möglichkeit, eine Vielzahl der zur Klasse des Problems des Handlungsreisenden zählenden Varianten zu lösen. Die eingeschränkte Version ist auf zwanzig Transportmittel und anzufahrende Stationen beschränkt. Die Verwendung von genetischen Algorithmen zur Implementierung der Bibliothek spiegelt sich in den zur Verfügung stehenden Parametern für die Optimierung wider. Über das Setzen von `JOptExitCondition.JOptGenerationCount` kann beispielsweise angegeben werden, nach wie vielen neu erzeugten Generationen der Algorithmus terminieren soll. Ebenso ist ein vorzeitiges Beenden des Algorithmus bei konvergierendem Wert der Kostenfunktion möglich, was durch die Angabe einer Generationenanzahl im Parameter `JOptExitCondition.JOptConvergencyCount` realisiert werden kann.

Generell ist der Ablauf für das Finden einer Route durch die folgenden fünf Schritte gekennzeichnet:

1. Konfiguration der anzufahrenden Orte
2. Konfiguration der Transportmittel
3. Setzen der Optimierungsparameter
4. Starten der Optimierung
5. Speichern und Rückgabe der Ergebnisse

Im ersten Schritt werden die anzufahrenden Orte an den Algorithmus übergeben. Zur Konfiguration gehören neben den Orten selbst auch die vorgegebenen Zeitfenster, vorhandene Relationen zwischen den Orten sowie die Menge ab- oder aufzuladender Frachtgüter welche jeder Station zugeordnet werden müssen. Daraufhin folgt im zweiten Schritt die Konfiguration und die Übergabe der Transportmitteldaten an den Algorithmus. Hierzu zählen unter anderem die aktuelle Position, die Durchschnittsgeschwindigkeit sowie die maximale Anzahl der fahrenden Stunden pro Tag. An dieser Stelle sind weitere Angaben für eine noch realistischere Modellabbildung möglich. So können seitens des JOpt.SDKs auch die Einhaltung der gesetzlich vorgeschriebenen Lenkzeiten oder der Transport von Gefahrgütern mit in die Routenplanung einfließen. Diese Möglichkeiten wurden auf

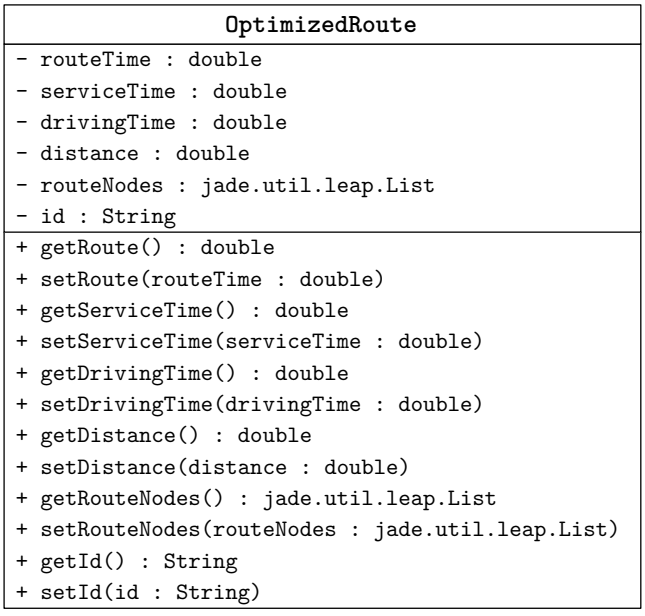

Abbildung 6.4: Klassendiagramm der `OptimizedRoute`-Klasse

Grund des Abstraktionsgrades des Fallbeispiels jedoch nicht mit berücksichtigt, können jedoch für die Implementierung eines realen Anwendungssystems benutzt werden. Im folgenden Schritt werden die Optimierungsparameter gesetzt. Dadurch kann Einfluss auf die zu findende Route genommen werden, in dem bestimmten Attributen eine hohe Gewichtung zugeordnet wird. Die eigentliche Optimierung und das damit verbundene Finden einer Route wird durch aufrufen von `startAsynchronousOptimizationRun()` der Klasse `Optimization` initiiert. Nachdem die Optimierung abgeschlossen ist, wird durch Aufrufen der Methode `onAsynchronousOptimizationResult()` eine Instanz der Klasse `OptimizationResult` zurückgeliefert, welches die Ergebnisse der Optimierung enthält. Die für die Realisierung des Fallbeispiels notwendigen Daten werden in einer Instanz der Klasse `OptimizedRoute` an den aufrufenden Transportmittelagenten zurückgegeben. Das dazugehörige Klassendiagramm ist in Abbildung 6.4 dargestellt. Die anzufahrenden Stationen werden dabei in einer Liste gespeichert, welche Instanzen der in 6.5 abgebildeten Klasse `RouteNode` enthält.

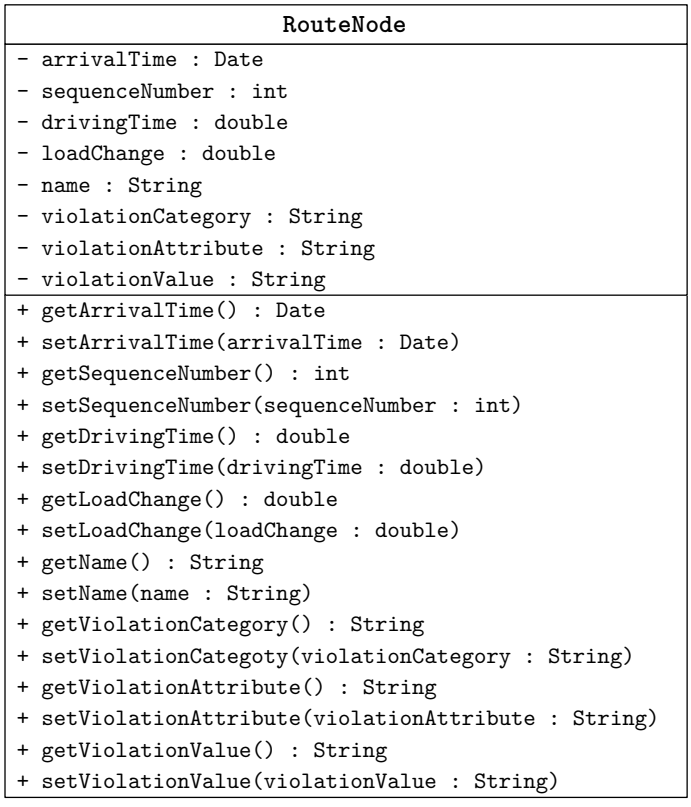

Abbildung 6.5: Klassendiagramm der `RouteNode`-Klasse

6.4 Implementierung des Frachtgutagenten

Der Frachtgutagent hat zwei Aufgaben zu erfüllen. Im ersten Schritt werden die für die Ausführung erforderlichen Parameter eingelesen. Diese werden in der Initialisierungs-Methode `setup()` des Agenten über eine Konfigurationsdatei, wie in Codebeispiel 6.1 beispielhaft zu sehen, zur Verfügung gestellt. Dazu gehören unter anderem der Auslieferungsort, die vorgeschriebenen Zeitfenster, der Typ des Frachtguts und der Maximalbetrag den der Agent bereit ist, für den Transport zu zahlen.

```
wareAgent.from=Hamburg
wareAgent.to=Berlin
wareAgent.maxPrice=1000.0
wareAgent.earliestDelivery=2008-07-31_10:00
wareAgent.latestDelivery=2008-07-31_20:00
wareAgent.earliestPickup=2008-07-31_08:00
wareAgent.latestPickup=2008-07-31_18:00
wareAgent.type=Reefer_Cargo
wareAgent.amount=60
```

Codebeispiel 6.1: Verwendete Konfigurationsdatei für Frachtgüter

Im zweiten Schritt müssen der generalisierte Überwachungsagent sowie der Transportagent initialisiert werden. Dazu bietet Jade die Möglichkeit, am zur Plattform gehörigen Kontroller `PlatformController` die Methode `createNewAgent(String nickname, String classname, Object[] args)` aufzurufen und anschließend zu starten. In Codebeispiel 6.2 ist dieser Vorgang beispielhaft für das Starten des Transportagenten durch den Frachtgutagenten dargestellt.

```
PlatformController container = getContainerController();
container.createNewAgent(
    getAID().getLocalName() + "_transportAgent",
    "agent.TransportAgent", wares).start();
```

Codebeispiel 6.2: Starten eines Agenten durch einen anderen Agenten

6.4.1 Implementierung der Überwachungsagenten

In der Implementierung des Fallbeispiels wird die Funktionsweise der in Kapitel 5.1.2 beschriebenen Überwachungsagenten anhand des Beispiels eines Kühlguts demonstriert. Dazu werden im Folgenden der Initialisierungsprozess sowie der Überwachungsablauf dargestellt. Das in Abbildung 6.6 gezeigte Sequenzdiagramm fasst die Funktionsweise der Agenten vereinfacht zusammen.

Zunächst wartet der vom Frachtgutagenten erzeugte und gestartete, generalisierte Überwachungsagent mittels des zyklischen Verhaltens `WaitAndDelegateMonitorings` auf eine eingehende Nachricht vom Transportagenten des Frachtguts. Erhält er diese Nachricht, werden die darin enthaltenen Informationen über die mit dem Transportmittel ausgehandelten Konditionen ausgelesen. Anhand dieser Daten entscheidet der Agent, welche, zusätzlich zu den eigenen, Überwachungsmaßnahmen getroffen werden müssen und welcher spezialisierte Überwachungsagent dafür benötigt wird. Für die Demonstration steht ein auf Kühlgüter spezialisierter Agent zur Verfügung. Dieser wird daraufhin erzeugt und gestartet. Seine Aufgabe besteht darin, mit den in der Realität vorhandenen Sensoren zu kommunizieren und die von ihnen zurückgelieferten Daten auszuwerten. Um den Prozess der Ausnahmebehandlung zu simulieren, wurde ein weiterer Agent, welcher die Rolle solch eines Sensors übernimmt, implementiert. Im Gegensatz zu einer realen Anwendung in der die Frachtgüter bereits mit solchen Hardware-Sensoren versehen sind, wird in dieser Implementierung der Sensor-Agent vom spezialisierten Überwachungsagenten erzeugt und gestartet. Daraufhin abonniert der spezialisierte Agent die Messdaten des Sensor-Agenten durch Senden einer Nachricht vom Typ `ACLMessage.SUBSCRIBE`. Dieses Senden wurde innerhalb des `SubscribeSignal`-Verhaltens, welches ein `OneShotBehaviour` darstellt, implementiert. Der Sensor bietet zum Empfangen solcher Nachrichten ein zyklisches Verhalten namens `WaitForSignalSubscription`. Innerhalb dieses Verhaltens wird ein zusätzliches `TickerBehaviour` hinzugefügt, welches die Messwerte in regelmäßigen Abständen an den Sender der Abonnent-Nachricht schickt, bis die Auslieferung des Frachtguts erfolgt ist. Die Kühlguttemperaturen werden durch eine Methode zufällig aus einem vorgegebenen Intervall generiert und in diesem Fall an den auf Kühlgüter spezialisierten Agenten gesendet. Dieser hat durch ein zyklisches Verhalten die Möglichkeit, diese Messdaten auszuwerten. Für die Realisierung wurde das Intervall, aus dem die Kühlguttemperaturen ausgewählt werden können so gewählt, dass es zu einem Überschreiten des Toleranzwertes für den Transport von Kühlgut kommen kann. Tritt dieser Fall ein, so sendet der spezialisierte Agent eine Nachricht vom Typ `ACLMessage.FAILURE` an den generalisierten Überwachungsagenten, in welcher der Grund für den Alarm enthalten ist.

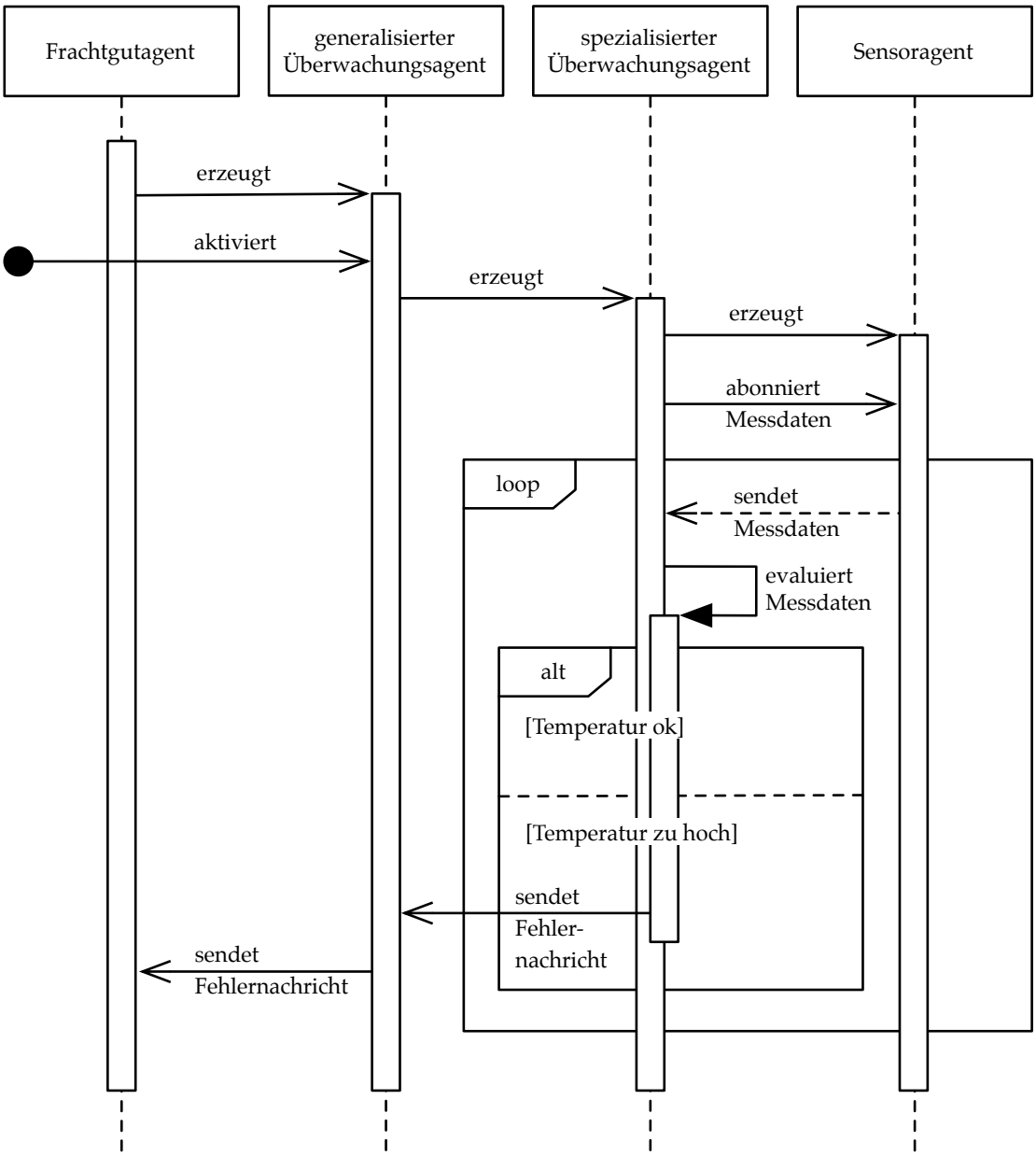

Abbildung 6.6: Sequenzdiagramm des Überwachungskreislaufs

Dadurch erhält dieser die Möglichkeit, auf Fehler zu reagieren, indem er seinerseits die dafür vorgesehene Fehlerbehandlung im zyklischen Verhalten `ProcessFailureMessage` startet.
Der Überwachungsablauf dieser Demonstration ist damit abgeschlossen, da weitere Behandlungen problemspezifisch erfolgen müssten.

6.4.2 Implementierung des Transportagenten

Der Transportagent muss die Suche nach geeigneten Transportmitteln realisieren und im darauf folgenden Schritt die Verhandlung über die Vergabe des Transportauftrag initiieren. In der konkreten Umsetzung benutzt der Transportagent den von der Jade Plattform zur Verfügung gestellten Gelbe-Seiten-Dienst zum Finden von Transportmitteln. Dazu wird eine Anfrage an den Dienst gesendet in der nach Transportmitteln gefragt wird, die den Transport vom Typ des Frachtguts unterstützen. Hierbei ist eine Limitierung der Ergebnisse auf maximal fünf Treffer vorgesehen. Diese Einschränkung dient der Begrenzung des Kommunikations- und Rechenaufwands. Wurden mögliche Transportmittel für das Frachtgut gefunden, so wird mit der Verhandlung über die Transportkonditionen mit jedem der zurückgelieferten Transportmittel begonnen. Wie in Abschnitt 5.1.1 beschrieben, basiert diese Verhandlung auf einer reversen Vickery Auktion. Realisiert wurde dieser Auktionsmechanismus unter Zuhilfenahme des Contract Net Protokolls. Dieses Protokoll wurde von Smith [Smi80] entwickelt und ist ein Verhandlungsprotokoll, das zur Lösung des Zuweisungsproblems bei der Arbeitsteilung eingesetzt werden kann. Gleichzeitig soll eine domainspezifische Kostenfunktion optimiert werden. Dazu legt dieses Protokoll den Inhalt und den Ablauf der Zusammenarbeit fest. Grundsätzlich sind hierbei zwei Arten von Agenten definiert: ein Initiator und die Teilnehmer. Die Ausgangssituation ist in Abbildung 6.7(a) dargestellt. Der Initiator der Verhandlung (grün gekennzeichnet) hat ein Problem beziehungsweise eine Aufgabe, welche auf mehrere Teilnehmer (blau dargestellt) aufgeteilt werden soll. Übertragen auf den konkreten Anwendungsfall ist der Initiator der Transportagent des Frachtguts und die über die Suche des Gelbe-Seiten-Dienstes zurückgelieferten Transportmittel stellen die Teilnehmer dar. Im ersten Schritt (Abbildung 6.7(b)) sendet der Initiator eine Ausschreibung des zu lösenden Problems an alle Teilnehmer. In diesem Fall wird der Transportauftrag des Frachtguts ausgeschrieben und die Auktion somit gestartet. Daraufhin wird die Ausschreibung von den Teilnehmern ausgewertet.

Ist ein Teilnehmer an der Übernahme der Aufgabe interessiert und in der Lage, diese auch zu erfüllen, wird ein Gebot an den Initiator zurückgesandt (Abbildung 6.7(c)). Angewendet auf die Transportauftragsvergabe müssen die Transportmittel in diesem Schritt die Kosten berechnen, welche ihnen durch die Übernahme des Austrags entstehen würden. Diese Kosten werden dann als bindendes Gebot an den Transportagenten gesendet. Im dritten und letzten Schritt, welcher in Abbildung 6.7(d) dargestellt ist, wertet der Initiator die bei ihm eingegangenen Gebote der Teilnehmer aus und wählt das Kostengünstigste aus. Der Teilnehmer, welcher dieses Gebot abgegeben hat, wird über den Gewinn der Ausschreibung informiert; alle anderen Teilnehmer bekommen eine Nachricht, dass sie

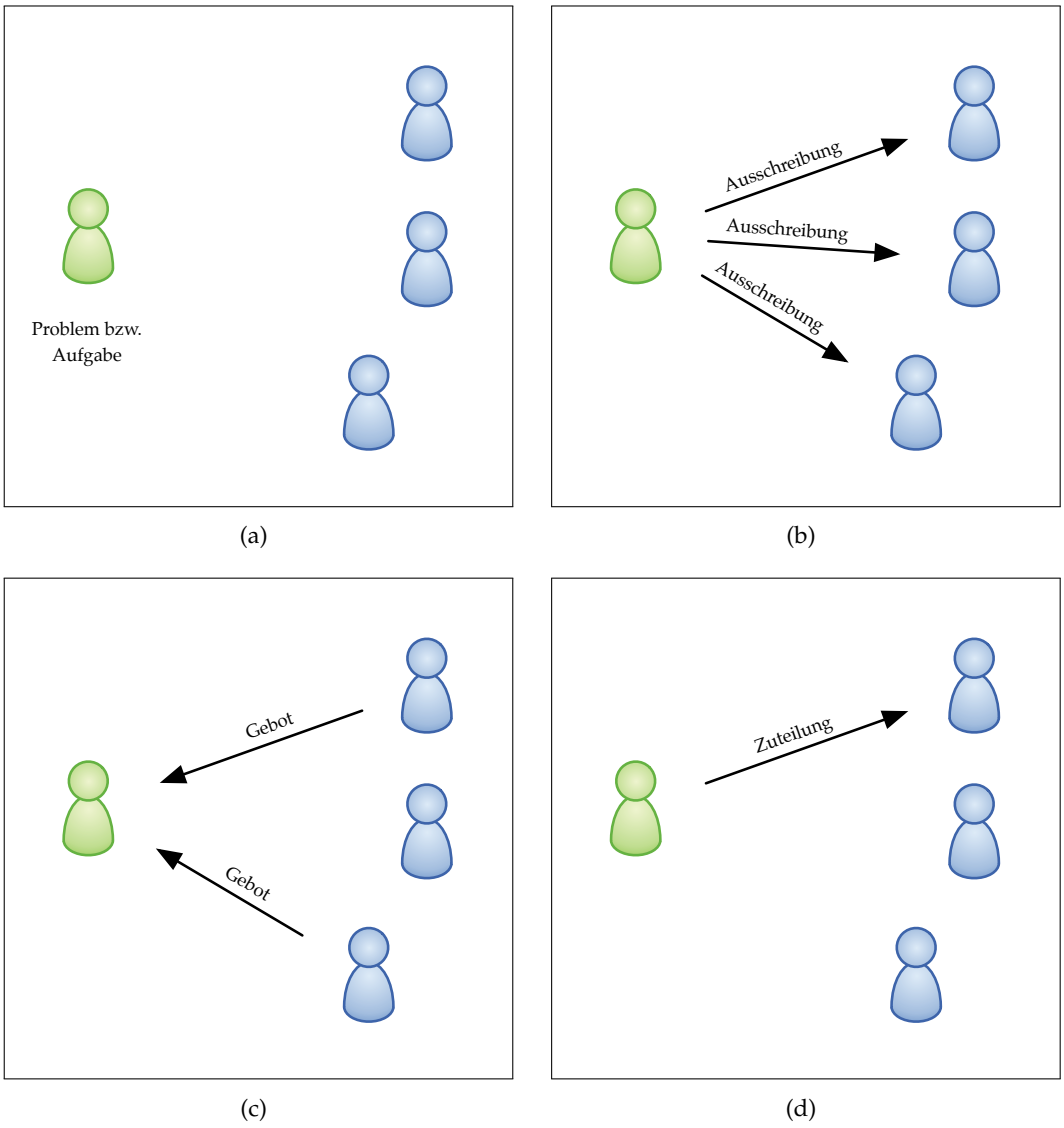

Abbildung 6.7: Die vier Phasen des Contract Net Protokolls: (a) Ausgangssituation; (b) Ausschreibung des Problems; (c) Abgabe der Gebote durch Teilnehmer; (d) Zuweisung der Aufgabe

diese Ausschreibung nicht gewonnen haben. Bezugnehmend auf das zu realisierende Fallbeispiel stellt dieser Schritt den Abschluss der Auktion dar. Das Transportmittel, welches das geringste Gebot abgegeben hat, gewinnt somit die Auktion und kann den Transport des Frachtguts übernehmen.

Die FIPA hat das Contract Net Protokoll aufgegriffen und in leicht abgewandelter Form in [Fou02c] standardisiert. Das Sequenzdiagramm des FIPA Contract Net Interaction

Protokolls ist in Abbildung 6.8 zu sehen. Dabei wurden die Typen der Sprechakte entsprechend des dafür gültigen Standards [Fou02b] festgelegt. Hinzugefügt wurde außerdem eine Rückmeldung des Teilnehmers, welcher die Ausschreibung gewonnen hat, an den Initiator über den Verlauf der Aufgabenausführung. So kann ein aufgetretener Fehler ebenso wie eine Informationsmeldung über die erfolgte Durchführung oder das Ergebnis der Durchführung der Aufgabe in Form einer ACL-Nachricht an den Initiator gesendet werden.
Da das Transportmittel in diesem Fall als Initiator agiert, wurde es innerhalb eines `ContractNetInitiator`-Verhaltens implementiert.

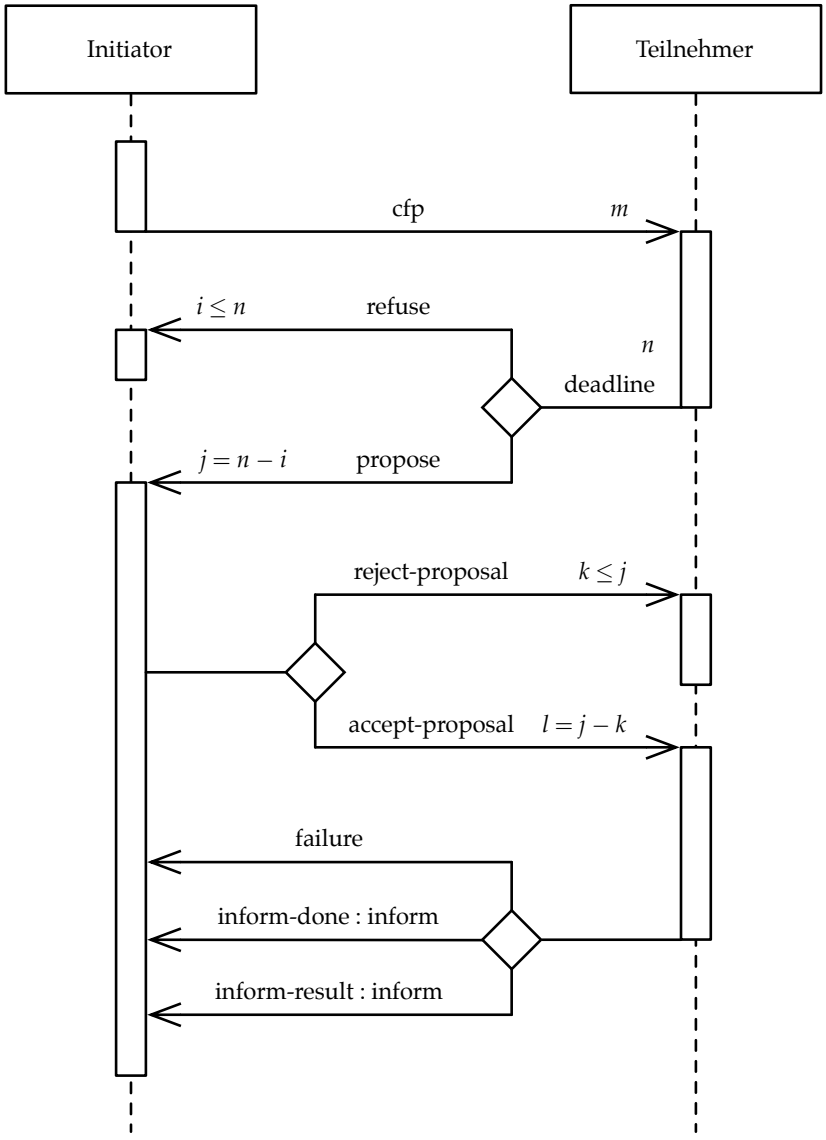

Abbildung 6.8: FIPA Contract Net Interaction Protocol nach [Fou02c]

6.5 Implementierung des Transportmittelagenten

Die Realisierung des Transportmittelagenten kann thematisch in fünf Teile gegliedert werden:

- Registrierung der eigenen Dienste im Transportmittelverzeichnis
- Antworten auf Ausschreibungen von Transportagenten
- Suche nach Transportmitteln zur Kooperation
- Fungieren als Initiator und Teilnehmer an Kooperationen mit weiteren Transportmittelagenten
- Simulation des Fahrens

Von diesen Teilen sind die vier zuletzt genannten als Verhalten modelliert, während die Registrierung der eigenen Dienste als Methode implementiert ist, die während des Initialisierens in der `setup()`-Methode aufgerufen wird. Dieser Registrierungsprozess ist in Codebeispiel 6.3 dargestellt.

```
DFAgentDescription dfAgentDescription = new DFAgentDescription();
dfAgentDescription.setName(getAID());

ServiceDescription serviceDescription = new ServiceDescription();
serviceDescription.setName("Cargo_Transport");
serviceDescription.setType("Transport");
serviceDescription.addOntologies(TransportOntology.getInstance().getName());
serviceDescription.addLanguages(new SLCodec());
serviceDescription.addProperties(new Property("type", "Reefer_Cargo"));
serviceDescription.addProperties(new Property("company", "Company"));

dfAgentDescription.addServices(sd);
DFService.register(this, dfAgentDescription);
```

Codebeispiel 6.3: Registrierung eines Transportmittel im Transportmittelverzeichnis

Zuerst wird ein neue Agentenbeschreibung `dfAgentDescription` erzeugt. Diese wird dem erstellenden Agenten als Agenten-Identifikator übergeben. Daraufhin wird der eigentliche Dienst, welchen der Transportmittelagent anbietet, durch Setzen von Parametern einer Dienstbeschreibung `serviceDescription` spezifiziert. Unter anderem werden dabei auch die Ontologie sowie der Codec angegeben, welche dem Agenten bekannt sind.

Nachdem alle notwendigen Parameter gesetzt sind, wird der Dienst zur Agentenbeschreibung hinzugefügt und abschließend beim Gelbe-Seiten-Dienst der Plattform (`DFService`) registriert.

Das Antworten auf Ausschreibungen von Transportmitteln erfolgt im Gegenpart zu dem dort implementierten `ContractNetInitiator`-Verhalten, einem `ContractNetResponder`-Verhalten. Trifft solch eine Ausschreibung als Inhalt einer ACL-Nachricht beim Transportmittelagenten ein, so werden die Kosten für eine eventuelle Übernahme des Transports mittels einer Anfrage an den Routingalgorithmus berechnet. Diese Ergebnisse werden in Form eines Tickets, welches eine Instanz der `TransportTicket` ist, an den anfragenden Transportagenten zurückgesendet und dient damit zugleich als Gebot für die Auktion, in der der Transportauftrag versteigert wird. Zu den in einem Ticket übertragenen Daten gehören die Höhe des Gebots, die Abholstation und die Auslieferungsstation, der Abholzeitpunkt und der Auslieferungszeitpunkt sowie der Typ der Ware, welcher transportiert werden soll. Darüber hinaus wird für jedes Ticket ein eindeutiger Identifikator unter Zuhilfenahme der Klasse `java.util.UUID` erzeugt. Dieser dient als Schlüssel zur Speicherung der Tickets in einer lokalen Map. Dadurch muss der Transportagent in der Benachrichtigung über den Ausgang der Auktion nur noch den Schlüssel an das Transportmittel senden. Somit kann der Nachrichten-Overhead verringert und die Kommunikation effizienter gestaltet werden. Eine gleichzeitige Teilnahme an mehreren Auktionen ist nicht vorgesehen, da durch den zwischenzeitlichen Gewinn einer anderen Auktion die Gültigkeit der abgegebenen Gebote und den diesen zu Grunde liegenden Routenberechnungen nicht mehr gewährleistet werden könnte. Aufgrund dieser Voraussetzung kann der Transportmittelagent im Falle des Gewinns einer Auktion seine aktuelle Route durch die zur Gebotsabgabe berechneten Route ersetzen. Ein Abgleich zwischen der aktuellen Route und der zum Zeitpunkt der Gebotsabgabe berechneten Route ist dennoch notwendig, da sichergestellt werden muss, dass in der Zeit zwischen Gebotsabgabe und der Benachrichtigung über den Gewinn derselbigen nicht schon weitere Stationen angefahren wurden. Sollte dieser Fall eingetreten sein, so muss die neue Route entsprechend angepasst werden. Bekommt das Transportmittel die Nachricht, dass es die Auktion nicht gewonnen hat, so löscht es den Eintrag aus der Map.

Die Suche nach Transportmittelagenten dient als Grundlage für eine darauf folgende Kooperation mit anderen Transportmittelagenten zum Zweck einer Frachtgutumverteilung. Eine Auswahl der für eine mögliche Kooperation in Frage kommenden Transportmittel erfolgt ebenfalls über den Gelbe-Seiten-Dienst, der von Jade zur Verfügung gestellt wird. Dieser bietet die Möglichkeit, durch das Setzen von `SearchConstraints` die Suche einzugrenzen und zu spezifizieren. Als Ergebnis werden die Agenten-Identifikatoren von den

Transportmitteln zurückgeliefert, welche den Suchparametern entsprechen. Über diese Identifikatoren ist es dem suchenden Agenten dann möglich, eine Kommunikation in Form von ACL-Nachrichten zu beginnen.

Im Transportmittelagenten spiegelt sich der Ansatz des Contract Net Protokolls wider, dass die Agenten sowohl die Rolle des Initiators als auch die Rolle des Teilnehmers einnehmen können. Die Ausschreibung wird, wie beim Transportagenten auch, in einem `ContractNetInitiator`-Verhalten implementiert, während ein Reagieren auf solche Ausschreibungen durch ein `ContractNetResponder`-Verhalten ermöglicht wird. Für die Implementierung der Kooperation von Transportmitteln untereinander wurde das in Kapitel 5.2.2 beschriebene Ein-Schritt-Protokoll gewählt, da es die gleichen Vorteile besitzt wie das monotone Zugeständnis-Protokoll. Eingeleitet wird die Kooperation durch ein einmalig auszuführendes Verhalten, welches nach dem Ersteigern einer Ware ausgelöst wird. Innerhalb dieses Verhaltens wird eine Instanz der Klasse `ProposalRequest` erzeugt. Sie dient als Ausgangspunkt der Verhandlungen und beinhaltet die momentan geladenen Frachtgüter, den aktuellen Standort, die aktuellen Kosten, die Durchschnittsgeschwindigkeit sowie die vorhandene Maximalkapazität. Im Anschluss daran wird das eigentliche `ContractNetInitiator`-Verhalten hinzugefügt und ausgeführt. Die Ausschreibung erfolgt durch das Senden einer ACL-Nachricht an die im vorherigen Schritt gefundenen Transportmittelagenten. Somit werden die Verhandlungen parallel mit diesen Transportmitteln geführt. Ihnen zugrunde liegt jeweils das Ein-Schritt-Protokoll. Auf die Nachrichten wird, wie bereits erwähnt, durch ein `ContractNetResponder`-Verhalten reagiert. In diesem Verhalten werden alle möglichen Kombinationen von Frachtgutumverteilungen berechnet und ausgewertet. Bevor dieses geschieht muss jedoch ein Ort ausgewählt werden, an dem sich die Transportmittelagenten zum Zweck des Umladens treffen würden, da die Fahrt zu diesem Umschlagplatz oder Depot mit in die Kostenberechnungen einfließen muss. Als Lösung dieses Problems wird in dieser Arbeit eine Station angenommen, welche auf der kürzesten Verbindungsstrecke zwischen den aktuellen Standorten der verhandelnden Transportmittelagenten liegt. Diese Strecke wird mit Hilfe des Dijkstra-Algorithmus, welcher in [Dij59] beschrieben wird, berechnet. Die Auswahl der Station, an der sich die Transportmittel im Falle einer Kooperation treffen würden, ist in Abbildung 6.9 dargestellt. Befinden sich eine ungerade Anzahl an Stationen auf der Strecke zwischen den aktuellen Standpunkten der beiden Transportmittel (Abbildung 6.9(a)), so wird die Station in der numerischen Mitte, in diesem Fall S_2 ausgewählt. Liegt hingegen eine gerade Anzahl an Zwischenstationen vor, so wird eine der mittleren beiden per Zufall ausgewählt. Im Fall der in Abbildung 6.9(b) gezeigten Situation würden sowohl S_2 als auch S_3 zur Auswahl stehen. Nachdem die Verladestation ausgewählt wurde, erfolgt die eigentliche Berechnung der Kosten für alle möglichen Umverteilungskonstellationen. Durch die Zusendung

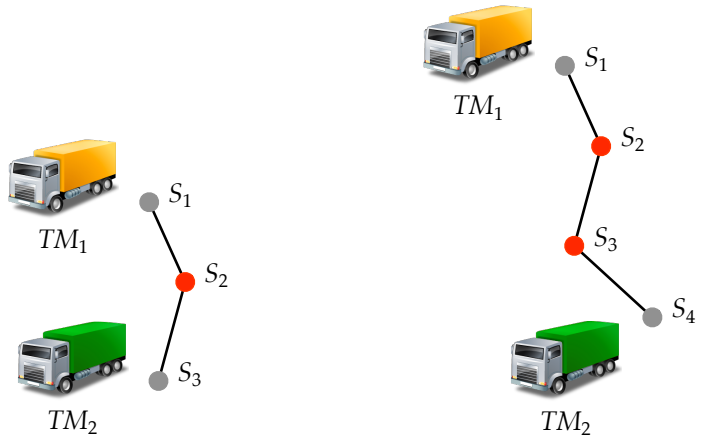

Abbildung 6.9: Auswahl des Umschlagplatzes: (a) bei einer ungeraden Anzahl an Zwischenstationen; (b) bei einer geraden Anzahl an Zwischenstationen

der Daten im `ProposalRequest` haben die Teilnehmer die Möglichkeit, sämtliche Kosten sowohl für sich selbst, als auch für den Initiator zu berechnen. Normalerweise würde jeder Agent seine eigenen Berechnungen durchführen, um nicht zuletzt den Kalkulationen anderer Agenten Glauben schenken zu müssen. In dieser Realisierung wurde jedoch auf eine zweimalige Berechnung verzichtet, da alle Agenten die gleichen Kostenfunktionen aufweisen und von einer bestehenden Vertrauensbasis ausgegangen wird. In einer realen Anwendung kann diese Basis jedoch nicht als gegeben angesehen werden. Sind die Kosten für jeden Agenten für jede Umverteilungsmöglichkeit der Frachtgüter berechnet worden, erfolgt die Auswertung. Der Teilnehmer wählt aus allen Ergebnissen dasjenige aus, welches der in Abschnitt 5.2.2 definierten Nash-Verhandlungslösung entspricht. Sollten mehrere Umverteilungsoptionen das gleiche Nash-Produkt ergeben, so wird jede dieser Optionen ausgewählt. Die so gefundenen Lösungen werden an den Initiator der Ausschreibung in Form eines `ProposalAnswer`-Objekts zurückgeschickt.
Abschließend hat der Transportmittelagent, der die Kooperation ausgeschrieben hat, die Aufgabe, den Transportmittelagenten und die Umverteilungsoption auszuwählen, welche das höchste Nash-Produkt aufweist. Sollte die Wahl auf einen Agenten fallen der mehrere Möglichkeiten zurückgesendet hat, entscheidet der Zufall über die durchzuführende Option. Daraufhin akzeptieren beide Agenten die Umverteilung der Frachtgüter und fahren zur ausgewählten Verladestation.

Das Fahren wird ebenfalls mittels eines Verhaltens simuliert. Dazu wird innerhalb einer Schleife über eine Liste, welche die anzufahrenden Stationen enthält, iteriert. In jedem

Schleifendurchlauf wird ein `WakerBehaviour` erzeugt, also ein Verhalten, welches einmalig nach Ablauf einer vorgegebenen Zeit ausgeführt wird. Diesem Verhalten wird als Auslösezeit die geplante Reisezeit bis zur aktuell anzufahrenden Station, welche vom Routingalgorithmus errechnete wurde, übergeben. Da die vom Routingalgorithmus errechneten Zeiten als Realzeiten ausgegeben werden, wurde zusätzlich ein Parameter eingeführt, der die Geschwindigkeit der Simulation steuern kann. Dazu wird die Reisezeit mittels Division durch diesen Parameter verkürzt. Theoretisch könnte sie, durch die Wahl eines Parameters zwischen 0 und 1 ebenso verlängert werden, allerdings ist dieses in der Simulation nicht erwünscht. Alle Berechnungen finden dabei auf Basis einer Referenzzeit statt. Wurde eine Station erreicht, wird sie aus der Liste der noch anzufahrenden Stationen gelöscht und der aktuelle Beladungszustand wird angepasst.

6.6 Durchführung von Tests

Zum Testen der Implementierung des Fallbeispiels wurden drei Transportmittelagenten sowie zwei Frachtgutagenten erzeugt. Die Transportmittelagenten wurden an zufällig aus der Menge an Orten, welche in Abbildung 6.10 zu sehen sind, ausgewählt.

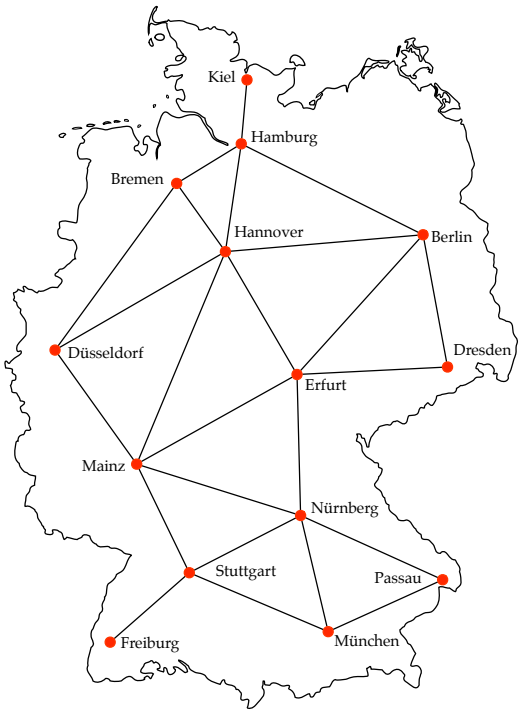

Abbildung 6.10: Für die Tests verwendetes Streckennetz

Das Streckennetz wurde mittels einer symmetrischen Distanzmatrix realisiert. In Tabelle 6.4 sind die für diese Matrix verwendeten Distanzen in Form einer Entfernungstabelle angegeben. Die Parameter, welche initial sowohl den Transportmittelagenten als auch den Frachtgutagenten zur Verfügung stehen müssen, wurden mit Hilfe einer Konfigurationsdatei an jeden erstellten Agenten übergeben. Zwei der drei Transportmittelagenten erhielten auf diese Weise ein bereits zum Systemstart geladenes Frachtgut vorgegeben. Ebenfalls beim Systemstart wurde ein Frachtgutagent erzeugt, welcher zum Transport bereit steht. Der zweite Frachtgutagent wurde erst gestartet, nachdem die Vergabe des Transportauftrags und eine anschließende Kooperation unter den Transportmittelagenten erfolgte. Das erste Frachtgut war somit auf dem Weg, wodurch der Verhandlungs- sowie der Kooperationsmechanismus getestet werden konnten.

	Berlin	Bremen	Dresden	Düsseldorf	Erfurt	Freiburg	Hamburg	Hannover	Kiel	Mainz	München	Nürnberg	Passau	Stuttgart
Berlin	0	∞	147	∞	208	∞	224	219	∞	∞	∞	∞	∞	∞
Bremen	∞	0	∞	227	∞	∞	82	93	∞	∞	∞	∞	∞	∞
Dresden	147	∞	0	∞	166	∞	∞	∞	∞	∞	∞	∞	∞	∞
Düsseldorf	∞	227	∞	0	∞	∞	∞	215	∞	152	∞	∞	∞	∞
Erfurt	208	∞	166	∞	0	∞	∞	156	∞	201	∞	153	∞	∞
Freiburg	∞	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	∞	∞	116
Hamburg	224	82	∞	∞	∞	∞	0	119	75	∞	∞	∞	∞	∞
Hannover	219	93	∞	215	156	∞	119	0	∞	252	∞	∞	∞	∞
Kiel	∞	∞	∞	∞	∞	∞	75	∞	0	∞	∞	∞	∞	∞
Mainz	∞	∞	∞	152	201	∞	∞	252	∞	0	∞	188	∞	133
München	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0	134	137	165
Nürnberg	∞	∞	∞	∞	153	∞	∞	∞	∞	188	134	0	175	138
Passau	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	137	175	0	∞
Stuttgart	∞	∞	∞	∞	∞	116	∞	∞	∞	133	165	138	∞	0

Tabelle 6.4: Entfernungstabelle des Streckennetzes

7 Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, ein autonomes Multiagentensystem zur Transportsteuerung zu entwerfen und anschließend beispielhaft zu implementieren. Zur Modellierung wurden Ansätze aus verschiedenen Forschungsbereichen verwendet. Den wichtigsten Bereich stellt die Spieltheorie dar. Ihr entnommen sind unter anderem die Ansätze und Strategien, die Zlotkin und Rosenschein für Vereinbarungen zwischen Agenten in verschiedenen Domänen verfolgen. Zudem wurde problemspezifisch beschrieben, wie Agenten mit fehlendem, verteiltem oder unsicherem Wissen umgehen können.

Die neben dem Modell durchgeführte Beispiel-Implementierung zeigt die konkrete Umsetzung der gefundenen Ansätze. Durch die an verschiedenen Punkten der Implementierung vorgenommenen Vereinfachungen und Abstraktionen ergibt sich die Möglichkeit, das Modell anhand weiterer Forschungen zu verfeinern und noch realistischer zu gestalten. Bei der Implementierung war die Umsetzung der Grundprinzipien am wichtigsten; die Performance nur zweitrangig. So beinhaltet dieser Bereich das größte Optimierungspotenzial. Weiterführende Arbeiten könnten sich mit der Verbesserung der Auswahlfilter für Verhandlungen über große Frachtgutmengen befassen. Ebenso wären Approximationen im Bereich der Tourenplanung von Vorteil, um die Kosten für das Einfügen oder Entfernen von Frachtgütern in beziehungsweise aus Routen abschätzen zu können. Die Verwendung von metaheuristischen Ansätzen für das Tourenplanungsproblem scheint jedoch sehr vielversprechend und effizient.

Die Verwendung von Jade als Agentenplattform erwies sich als gute Wahl, da aufgrund der guten Dokumentation ein schnelles Einarbeiten möglich war. Beim Einsatz von Ontologien muss jedoch beachtet werden, dass nicht alle von Java zur Verfügung gestellten Klassen benutzt werden können. Neben den Grundtypen werden Konstrukte des Jade-LEAP, der Laufzeitumgebung für mobile Endgeräte, sowie Daten unterstützt. Die Benutzung von Argumenten zur Konfiguration der Agenten ist nur eingeschränkt möglich, da hierzu durch Kommata separierte Zeichenketten verwendet werden. Eine Unterstützung von XML wäre an dieser Stelle wünschenswert.

Um die organisationale Sichtweise auf Agentensysteme weiterzuführen, wäre die Öffnung des Systems für Agenten unter dem Belief-Desire-Intention Modell aus [Bra87] interessant. Es beschreibt das praktische Denken von Menschen und diente als Grundlage für BDI-Agenten. *Beliefs* beschreiben hierbei die Annahmen über die Umwelt, während *desires* die Wünsche des Agenten widerspiegeln und *intentions* die festen Absichten darstellen. Ein Vertreter dieser BDI-Agentenplattformen ist das auf Jade basierende Jadex[1]. Durch die Möglichkeit, Jadex Agenten auf Jade als Middleware laufen zu lassen, müssten keine Anpassungen an das Modell beziehungsweise die Implementierung gemacht werden.

Für den Einsatz in einem realen Logistikumfeld wären jedoch Adaptionen und Erweiterungen der Agenten notwendig. Unter anderem müssten sie an die vorzufindende Infrastruktur angepasst werden. Beispielhaft sei hier der Überwachungsagent genannt, welcher die Daten der Hardware-Sensoren der Frachtgüter evaluiert. Ebenfalls müssten Konzepte erstellt werden, wie die Mobilität der Agenten in das System zu integrieren wäre. Einen Ansatz, wie die Migration der Agenten erfolgen könnte, gibt beispielsweise [BBG+06].

[1] `http://vsis-www.informatik.uni-hamburg.de/projects/jadex`

Literaturverzeichnis

[AD07] AUREN ; DÍAZ, Bernabé D.: *The VRP Web*. `http://neo.lcc.uma.es/radi-aeb/WebVRP`, März 2007

[Baa03] BAADER, Franz: *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, 2003. – ISBN 0521781760

[BBG+06] BEHRENS, C. ; BECKER, M. ; GEHRKE, J. D. ; JEDERMANN, R. ; GÖRG, C. ; HERZOG, O. ; LANG, W. ; LAUR, R.: Ein Multiagentensystem für Selbststeuerung in der Transportlogistik. In: *Fachtagungsberichte VDE Kongress 2006. Innovations for Europe*. Aachen : VDE Verlag, 2006, 29-34

[BCG07] BELLIFEMINE, Fabio L. ; CAIRE, Giovanni ; GREENWOOD, Dominic: *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons, 2007. – ISBN 0470057475

[BDDFS05] BORDINI, Rafael H. (Hrsg.) ; DASTANI, Mehdi (Hrsg.) ; DIX, Jürgen (Hrsg.) ; FALLAH-SEGHROUCHNI, Amal E. (Hrsg.): *Multiagent Systems, Artificial Societies, and Simulated Organizations*. Bd. 15: *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005. – ISBN 0–387–24568–5

[BME+07] BOOCH, Grady ; MAKSIMCHUK, Robert A. ; ENGEL, Michael W. ; YOUNG, Bobbi J. ; CONALLEN, Jim ; HOUSTON, Kelli A.: *Object-Oriented Analysis and Design with Applications (3rd Edition)*. Addison-Wesley Professional, 2007. – ISBN 020189551X

[Bra87] BRATMAN, M. E.: *Intention, Plans, and Practical Reason*. Cambridge, MA : Harvard University Press, 1987

[Bun03] BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK: *Kommunikations- und Informationstechnik 2010+3: Neue Trends und Entwicklungen in Technologien, Anwendungen und Sicherheit*. `http://www.bsi.bund.de/literat/studien/trend2010`, 2003

[BZW98] BRENNER, Walter ; ZARNEKOW, Rüdiger ; WITTIG, Hartmut: *Intelligente Softwareagenten: Grundlagen und Anwendungen*. Berlin : Springer, 1998

[CC04] CAIRE, Giovanni ; CABANILLAS, David: *JADE Tutorial: Application-defined content languages and ontologies*. `http://jade.tilab.com/doc/CLOntoSupport.pdf`, 11 2004

[CW64] CLARKE, G. ; WRIGHT, J. W.: Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. In: *Operations Research* 12 (1964), Nr. 4, S. 568–581

[DDFS04] DASTANI, Mehdi (Hrsg.) ; DIX, Jürgen (Hrsg.) ; FALLAH-SEGHROUCHNI, Amal E. (Hrsg.): *Programming Multi-Agent Systems, First International Workshop, PROMAS 2003, Melbourne, Australia, July 15, 2003, Selected Revised and Invited Papers*. Bd. *3067*. Springer, 2004 (Lecture Notes in Computer Science). – ISBN 3–540–22180–8

[Det03] DETTMANN, Kai: *Eine generische Online-Beschaffungsplattform: Anforderungsanalyse, objektorientierter Entwurf, Realisierung*, Universität Hamburg, Diplomarbeit, Januar 2003

[Dij59] DIJKSTRA, Edsger W.: A note on two problems in connexion with graphs. In: *Numerische Mathematik* 1 (1959), S. 269–271

[DNA08] DNA EVOLUTIONS: *Jopt.SDK - Vehicle Routing Software Library*. `http://www.dna-evolutions.com`, Juli 2008

[FHMV03] FAGIN, Ronald ; HALPERN, Joseph Y. ; MOSES, Yoram ; VARDI, Moshe Y.: *Reasoning about Knowledge*. The MIT Press, 2003

[FM96] FISCHER, Klaus ; MÜLLER, Jörg: A decision-theoretic model for cooperative transportation scheduling. In: *MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 1996. – ISBN 3–540–60852–4, S. 177–189

[FMPS95] FISCHER, Klaus ; MÜLLER, Jörg P. ; PISCHEL, Markus ; SCHIER, Darius: A Model for Cooperative Transportation Scheduling. In: *Proceedings of the First International Conference on Multiagent Systems.* Menlo park, California : AAAI Press / MIT Press, June 1995, 109–116

[Fou02a] FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS: *FIPA ACL Message Structure Specification*. `http://www.fipa.org/specs/fipa00061`, 12 2002

[Fou02b] FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS: *FIPA Communicative Act Library Specification*. `http://www.fipa.org/specs/fipa00037`, 12 2002

[Fou02c] FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS: *FIPA Contract Net Interaction Protocol Specification*. `http://www.fipa.org/specs/fipa00029`, 12 2002

[Fou02d] FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS: *FIPA SL Content Language Specification*. `http://www.fipa.org/specs/fipa00008`, 12 2002

[HI05] HOLLER, Manfred J. ; ILLING, Gerhard: *Einführung in die Spieltheorie*. Springer, 2005. – ISBN 354027880X

[Hol75] HOLLAND, John H.: *Adaptation in Natural and Artificial Systems*. Ann Arbor : The University of Michigan Press, 1975

[KCO04] KURUMATANI, Koichi (Hrsg.) ; CHEN, Shu-Heng (Hrsg.) ; OHUCHI, Azuma (Hrsg.): *Multi-Agent for Mass User Support, International Workshop, MAMUS 2003 Acapulco, Mexico, August 10, 2003 Revised and Invited Papers*. Bd. *3012*. Springer, 2004 (Lecture Notes in Computer Science). – ISBN 3–540–21940–4

[KK01] KORKMAZ, Turgay ; KRUNZ, Marwan: Multi-Constrained Optimal Path Selection. In: *INFOCOM*, 2001, 834-843

[Klu99] KLUSCH, Matthias: *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 1999. – ISBN 3540651128

[McC97] MCCLURE, Carma: *Software reuse techniques: adding reuse to the system development process*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1997. – ISBN 0–13–661000–5

[Nas50] NASH, John: The bargaining problem. In: *Econometrica* 18 (1950), April, Nr. 2, S. 155–162

[Nas53] NASH, John: Two person cooperative games. In: *Econometrica* 21 (1953), January, Nr. 1, S. 128–140

[NG] NIEDERBERGER, Christoph ; GROSS, Markus H.: *Towards a Game Agent*. `http://citeseer.ist.psu.edu/niederberger02towards.html`

[OPB01] ODELL, James J. ; PARUNAK ; BAUER, Bernhard: Representing Agent Interaction Protocols in UML. In: *First International Workshop, AOSE 2000 on Agent-Oriented Software Engineering*, Springer-Verlag New York, Inc., 2001, S. 121–140

[RG95] RAO, A. S. ; GEORGEFF, M. P.: *BDI-agents: from theory to practice*. `http://citeseer.ist.psu.edu/rao95bdi.html`. Version: 1995

[RN02] RUSSELL, Stuart J. ; NORVIG, Peter: *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002. – ISBN 0137903952

[Ros08] ROSENFELD, Maurice: *Übersicht aktueller Agentenplattformen*. Juli 2008

[RZ94] ROSENSCHEIN, Jeffrey S. ; ZLOTKIN, Gilad: *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. Cambridge, Massachusetts : MIT Press, 1994

[San93] SANDHOLM, T.: An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In: *Eleventh National Conference on Artificial Intelligence* (1993), January, 256-262. `http://mas.cs.umass.edu/paper/66`

[San98] SANDHOLM, T.: *Contract types for satisficing task allocation: I theoretical results*. `http://citeseer.ist.psu.edu/sandholm98contract.html`. Version: 1998

[SKMT03] SCHILLO, Michael (Hrsg.) ; KLUSCH, Matthias (Hrsg.) ; MÜLLER, Jörg P. (Hrsg.) ; TIANFIELD, Huaglory (Hrsg.): *Multiagent System Technologies, First German Conference, MATES 2003, Erfurt, Germany, September 22-25, 2003, Proceedings*. Bd. *2831*. Springer, 2003 (Lecture Notes in Computer Science). – ISBN 3–540–20124–6

[SL95] SANDHOLM, Tuomas ; LESSER, Victor: Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In: LESSER, Victor (Hrsg.): *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*. San Francisco, CA, USA : The MIT Press: Cambridge, MA, USA, 1995, 328–335

[Smi80] SMITH, R. G.: The contract net protocol: High-level communication and control in a distributed problem solver. In: *IEEE Transactions on Computers* 29 (1980), Nr. 12, S. 1104–1113

[Wei99] WEISS, Gerhard: *Multiagent systems. A modern approach to distributed artificial intelligence*. The MIT Press, 1999. – ISBN 0–262–23203–0

[WJ95] WOOLDRIDGE, Michael ; JENNINGS, Nicholas R.: Intelligent Agents: Theory and Practice. In: *Knowledge Engineering Review* 10 (1995), Nr. 2, 115-152. `http://citeseer.ist.psu.edu/97055.html`

[WJ05] WEISS, Gerhard ; JAKOB, Ralf: *Agentenorientierte Softwareentwicklung: Methoden und Tools*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2005. – ISBN 3540000623

[Woo01] WOOLDRIDGE, Michael: *Introduction to Multiagent Systems*. New York, NY, USA : John Wiley & Sons, Inc., 2001. – ISBN 047149691X

[WP04] WINIKOFF, Michael ; PADGHAM, Lin: *Developing Intelligent Agent Systems: A Practical Guide*. New York, NY, USA : Halsted Press, 2004. – ISBN 0470861207

[ZR94] ZLOTKIN, Gilad ; ROSENSCHEIN, Jeffrey S.: Designing Conventions for Automated Negotiation. In: *Artificial Intelligence* 15 (1994), Nr. 3, S. 29–46

[ZR96a] ZLOTKIN, Gilad ; ROSENSCHEIN, Jeffrey S.: Compromise in Negotiation: Exploiting Worth Functions over States. In: *Artificial Intelligence* 84 (1996), Nr. 1-2, 151-176. `http://citeseer.ist.psu.edu/622909.html`

[ZR96b] ZLOTKIN, Gilad ; ROSENSCHEIN, Jeffrey S.: Mechanism Design for Automated Negotiation, and its Application to Task Oriented Domains. In: *Artificial Intelligence* 86 (1996), Nr. 2, 195-244. `http://citeseer.ist.psu.edu/34076.html`

[ZR96c] ZLOTKIN, Gilad ; ROSENSCHEIN, Jeffrey S.: Mechanisms for Automated Negotiation in State Oriented Domains. In: *Journal of Artificial Intelligence Research* 5 (1996), 163-238. `http://citeseer.ist.psu.edu/zlotkin96mechanisms.html`

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

A Struktur der Begleit-CD

In der folgenden Abbildung ist die Verzeichnisstruktur der beiliegenden CD-ROM sichtbar. Im Stamm-Verzeichnis befindet sich eine Datei mit dem Namen „Liesmich.txt“mit Hinweisen zu den auf der CD befindlichen Dateien.

Abbildung 7.1: Verzeichnisstruktur der Begleit-CD

B CD-ROM