Flexible Kooperation zwischen Autonomen Agenten in Dynamischen Umgebungen

Dissertation

zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes von

Dipl.-Inform. Andreas Gerber

Saarbrücken, 11.Mai.2004

Kolloquium	8.Februar.2005
Vorsitz	Prof. Dr.-Ing. P. Slusallek, Universität des Saarlandes
Gutachter	Prof. Dr. (PhD) J. Siekmann, Universität des Saarlandes
	Prof. Dr. W. Wahlster, Universität des Saarlandes
Beisitzer	Dr. M. Klusch, Deutsches Forschungszentrum für Künstliche Intelligenz
Dekan	Prof. Dr. J. Eschmeier, Dekan der NTF1, Universität des Saarlandes

Meinen Eltern gewidmet.

Kurzzusammenfassung

Die vorliegende Dissertation präsentiert ein Schema zur Holonenformierung in dynamischen Umgebungen eines holonischen Multiagenten-Systems. Holone werden dazu eingesetzt, um eine Gruppe von Agenten für einen begrenzten Zeitraum zu einem strukturierten Verbund zusammenzufassen. Unter der Führung eines ausgezeichneten Agenten, dem Holonenführer bzw. Kopf des Holons, können gemeinsam komplexe Aufgaben gelöst werden. Die Holonenführer, wie auch die restlichen Agenten eines Multiagenten-Systems, besitzen dabei nur ein eingeschränktes, vages und zum Teil fehlerhaftes Wissen über die anderen Agenten. Das Schema beschreibt, wie jeder Agent sein Wissen erlernen und davon ausgehend potentielle Holonenstrukturen simulationsbasiert aufbauen kann. Anschließend werden diese hypothetischen Holonenstrukturen in Verhandlungen realisiert. Zur Umsetzung des Schemas und der daraus resultierenden Algorithmen wird ein mathematischer Formalismus zur Abbildung eines Multiagenten-Systems auf einen Vektorraum vorgestellt, der es ermöglicht, die Entitäten des Multiagenten-Systems und die Funktionen des Schemas formal zu beschreiben. Aufbauend auf diesen Arbeiten und des Schemas werden abschließend vier Algorithmen, von einem einfachen randomisierten Verfahren bis hin zu komplexen wissensbasierten Verfahren zur Holonenformierung, exemplarisch entwickelt, diskutiert und evaluiert.

Abstract

This work proposes a scheme for the formation of holonic agents in a multiagent-system acting in a dynamic environment. This model captures cooperative holonic multiagent-systems in which each agent has incomplete, vague and erroneous information about the other agents and about its dynamic and uncertain world. Holons are temporarily formed to fulfill complex tasks, that a single agent cannot handle alone. To find assistance the head of a holon continuously tries to improve its holonic structure. Therefore it builds a set of hypothetical holons, rating them and if a higher ranked holon structure than the current one is found, it starts with the (re-)negotiation of the holon. Also if the holon is for some reason not longer capable to achieve its goals, re-negotiation starts. The scheme describes in particular, how an agent learns the properties of other agents. Based on this knowledge the agents are then able to build hypothetical holons by simulations. In the following these hypothetical holons are realised by bilateral negotiations with the agents of that pre-computed holonstructure. For describing this scheme and the algorithms based on this scheme, I first present a formalism to map a multiagent-system onto a vector space, such that it is possible to give a formal description of the entities of a multiagenten-system and of the functions of the proposed scheme. Finally I present four algorithms to form dynamically holons based on that scheme. These algorithms consist of a simple randomised agent selection to a complex knowledge-based selection of relevant agents to solve the given tasks. These algorithms are in the end discussed and evaluated in detail.

Zusammenfassung

Kooperationen zwischen Agenten und Gruppen von Agenten sind wesentliche Mechanismen in benevolenten Multiagenten-Systemen, um komplexe Aufgaben zu lösen. Jedoch führt die Annahme, dass die Agentengesellschaft nicht notwendigerweise benevolent ist, zu Problemen während der Kooperation. Kooperationen in dynamischen Umgebungen, in denen Störungen (Ausfall von Ressourcen, Netzwerk, Korrumption von Daten, etc.) jederzeit auftreten können, erschweren zudem den koordinativen Prozess. Systeme, die in solch dynamischen Umgebungen eingesetzt werden, müssen mit vagem und unvollständigem Wissen umgehen können. Zur Modellierung dynamischer Umgebungen eignen sich Multiagenten-Systeme in besonderer Weise, da sie diesbezüglich mehrere Besonderheiten besitzen: Multiagenten-Systeme sind in der Regel *offen* konzipiert, so dass zur Laufzeit neue Softwarekomponenten (wie zum Beispiel Agenten) in das System integriert, aber auch bestehende Einheiten aus dem System entnommen werden können. Weiterhin werden Multiagenten-Systeme entwickelt, um permanente Kontrolle auszuführen. Die Offenheit und das dynamische Auftreten von Störeffekten resultieren in Veränderungen der Systemumgebung. So können die berechneten Aktionspläne der Agenten so stark gestört werden, dass sie gegebenenfalls nicht mehr korrekt ausführbar sind. Die Erhaltung von Effizienz hinsichtlich der Ausführbarkeit der Agentenpläne ist daher ein wichtiges Thema und die Voraussetzung für den erfolgreichen Einsatz von Multiagenten-Systemen.

Ziel dieser Dissertation ist es, ein Verfahren zu entwickeln, das es Agenten eines Multiagenten-Systems erlaubt, in einer nicht-benevolenten, dynamischen Umgebung stabile Kooperationen mit anderen Agenten einzugehen, um komplexe Aufgaben in der Gemeinschaft zu lösen. Zur Bildung von Kooperationen werden holonische Multiagenten-Systeme eingesetzt. Die Anforderung an ein solches System ist, dass es sich trotz der Einwirkung von Störungen flexibel und adaptiv den aktuellen Gegebenheiten anpassen kann. Aufbauend auf einem Formalismus zur Beschreibung von Arbeitsabläufen, wird ein Schema entwickelt, das es den Agenten ermöglicht, sich flexibel und eigenständig den Veränderungen in ihrer Umgebung anzupassen. Arbeitsabläufe dienen dabei als strukturelle Grundlage für die Bildung von Holonen. Eine Analyse aktueller Fragestellungen im Bereich Logistik zeigt einen verstärkten Bedarf an solchen adaptiven Systemen. Dieser Trend lässt sich an den Systemen TeleTruck, CASA und AGRICOLA.NET veranschaulichen, an denen ich seit Beginn meiner Promotion gearbeitet habe. Damit eine Planung und Koordination von unterschiedlichen Abläufen realisiert werden kann, muss eine Vorgehensweise zur Abbildung von Arbeitsabläufen entwickelt werden. Die Verwendung von Arbeitsabläufen ermöglicht eine Definition einer logischen, sowie chronologischen Abfolge von komplexen Arbeitsabläufen. Für die Bearbeitung solcher Abläufe werden spezielle Agenten eingesetzt, die die Verwaltung, Zuweisung von Aufgaben auf ausführende Agenten, sowie die Planüberwachung durchführen. Für die Umsetzung dieser Vorgehensweise durch ein Multiagenten-System wird zunächst ein geeigneter Formalismus zur Beschreibung von Aufgaben, Diensten, Agenten und Holonen angegeben. Dieser dient

dann als Grundlage für die Entwicklung eines Verfahrens zur automatisierten Planung und Koordination.

Aus der Definition der Agenten und einer geeigneten Verknüpfungsvorschrift wird abgeleitet, dass unter bestimmten Voraussetzungen einem Multiagenten-System Vektorcharakter zugesprochen werden kann. Somit kann die gesamte Theorie der Vektorrechnung zur weiteren Bearbeitung verwendet werden. Verfahren interdisziplinärer Forschungsrichtungen, wie z.B. Data-Mining, die auf der Vektorrechnung aufbauen, können so in Multiagenten-Systemen eingesetzt werden. Beispielsweise kann unter Verwendung einer Metrik der Abstand zwischen gegebenen Agentenvektoren und einem Aufgabenvektor berechnet werden, um so den nächstliegenden (am meisten geeigneten) Agenten zu ermitteln, der die Aufgabe am besten lösen kann. Existiert eine Abbildungsvorschrift zweier Agenten auf je einen Vektor und kann eine Verknüpfungsvorschrift angegeben werden, dann ist es möglich, Agenten auch anhand einer Menge von Attributen zu vergleichen und Gruppen von Agenten und Holone auf Vektoren abzubilden. Zusätzlich werden Lernverfahren eingesetzt, um sich den Auswirkungen einer dynamischen Umgebung anzupassen. Darauf aufbauend gebe ich ein Schema zur Holonenformierung in dynamischen Umgebungen an.

Das Schema beschreibt, wie die Agenten lokales Wissen über die Gesellschaft erwerben, das als Grundlage für die Agentenauswahl dient. Dieses Wissen dient der Selektion von Agenten zur Bearbeitung einzelner Aufgaben. Zur Effizienzsteigerung sieht das Schema optional einen globalen Nachrichtendienst vor, der den Agenten eine allgemeine Einschätzung der restlichen Agenten anbietet. Der Nachrichtendienst erwirbt wiederum Wissen durch die Bewertungen der Agenten bezüglich der restlichen Agenten. Agenten können mit Hilfe dieses Dienstes zusätzlich Informationen über potentielle Agenten erwerben, wodurch das Risiko einer Fehleinschätzung sinkt. Der Nachrichtendienst bildet die Basis für verteiltes Lernen über die Agenteneigenschaften. Bevor jedoch konkrete Verhandlungen zur Bildung von Holonen für die gemeinschaftliche Bearbeitung komplexer Aufgaben durchgeführt werden, wird durch wiederholtes Simulieren verschiedener Holonenstrukturen die momentan 'beste' Struktur ermittelt. Anschließend instanziiere ich aus diesem Schema exemplarisch vier Algorithmen zur Holonenformierung, die von randomisierten Ansätzen bis hin zu Algorithmen mit einer detaillierten Analyse und Pflege der Wissensbasis eines Agenten reichen. Anschließend habe ich Testläufe durchgeführt und ausgewertet, um Aussagen über die Stabilität von Agentengesellschaften und die Effizienz des Verfahrens treffen zu können. Eine Analyse der Testergebnisse zeigt schließlich, in welchem Verhältnis der Aufwand der Pflege einer Wissensbasis zur Reduktion der ausgeführten Verhandlungen in Bezug auf die Größe der Agentengesellschaft und Aufgabenvielfalt steht. Zusammenfassend baue ich einen Formalismus auf, der als Grundlage für die Entwicklung von Algorithmen dienen kann, um das Problem der Holonenformierung in dynamischen und offenen Umgebungen lösen zu können.

Veröffentlichungen

Die generische Erweiterung der InteRRaP-Architektur, sowie der Aufbau holonischer Multiagentenarchitekturen für logistische Anwendungen, der Abschnitte 2.1.8 und 2.2.1 basieren auf den Arbeiten [GVZ00], [GKK02] und [GR00].

Forschungsergebnisse für den Einsatz von holonischen Multiagentensystemen zur Integration von mobilen Diensten in ein agentenbasiertes Informations- und Handelsnetzwerk wurden in mehreren Veröffentlichungen [GR01b], [KBF+02], [GRK02] präsentiert. Die Anwendung dieser Techniken im Projekt CASA wurde in [GK01], [KG01] und [GK02] beschrieben.

Die Grundlage für die in Kapitel 4 beschriebene Methodik zur Abbildung von komplexen Arbeitsabläufen auf holonische Multiagenten-Systeme stellt die Publikation [GR01c] dar. Der daraufbasierende Formalismus aus Kapitel 5 zur Abbildung eines Multiagenten-Systems auf einen Vektorraum wurde in Projekt AGRICOLA erfolgreich eingesetzt.

Koordinations- und Kooperationsmechanismen für holonische Multiagenten-Systeme, die zum Einsatz in dynamischen Umgebungen geeignet sind, wie sie Kapitel 6 beschrieben sind, wurden in [GKRZ01], [GR01a] und [GRK03] veröffentlicht.

Das Schema zur Bildung von Verbünden in dynamischen, nicht-benevolenten Umgebungen, das in Abschnitt 7.2 vorgestellt und diskutiert wird, basiert auf den Arbeiten [KG02c], [KG02b], [KG02a] und [GK03]. Die Beschreibung einer Instanz des Schemas, sowie die Auswertung der Testergebnisse in der AGRICOLA-Anwendungsdomäne wurden in [GK04] veröffentlicht.

Danksagung

An erster Stelle möchte ich mich bei Herrn Prof. Dr. Siekmann nicht nur für die Unterstützung während meiner Forschungsarbeit am DFKI bedanken, sondern auch für die Begeisterung an dem Thema Künstliche Intelligenz und die Führung meiner wissenschaftlichen Laufbahn seit Beginn meines Hauptstudiums vor nun über neun Jahren.

Besonders bedanke ich mich bei meinem Betreuer Herrn Dr. Matthias Klusch für die vielen hilfreichen Ratschläge und die wissenschaftliche Führung, die meine Arbeitsweise stark geprägt hat. Ebenfalls bedanke ich mich bei Herrn Dr. Klaus Fischer für die Möglichkeit in einer solch erfolgreichen und international anerkannten Forschungsgruppe arbeiten zu dürfen.

Ich möchte auch all meinen Kollegen der Multiagenten-System Gruppe am DFKI für die fruchtbare Forschungsumgebung danken. Ein ganz besondere Dank hierbei gilt einem sehr guten Freund und Zimmerkollegen Herrn Christian Ruß für die konstruktive Zusammenarbeit, Motivation und für das sehr harmonisches Arbeitsklima in den letzten vier Jahren. Ebenso herzlich möchte ich mich bei meinem Kollegen Herrn Dr. Helmut Lohmann bedanken, mit dem die Zusammenarbeit in den Projekten CASA und AGRICOLA immer eine besondere Freude war. Auch habe ich in vielerlei Hinsicht sehr viel von ihm gelernt.

Ganz besonders möchte ich mich bei meinem Bruder Dr. Christian Gerber bedanken, dafür dass er immer für mich uneingeschränkt zur Verfügung stand und ich immer ein großes Vorbild in ihm hatte.

Den größten Dank schulde ich meinen Eltern, ohne deren großartige Unterstützung und Verständnis ich mich nicht in dieser Weise meinem Studium und meiner Promotion hätte widmen können. Worte können nicht meine Dankbarkeit ausdrücken.

Saarbrücken, 11.Mai.2004 Dipl.Inform. Andreas Gerber

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

In *E-Commerce* Anwendungen wachsen die Anforderungen an Flexibilität und Dynamik der computergestützten Systeme. Durch die Modellierung der operativen Einheiten eines Betriebs bzw. der Teilnehmer eines Unternehmensnetzwerks durch intelligente, autonome Agenten, kann man diesen Anforderungen leichter gerecht werden. Der Einsatz von spezialisierten Agenten, die komplexe Aufgaben eigenständig koordinieren können, verleiht einem Multiagenten-System mehr Flexibilität, Effizienz und die Möglichkeit der Anpassung, insbesondere bei verteilten Systemen. Die Agenten müssen dazu jederzeit in der Lage sein, miteinander zu kooperieren, obgleich sie sich vielleicht gerade in Verhandlungen befinden oder Aufgaben ausführen. Die Kooperation und das Teilen von Ressourcen sind bei der Erzeugung von temporären (stabilen) Verbünden autonomer, eigennütziger und beschränkt rationaler Agenten von großer Bedeutung für die Ausführung von komplexen Aufgaben, die die Möglichkeiten einzelner Agenten überschreiten. Zudem können die Agenten durch die Kooperation Kosten einsparen und so ihren Gewinn steigern.

Zur Bildung von solchen Zweckgemeinschaften berechnet jeder Agent den Nutzen seiner Aktionen und seiner Ressourcen in Relation zu anderen Agenten unter Verwendung einer individuell angepassten Nutzenfunktion. Analog ermittelt jeder Führer eines Verbunds seinerseits den Nutzen seiner Gemeinschaft und die Auswirkungen der Kooperation mit anderen Agenten und Verbünden, basierend auf einer eigenen Nutzenfunktion. Als Grundlage für solche Berechnungen dient den Agenten einerseits globales Wissen über die Agentengesellschaft und andererseits lokales Wissen, das sie während der Ausführung von Aktionen über die anderen Agenten erworben haben. Eine Schwierigkeit bei der Bildung von Verbünden liegt in den Voraussetzungen der Umgebung. Handelt es sich um eine geschlossene Welt mit benevolenten Agenten, dann können die Agenten *offen* miteinander verhandeln und ihre Eigenschaften, Pläne und Ziele den anderen Agenten offen legen. Somit ist es möglich, dass ein ausgezeichneter Agent eine global optimale Lösung berechnen kann. Jedoch sind solche Voraussetzungen in der Realität nur selten anzutreffen. Stattdessen überwiegen Szenarien, in denen (teil-)konkurrierende Agenten miteinander kooperieren müssen, um ihre Ziele zu erreichen. Das heißt, die Annahme einer benevolenten Gesellschaft trifft hier nicht zu und Verfahren, die für benevolente Gesellschaften entwickelt wurden,

können nicht oder nur eingeschränkt mit Modifikationen übernommen werden. Die Agenten müssen allgemein in der Lage sein, in einer offenen, verteilten und heterogenen Umwelt Verbünde in einer annehmbaren Zeit bilden zu können. Das beinhaltet auch Szenarien, in denen dynamische Ereignisse auftreten können, die den Formierungsprozess stören können. Störeinflüsse beeinflussen dabei nicht nur die zu bearbeitenden Aufgaben, sondern auch die Verfügbarkeit von Informationen und Ressourcen, wie auch die temporäre Trennung von Verbundpartnern innerhalb des Agentennetzwerks.

Als Grundlage für die Entwicklung von Algorithmen zur Bildung von Verbünden in dynamischen, nicht-benevolenten Umgebungen, dienen Verfahren zur Bildung von Holonen [GSV99a]. Der Unterschied zu den bisherigen Verfahren zur Erzeugung von Holonen liegt darin, dass nunmehr Informationen unsicher und die Agenten nicht notwendigerweise benevolent[1] sind. Daher müssen die bekannten Verfahren auf ihre Verwendbarkeit hin geprüft und gegebenenfalls angepasst werden. Die Entwicklung und Verwendung von Formierungsmethoden, die es potentiellen Partnern ermöglicht, zeitlich beschränkte Verbünde nach Bedarf jederzeit zu bilden, beinhaltet auch die Herausforderung, effizient und zeitnah zeitlich begrenzte, profit-orientierte Verbünde zu bilden. Solche Verfahren können beispielsweise für optimale Verhandlungen, Anschaffungen und gemeinsame Nutzung von Ressourcen während der gleichzeitigen Teilnahme an mehreren elektronischen Märkten eingesetzt werden. Erste Arbeiten in dieser Richtung wurden von [TS00b], [YkS01], [She01] vorgelegt. Als Grundlage für den Aufbau neuer Kooperationsverfahren wird in dieser Arbeit ein Formalismus zur Abbildung eines Multiagenten-Systems auf einen Vektorraum vorgestellt. Dadurch eröffnet sich die Möglichkeit, Sätze und Verfahren der klassischen Mathematik auf Multiagenten-Systemen zu übertragen. Durch die Abbildung von Agenten auf Vektoren besteht die Möglichkeit, Gruppen von Agenten durch einen Vektor zu beschreiben[2]. Diese Darstellung der Entitäten eignet sich speziell für vektorbasierte Verfahren, wie zum Beispiel dem Einsatz von *Support Vector Machines* (SVM) [CV95a] zur Selektion geeigneter Agenten bzw. Gruppen von Agenten anhand einer Nachfrage.

Ist diese Abbildung möglich, bietet diese Theorie unterschiedlichste Einsatzmöglichkeiten, so z.B. bei der Analyse von Relationen zwischen den Agenten, basierend auf einer Metrik für Matchmakingverfahren, Bestimmung alternativer Agenten zur Aufgabenbearbeitung über das Ähnlichkeitskriterium, oder bei der Ermittlung des Ressourcenbedarfs komplexer Aufgaben, für deren Bearbeitung unterschiedliche Agenten eingesetzt werden müssen, die mit Hilfe einer Linearkombination gelöst werden kann. Beispielsweise finden bereits am *Max-Planck-Institut für Informatik* (MPII) erste Bestrebungen statt[MPI], für nicht graphische vektorielle Berechnungen die GPUs (Graphic Processing Unit) der Grafikkarten zu nutzen, da diese für solche Berechnung hoch optimiert und deutlich effizienter als herkömmliche CPUs sind. Die ernorme Leistungssteigerung in den letzen Jahren im Bereich der Computergrafik (Verfahren, wie auch Hardware) legt eine solche Abbildung nahe. Aus dem Bereich Computergrafik kön-

[1] Nicht betrügerisch.

[2] Basierend auf einer durch den Vektorraum definierten Verknüpfungsvorschrift.

nen zum Beispiel Algorithmen zur Kollisionsberechung eines grafischen Objekts mit der Umwelt eingesetzt werden, um Agenten mit bestimmten Eigenschaften auszuwählen. Verfahren zur Einschränkung der Komplexität, wie zum Beispiel der Einsatz von sogenannten Boundingboxen in der Computergrafik, können in Multiagenten-Systemen möglicherweise ebenso sinnvoll verwendet werden. Solche Verfahren sind besonders in Szenarien mit einer sehr hohen Anzahl von Agenten sinnvoll einsetzbar.

1.1 Zielsetzung dieser Arbeit

In dieser Arbeit schlage ich einen generischen Ansatz vor, wie ein Multiagenten-System zu entwerfen ist, das Holone in einer instabilen und dynamischen Umgebung bilden kann. Basierend auf drei Anwendungsbeispielen wird ein Verfahren entwickelt, welches eine effiziente Möglichkeit zur Bildung von Holonen beschreibt und es den Agenten so ermöglicht, sich selbst den dynamischen Änderungen der Umgebung anzupassen. Jeder Agent ist mit einem Holonenformierungsmechanismus ausgestattet, um den Flaschenhals-Effekt durch Einsatz eines zentralen Koordinators zu vermeiden, der in vielen Arbeiten immer noch Einsatz findet [SSJ98], [SK99], [SLA+99], [TS00a]. Dieser Mechanismus liefert einen dezentralen Ansatz, um das *dynamic holon formation* (DHF) Problem zu lösen. Dazu werden in dieser Arbeit sowohl Koalitionsformierungsalgorithmen, wie auch Teamformierungsalgorithmen in Bezug auf theoretische, wie auch praktikable Aspekte untersucht. Die Kooperation zwischen den Agenten geschieht auf Grundlage ihrer individuellen Nutzenberechnung unter Berücksichtigung von Unsicherheiten und Wahrnehmung der sich dynamisch verändernden Umwelt. Agenten müssen dabei zusätzlich mit Problemen, die dynamisch in der Arbeitsumgebung auftreten, umgehen können, wie:

- Informationen sind unvollständig
- Informationen sind unsicher und fehlerhaft
- Agenten verlassen die Agentengesellschaft bzw. treten ihr jederzeit bei
- Agenten sind nicht notwendigerweise benevolent

Welche Auswirkungen dies auf die Agentengesellschaft hat und welche Konsequenzen für die Bildungen von Koalitionen, Teams bzw. Holonen daraus entstehen, werde ich in dieser Arbeit diskutieren und entsprechende Verfahren entwickeln, um trotz dieser Schwierigkeiten zu akzeptablen Lösungen zu kommen.

1.2 Gliederung

Diese Arbeit ist folgendermaßen strukturiert:

Kapitel 2 gibt einen Überblick und Einführung in die Grundlagen der Multiagenten-System-Technologie und der am DFKI entwickelten Agentenarchitektur InteRRaP, die als Grundlage dieser Arbeit dient. Aufbauend auf den Arbeiten von [Vie00] und [Ger99b] wird das Paradigma holonischer Agenten beschrieben.

Kapitel 3 beschreibt Anwendungen von holonischen Multiagenten-Systemen in dynamischen Umgebungen anhand von drei Anwendungsbeispielen und die Auswirkungen von Störungen auf die Kooperationen der Agenten.

Kapitel 4 entwickelt die konzeptionellen Grundlagen zur Organisation von komplexen Arbeitsabläufen unter Einsatz eines Ablaufmanager-Agenten zur Koordination und Kontrolle der korrekten Bearbeitung von komplexen Aufgaben.

Kapitel 5 wendet die Vektorraumtheorie zur Repräsentation von Agenten, Holonen, Aufgaben und Diensten an. Dazu wird gezeigt, dass ein Multiagenten-System als metrischer Vektorraum angesehen werden darf und wie Relationen zwischen den Agenten anhand dieser Theorie beschrieben werden können.

Kapitel 6 charakterisiert die Arbeitsumgebung dynamischer Systeme und Möglichkeiten von Kooperationsformen in Multiagenten-Systemen. Folgend wird der Begriff der Stabilität für diese Anwendungsdomäne definiert, der zur Bewertung unterschiedlicher Kooperationsansätze notwendig ist. Das Erstellen von Prognosen und der Einsatz von Lernverfahren (basierend auf dem in Kapitel 5 definierten Agenten-Vektorraum) sind Grundlage für die Adaptionsfähigkeit von Agenten. Abschließend wird die Funktionsweise und Einsatzmöglichkeit von *Support-Vector-Machines* zur Verbesserung der Lernfähigkeit eines Agenten beschrieben.

Kapitel 7 präsentiert und diskutiert ein Schema zur Bildung von Holonen in dynamischen Umgebungen. Exemplarisch werden aus dem Schema vier Algorithmen entwickelt, deren Korrektheit, Terminierung, Laufzeit und Komplexität im Verlauf gezeigt wird. Die Algorithmen basieren dabei auf den Verfahren, die in den vorherigen Kapitel 4, Kapitel 5 und Kapitel 6 eingeführt wurden.

Kapitel 8 erörtert die Anwendung der in Kapitel 7 beschriebenen Algorithmen anhand von drei Testszenarien. Eine Evaluierung und Interpretation der Testergebnisse zeigt die Performanz von Stabilität und Effizienz hinsichtlich der Anzahl von notwendigen Verhandlungen und der Kosten bzw. Nutzen der Holone.

Kapitel 9 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf mögliche Weiterführungen.

Kapitel 2

Kooperation in Multiagenten-Systemen

Die Künstliche Intelligenz (KI) ist eine noch junge Forschungsdisziplin, die vor etwa fünfzig Jahren am Dartmouth Colleg erstmals unter dem neuen Namen *artifical intelligence* ihre Arbeiten präsentierte: 1956 traf sich eine Gruppe von amerikanischen Wissenschaftlern unter der Führung von John McCarthy in Vermont zum 'The Dartmouth summer research project on artificial intelligence'. Später wurde dieses Treffen bekannt als Darthmouth College Conference, bei dem die Teilnehmer die Zukunft maschineller Intelligenz diskutierten. Unter den Teilnehmer dieses Treffens, waren bekannte Forscher wie Claude Shannon, Marvin Minsky, John McCarthy, Allen Newell und der spätere Nobelpreisträger Herbert Simon [NAS00]. Erste Forschungszentren für künstliche Intelligenz bildeten sich später an der Carnegie Mellon University (CMU) und am Massachusset Institute of Technology (MIT). Einen ausführlichen Überblick über die Entstehung der KI gibt [His]. Die Entwicklung und Erforschung intelligenter Agenten begann in den siebziger Jahren des letzten Jahrhunderts. Erst in den achtziger Jahren erweiterte sich dieses Forschungsgebiet um den Bereich der Multiagenten-Systeme. Caglayan et al. [CH98] und Ferber [Fer01] geben einen zeitlichen Überblick der Agenten- und Multiagentenforschung.

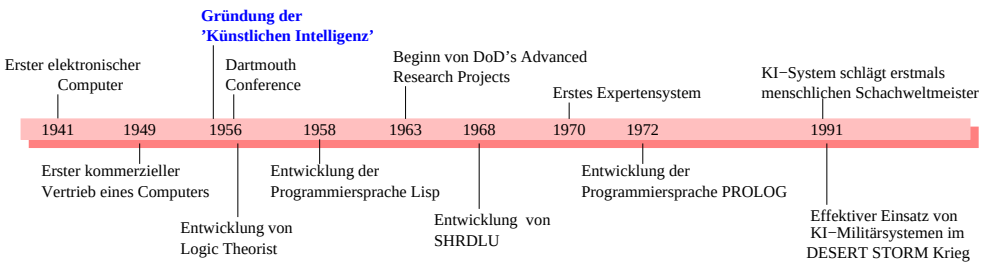

Abbildung 2.1: Entwicklung der künstlichen Intelligenz

2.1 Grundlagen von Multiagenten-Systemen

Multiagenten-Systeme entwickeln sich immer mehr zu einer interdisziplinären Forschungsrichtung, so dass zur Entwicklung von solchen Systemen auch Kenntnisse aus anderen Forschungsbereichen erforderlich sind. So sind zu Multiagenten-Systemen korrelierende Bereiche die Kommunikationsprotokolle und Agenten-Kommunikationssprachen, Agentenarchitekturen, verteilte Programmierung, sowie verteiltes Problemlösen, Analyse von komplexen adaptiven Systemen und Organisationstheorie und Sozionik, um nur die Wichtigsten zu nennen. Russel und Norvig [RN95] definieren speziell einen Agenten wie folgt:

'An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.'

Diese Definition eines Agenten ist noch sehr vage. Agenten können Roboter, Internetagenten, Soft Bots, Auktionsagenten, etc. sein, die von einfachen bis zu hoch komplexen Einheiten reichen. Corsten und Gössinger [CG97] klassifizieren Agenten anhand der implementierten Intelligenz in primitive, technische, kognitive und soziale Agenten. Ferner definieren sie einen intelligenten Agenten als Repräsentanten seines Nutzers, der autonom und beschränkt rational handelt. Abbildung 2.2 zeigt einen Überblick der verschiedenen Einflussbereiche auf Agenten in verteilten Systemen. Hieraus wird ersichtlich, wie umfangreich und komplex die Realisierung eines einzelnen Agenten sein können. Vom besonderen Interesse ist hierbei jedoch der Einfluss der *'Verteilten Künstlichen Intelligenz'* (VKI) auf die Agentenforschung. Systeme zur verteilten Problemlösung werden durch einen zentralen Top-Down-Lösungsprozess realisiert. Bei diesem Ansatz werden die Probleme in Teilprobleme dekomponiert, gefolgt von einer Analyse dieser Teilprobleme. Gegebenenfalls müssen diese nochmals in weitere Teilprobleme zerlegt werden, bis sie einem spezialisierten Agenten zugewiesen werden können. Der Vorgang gliedert sich in vier Schritte:

1. Problemdekomposition
2. Verteilung der Teilprobleme
3. Lösen jedes Teilproblems
4. Zusammenfügen aller Teillösungen entsprechend der Aufteilung zu einer Gesamtlösung

Damit dieser Ablauf möglich ist, müssen die Agenten einerseits benevolent sein, d.h. sie kooperieren uneingeschränkt miteinander, und zum anderen müssen die Probleme sich in unabhängige Teilprobleme zerlegen lassen, damit eine verteilte Problemlösung überhaupt erst möglich ist. Probleme können mit Hilfe eines solchen Verfahrens bezüglich regionaler, funktionaler oder zeitlicher Abhängigkeiten dezentral gelöst werden. Dabei kann die VKI in folgende drei Untergruppen gegliedert werden:

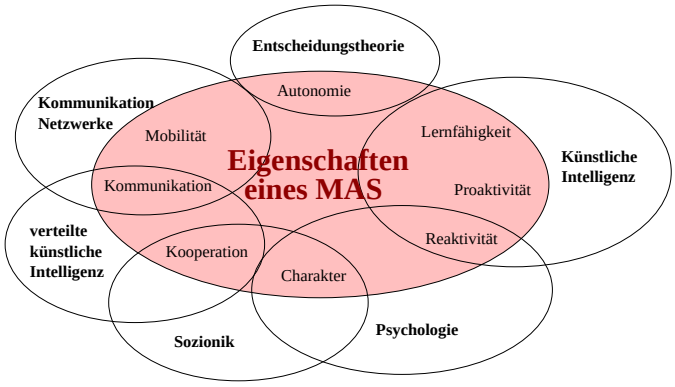

Abbildung 2.2: Einflussgebiete von Agenten- und Multiagenten-Systemen

- Parallele KI,
- Verteiltes Problemlösen und
- Multiagenten-Systeme

Diese Klassifikation ist zwar weit verbreitet, aber dennoch verschwimmen die Grenzen zwischen diesen Teilgebieten oftmals sehr stark; oft werden Multiagenten-Systemen als eine spezielle Form des 'Verteilten Problemlösens' angesehen [CG97]. Der wesentliche Unterschied zwischen kooperativ verteiltem Problemlösen und Multiagenten-Systemen liegt in der Flexibilität der Systeme zur Laufzeit. Während beim Verteilten Problemlösen zu Beginn des Systementwurfs die Aufgabenstellung bekannt ist, verfügen kooperierende Agenten in Multiagenten-Systemen über eine explizite Repräsentation ihrer Ziele, die es ihnen ermöglicht, auch vorab nicht definierte Aufgaben zu lösen [SGR98]. Bond und Gasser [BG98] definieren ein Multiagenten-System als:

'*Multiagent Systems are concerned with coordinating intelligent behaviour among a collection of autonomous intelligent agents, how they coordinate their knowledge, goals, skills, and plans joinly to take action or solve problems.*'

Im Wesentlichen stellt sich also die Frage, wie Agenten ihr Wissen, ihre Ziele, Fähigkeiten und Pläne koordinieren und somit ihr gemeinsames Handeln regeln können. Wesentlich für gemeinsames Handeln ist die Repräsentation der Welt durch die Agenten. Weiss [Wei99] definiert dazu drei Agententypen mit unterschiedlicher Weltrepräsentation (0-level-agent, 1-level-agent and 2-level-agent), mehr oder minder fähig über das Verhalten der anderen Agenten zu schließen. Agenten sollen dem Anwender Arbeit abnehmen und diese autonom durchführen [Che99]. Autonom bedeutet, dass der Agent nicht unter unmittelbarer Kontrolle des Anwenders agiert, sondern sein Verhalten wird durch seine eigene Erfahrung bestimmt. Ein Agent nimmt seine Umwelt durch Sensoren wahr und reagiert durch seine Handlungen auf diese Umwelt. Die Aktionen eines Agenten sind dabei auf ein bestimmtes Ziel hin ausgerichtet. Russell und Norvig [RN95]

unterscheiden hierzu vier verschiedene Typen von Softwareagenten:

- *Einfache Reflex-Agenten:* Einfache Reflex-Agenten handeln nach fest vorgegebenen Regeln. Die Wahrnehmungen werden mit den in den Regeln beschriebenen Zuständen verglichen, bei Übereinstimmung wird die entsprechende Handlung gesetzt.

- *Agenten, die ihre Umwelt repräsentieren:* Im Gegensatz zu den nach Reflexen handelnden Agenten verfügen diese Agenten über ein internes Modell, wie sich die Umwelt entwickelt und welche der eigenen Aktionen die Umwelt beeinflussen. Durch dieses Modell ist der Agent in der Lage, unterschiedliche Umweltzustände hinsichtlich ihrer Auswirkungen zu unterscheiden, auch wenn diese Zustände die gleichen Sensorreize produzieren.

- *Zielgerichtete Agenten:* Zielgerichtete Agenten haben ein Ziel und versuchen dieses zum Beispiel durch Planungsverfahren zu erreichen. So können Aktionen ausgewählt werden, die im Hinblick auf die gestellte Aufgabe zielgerichtet sind, d.h. den Agenten seinem Ziel näher bringen.

- *Nutzenorientierte Agenten:* Nutzenorientierte Agenten können zwischen verschiedenen Zielen differenzieren, indem sie versuchen, eine Nutzenfunktion über den Zielen zu maximieren. Die Nutzenfunktion bildet verschiedene Umweltzustände auf einer Skala ab, die Präferenzen hinsichtlich verschiedener Zustände beschreibt. Der Agent wird so in die Lage versetzt, Zielkonflikte zwischen mehreren wünschenswerten Zielen zu bewerten.

Zurückgreifend auf die Definition von Bond und Gasser [BG98], ist die Kooperation zwischen Agenten ein wesentlicher Bestandteil von Multiagenten-Systemen, da manche Probleme überhaupt nur durch den Einsatz mehrerer Agenten lösbar sind. Ferner resultieren durch die Kooperation der Agenten untereinander eventuell quantitative und/oder qualitative Vorteile, hinsichtlich Ressourcenverbrauch bzw. Zielerfüllung. Insbesondere die unterschiedliche Spezialisierung der Agenten (Planen, Informationssuche, Verhandeln, Manager, usw.) unterstreicht den Sinn der Zusammenarbeit unterschiedlichster Agenten zur effizienten Lösung einer komplexen Aufgabe. Entsprechend dem Grundsatz der VKI, dass unter Einsatz vieler Einheiten ein effizienteres Problemlösen gegenüber einer Einheit möglich ist, wird die Notwendigkeit einer Koordination deutlich. Charakteristisch für Multiagentensysteme ist ihre meist offene Architektur (Veränderung der Agentengesellschaft zur Laufzeit), Autonomie jedes einzelnen Agenten und ihre in der Regel dezentrale Realisierung (Verteilung der Agentengesellschaft auf ein physikalisches Netzwerk). Dadurch können bei Bedarf dem System weitere Agenten problemlos hinzugefügt werden. Kommunikation bildet die Grundlage der Koordination. Prinzipiell ist zwar eine Koordination auch ohne Kommunikation möglich (siehe [BG98]), dazu müssten die Agenten jedoch in der Lage sein, ihre Umwelt vollständig zu erfassen. Diese Form von Koordination ist dann sinnvoll, wenn eine direkte Kommunikation nicht möglich ist [NLJ96]. Sandholm und Volkan zeigen, wie sich aus dem Verhalten der anderen Verhandlungsteilnehmer Rückschlüsse auf deren Strategien und Verhandlungsfristen ziehen lassen [SV98].

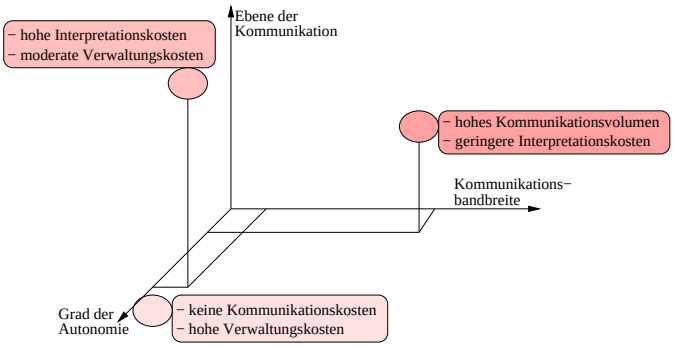

Abbildung 2.3: Nutzen-Verlust Balance der Kommunikation

2.1.1 Kommunikation

Kommunikation ist eine Grundeigenschaft der Agentendefinition. Der Austausch von Nachrichten zwischen den Agenten verbraucht jedoch einen Teil ihrer zur Verfügung stehenden Ressourcen (z.B. Rechenzeit), so dass der Kommunikationsnutzen gegenüber dem Ressourcenverbrauch abgewogen werden muss. Für die Realisierung eines Kommunikationsmodells gibt es verschiedene Möglichkeiten[SF96], aus denen eine aufgrund des nicht unerheblichen Ressourcenbedarfs für die Kommunikation je nach Anwendungsszenario zu wählen ist. Die Kommunikationsmechanismen lassen sich in verschiedene Gruppen entsprechend der zugrunde liegenden Verfahrensweise aufteilen. Ollmert und Schinzer [OS01] beschreiben zwei Gruppen: a) Blackboard- und b) nachrichtenorientierten Kommunikationsmechanismen. Bei der Verwendung eines Blackboard-Mechanismus führen die Agenten eine indirekte Kommunikation, sie lesen und schreiben dazu auf einen für alle zugänglichen Speicherbereich (shared memory). Dieser Mechanismus ist zwar einfach zu realisieren, jedoch hat er auch Nachteile:

1. Der zentrale Ansatz birgt die Gefahr eines Kommunikationsflaschenhalses hinsichtlich der Performanz des Systems.

2. Die Robustheit eines Multiagenten-System mit Blackboard-Kommunikation hängt sehr stark von der Zuverlässigkeit des Blackboards ab (*single point of failure*).

3. Die Sicherheit der Daten ist nicht gegeben. Alle Informationen auf dem Blackboard sind allen Agenten jederzeit zugänglich und daher nicht mehr geheim.

Bei der nachrichtenorientierten Variante wird eine direkte Peer-to-Peer Kommunikation zwischen den Agenten geführt. Broad- und Multicasting sind dabei Kommunikationsmethoden, in denen die Nachrichten gruppenorientiert gesendet bzw. empfangen werden. Diese Variante führt jedoch zu einer hohen Kommunikationslast im physikalischen Netzwerk, die es zu vermeiden gilt. Um zum einen die Kommunikationslast zu reduzieren und zum anderen die Aktionen von Agenten zu synchronisieren, werden dem Problem entsprechend spezielle Kommunikationsprotokolle entwickelt. Sie bilden eine feste Abfolge von Anfragen

und Antworten in einer zeitlich vorgegebenen Reihenfolge. Bei der so genannten 'Master-Slave' Variante gibt der Master konkrete Befehle an den Slave, der sie ohne Abweichungen auszuführen hat. Diese Strategie ist sehr hierarchisch geprägt und verringert die Kommunikationslast, da zum einem keine Abstimmung der Befehlsinhalte möglich ist und andererseits im Allgemeinen keine Antwort erforderlich ist.

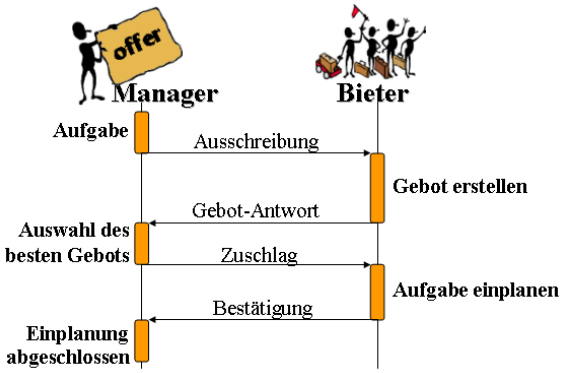

Abbildung 2.4: Ablauf des Contract-Net Protokolls

Das bekannteste Kommunikationsprotokoll ist das *Contract-Net Protokoll*. Das Protokoll eignet sich in besondere Weise zur Verteilung von Aufgaben auf Agenten. Das Protokoll sieht einen speziellen Manageragenten vor, der Aufgaben an eine Menge von Bieteragenten ausschreibt (siehe Abbildung 2.4). Basierend auf den lokalen Kosten-/Nutzenberechnungen der einzelnen Bieteragenten, werden Gebote ermittelt, die dem Manageragenten übersandt werden. Sind alle Antworten eingegangen, selektiert der Manager das 'beste' Gebot[1] und schickt dem entsprechenden Bieter einen Zuschlag. Daraufhin plant dieser Bieter die Aufgabe in seinem Plan fest ein und informiert anschließend den Manager über den Erfolg des Einplanens.

Formal basiert die Kommunikation unter Agenten auf der Sprechakttheorie [Wei99]. Sprechakte entkoppeln Absicht und Effekt und werden wie Aktionen behandelt. Eine Konversation entsteht dann durch die koordinierte Aneinanderreihung einzelner Sprechakte. Demnach werden mit jeder Äußerung drei Arten von Sprechakten ausgeführt:

- *Lokutionärer Akt* bezieht sich auf das Sprechen selbst.
- *Illokutionärer Akt* ist der beabsichtigte Sinn der Meinung des Sprechers.
- *Perlokutionärer Akt* ist die aus dem illokutionären Akt resultierende Aktion.

Weiterhin benötigt man für eine reibungslose Verständigung ein geeignetes Kommunikationssprache. Ein bekannter und häufig verwendeter Sprachstandard ist

[1] Die Bewertung der Gebot ist abhängig von der lokalen Nutzenfunktion des Manageragenten.

type:	<Nachrichtentyp - Performative>
sender:	<Sender der Nachricht>
receiver:	<Empfänger der Nachricht>
reply_with:	<Kennung mit der geantwortet werden muss>
in_reply_to:	<Kennung des Bezugssprechakts>
ontology:	<Thema der Nachricht>
language:	<für den Inhalt verwend. Sprache(KIF,Prolog...)>
content:	<eigentlicher Inhalt>

Abbildung 2.5: Aufbau eines Sprechakts

die *Knowledge Query and Manipulation Language* (KQML) [Lux95]. Entsprechend dem KQML-Standard besitzt der Kopf eines Sprechakts (siehe Abbildung 2.5) folgende Attribute: *performative, sender, receiver, reply_with, in_reply_to, content, language, ontology*. Das '*performative*' klassifiziert den Sprechakt, wodurch ein Sprechakt Protokollen zugewiesen werden kann. Das 'performative' wird dabei entsprechend dem zu übertragenden Inhalt gewählt. Für eine bidirektionale Nachrichtenart stehen so beispielsweise zwei Sprechakttypen *ask_if* und *inform* zur Verfügung. Ein *ask_if* Sprechakt fordert implizit eine Antwort, die durch einen anschließenden *inform*-Sprechakt zurückgegeben wird. Ein *inform* kann auch ohne *ask_if* abgesandt werden, wenn ein anderer Agent informiert werden soll. Hingegen enthält z.B. ein *tell* Sprechakt eine Anweisung, die ausgeführt werden muss, jedoch nicht durch ein *inform* bestätigt werden muss (unidirektionale Nachricht). Durch die Attribute *sender* und *reply_with, in_reply_to* wird die Anschrift des sendenden Agenten, sowie ein Identifikator übermittelt. *reply_with* und *in_reply_to* enthalten einen eindeutigen Bezeichner, der angibt auf welchen Sprechakt Bezug genommen werden soll bzw. Bezug genommen wird. Durch diese Parameter ist es möglich, den Sprechakt eindeutig einer Aktion eines Agenten zuzuweisen. Der Eintrag der Adresse des Empfängers (*receiver*) dient zur eindeutigen Zuordnung des Sprechakts durch eine Send-Methode in der Kommunikationsebene eines Agenten. *Sender* und *receiver* sind von der gleichen Datenstruktur. Falls Agenten unterschiedlicher Architekturen miteinander interagieren müssen, ist es notwendig, die Sprache (*language*) und die Ontologie (*ontology*) im Sprechakt anzugeben, damit ein Empfänger in der Lage ist, den Inhalt zu lesen und zu interpretieren. Die Sprache benennt ein Regelwerk für die Syntax, die Ontologie gibt eine Begriffserläuterung an, die einem Wort eine Bedeutung zuweist. Der Inhalt des Sprechakts wird in dem Attribut *content* abgelegt.

2.1.2 Koordination

In Multiagenten-Systemen ist es wichtig, nicht nur die Ziele zu koordinieren, sondern auch die Aktivitäten und das Verhalten der Agenten. Gemäß Malone and Crowston [MC93] kann Koordination als Prozess zur Verwaltung von Abhängigkeiten zwischen den Aktivitäten von Agenten bezeichnet werden. Bei der Analyse der Abhängigkeiten zwischen den einzelnen Prozessen wird speziell nach gemeinsam genutzten Ressourcen, Abhängigkeiten zwischen einzelnen Aufgaben, Verfahrens- und Kundenbindungen gesucht. Abhängigkeiten zwischen

einzelnen Aufgaben existieren zum Beispiel, wenn die Aufgabe Teil einer komplexen Aufgabe ist, so kann z.B. ein LKW erst dann beladen werden, wenn ein Gabelstapler zum gleichen Zeitpunkt zur Verfügung steht. Verhandlungen sind essentiell für die Koordination in Multiagenten-Systemen. Die Ansätze in den Verhandlungen basieren in der Regel auf spiel- und entscheidungstheoretischen Grundlagen [Var99]:

- *Spieltheorie*
  Das Gesamtresultat der Bearbeitung von (komplexen) Aufgaben ist abhängig von den Entscheidungen mehrerer Akteure, so dass ein Einzelner das Ergebnis nicht unabhängig von der Wahl anderer Agenten bestimmen kann.

- *Entscheidungstheorie*
  Entscheidungen werden unter einem abschätzbaren Risiko (eine Wahrscheinlichkeit kann den einzelnen Zuständen zugeordnet werden) oder unter Unsicherheit (eine Wahrscheinlichkeit kann den Zuständen nicht zugeordnet werden) getroffen.

Zentrale Bewertungskriterien für eine erfolgreiche Verhandlungsform sind:

- *Allgemeine Nutzenverteilung* ist als Vergleichskriterium zur Bewertung verschiedener Verhandlungsformen geeignet. Zum Beispiel kann die Summe über alle agentenindividuellen Nutzenfunktionen gebildet werden. Die Ergebnisse lassen sich dann z.B. für Szenarien verwenden, in denen Koalitionen von Agenten konkurrieren.

- *Pareto-Effizienz* ist auch ein Vergleichskriterium, welches die Verhandlungsformen nach der Effizienz pareto-optimaler Lösungen bewertet. Es kann aber auch als Ausschlusskriterium eingesetzt werden, so das nur Verhandlungsformen in die engere Auswahl gelangen, die pareto-optimal (*'the best that could be achieved without disadvantaging at least one group.'* [Sch70]) sind.

- *Individuell rationale Verhandlungsformen*, d.h. ein Agent willigt nur in Verhandlungen ein, wenn dadurch seine Auszahlung nicht geringer wird. Dieses Kriterium muss nicht notwendigerweise erfüllt sein, insbesondere wenn die Verbesserung der allgemeinen Situation wichtiger ist, als das Wohl eines einzelnen Agenten.

- *Stabilität* existiert, sobald ein Agent keine dominante Strategie verfolgen kann, sondern die eigene Strategie von der anderer Agenten abhängt. Zur Beschreibung der Stabilität muss ein geeignetes Kriterium eingeführt werden, das unter Umständen mit dem Effizienz-Kriterium in Konflikt stehen kann. Hier kommen spieltheoretische Überlegungen in Betracht. Beispielsweise kann ein Nash-Gleichgewicht [Nas50] in der Situation eines Gefangenendilemmas die pareto-effizienteste Lösung nicht erreicht werden. Das *Nash*-Gleichgewicht stellt eine Standardlösung in der Spieltheorie dar und wird beispielsweise zur Analyse von politische Auseinandersetzungen verwendet.

- *Effiziente Berechnungen*, d.h. optimale Ausnutzung der Rechner und Vermeiden unnötiger Rechenzeit.
- *Effiziente Kommunikation*, also möglichst geringe Belastung der Kommunikationskanäle, aber ausreichend um alle notwendigen Informationen zu übermitteln.

Die Durchführung von koordinativen Prozessen teilt sich in zwei Klassen:

- *zentrale Verhandlungen*: ein ausgezeichneter Agent sammelt alle Informationen, Nachfragen und Angebote von Ressourcen und Aufgaben und berechnet eine Lösung, die durch die teilnehmenden Agenten umgesetzt werden müssen.
- *dezentrale Verhandlungen*: jeder Teilnehmer behält seine Autonomie und tritt mit den relevanten Agenten in direkte Verhandlungen. Ein Agent führt so unter Umständen mehrere Verhandlungen bezüglich der gleichen Fragestellung (parallel) aus, bis eine Lösung gefunden wurde.

Beide Vorgehensweisen haben Nachteile. Die erste reduziert die Eigenverantwortung der Agenten maßgeblich und fordert von den Teilnehmern Vertrauen in die Integrität des zentralen Koordinators. Auch kann diese Einheit zum Flaschenhals werden, so dass alle weiteren Aktionen der Agenten von ihr abhängen. Bei einem Ausfall der zentralen Instanz kann dies zu einer Blockade führen. Die zweite Vorgehensweise erhält die Autonomie der Agenten und reduziert so das Ausfallrisiko, jedoch auf Kosten der Performanz und einer höheren Kommunikationslast. Arbeiten zu speziellen Entscheidungsschemata [TF99] [JT01] für konkurrierende Agenten mit gegensätzlichen Interessen können dabei helfen, Entscheidungskonflikte zu lösen.

2.1.3 Kooperation

Eine Vielzahl unterschiedlicher wissenschaftlicher Disziplinen (Soziologie, Psychologie, Organisationstheorie, Arbeitswissenschaften, etc.) beschäftigen sich mit dem Gebiet des *kooperatives Arbeiten*. Es existieren daher unterschiedliche Definitionen darüber, was unter 'Kooperation' zu verstehen ist und welche Voraussetzungen an die Umwelt gestellt werden. Einige Autoren verstehen unter kooperativem Arbeiten die gemeinsame Arbeit mehrerer Aufgabenträger an einem Produkt oder einer Dienstleistung. Dabei hat jeder einzelne Teilnehmer individuelle Teilaufgaben und -bewertungen wahrzunehmen. Insgesamt soll jedoch das gemeinschaftliche Hauptziel erfüllt werden. Die gemeinsame Erfüllung von Aufgaben erfordert eine gute Koordination und Kommunikation unter den Gruppenmitgliedern, um ein effizientes Zusammenarbeiten zu gewährleisten. Denn nur auf diese Weise lassen sich inkompatible Teilergebnisse und mehrfaches Bearbeiten der gleichen Teilaufgabe durch unterschiedliche Teilnehmer verhindern. Die Zusammenarbeit soll zudem unabhängig von Zeit und Raum möglich sein; es lassen sich somit vier Kooperationssituationen unterscheiden:

1. gleiche Zeit/gleicher Ort,
2. gleiche Zeit/verschiedene Orte,
3. verschiedene Zeiten/gleicher Ort und
4. verschiedene Zeiten/verschiedene Orte

Als erste Form der Zusammenarbeit wird die **Koordination** beschrieben, durch welche verschiedene Tätigkeiten unter einem höheren Gesichtspunkt vereinigt werden sollen. Jede neue Tätigkeit soll den vorherigen Tätigkeiten Sinn geben und auf diesen aufbauen. Wesentlich für die Koordination ist die Abstimmung der einzelnen Teilaufgaben (*Synchronisation*). In der **Kollaboration**, d.h. der Zusammenarbeit mehrerer Personen, wird die zweite Form der Kooperation gesehen. Wesentlich für diese Kooperationsform ist die gemeinsame Zielvorstellung aller Gruppenmitglieder, so dass sie ihre Einzelaufgaben im Sinne dieses Ziels erfüllen können. Ein reger Informationsaustausch zwischen den Gruppenmitgliedern ist für die effiziente Zusammenarbeit unbedingt erforderlich. Als dritte Form der Kooperation wird die **Gruppenentscheidung** angesehen, bei welcher mehrere Personen zusammenwirken, um gemeinschaftlich eine Entscheidung zu finden. Hierbei können die Gruppenmitglieder gleichrangig sein oder aber unterschiedliche Rollen einnehmen. Wichtig für eine Gruppenentscheidung ist ein gemeinsames Grundverständnis der Sache und gegenseitiges Vertrauen, da andernfalls die Akzeptanz der getroffenen Entscheidung gefährdet sein könnte. Zusammenarbeit kann in analoger Weise daher in vier Ebenen aufgeteilt werden. Die erste Ebene der Zusammenarbeit ist demnach das **Informieren**, wobei eine Bekanntschaft zwischen Sender und Empfänger nicht unbedingt erforderlich ist. Die zweite Ebene wird in der **Koordination** gesehen, welche durch die gemeinsame Ressourcennutzung mehrerer Personen gekennzeichnet ist. Die Bekanntschaft dieser Personen ist eher flüchtig. Das **Zusammenwirken** stellt die dritte Ebene der gemeinsamen Arbeit dar. Es wird gemeinsam an der Erstellung eines Produktes gearbeitet, wobei es jedoch individuelle Beiträge und Bewertungen zur Zielerfüllung gibt. Erst die vierte Ebene wird als **Kooperation** bezeichnet. Hier arbeiten die Teilnehmer gleichberechtigt zusammen und die Ziele des Einzelnen werden dem gemeinsamen Gruppenziel untergeordnet. Das Gesamtergebnis wird durch die Gruppe bewertet. Die Analyse zeigt, dass das allgemeine Verständnis unterschiedlicher Quellen hinsichtlich des Begriffs 'kooperative Arbeit' überwiegend sehr ähnlich ist. Es sind jeweils mehrere Personen beteiligt, die zur Erfüllung einer gemeinsamen Aufgabe individuelle Beiträge zu leisten haben. Um eine erfolgreiche Zusammenarbeit zu gewährleisten, muss die Erledigung der einzelnen Teilaufgaben koordiniert werden, wozu wiederum eine gute Kommunikation der Gruppenmitglieder untereinander und eine gemeinsame Vorstellung des zu erreichenden Gruppenziels notwendig sind.

In kooperativen Szenarien [Wei99], [SF96] wird im Allgemeinen von benevolenten Agenten ausgegangen. Als Basisstrategie ist hier das Aufteilen einer Aufgabe in verschiedene Teilaufgaben zu sehen, die jeweils getrennt gelöst und deren Lösungen anschließend zu einer globalen Lösung zusammengesetzt werden. Dies ist jedoch nur bei Aufgaben möglich, bei denen die unabhängige Bearbeitung der Teilaufgaben möglich ist. Existieren nach der Aufteilung dennoch

Abhängigkeiten zwischen den Teilaufgaben, müssen die Aktionen der Agenten koordiniert werden. Das kann durch im Vorfeld festgelegte Regeln, durch einen alles überwachenden Agenten oder durch regelmäßige, gegenseitige Abstimmung geschehen.

2.1.4 Konkurrenz

Konkurrierende, eigennützige Agenten [Wei99] verfolgen im Allgemeinen unterschiedliche Ziele. Beispielweise benötigen sie zum Lösen zweier unabhängiger Aufgaben gemeinsame, begrenzte Ressourcen zur Verfolgung ihrer persönlichen Ziele. Obwohl die Agenten in einem Konkurrenzverhältnis zueinander stehen, müssen sie gegebenenfalls miteinander kooperieren, um einen reibungslosen Ablauf ihrer Prozesse zu finden. Die Agenten werden versuchen, ohne Rücksichtnahme auf globale Ziele, ihr eigenes lokales Ziel zu maximieren. So lässt sich zum Beispiel in Situationen, in denen die Aufgaben unabhängig voneinander zu lösen sind, durch Konkurrenz den am besten geeigneten Agenten für eine bestimmte Aufgabe finden. Dieses Verhalten ist nicht ungewöhnlich und häufig in der Realität anzutreffen, z.B. wenn mehrere Unternehmen miteinander in Verhandlung stehen. In der Mikroökonomie existieren Methoden, die zu einer rationalen Entscheidungsfindung beisteuern. Analog der Kooperation können auch in offenen Systemen konkurrierende Agenten aufeinander treffen, die unterschiedliche Gruppen der realen Welt repräsentieren und gegebenenfalls von unterschiedlichen Personen entwickelt worden sind.

2.1.5 Planung

Um ein Ziel zu erreichen, muss ein Agent eine Reihe von Aktionen ausführen, d.h. ein Planverfahren muss zum einen eine effiziente, zielgerichtete Suche durchführen, um einen Plan (eine Sequenz von Aktionen) zu finden und zum anderen eine nützliche Repräsentation der Planschritte bzw. der Aktionen des Agenten verwenden, um Aussagen über die Vorbedingungen und Auswirkungen der geplanten Aktionen bestimmen zu können. Einer der bekannteste Planungsverfahren ist der *STRIPS*-Planer [FN71].

Für die Erzeugung eines Plans sind drei Aspekte von besonderer Bedeutung:

1. *Granularität* der (Teil-)Plänen - Je häufiger ein Plan verändert wird, desto notwendiger ist es, den Plan in kleine Segmente zu unterteilen, so dass der Plan nicht als Ganzes von den Modifikationen betroffen ist. Die Granularität ist auch bei der Wiederverwendbarkeit von Teilplänen von großer Bedeutung. Zu grob gefasste Teilpläne können seltener wieder verwendet werden als kleinere Teilpläne, die unter Umständen weniger Vorbedingungen haben. Jedoch steigt mit der Verfeinerung der Granularität die Komplexität des Planungsprozesses.

2. *Wiederverwendbarkeit* von (Teil-)Pläne - Bei der erfolgreichen Ausführung von Plänen, werden diese für den erneuten Einsatz gespeichert. Um die Einsatzbedingungen zu einem späteren Zeitpunkt prüfen zu können, wer-

den zu den Plänen die jeweiligen Vor- und Nachbedingungen mitgespeichert[2].

3. *Partielle Pläne* - Zuerst wird ein abstrakter generischer Plan erzeugt, der die wichtigsten Randbedingungen der Aufgabenstellung berücksichtigt. In nachfolgenden Iterationen werden weitere Randbedingungen der Aufgabenstellung hinzugenommen und der Plan daraufhin durch weitere notwendige Planschritte verfeinert bis schließlich alle Randbedingungen in den Plan aufgenommen sind. Der Vorteil dieses Verfahrens ist, dass Pläne wesentlich schneller erzeugt werden können, da zunächst die Komplexität durch die eingeschränkte Auswahl von Randbedingungen stark reduziert werden kann. Zum anderen müssen nur die Planschritte verfeinert werden, die zur Ausführung anstehen. So reduziert sich zusätzlich der Planungsaufwand, falls Störungen auftreten und Umplanungen durchgeführt werden müssen [RN95].

Um jedoch eine korrekte Planung, insbesondere in dynamischen Umgebungen, zu gewährleisten, ist es notwendig, ständig einen Abgleich der Ausführung mit dem Plan durchzuführen. Wird der Plan aufgrund von Störungen fehlerhaft, muss eine Neuplanung angestoßen werden. Konzeptionell wird das Multiagenten-Planen im Allgemeinen nach zentraler und verteilter Planung unterschieden [SF96].

- Die zentrale Planung sieht einen ausgezeichneten Agenten als zentrale Koordinationsstelle vor, der alle notwendigen Information von Agenten sammelt, wodurch effizient eine global optimale Lösung gefunden werden kann.

- Bei der dezentralen Planung werden die Nachrichten direkt unter den Agenten (Peer-to-Peer Kommunikation) ausgetauscht und die Planung so individuell autonom von den einzelnen Agenten ausgeführt. Ein häufig eingesetztes Verfahren hierbei ist das Contract-Net-Protokoll.

2.1.6 Wissen

Aktuelles Wissen über die Umwelt eines Agenten ist für die Planung und Koordination von Prozessen von großer Bedeutung. Neben der Fragestellung, wie das Wissen aktualisiert wird und Inferenzen über dem Wissen ausgeführt werden, ist die Art der Wissensrepräsentation ein weiteres zentrales Problem. Die Repräsentation von zeitabhängigem Wissen und der Beschreibung eines Zustands im Zeitkontinuum ist eine aktive Forschungsrichtung in der KI und Logik. Eine hierbei vielgenutzte Möglichkeit ist es, den Verlauf der Zeit in Intervallen aufzuteilen, bzw. zuvor definierte, explizite Zeitpunkte zu betrachten. Die Modellierung von zeitabhängigen Beziehungen basierend auf Intervallen kann unter Verwendung einer *Allen Algebra* [All83] realisiert werden. Dabei werden die Beziehungen in 13 Klassen unterteilt [All]. Aber es ist zu beachten, dass das *reasoning* in einer Allen Algebra NP-vollständig ist; jedoch *reasoning* in einigen

[2]Die Notation der STRIPS-Operatoren diente als Grundlage für die Definition des PDDL-Standards zur Beschreibung einer einheitlichen Plan-/Aktions-/Weltbeschreibungssprache.

speziellen Unterklassen dieser Algebra können nachweislich in polynomialer Zeit durchgeführt werden [BN95]. Eine weitere Möglichkeit Zeit zu repräsentieren, ist die explizite Beschreibung einzelner Zeitpunkte, zum Beispiel im *Situations-Kalkül* [MH69]. Eine große Vielfalt von temporalen und räumlichen Spezifikationen können auch durch lineare Ungleichungen beschrieben werden, die sich auf das Ende von Ereignissen oder respektive auf begrenzte Oberflächen beziehen [MB83]. In vielen Fällen genügt es, die Repräsentation der Zeit durch diskrete Intervalle oder explizite Zeitpunkte festzulegen. Gerade die Bearbeitung von Aufgaben geschieht immer innerhalb eines Zeitintervalls, so dass es oftmals ausreicht, die Start- und Endzeit der Bearbeitung durch konkrete Zeitpunkte festzuhalten [Vie00] [Ger99a]. Eine Überprüfung der Ausführbarkeit eines Plans geschieht dann durch die Prüfung auf Überlappung der Aufgabenzeitintervalle. Bei diesem Ansatz wird also keine explizite Repräsentation von Situationen vorgenommen. Die Zustandsbeschreibung zu einem gegebenen Zeitpunkt ergibt sich dann durch das Zusammenfassen der Einzelanalysen der Pläne der Agenten zu diesem Zeitpunkt.

2.1.7 Lernen

Adaption und *Lernen* werden parallel zur Planung betrieben. *Adaption* beschreibt das Verhalten eines Agenten, auf neue Situationen zu reagieren und seinen Plan geeignet zu anzupassen, so dass die Inkonsistenzen im Plan behoben sind. *Lernen* hingegen beschreibt die Fähigkeit, Wissen zu erlangen und dieses für die Planung nutzbar zu machen. Lernen [Wei99] in Multiagenten-Systemen ist ein relativ neuer Forschungsbereich der Künstlichen Intelligenz. Lernen in einem Multiagenten-System umfasst einerseits isoliertes Lernen eines einzelnen Agenten unabhängig von den anderen Agenten, andererseits kann ein einzelner Agent durch interagierendes Lernen sein erworbenes Wissen durch Informationsaustausch mit anderen Agenten erweitern. Beim Lernen kann zwischen folgenden Ansätzen der Verfahren unterschieden werden:

- *Caching*: Cachen ist das einfachste Verfahren, Wissen über einen kurzen Zeitraum zu sammeln. Diese Methode speichert die gesammelten Informationen, so dass sie bei Bedarf wiedergegeben werden können. Caching ist nur in Situationen sinnvoll, in denen die Umwelt stabil ist und sich die Voraussetzungen nicht ändern.

- *Feedback*: Die zugrunde liegende Idee ist, durch Interaktion mit anderen Agenten zu lernen. Diese Methode wird auch als *reinforcement learning* [SB98] [WST01] bezeichnet. Sie beschreibt ein sehr zielgerichtetes Lernen. Dabei werden dem Ziel dienliche Aktionen belohnt und die anderen bestraft. Im Wesentlichen stützt sich dieses Verfahren auf vier Funktionen:

  - Taktik
  - Belohnungsfunktion (kurze Zeitdauer)
  - Nutzenfunktion (große Zeitdauer)
  - Modell der Umwelt (optional)

Q-learning ist ein Sonderfall des reinforcement learning. Dabei existiert eine Funktion, die jedem Status-Aktionspaar den bestmöglichen Wert zuweist, so dass ein Vergleich zulässiger Aktionen zur Problemlösung möglich ist [Mit97]. Entsprechend Abschnitt 6.4.2 können Lernverfahren auch auf stochastische Verfahren aufbauen, um so das Verhalten von Agenten in der nahen Zukunft 'vorausberechnen' zu können. Neben den Lernverfahren sind für die Auswertung von gesammelten Daten auch statistische Methoden hilfreich, insbesondere die Prognose, wann bestimmte Ereignisse wieder eintreten werden (siehe 6.4.1).

2.1.8 InteRRaP Agentenarchitektur

Es existiert bereits eine Fülle von Agentenarchitekturen, die die genannten Eigenschaften und Fähigkeiten eines Agenten implementieren, wie z.B. BDI-, PRS-Agenten. Eine Architektur, die am DFKI in der Multiagentengruppen entwickelt wurde, ist die so genannte *InteRRaP*-Agentenarchitektur [FK93], [Mül96] (*INTEgration of Reactivity and RAtional Planning*). InteRRaP ist eine geschichtete Architektur (siehe Abbildung 2.6), in der jede höhere Ebene eine höhere Abstraktion als die untere aufweist [Mül94] [MP94] [MPT94]. Zudem werden die drei Ebenen in zwei Module geteilt: Ein Modul enthält die Wissensbasis jeder Ebene und das andere enthält verschiedene Kontrollkomponenten, die mit den korrespondierenden Ebenen der Wissensbasis interagieren.

Die unterste Ebene der InteRRaP-Architektur besteht aus einer *world interface control* Komponente mit einer entsprechenden *world model* Wissensbasis. Das Weltinterface dient als Schnittstelle zwischen dem Agenten und seiner Umwelt, um Aktionen auszuführen, Perzepte zu empfangen und mit anderen Agenten zu kommunizieren. Über der world-interface-Komponente steht die behaviour-based-Komponente. Die Aufgabe dieser Ebene ist die Umsetzung und Kontrolle der reaktiven Fähigkeiten des Agenten. Diese Ebene manipuliert eine Menge von Verhaltensmustern (patterns of behaviour - PoB). Eine PoB besteht aus einer Struktur, die eine Vorbedingung enthält, welche angibt, wann diese PoB aktiviert wird. Diese Struktur enthält weitere Bedingungen, die beschreiben, unter welchen Umständen angenommen wird, dass die Ausführung der PoB erfolgreich ist. Als letzte Elemente enthält eine PoB eine *post-condition* (ähnlich STRIPS [FN71]) und ein Funktion, die beschreibt welche Aktionen ausgeführt werden, falls diese PoB angewandt wird. Die darüber liegende Ebene der InteRRaP-Architektur enthält die *plan-based* Komponente. Diese enthält neben einen Planer alle Komponenten, die zur Verwaltung der lokalen Pläne notwendig sind. Die oberste Ebene besteht aus einer *co-operation* Komponente, mit deren Hilfe es möglich ist, mit anderen Agenten zu interagieren und gemeinsame Pläne zu entwickeln. Diese Pläne werden unter Einbeziehung der *plan-based* Komponente erzeugt.

Der Kontrollfluss in InteRRaP ist sowohl daten-, wie auch zielgesteuert. Der Eingang von Perzepten wird durch das world-interface organisiert und führt zu einer Veränderungen des Weltmodells. Als Reaktion von Veränderungen des Weltmodells, werden unterschiedliche Verhaltensmuster aktiviert, beendet oder ausgeführt. Die Ausführungen einer PoB werden durch die plan-based und die

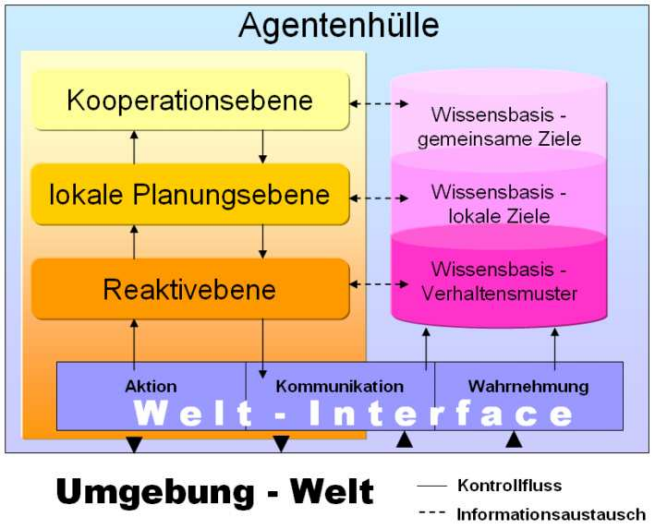

Abbildung 2.6: Aufbau der InteRRaP-Agentenarchitektur

co-operation Komponenten angestoßen, neue eigene Pläne und daraus resultierend neue agentenübergreifende Pläne zu erzeugen, um die Ziele des Agenten wieder zu erreichen. Dies führt schließlich zur Ausführung von Aktionen und zur Erzeugung von Nachrichten, ausgeführt durch das world-interface.

Generische InteRRaP-Agentenmodellierung

Agententechnologie stellt unter anderem auch hilfreiche Konzepte für das Design und die Implementierung von großskalierbaren, verteilten Softwaresystemen zur Verfügung [Wei99]. Ähnlich dem objektorientierten Ansätzen, stellt der agentenorientierte Entwicklungsansatz eine logische Weiterführung von Objekten dar. Objekte besitzen keine Eigenverantwortung und Autonomie. Agenten erweitern Objekte, indem sie wie Objekt reaktiv sind, zusätzlich jedoch auch pro-aktiv, autonom und kommunikativ handeln können. Agenten eignen sich daher besonders für den Einsatz in komplexen, verteilten Anwendungsdomänen, bei denen ein hohes Maß an Robustheit gegen Ausfälle, Flexibilität und realistisches Verhalten gefordert werden. Aufgrund der hohen Modularität, bedingt durch die Zuweisung einzelner Funktionen zu verschiedene Agenten und Offenheit in solchen Systemen (Agenten können jederzeit der Gesellschaft beitreten bzw. aus ihr herausgenommen werden) eignen sich Multiagenten-Systeme zudem auch besonders gut für die Wiederverwendbarkeit von Programmcode in neuen Systemen. Die Kommunikation der Agenten bietet so eine standardisierte Schnittstelle, weshalb die Erweiterung und Wiederverwendbarkeit besser als in herkömmlichen Systemen möglich ist.

Eigenschaften, wie Reaktivität, Planung, Kommunikation und Mobilität werden sehr häufig von Agenten eines Multiagenten-Systems gefordert. Können diese Funktionalitäten der Agenten generalisiert werden, dann steht den Entwicklern ein System zur Verfügung, welches sich lediglich durch domänen-

spezifische Anpassungen einiger Funktionen die Entwicklungszeit stark reduzieren lässt. Agenten-Designmethoden und -Architekturen bieten konzeptionelle Hilfsmittel für die Entwicklung von agentenorientierten Anwendungen, jedoch unterstützen sie nicht den Implementierungsprozess an sich. Agentengesellschaften können entsprechend dem holonischen Grundprinzip strukturiert werden [GSV99b], wodurch auch die Anwendungsdomäne in verschiedene Abstraktionsebenen aufgeteilt wird. Die Agenten an sich können beispielsweise unter Verwendung der InteRRaP-Agentenarchitektur [Mül96] entwickelt werden, die sich durch die Aufteilung in reaktives Verhalten, deliberatives Planen und Interaktionen mit der Umwelt besonders für die Wiederverwendbarkeit von Softwaremodulen eignet.

Für Anwendungen, die auf ähnlichen Konzepten beruhen, wurde eine hierarchisches objektorientiertes Modell entwickelt, um die Systementwicklung weitestgehend auf anwendungsspezifische Anpassung zu reduzieren. An der Spitze der Hierarchie stehen generische Objekte, die komplett unabhängig von der Anwendungsdomäne sind, wie z.B. Techniken zur Kommunikationsinfrastruktur (Versenden von Sprechakten) etc. Jede weitere Ebene der Hierarchie wird domänenspezifischer. So teilt sich die zweite Ebene beispielsweise in die Bereiche Logistik, e-commerce, u.a. [GVZ00]. Mit Hilfe dieses Modells steht den Entwicklern ein Baukasten zur Verfügung, der für die unterschiedlichsten Anwendungen Programmmodule bereit hält. Wurde noch kein System für ein spezielles Anwendungsszenario entwickelt, so stehen den Entwicklern zumindest die grundlegenden Module bereit, wodurch der Entwicklungsaufwand für die Agenten dennoch reduziert wird.

2.2 Holonische Agenten

Das Wort Holon [GSV99a] leitet sich aus dem griechischem Wort holos (Ganzes) und dem Suffix on (Teil) ab. Ein Holon besteht entweder aus einem Verbund von weiteren Holonen oder ist ein einzelnes Objekt. Das Holon kann dadurch eine Gruppe von Objekten als ein Ganzes repräsentieren und kann seinerseits wiederum Teil eines übergeordneten Holons sein. Zum Beispiel können Einzelaktionen als Elementarholone angesehen werden. Diese können in einer sinnvollen Reihenfolge eine komplexe Ausführung beschreiben, die dann wiederum als Ganzes ein Holon darstellen kann. Umgekehrt kann eine Folge von Planschritten durch einen holonischen Planschritt dargestellt werden, der bei Bedarf wieder in die einzelnen Aktionen dekomponiert werden kann.

2.2.1 Holonische Multiagenten-Systeme

In Multiagentensystemen wird diese Technik eingesetzt, um eine Gruppe von Agenten für einen gewissen Zeitraum zu einem Verbund unter der Führung eines ausgezeichneten Agenten, dem Holonenführer bzw. Kopf, zusammenzufassen. Diese Agenten verfolgen zu diesem Zeitpunkt mindestens ein gemeinsames Ziel. Ein Holonagent kann u.a. auf folgende zwei Arten realisiert werden:

- Eine Möglichkeit besteht darin, einen neuen Agenten zu kreieren, dessen ausschließliche Aufgabe es ist, die Verwaltung der Subagenten durchzuführen und somit das Holon gegenüber den restlichen Agenten als eine Einheit zu repräsentieren.
- Eine zweite Möglichkeit ist, einen Agenten dieser Gruppe auszuwählen, der diese Gruppe repräsentiert. In diesem Fall erbt dieser Agent Funktionen, die es ihm ermöglichen, die Funktionalität der Subagenten aufzunehmen bzw. die Anfragen an seine Subagenten zu delegieren.

Bei der zeitweisen Vereinigung zu einem Holon behalten die Subagenten weiterhin ihre Individualität. Die Agenten können das Holon jederzeit wieder verlassen, um sich einem anderen anzuschließen bzw. um eigenständig weiterzuhandeln. Nach Ablauf der 'Zusammenarbeit' verliert der ausgezeichnete Agent wieder seine zusätzlich erworbenen Fähigkeiten oder, falls es sich um einen eigens dafür erzeugten Holonagenten handelt, hört dieser danach auf zu existieren, da er nur einen temporären Agenten darstellt. Multiagenten-Systeme, die holonischen Agenten enthalten, werden als *holonic MAS* [BFV98a] bezeichnet. In einem holonischen Multiagentensystem sind autonome Agenten in der Lage, mit anderen Agenten gemeinsam ein neues Holon zu bilden. Dieses Holon ist wiederum ein autonomer Agent, weshalb eine rekursive Definition von holonischen Agenten möglich ist.

2.2.2 Einsatz holonischer Agenten in der Logistik

Unter den Einsatzgebieten für Multiagenten-Systeme existiert eine Fülle von Anwendungsdomänen, in denen eine 'intelligente verteilte Problemlösung' sinnvoll eingesetzt werden kann. Mit Hilfe eines Multiagenten-Systems können die komplexen Zusammenhänge der realen Welt auf einzelne Agenten abgebildet werden. Eine Domäne, die sich besonders für den Einsatz autonomer Agenten eignet, ist der Bereich Logistik. In Situationen, in denen ein Unternehmen seine Logistik nicht ohne fremde Hilfe durchführen kann, müssen Kooperationen mit anderen Logistikunternehmen geschlossen werden. Durch den Einsatz von holonische Agenten ist es somit zum einen möglich, ein Unternehmen mit all seinen Einheiten und Untereinheiten holonisch durch Agenten zu repräsentieren. Zum anderen können auch die Kooperationen von mehreren Unternehmen zu einem Holon zusammengefasst werden, die zu diesem Zeitpunkt ein gemeinsames Ziel verfolgen. Im nachfolgenden Kapitel beschreibe ich drei Logistikanwendungsbeispiele, die ich unter Verwendung eines holonischen Multiagenten-Systems realisiert habe.

Kapitel 3

Holonische Multiagenten-Systeme: Anwendungen und Probleme

Im Folgenden werden drei Anwendungen für Multiagenten-Systeme vorgestellt, bei denen holonische Agenten im Bereich der Logistik zum Einsatz kommen. Während ich beim ersten System für die Entwicklung und Umsetzung der fachlichen Konzepte verantwortlich war, habe ich maßgeblich das Systemdesign, die Systemarchitektur und die zugrundliegenden Konzepte für die Entwicklung der letzten beiden Systeme entwickelt. Insbesondere die Konzepte dieser Arbeit zur Koordination von komplexen Aufgaben in dynamischen Umgebungen habe ich erfolgreich im letzten Projekt umgesetzt. All diesen Systemen ist gemein, dass Koordinator-Agenten eingesetzt werden, die jeweils eine Ressourcenart (Bearbeitungseinheit, Transporter, Lager) verwalten und als Broker Angebot und Nachfrage anhand von Interaktionsregeln zusammenführen. Für die durch den Koordinator-Agenten zusammengeführten Agenten besteht so die Möglichkeit, sich zu einem Holon zusammenzuschließen, um komplexe Aufgaben in Kooperation lösen zu können. Die Subagenten des Holons verfolgen weiterhin aufgrund ihrer (Teil-) Autonomie ihre eigenen Interessen und versuchen ihren lokalen Gewinn zu maximieren. Beispielsweise können Ressourcenagenten eine möglichst gleichmäßige oder eine hohe Auslastung, aber auch geringe Rüstkosten als egoistisches Ziel verfolgen. Somit stehen die Agenten erstmals in Konkurrenz zueinander. Die Ressourcenagenten stehen oftmals in Konkurrenz zueinander, müssen trotz der Konkurrenzsituation gegebenenfalls auch miteinander kooperieren, um ihre Ziele zu erreichen. Die Kooperationsform kann dabei von einer strikt hierarchischen *Master-Slave*-Beziehung bis zu gegenseitiger gleichwertiger Kooperation reichen. Dem gegenüber können beispielsweise spezielle Agenten zur Verwaltung von Lagern konkurrierende Strategien zu anderen Lageragenten entwickeln, wenn in einer Situation zuviel freier Lagerplatz existiert. Andererseits sind gegenseitige Kooperationen von Transportagenten und Lageragenten denkbar, wenn diese gemeinsam ein besseres Angebot abgeben können. Die folgenden Anwendungsbeispiele des Logistikmanagements zeigen, wie in einem System Agenten mit unterschiedlichen Koordinierungsstrategien zusammenarbeiten.

3.1 TeleTruck

Die zunehmende Anzahl von Fahrzeugen auf den Straßen Europas, unter anderem bedingt durch die europäische Integration und die politischen Öffnung Osteuropas, führen zu einer mehr und mehr angespannten Verkehrssituation. Viele Autobahnen sind überlastet und zusätzliche Behinderungen, wie z.B. straßenerhaltende Maßnahmen verschärfen die Situation. Dabei wird es für produzierende Firmen immer wichtiger, sehr schnell die Ware zum Kunden zu liefern. *Just-in-Time*-Produktionen sind die empfindlichsten Varianten der Transportabwicklung. Bei dieser Planung der Produktfertigung wird nicht nur der zeitliche Ablauf der Montage berechnet, sondern hier ist auch die benötigte Zeit zur Lieferung des Produkts zum Kunden sehr bedeutsam. Treten bei der Auslieferung unerwartete Veränderungen bei den Vorgaben auf, wie Ausfall eines Lkws, Staus, Änderungen bei den Verkehrsanbindungen, ist es sehr wichtig, schnell darauf reagieren zu können. Aufgrund der Globalisierung und Öffnung neuer Märkte wird es für produzierende Firmen immer wichtiger, kostengünstig ihre Produkte herzustellen. Folge ist die Verlagerung der Produktion an günstigere Standorte, wodurch der Transport von Bauteilen und schließlich des Endprodukts einen immer höheren Stellenwert erlangt. Eine gute Auslastung der Transportflotte kann die Kosten des Endprodukts erheblich beeinflussen. Die Planung der Transporte von Gütern gewinnt somit eine immer höhere Bedeutung und muss sich immer flexibler an die Gegebenheiten anpassen.

Um den Anforderungen an Flexibilität, maximaler Auslastung, termingerechter Auslieferung usw. gerecht zu werden, ist die Qualität der Disposition von entscheidender Bedeutung für die Wettbewerbsfähigkeit einer Spedition. Durch die Globalisierung wachsen Speditionen zunehmend, wodurch die Planung und Koordination der Ressourcen komplexer und unüberschaubarer wird. Kann ein kleines Unternehmen mit wenigen Lkws seine Planung noch komplett manuell durchführen, ist es bei größeren Speditionen ohne Einsatz von modernen Kommunikationsmöglichkeiten und Planungssystemen nicht mehr möglich, eine kaufmännisch optimierte Planung durchzuführen. Auftragseingang, Transportplanung und -zusammenstellung, Wareneinn- und ausgang, sowie Verwaltungsaufgaben gehören zum Aufgabenbereich der Disposition einer Spedition. Bei diesen Aufgaben gilt es, seitens der IT Speditionen mit entsprechenden Programmpaketen zu unterstützen. Das TeleTruck-System [BFV98b] wurde hierbei entwickelt, die Bearbeitung der Aufgaben der Disposition zu unterstützen. Viele Speditionen nutzen die IT bereits sehr ausgiebig, jedoch geschieht die Disposition meist ausschließlich „von Hand“. Durch den immer stärker werdenden Konkurrenzdruck ist es mehr denn je wichtig, kurzfristig planen zu können und auf unvorhergesehene Ereignisse, wie Ausfall eines Lkws, Staus usw., reagieren zu können. Diesen Anforderungen gerecht zu werden ist die Zielsetzung, unter der das TeleTruck-System entwickelt wurde. Es soll den Disponenten während der Planung unterstützen, indem es Lösungsvorschläge generiert, die Verwaltung der Ressourcen der Spedition vornimmt und die Ausführung der Pläne überwacht, soweit es durch die Rückgewinnung von Information über Telefon, GPS u.a. möglich ist.

3.1.1 Anwendungsszenario

Der Einsatzbereich des TeleTruck-Systems umfasst die Unterstützung der Disposition bei der Verteilung der Aufträge auf die Lkws einer Spedition. Dazu verwaltet dieses Transportplanungssystem sowohl die eingehenden Aufträge, Pläne der Lkws, wie auch die Ressourcen der Spedition, die zur Ausführung eines Transports notwendig sind, wie die Fahrer, Zugmaschinen, Hänger, usw.

Das TeleTruck-System führt nach Auftragseingang eine vollautomatische Ressourcenverteilung und Disposition der Aufträge auf die Lkws durch. Dabei werden die Aufträge bestehenden oder bei Bedarf neu zusammengestellten Lkw-Zügen zugewiesen. Ein Lkw-Zug ist für das Planungssystem keine fixe Einheit, sondern wird aus Komponenten zusammengestellt und bei Bedarf auch umkonfiguriert. Der Begriff „Lkw-Zug“ ist lediglich ein Bezeichner für eine Zusammenstellung von Komponenten zu einem bestimmten Zeitpunkt, die in der Realität einen einsatzfähigen Lkw-Zug repräsentieren. Die Komponenten umfassen dabei Zugmaschine, Hänger, Wechselbrücke oder Container wie auch Fahrer. Dabei besteht in allen Schritten der Disposition die Möglichkeit des manuellen Eingreifens durch den Benutzer. Er ist in der Lage, nachträgliche Veränderungen an den Zugzusammenstellungen durchzuführen, sowie Ein-, Aus- und Umplanungen von Aufträgen auszuführen. Weiterhin besteht die Möglichkeit, schon eingeplante Aufträge zu modifizieren.

Das TeleTruck-System ist eine Weiterentwicklung des *MAS-MARS*-Systems (**M**ulti**A**gents**S**ystem - **M**odeling **A**utonomous coope**R**ating **S**hipping companies) [Sch95], das erstmals auf einem holonischen Multiagenten System aufbaut. Die Überarbeitung dieses Systems wurde notwendig, da das *MAS-MARS*-System lediglich einen kompletten Lkw-Zug als kleinste Einheit behandelt. Wie sich bei einer Zusammenarbeit mit einer in Saarbrücken ansässigen Spedition herausstellte, ist diese Granularität jedoch zu grob. Um die Realität besser abbilden zu können, war eine Umstrukturierung und Neuimplementierung des Transportplanungssystems notwendig. Bei dem TeleTruck-System setzt sich nun ein Lkw-Zug aus mehreren Komponenten, beispielsweise einem Fahrer, einer Zugmaschine, einem Hänger zusammen (siehe Abbildung 3.1). Diese Erweiterungen ermöglichen es, eine genauere Abbildung der Realität durchzuführen. Dabei können wesentlich mehr Randbedingungen, z.B. Fahrzeit, Laderaum, Ladungsunverträglichkeiten berücksichtigt werden. Ferner ist es durch diese Erweiterung möglich, die Zusammenstellung von Lkw-Zügen nachträglich noch zu verändern, bzw. Komponenten zu unterschiedlichen Zeiten unterschiedlichen Lkw-Zügen zuzuweisen. Dadurch wird eine zusätzliche Optimierung der Lkw-Züge nach der Größe der Aufträge möglich. Die verwendeten Verhandlungsprotokolle sind im Einzelnen das *Extended-Contract-Net*-Protokoll [San93] zur Berechnung einer Initiallösung und der Optimierungsalgorithmus *Simulated-Trading* [BHM92] zur Nachoptimierung der Initiallösung.

3.1.2 Systemarchitektur

Das TeleTruck-System besitzt aufgrund der Strukturierung und durch Einsatz von Multiagententechnologie eine sehr offene Architektur, die sich

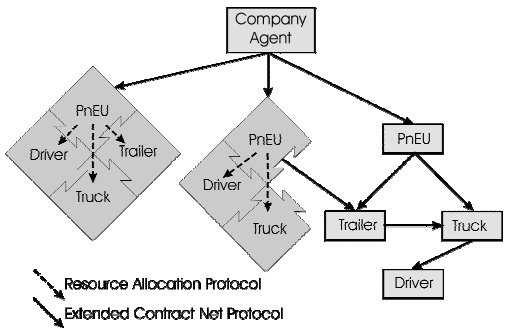

Abbildung 3.1: Holonische Struktur der Repräsentation eines LKWs

sehr leicht erweitern lässt. Durch Einsatz von Sensoren (GPS) ist eine aktive Planüberwachung möglich, d.h. der Agent kann völlig autonom Daten von der Umwelt empfangen und diese mit seinem berechneten Plan abgleichen. Die Anbindung solcher zusätzlicher Eingabemöglichkeiten kann individuell je nach dem Aufgabenfeld und den finanziellen Möglichkeiten einer Spedition ausfallen. So ist es nicht zwingend notwendig, dass jeder Lkw mit einem GPS ausgestattet wird. Durch die Ressourcenbeschreibung, die für jeden einzelnen Agenten vorgenommen werden muss, kann das TeleTruck-System sehr genau an die realen Komponenten angepasst werden, wodurch die Planung effizienter bzw. genauer vorgenommen werden kann.

Der Einsatz von zwei verschiedenen Planungsalgorithmen erlaubt eine flexible Anpassung an die Situation. Je nach dem, ob neu berechnete Pläne sofort zur Verfügung stehen sollen, wird das *Contract-Net*-Protokoll verwendet, um weitere Aufträge einzuplanen. Steht genügend Zeit zur Verfügung, kann einerseits die bereits existierende Lösung mit Hilfe des *Simulated-Trading* Optimierungsalgorithmus verbessert werden. Andererseits können auch zusätzliche Aufträge mittels dieses Algorithmus eingeplant werden. Beide Algorithmen führen zu einem zulässigen Plan, der erste bietet sehr schnell eine Lösung, der zweite benötigt mehr Zeit als der erste, erzeugt jedoch bessere Lösungen.

Das Teletruck-System hat folgende Eigenschaften:

- holonische Agenten
- offene Architektur
- individuelle Anpassung des Systems an die abzubildende Spedition
- inkrementelle Bearbeitung der Pläne mittels des *Simulated-Trading*
- ein Abbruch der Optimierung kann jederzeit durchgeführt werden (*anytime*-Algorithmus)

- dynamische Anpassung der Pläne an Umweltveränderungen
- Möglichkeit der manuellen Ausführen und Überwachen aller Funktionen durch den Benutzer

Das TeleTruck-System ist aufgrund seiner sehr hohen Reaktivität und seines dynamischen Verhaltens sehr gut geeignet, kurzfristige Planungen durchzuführen. Situationen, in denen ständig Veränderungen auftreten und Neuplanungen angestoßen werden müssen, sind die Hauptdomäne des TeleTruck-Systems.

3.1.3 Agentenbasierte Dienste

In der Regel treffen alle Aufträge, die zum nächsten Ausliefertermin ausgefahren werden sollen, frühzeitig in der Spedition ein, so dass eine Planung über alle Aufträge durchgeführt werden kann. Es bleibt also noch genügend Zeit, um nach der Berechnung der Initialpläne diese nachträglich zu optimieren. Ferner werden die Pläne im Allgemeinem so ausgeführt, wie es die Disposition vorgibt. Dies beschreibt den Idealfall, der häufig nicht der Regelfall ist. Daher muss die Disposition in der Lage sein, sehr flexibel auf Veränderung der Planung reagieren zu können. Durch die Aufteilung des Gesamtplans der Spedition in viele Teilpläne der einzelnen Agenten ist es bei einer „kleineren“ Modifikation des Gesamtplans nicht notwendig, ihn komplett neu zu berechnen. Kleine Modifikationen des Gesamtplans sind Änderungen, die an nur sehr wenigen Transportplänen durchgeführt werden müssen. Dabei ist der Plan eines Agenten nur ein kleiner Bestandteil der Gesamtlösung und bedarf daher eines wesentlich geringeren Rechenaufwands gegenüber einer kompletten Neuberechnung einer Gesamtlösung. Aus diesem Grund kann das TeleTruck-System sehr schnell einen Ersatzplan anfertigen, der aufgrund der geringfügigen Änderungen sehr „nah“ an der optimalen Lösung bleiben wird, falls die vorherige Variante optimal war. Die unbetroffenen Agenten besitzen weiterhin optimierte Pläne. Solche *kleineren* Änderungen im Plan eines Agenten sind die Folge von nachträglichen Modifikationen an bereits eingeplanten Lieferungen, wie z.B. die Terminänderungen oder Veränderung der Menge der zu transportierenden Fracht usw.

Schwerwiegender sind Änderungen, die sich auf Pläne beziehen, die bereits ausgeführt werden und daher auch schon für Änderungen gesperrt sind. Die Sperre eines Teilplans verbietet dem System im allgemeinem, diese nachträglich zu verändern. Dennoch gibt es Situationen, in denen es unumgänglich ist, auch diese Pläne zu modifizieren. Dies ist beispielsweise notwendig, falls ein LKW während der Ausführung aufgrund einer Panne ausfällt oder infolge eines Staus seinen Restplan nicht mehr termingerecht erfüllen kann. In solchen Fällen müssen alle betroffen Lieferungen dieser blockierten Lkws auf andere Lkws verschoben werden, die eventuell schon unterwegs sind. Daher muss genau angegeben werden, an welchen Orten eine Planänderung dem Fahrer mitgeteilt werden kann. Im Idealfall sind die Fahrer ständig über Mobiltelefon oder Funk erreichbar. Allerdings sollte diese Möglichkeit der Planänderung während der Planausführung nur in oben genannten Ausnahmefällen genutzt werden, um den Fahrern eine gewisse eigene Planung, z.B. ihrer Pausen, zu ermöglichen.

Ferner erzeugt eine Änderung eines Plans, der schon ausgeführt wird, eine Menge von Folgeaktionen, die anschließend ausgeführt werden müssen. Werden diese Teilpläne geändert, bedeutet dies, dass sich auch die entsprechenden Transporte ändern. Das kann sich auf die komplette Folgeverarbeitung der Transporte in der Buchhaltung auswirken, die unter Umständen komplett neu durchgeführt werden muss. Geschäftliche Vereinbarungen, wie Kosten und der damit verbundene Endpreis eines Produkts, bleiben eventuell unverändert, aber die Anlieferungstermine können sich stark verändern, wodurch die Folgeverarbeitung der Fracht auch neu geplant werden muss.

3.2 CASA

Das CASA-System ist ein weiteres Multiagenten-System aus dem Bereich der Logistik, das auf holonischen Agenten basiert. Anders als im TeleTruck-System, das als Anwendungsbeispiel die Disposition einer Spedition vorsieht, wurde ein Anwendungsfall aus dem Bereich der forstwirtschaftlichen Logistik und Vertrieb gewählt. Die Fähigkeit, Daten und Informationen mobil zu empfangen und zu versenden, unabhängig von Standort und Zeitpunkt, eröffnet neue Möglichkeiten, den Informationsfluss und die Zugänglichkeit von Daten gerade im Bereich der Forstwirtschaft zu erleichtern und zu verbessern. Besonders die Erleichterung beim Versenden von Statusmeldungen während der Holzproduktion erlaubt eine flexible, zeitnahe und exakte Anpassung an aktuelle Ereignisse und Optimierung der Geschäftsabläufe in der betreffenden Wertschöpfungskette. In der Forstwirtschaft besteht ein steigender Bedarf, die Produktion und den Verkauf der Nachfrage *Just-in-Time* anzupassen und somit flexibel auf Veränderungen reagieren zu können. Dies betrifft vor allem eine optimale und dynamische Planung der Zuteilung von Ressourcen. Das CASA Informations- und Handelnetzwerk (*CASA-ITN*) ist ein Projekt im Bereich des computergestützten integrierten Handels (*I-Commerce*). Diese neue Handelsform ist eine konsequente Weiterentwicklung der Grundideen des elektronischen Handels im Internet (E-Commerce) und hat zum Ziel:

- den Kunden mehr in die Abläufe der Bearbeitung seiner Aufträge aktiv einzubinden, auch wenn an der Auftragsbearbeitung mehrere Subunternehmen beteiligt sind,
- verwandte Prozesse während des Aufbaus und Durchführung einer Wertschöpfungskette (Supply-Chain) in der Praxis besser zu integrieren.

Dabei fallen eine Fülle von Daten an, die möglicherweise unsicher und zum Teil fehlerhaft sind. Die Herausforderung besteht darin, derartige Informationen in der Planung von Holzproduktion und im elektronischen Handel flexibel, effektiv und effizient zu verarbeiten. Das CASA ITN bietet den Teilnehmern Dienste auf der Basis von Agententechnologien an, um die Durchführung von Verhandlungen, Kommunikation und Austausch von Dokumenten, Informationen und Daten zu erleichtern und zu verbessern. Daher eignen sich solche Dienste zur Verbesserung der inner- und zwischenbetrieblichen Abläufen. Zum Beispiel wird durch die Integration von logistischen Diensten während der Planungs-

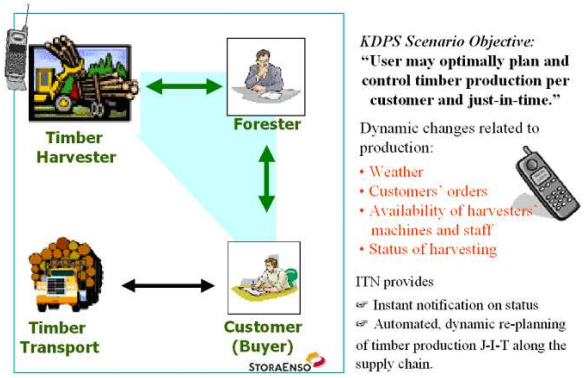

Abbildung 3.2: Zusammenfassung des KDPS Anwendungsszenarios

und Verhandlungsphasen eine bessere Abstimmung der Geschäftsabläufe und Produktionen zwischen den Partnern im ITN erreicht. Um die Verfügbarkeit der Systemdienste zu erhöhen, stehen alle Funktionen und Dienste des CASA-Systems jederzeit über das Internet zur Verfügung. Zudem wurden zusätzliche Schnittstellen zu mobilen (WAP-fähigen) Endgeräten eingerichtet, die den Teilnehmern des CASA ITN ebenfalls einen (entsprechend den Beschränkungen des WAP-Standards) eingeschränkten Zugriff auf Daten, Funktionen und Diensten ermöglichen.

3.2.1 Anwendungsszenarien

Als Grundlage für die Entwicklung von anwendungsorientierten und agentenbasierten Diensten des CASA ITN wurden verschiedene Anwendungsszenarien diskutiert. Eine kurze Übersicht dieser Szenarien wird im Folgenden gegeben:

- *Kundenbezogene, dynamische (Holz-)Produktionssteuerung (KDPS)*: Forst- und Einschlagsunternehmen kooperieren miteinander unter Zuhilfenahme von pro-aktiven Agenten, die spezielle Dienste für Logistik und Koordination im Bereich der Holzproduktion bereitstellen. Die Bearbeitung eines Auftrags hängt dabei von einer Vielzahl von Faktoren ab. So ist z.B. der Einsatz von Maschinen stark abhängig vom Wetter und dem Zustand des Bodens. Solche Faktoren wirken sich direkt auf den Produktionsablauf aus und müssen daher sofort nach Eintreten bekannt sein und in der aktuellen Planung berücksichtig werden. Dazu ist eine ständige Planüberwachung notwendig, die gegebenenfalls eine Neuplanung initiiert, die Einsatzpläne optimiert und die Zuteilung von Ressourcen neu koordiniert (siehe Abbildung 3.2). Die Integration des (Holz-)Ernteprozesses in die Planung wird im CASA ITN durch den Einsatz eines Multiagentensystems realisiert. Zur Eingabe von Wetter- und Statusmeldungen bietet das CASA ITN den Teilnehmern die Möglichkeit, diese zusätzlich auch mit WAP-fähigen Endgeräten zu versenden.

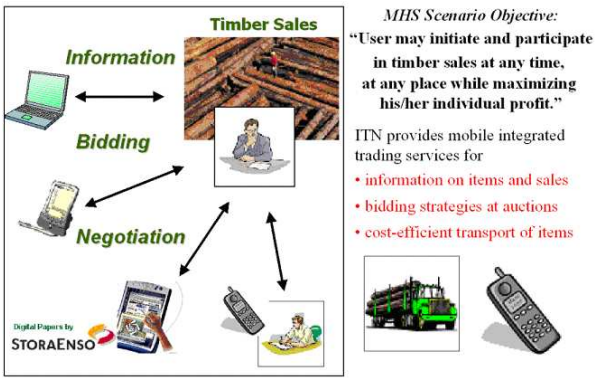

Abbildung 3.3: Zusammenfassung des MHS Anwendungsszenarios

- *Mobile Holzsubmissionen (MHS) im Internet*: Registrierte Benutzer des CASA ITN können zum Verkauf ihrer forstwirtschaftlichen Produkte Auktionen verschiedenen Typs starten bzw. zu deren Einkauf an Auktionen teilnehmen. Dabei wird auch hier dem Anwender zusätzlich die Möglichkeit gegeben, seine Eingaben nicht nur über einen am Internet angeschlossenen PC, sondern auch über mobile WAP-fähige Endgeräte zu versenden (siehe Abbildung 3.3).

Die beiden Szenarien sind im Bereich der Forstwirtschaft angesiedelt und zielen auf die integrierte Realisierung einer Supply-Chain ab, beginnend mit der Produktionsplanung, deren Ausführung, Handhabung der Probleme während der Ausführung, Durchführung der logistischen Aufgaben bis hin zum Verkauf der produzierten Ware.

3.2.2 Systemarchitektur

Die agentenbasierte Architektur des CASA ITN unterscheidet zwischen folgenden Anwendergruppen:

- Hersteller
- Käufer
- Zwischenhändler
- Logistikfirmen

Jeder Teilnehmer des CASA ITN wird durch geeignete, speziell für die Zielgruppe entwickelte Softwareagenten repräsentiert. Da die Unternehmensstrukturen der avisierten ITN-Teilnehmer unter Umständen sehr komplex sind, ist es generell nicht möglich, jeden dieser Teilnehmer jeweils nur durch einen einzelnen Agenten zu repräsentieren (siehe Abbildung 3.4). Trotzdem müssen individuelle Teilnehmer im ITN eindeutig repräsentiert werden. Um eine hierfür geeignete Kapselung von Agenten zu erreichen, die somit nach außen als ein einzelner Agenten wirkt, wurden beim CASA ITN holonische Agenten eingesetzt. Ein

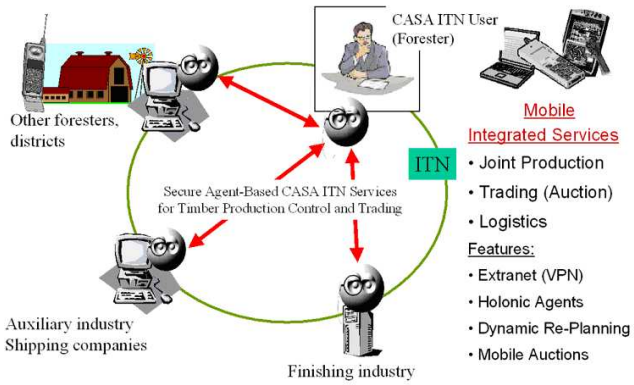

Abbildung 3.4: Makrostruktur des CASA Informations- und Handelsnetzwerks

Benutzer im CASA ITN wird durch einen speziellen Agententyp dargestellt, der für dessen Repräsentation im ITN zuständig ist. Dieser Agent wird 'personal agent' genannt und bildet eine Schnittstelle zwischen Benutzer und System. Nachdem der Benutzer ihm seine individuellen Präferenzen übertragen hat, ist dieser Agent in der Lage, an Stelle des Benutzers dessen Anweisungen autonom und pro-aktiv auszuführen. Dabei ist es nicht notwendig, dass der Benutzer im System angemeldet bleibt.

Der Agent existiert somit unabhängig von dem Einwahlverhalten seines zugewiesenen Benutzers. Ist der 'personal agent' einmal gestartet, bleibt er solange im System aktiv, wie sein zugehöriger Benutzer als Mitglied des ITN registriert ist. Für die Bearbeitung seiner Aufgaben kooperiert er auf geeignete Weise mit anderen spezialisierten Agenten, die für Verhandlungen, Teilnahme an Auktionen, Auffinden relevanter Informationen und Evaluierung von bestmöglichen Bietstrategien verantwortlich sind. Ein Agentenholon für ein im CASA ITN registriertes Unternehmen umfasst eine Menge von (wiederum holonisch strukturierten) Agenten, die jeweils eine separate Abteilung des Unternehmens funktional abbilden. Dienste werden dem jeweilig übergeordneten Holon zur Verfügung gestellt. Da die Rolle eines Teilnehmers des CASA ITN zwischen Anbieter/Produzent und Käufer/weiterverarbeitender Betrieb ständig wechseln kann, werden beide Rollen durch dieselbe holonische Agentenstruktur repräsentiert. So kann ein weiterverarbeitender Betrieb erst als Einkäufer auftreten und dann, nachdem er seine Produktion abgeschlossen hat, als Verkäufer.

Im CASA ITN wird zwischen Unternehmen mit und solche ohne eigene Logistik unterschieden. Generell ist die Logistik als eigenständige Einheit zu betrachten, die entweder als fest mit einem Unternehmen assoziiert betrachtet wird, oder je nach Bedarf unter Vertrag genommen werden kann. Ein Outsourcing logistischer Funktionen erspart einem Unternehmen mit stark schwankendem Logistikbedarf hohe Kosten in einer Leerlaufphase, erhöht jedoch den Koordinationsaufwand mit entsprechender Unterstützung durch geeignete Softwaresysteme. Das CASA ITN bietet zudem seinen Benutzern agentenbasierte

Dienste an, die es ihnen ermöglichen, einen virtuellen Marktplatz zu installieren und unabhängig vom jeweiligen Standort zu betreiben bzw. zu nutzen. Insbesondere Dienste zu Verhandlungsmechanismen erlauben den Handel zwischen Teilnehmern des CASA ITNs auf verschiedene Art und Weise: Waren können über verschiedene Arten von Auktionen gehandelt oder mit einem festen bzw. variablen Preis zum Kauf angeboten werden. Letzteres entspricht eher einer passiven Form der Vermarktung (die kaum eine Automatisierung zulässt); dagegen sind die Handelsaktivitäten über Auktionen in hohem Grade automatisierbar.

3.2.3 Agentenbasierte Dienste

Der Einsatz von Agententechnologie ermöglicht die effiziente Koordination der Ausführung einer Reihe von anwendungsorientierten Diensten, die allen Teilnehmern des CASA ITN angeboten werden können. Dabei bezieht sich die Anwendungsorientierung vor allem auf die nachfolgend beschriebenen Anwendungsszenarien.

- Diverse Auktionsmechanismen (Holländische, Englische, Vickrey und First-Price-Sealed-Bid Auktionen - siehe Abschnitt 6.2.2).

- Integrierte Dienste zur dynamischen Kostenbestimmung. Agenten sammeln zusätzliche Informationen bezüglich Transportkosten und den Rahmenbedingungen des Transports (Zeit, Dauer, Umfang, Transportkapazität und Kosten), die eine Entscheidungshilfe während einer laufenden Verhandlung sein kann. Durch Verwendung dieser Dienste ist ein Teilnehmer einer Auktion in der Lage, sich über die aktuellen Transportmöglichkeiten und -kosten zu informieren und somit sein Bietverhalten der aktuellen Situation anzupassen.

- Logistik-Dienste ermöglichen eine dynamischen und approximativ optimale Berechnung von Einsatzplänen und die Verteilung der existierenden Ressourcen auf eingehende Aufträge. Für die Berechnung einer Initiallösung, die nicht notwendigerweise optimal ist, wird das Contract-Net Protokoll verwendet. Um die Optimalität der Pläne zu gewährleisten, muss eine automatische Nachoptimierung durchgeführt werden. Dazu wird eine geeignete Abwandlung des Simulated-Trading Optimierungsalgorithmus verwendet, wie sie in [Tic03] beschrieben ist.

- Spezielle Agenten zur Informationsmanagement bereiten relevante Informationen eines Geschäftsprozesses entsprechend dem anfragenden Benutzer auf und bieten ihm somit eine rollenspezifische Informationshandhabung an.

- Mobile Dienste ermöglichen es den Teilnehmern, wesentliche Dienste des CASA ITNs zu verwenden, unabhängig von ihrem Aufenthaltsort, durch den Einsatz von WAP-tauglichen Endgeräten.

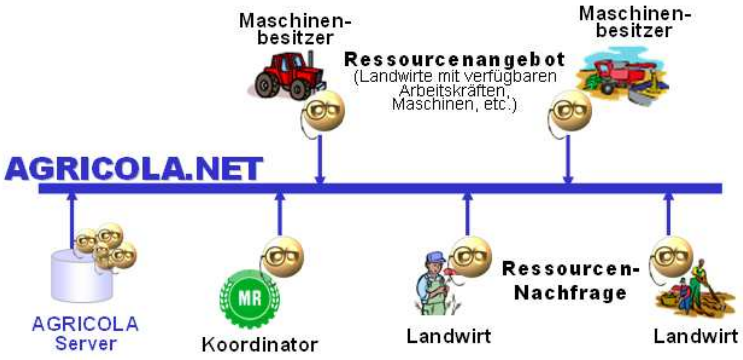

Abbildung 3.5: Teilnehmer im AGRICOLA.NET

3.3 AGRICOLA

Das AGRICOLA.NET Netzwerk bietet Landwirten eine innovative, softwaretechnische Unterstützung für bedarfsgerechte Just-in-Time Einsatzplanung von Maschinenparks in der landwirtschaftlichen Wertschöpfung. Die lokalen Maschineneinsatzplanungen von Landwirten vor Ort werden mit den partiell globalen Planungen der Maschinenbesitzer unter sich dynamisch verändernden Bedingungen hinsichtlich Witterung, sowie Ausfall von Maschinen, Personal und sonstigem Material bedarfsgerecht und Just-in-Time aufeinander abgestimmt. Eine Maschineneinsatzplanung durch Maschinenbesitzer umfasst die Faktoren Personal, Wartung und geographische Positionierung von Maschinen; sie ist während der Erntezeit zeitkritisch und kostenaufwendig. Ein für eine bestimmte Region zuständiger Maschinenring vermittelt zwischen Landwirten, die Maschinen für ihre operativen Prozesse benötigen, und einem oder mehreren Maschinenbesitzern mit entsprechend verfügbaren Maschinenparks. Die agentenbasierten Dienste des AGRICOLA.NET erlauben es, eine dynamische Umplanung von Maschineneinsätzen durch den Landwirt als auch den Maschinenbesitzer zeit- und ortsunabhängig durchzuführen und dabei die individuellen Planungen der betreffenden Teilnehmer automatisch mit einzubeziehen. Für die Realisierung der Dienste des Netzwerks AGRICOLA.NET werden Agententechnologie, neue Planungsverfahren der künstlichen Intelligenz, Internet und mobile Telekommunikation integrativ eingesetzt.

3.3.1 Anwendungsszenario

Im Projekt AGRICOLA wird von einer Aufteilung eines landwirtschaftlichen Betriebs in landwirtschaftliche Planungsprozesse, sowie Verwaltung und Koordination des Maschinenparks und des Personals als Dienstleister ausgegangen. Die Einsatzplanung findet auf drei Ebenen statt:

- Planung auf Aufgabenebene
- Koordination innerhalb eines Prozesses

- Übergreifende Planung (Koordination der Abhängigkeiten zwischen den Prozessen)

Das Anwendungsszenario von AGRICOLA befasst sich mit der Organisation der Arbeitsschritte, die zur Bearbeitung eines Feldes während der Ernte notwendig sind. Dazu sind allen registrierten Teilnehmer im Dienstnetzwerk AGRICOLA.NET spezielle Rollen zugewiesen (siehe Abbildung 3.5). Um eine konzeptionelle Trennung zwischen Koordination, Ressourcen- und Aufgabenverwaltung zu erreichen, wird eine Aufteilung der Funktionsgruppen in entsprechende Rollen wie folgt vorgenommen:

- *Landwirte* führen die Aufgabenverwaltung ihrer Prozesse durch. Sie besitzen dabei keine eigenen Ressourcen zur Bearbeitung dieser Aufgaben.

- *Maschinenbesitzer* bieten Ressourcen für die Ausführung von Aufgaben den Landwirten an.

- *Koordinatoren* vermitteln Teilnehmer mit Ressourcenbedarf und Ressourcenangebot.

Im internetbasierten AGRICOLA.NET werden alle Teilnehmer und Ressourcen durch geeignete Agenten repräsentiert. Persönliche AGRICOLA Agenten unterstützen die Teilnehmer in ihrer Planung von Ausleihe und Einsatz benötigter Nutzfahrzeuge für die Ernte, wie beispielsweise Drillmaschinen zur Aussaat, Mähdrescher, Traktoren zum Schwaden und Wenden, oder Schlepper zum Transport der Ernte oder sonstiger Güter. Alle agentenbasierten Dienste von AGRICOLA sind über mobile Telekommunikationsgeräte jederzeit an jedem Ort verfügbar. Der Landwirt wird unter Einsatz des AGRICOLA Systems in die Lage versetzt, seine individuelle Planung von ernterelevanten Arbeitsverfahren auf seinen Feldern in sogenannten Verfahrensketten (teil-) automatisiert durchzuführen. Die Planung einer solchen Verfahrenskette setzt sich aus der Auswahl einzelner Aufgaben (wie z.B. drillen, ernten, schwaden, usw.) und der Festlegung ihrer chronologischen Reihenfolge zusammen. Steht die Verfahrenskette fest, müssen geeignete Dienste von Ressourcenanbietern zur Bearbeitung der einzelnen Aufgaben gefunden werden. Dabei stehen dem Anwender folgende Möglichkeiten zur Verfügung.

- Eigenhändige Auswahl und Zuweisung von gewünschten Ressourcen zu den Aufgaben. Diese Vorgehensweise beschränkt sich auf Ressourcen, auf die dieser Teilnehmer uneingeschränkten Zugriff besitzt. In der Regel sind dies seine eigenen Ressourcen, bzw. Ressourcen von Teilnehmern (im System als Freundemarkiert), die ihm die vollständige Kontrolle ihrer Ressourcen überlassen.

- Zuweisung von Aufgaben zu Gruppen von Ressourcen, die für die Bearbeitung dieser Aufgabe relevant sind und auf die zugegriffen werden kann. Das System führt dann automatisch eine Ausschreibung der Aufgaben an diese Gruppen aus und plant sie ein.

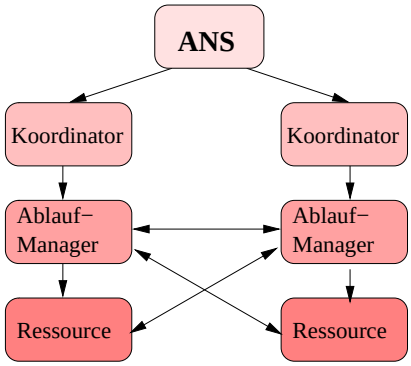

Abbildung 3.6: Koordinationsstruktur von AGRICOLA.NET Agenten

- Globale Ausschreibung von Aufgaben an alle im Netz vertretenen Teilnehmer, die Ressourcen besitzen. Je nach dem Grad des Vertrauens der Teilnehmer im AGRICOLA.NET untereinander, können eingehende Anfragen durch das System entweder direkt beantwortet oder lediglich als Anzeige für eine spätere manuelle Bearbeitung durch den Ressourcenanbieter bereitgehalten werden.

Im AGRICOLA.NET können Ressourcen und Aufgaben gezielt an Teilnehmer des Netzwerks ausgeschrieben und ihnen jederzeit die Zugriffsrechte darauf wieder entzogen werden. Darüber hinaus können einzelne Teilnehmer zwischen Angebot und Nachfrage von Ressourcen verschiedener Teilnehmer vermitteln. Ist die Zuweisung abgeschlossen, müssen die Ressourcen real gebucht werden. Das System unterstützt den Anwender hierbei durch automatisierte Verhandlungen nach einem erweiterten Kontraktnetzprotokoll für Multiagenten-Systeme.

3.3.2 Systemarchitektur

Die Anwendung von AGRICOLA.NET wird auf verschiedene über das Internet verbundene Computersysteme (PC, Notebook, und PDA) der Anwender verteilt. Die AGRICOLA Agenten werden auf den lokalen Systemen jeweils als eine Art *'fat Clients'* von AGRICOLA.NET installiert, die eigenständig umfangreiche Funktionen ausführen können. Jeder Agent kann über einen definierten, pro Agent eindeutigen Port mit anderen Agenten kommunizieren und verfügt über einen Gelbe-Seiten Dienst für die ihm bekannten Agenten. Ein zentraler Agent-Name-Service Agent (ANS) (siehe Abbildung 3.6) auf dem sog. AGRICOLA Server in AGRICOLA.NET verwaltet zudem alle Adressen der aktiven Agenten des Multiagenten-Systems und kann von jedem Agenten im Netz erreicht werden. Dies ermöglicht die Einbeziehung von im Einsatz befindlichen Ressourcen in den kooperativen Planungsprozess über eine satellitengestützte Positionsbestimmung (GPS). So können z.B. die Daten von PDA oder GPS-Agenten zunächst an den zentralen und statischen ANS-Agenten geschickt werden, mit der Zusatzinformation, diese Daten an den jeweiligen Ressourcen-Agenten des GPS-Agenten weiterzuleiten.

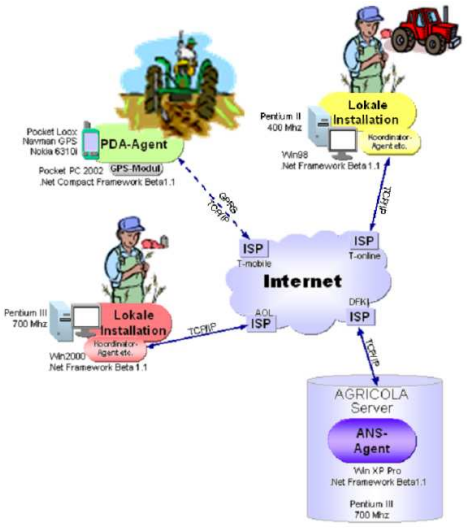

Abbildung 3.7: Technische Infrastruktur von AGRICOLA.NET

Die Topologie von AGRICOLA.NET kann sich ohne Einschränkung der Dienste verändern, beispielsweise bei Ausfall oder Neustart von Systemen registrierter Teilnehmer mit eventuell veränderter Lokation und dynamischer Vergabe von IP Adressen durch den Internet Service Provider (ISP) (siehe Abbildung 3.7).

3.3.3 Agentenbasierte Dienste

Zur mobilen Datenerfassung und -übertragung werden speziell ausgestattete PDAs mit integrierten GPS-Empfängern und einer integrierten Mobilfunkeinheit zur Datenübertragung eingesetzt. Für einen Datenabgleich werden die betreffenden Daten per Funk über eine TCP/IP-Verbindung zu den jeweiligen Agenten übertragen. Analog können Agenten auf stationären PCs Daten an die Agenten auf den mobilen Endgeräten senden und so z.B. einen Anwender vor Ort rechtzeitig von einer Planänderung informieren. Der Einsatz von GPRS / GSM-Modems in Kombination mit PDA ermöglicht, dass die Teilnehmer sich jederzeit und von jedem Ort mit dem Internet verbinden können und auf die Dienste des Systems direkt zugreifen können. In dem Anwendungsszenario wird eine lokale Bedarfs- und Maschineneinsatzplanung von Landwirten vor Ort mit aktuell verfügbaren, partiell globalen Planungen von geeigneten Anbietern bedarfsgerecht und Just-in-Time aufeinander abgestimmt. Die Maschineneinsatzplanung wird von den Anbietern durchgeführt und die Ergebnisse den Teilnehmern zugänglich gemacht. Die Planung einer Ressourcenverteilung kann durch eine Menge von sich dynamisch verändernden Bedingungen stark beeinflusst werden. Betrachtete Störfaktoren für die Planung in AGRICOLA sind

- Witterung
- Ausfall von Personal

- Ausfall von Maschinen
- Ausfall sonstiger notwendiger Materialien
- Biologische Störungen der Pflanzenproduktion

Auf Anfrage kann ein für eine Region zuständiger Koordinatoragent als eine globale Koordinationsstelle vermittelnd zwischen Landwirten und Maschinenbesitzern eingreifen. Eine derartige Vermittlung reicht von einer Auswahl von geeigneten Anbietern landwirtschaftlicher Nutzmaschinen bis hin zu Vorschlägen für die Maschineneinsatzplanung. Als Ergebnis von Bedarfs- und Einsatzplanungen der Teilnehmer werden entsprechende aufgabenorientierte Kooperationen gebildet. Eine orts- und zeitgenaue GPS-basierte Datenerfassung ermöglicht zudem eine präzise Abrechnung zwischen den Kooperationspartnern nach Abschluss eines Wertschöpfungsprozesses. Das AGRICOLA.NET ist in der Forschungsgruppe Multiagentensysteme am Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) in Saarbrücken entwickelt und vollständig implementiert worden. Dieses System erweitert das agentenbasierte Informations- und Handelsnetzwerk für die Forstwirtschaft CASA und ist von Anwendern in der Region erfolgreich getestet worden. Das Multiagentensystem und die Dienste des AGRICOLA.NET sind in der Programmiersprache C# unter .NET für die Betriebssystemplattformen Windows 98/2000/XP entwickelt worden. Alle Dienste des AGRICOLA.NET sind nach Installation der AGRICOLA Software auf dem jeweiligen Client über das Internet verfügbar. Die mobilen Dienste sind exemplarisch für den PDA 'Pocket Loox' der Firma Fujitsu-Siemens AG entwickelt worden (siehe Abbildung 3.8). Die mobile Datenübertragung vom PDA zum Netzwerk erfolgt mit Hilfe eines per Bluetooth mit dem PDA verbundenen GPRS-fähigen Mobiltelefons Nokia 6310i. Für die ortsabhängigen mobilen Dienste von AGRICOLA.NET ist der PDA mit einem zusätzlichen GPS-Empfänger ausgerüstet worden. Um das Problem der flexiblen Koordination von komplexen Abläufen und ihrer Planung in sich stetig ändernden Umgebungen zu lösen, wurden im Rahmen des Projekts effiziente Verfahren zur dynamischen Holonenbildung entwickelt und getestet. Ziel dieser Verfahren ist es, den einzelnen Agenten zu erlauben, rational kooperative Verbände (Teams, Koalitionen, Holonen) zu bilden, mit deren Hilfe sie komplexe Aufgabenstellungen gemeinsam bearbeiten, die sie allein nicht oder nur kaum lösen können. In offenen Umgebungen können jedoch

- die Agenten jederzeit einem Holon beitreten bzw. sie verlassen,
- die Aufgabenbeschreibung, wie auch die Ressourcen der Agenten sich fortwährend ändern und
- die kommunizierten Informationen zum Teil unsicher bzw. vage sein.

Es besteht daher das Problem, stabile Holonen zu bilden, ohne im Fall von Störungen des Systems ständig alle Holone wieder neu verhandeln zu müssen.

Abbildung 3.8: Funktionen eines AGRICOLA Agenten auf einem PDA

3.4 Probleme der Kooperation in anwendungsorientierten dynamischen Umgebungen

Die Entwicklung der drei aufgeführten Systeme ist aufeinander aufbauend und zeigt einen deutlichen Trend hin zu immer flexibleren Systemen. Sowohl *pervasive computing*, wie auch die Generalisierung von Koordinations- und Kooperationsprozessen an sich, werden von immer größerer Bedeutung für die Anwendungen. Ein besonderer Augenmerk bei der Entwicklung des letzten Systems lag für mich darin, ein Baukastensystem für Multiagenten-Systeme in dynamischen, nicht-benevolenten Umgebungen im Bereich Logistik zu entwickeln. Mit Hilfe eines solch modularen Systems, das zur Laufzeit an die entsprechenden Fallbeispiele angepasst werden kann, besteht im Gegensatz zu den ersten beiden Systemen die Möglichkeit, mit geringerem Programmieraufwand die Agentengesellschaft an die Anwendungsdomäne anzupassen. Das Customizing geschieht dann über ein Konfigurationswerkzeug, das Bestandteil von AGRICOLA ist. Dies betrifft neben der Definition der Agenten, Aufgaben und Dienste, auch die Zusammensetzung der Hierarchien innerhalb der Gesellschaft und die freie Definition der Abläufe zur Bearbeitung bestimmter Aufgabentypen. Dazu war es notwendig die Modellierung der Arbeitsabläufe zu generalisieren und für die Beschreibung der Welt und ihrer Objekte (Agenten, Holonen, Aufgaben und Dienste) einen Formalismus zu finden, der zur Laufzeit Modifikationen erlaubt. Die teils sehr statischen Ansätze der vorherigen Systeme konnten hierbei nicht angewandt werden. Im folgenden Kapitel habe ich eine Methodik zum Entwurf generischer Multiagenten-Systeme entwickelt, die speziell für Planungs-, Koordinations- und Kooperationsprozessen einsetzbar ist. Die Wiederverwendbarkeit nicht nur von einzelnen Komponenten, sondern auch von kompletten Multiagenten-Systemen steht hierbei im Vordergrund.

Kapitel 4

Dynamische Koordination von Arbeitsabläufen

Eine Analyse der Anwendungsbeispiele aus dem vorherigen Kapitel zeigt, dass ein reeller Bedarf an Planungssystemen zur dynamischen Koordination von komplexen Arbeitsabläufen existiert. Arbeitsabläufe beschreiben hierbei eine definierte logische sowie chronologische Abfolge von einzelnen Aufgaben, die zum Teil von einander abhängig sind. Zur Verwaltung und Organisation von Arbeitsabläufen habe ich speziell im Projekt Agricola ein System entwickelt, das dem Anwender die Möglichkeit gibt, eigene Arbeitsabläufe zusammenzustellen, die Planung und Ausführung der einzelnen Aufgaben zu überwachen und die Ressourcenverteilung dynamisch und *Just-in-Time* zu koordinieren.

4.1 Konzeptionelle Grundlagen - Arbeitsabläufe und Workflow-Management Systeme

In Systemen zur Verwaltung und Organisation von Arbeitsabläufen wird der Informationsfluss so automatisiert, dass die Weitergabe, Bearbeitung und der Abschluss von Arbeitsaufträgen innerhalb strategischer, funktionaler oder administrativer Geschäftsabläufe erleichtert wird. Je schneller und reibungsloser die Kommunikation abläuft, umso geringer sind die administrativen Kosten. Eine leistungsfähige Lösung erfasst alle Geschäftsprozesse, die bei der Umsetzung von Arbeitsabläufen anfallen. Dem Mitarbeiter stehen dabei zu jedem Schritt eines Arbeitsablaufs Informationen in der richtigen Aufbereitung und Anwendung zur Verfügung. Routinearbeiten und Rückfragen beschränken sich auf ein Minimum. Die einzelnen Arbeitsschritte sind automatisierbar, kontrollierbar und jederzeit abfragbar, es entsteht eine prozessorientierte computergestützte Vorgangsbearbeitung. Die Anwender können den Ablauf, sowie Funktionalität der einzelnen Arbeitsschritte eigenständig definieren und ihre individuellen Daten, gewünschte Statusinformationen und Laufwege ergänzen. Automatisiert kann der Anwender einen Vorgang an den nächsten Teilnehmer des Arbeitsablaufs weiterleiten. Dieser erhält dann seine Aufgaben elektronisch in seinem persönlichen Eingangskorb. Analog den Workflow-Management-Systemen, wie sie beispielsweise durch SAP im R/3-System [R39] und in Zope [LP01] zum Einsatz kommen, habe ich

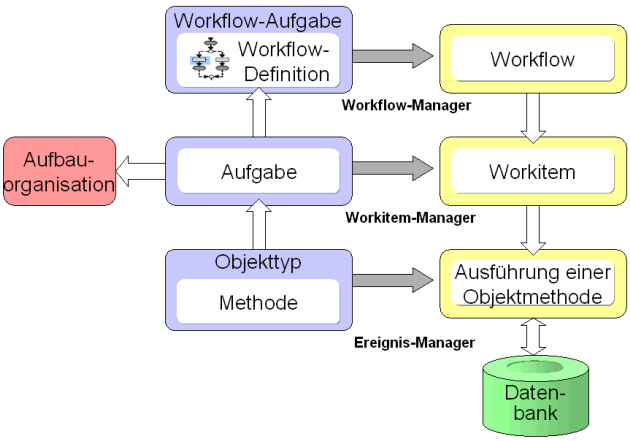

Abbildung 4.1: Allgemeiner Aufbau eines Workflows zur Abbildung von Geschäftsprozessen

eine Variante für Multiagenten-Systeme entwickelt, die die Organisation von Arbeitsläufen unter Einsatz von Agenten und deren Diensten ermöglicht. Abbildung 4.1 zeigt zunächst den allgemeinen Aufbau eines Workflow-Management Systems, das als Grundlage für die Entwicklung eines Ablaufmanager-Agenten dient.

Eine Analyse der Vorgehensweise in Workflow-Management-Systemen dient als Ausgangspunkt für die weiteren Betrachtungen. Zur Definition einer Aufgabe werden aus einer Menge von Objekttypen den Aufgaben Objekte zugewiesen, so dass die Aufgabenstellung durch die Methoden der Objekte bearbeitet werden kann. Aufgaben bilden die Basis für die Erstellung von komplexen Arbeitsabläufen, die in so genannten *Workflow-Definitionen* zusammengefasst sind. Aus einer Workflow-Definition können so beliebig viele *Workflows* als Laufzeitinstanzen abgeleitet werden. Dem *Workflow-Manager* obliegt die Steuerung und Ablaufkontrolle des Prozesses. Durch die Auswertung von Bedingungen über Attribute und die Berücksichtigung der Ergebnisse einzelner Schritte bestimmt er die Bearbeitung der anstehenden nächsten Schritte, deren korrekte Abarbeitung durch ihn kontrolliert und koordiniert wird. Ein Workflow steht hierbei als Bezeichnung für arbeitsteilige Prozesse, die ein Spektrum von einfachen Prozessen bis zu komplexen, organisationsweiten bzw. organisationsübergreifenden Vorgängen erfasst.

Jeder Workflow besteht aus einer Folge von Workitems, die zur Laufzeit erzeugt werden. Die einzelnen Schritte einer Workflow-Definition werden durch je ein Workitem beschrieben. Workitems werden dabei entweder durch menschliche Bearbeiter oder durch Systeme im Hintergrund ausgeführt. Demnach nennen sie sich entweder *Dialog-Workitems* oder *Hintergrund-Workitems*. Dialog-Workitems können nur von Mitarbeitern, die organisatorisch für ihre Bearbeitung vorgesehen sind, ausgeführt werden. Die Abwicklung der Ausführung einzelner Workitems einschließlich der Zuordnung zu Bearbeitern und der Terminüberwachung ist Aufgabe des Workitem-Managers. Über Ereignisse werden

Informationsänderungen an Anwendungsobjekten systemweit bekannt gemacht. Ereignisse sind durch den Objekttyp, dem sie etwas mitteilen, und durch ihren Namen identifiziert. Außerhalb der Standardanwendung ist definiert, welche Reaktionen auf das Auftreten eines Ereignisses folgen. Ereignisse können Workflows starten oder bereits „laufende“ Workflows, die auf spezielle Ereignisse warten, reaktivieren. Der Ereignis-Manager nimmt die Zuordnung der Aufgaben zu Ereignissen vor.

4.2 Aufbau eines Arbeitsablauf-Management-Systems

Für den Aufbau eines Arbeitsablaufsmanagement-Systems müssen zuerst die Datenklassen und Agententypen definiert werden. Die Abbildung der Verfahren und Abläufe einer Anwendungsdomäne auf Multiagenten-Systeme wird dabei in Analogie zu Workflow-Systemen durchgeführt (siehe Abbildung 4.2). Dazu muss das Modell eines Workflow-Systems um eine zusätzliche Ebene erweitert werden. Zwischen den Elementen und der Struktur eines Workflows ist es notwendig, eine weitere Ebene einzuführen, die die Dienste und resultierenden Aktivitäten der Agenten geeignet darstellt. Entsprechend ihren Fähigkeiten werden Objekttypen den Agenten zugewiesen, die zur Realisierung von Diensten eingesetzt werden.

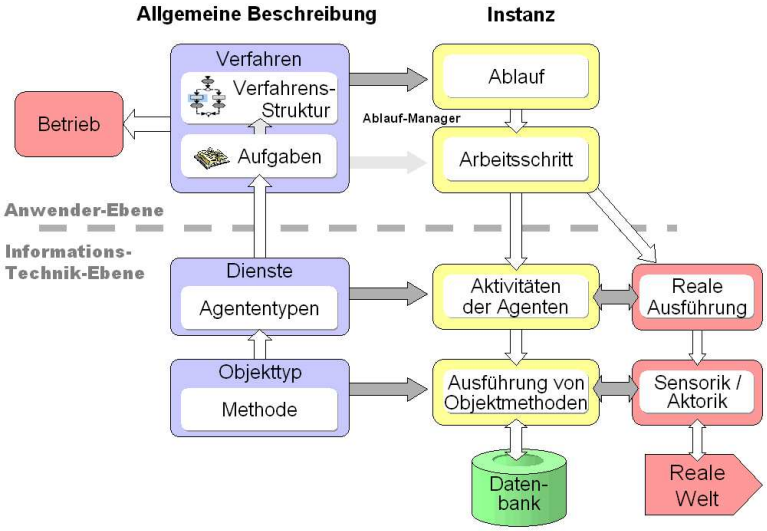

Abbildung 4.2: Anpassung eines Workflow-Systems auf Verfahrensstrukturen am Beispiel der Landwirtschaftdomäne

Dienste klassifizieren bereits kleinere Organisationseinheiten von Agenten, wo Agenten Bereiche ihrer Fähigkeiten in einem Dienst zusammenfassen und der Aufgabenbearbeitung zur Verfügung stellen. Die Ausführung von Diensten führt zu Aktivitäten der Agenten, die mit der Ausführung in der realen Welt synchronisiert werden müssen. Der Austausch von elektronischen Daten geschieht durch Sensoren, die es dem Agenten ermöglichen, eigenständig Informationen

zu erlangen und bei Einsatz von Aktoren sogar Entitäten der realen Welt zu steuern.

Aufgaben können mit Hilfe von Verfahrensstrukturen zu komplexeren Abläufen zusammengestellt werden, wodurch zeitliche wie auch bedingte Abhängigkeiten zwischen der Ausführung verschiedener Aufgaben beschreibbar sind. Verfahrensstrukturen und Aufgaben zusammen bilden ein Verfahren, das zu einem konkreten Ablauf analog einem Workflow instanziiert wird. Der Ablauf führt in der Ausführung zu Arbeitsschritten, die durch die einzelnen Schritte eines Ablaufs repräsentiert werden.

4.2.1 Dienste

Dienste werden im Allgemeinen von den Teilnehmern angeboten und durch Agenten ausgeführt. Sie stellen die Grundlage zur Bearbeitung einer Aufgabe dar. Ein Dienst organisiert und verwaltet die speziellen Einsatzmöglichkeiten der Agenten (Ressourcen und Methoden) so, dass komplexere Aufgaben im Verbund aus mehreren Ressourcen gelöst werden können. Die Koordination der Detailabläufe obliegt dem Dienstanbieter und nicht dem Aufgabensteller, wodurch eine Trennung zwischen Maßnahmenplanung und Ressourcenplanung vollzogen werden kann.

Definition 4.1. Die Datenstruktur eines *Dienstobjekts* wird durch folgendes 5-Tupel repräsentiert:

- Dienstbezeichner
- Dienstanbieter
- Menge zugeordneter Aufgabenklassen
- Zustandsbeschreibung des ausführenden Agenten
- textuelle Aufgabenbeschreibung

Beispiel 4.1. Ein Landwirt, der eine Maschine besitzt, kann diese Ressource anderen Landwirten anbieten kann. Damit jedoch klar ist, welche Funktionen mit dieser Ressource möglich sind, werden Dienste definiert und Ressourcen zugewiesen. Dies kann zum Beispiel das Häckseln von Stroh mit einem mobilen Feldhäcksler sein. Der Besitzer eines solchen mobilen Feldhäckslers kann somit die Dienstleistung 'Häckseln' bereitstellen, die von den anderen Landwirten für eine entsprechende Aufgabe bei Bedarf eingekauft werden kann.

4.2.2 Aufgaben

Aufgaben gliedern sich in zwei Aufgabentypen (siehe Abbildung 4.3):

- *Mehrschrittaufgaben* liegen weitere Verfahren zu Grunde und werden zur Schachtelung von Prozessen eingesetzt.
- *Einzelschrittaufgaben* hingegen referieren auf einen einzigen Dienst.

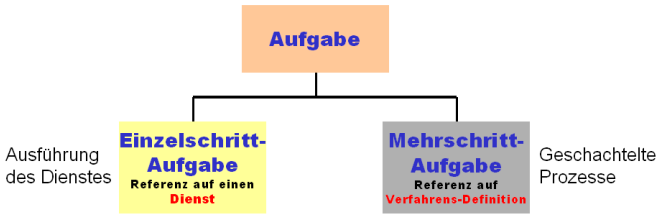

Abbildung 4.3: Strukturierung von Aufgabenklassen

Einzelschrittaufgaben

Eine Einzelschrittaufgabe ist ein elementarer Datentyp und kann nicht in weitere Teilaufgaben unterteilt werden. Eine Einzelschrittaufgabe kann direkt von einem Agenten ausgeführt werden, ohne dass eine Kooperation mit anderen Agenten erforderlich ist. Der Datentyp Einzelschrittaufgabe besteht aus so genannten Kopfdaten und einer Beschreibung. Die Kopfdaten bestehen aus einer eindeutigen Aufgabenbezeichnung, Aufgabensteller, Ausführer und einer Beschreibung der einsetzbaren Dienstklasse, die für die Bearbeitung dieser Aufgabe zulässig ist. Die Aufgabenbeschreibung enthält zusätzlich noch weitere Informationen über die Vorgaben und Restriktionen bezüglich des Zustands, den ein Agent erfüllen muss, damit er die Aufgabe erfolgreich bearbeiten kann. Diese domänenspezifischen Parameter beschreiben den Zustand eines Agenten und sind in Abhängigkeit des jeweiligen Anwendungsfalls zu wählen. Der Aufgabe kann zusätzlich eine textuelle Beschreibung beigefügt werden, um die Verständlichkeit für den Anwender zu erhöhen. Diese Angaben werden weder von den Agenten analysiert noch verwendet.

Definition 4.2. Eine *Einzelschrittaufgabe* wird durch folgendes 6-Tupel repräsentiert:

- Einzelschrittaufgabenbezeichner / Indikator
- Aufgabensteller
- Ausführer
- Notwendiger Dienst
- Zustandsbeschreibung eines möglichen Agenten
- Textuelle Aufgabenbeschreibung

Beispiel 4.2. In der Landwirtschaft fallen während des Ernteprozess mehrere Tätigkeiten an, die zum Teil parallel oder nacheinander ausgeführt werden müssen. Eine Einzelschrittaufgabe innerhalb eines solchen komplexen Prozesses ist beispielsweise der Transport der Ernteguts von Acker zum Lagersilo.

Mehrschrittaufgaben

Mehrschrittaufgaben sind analog der Einzelschrittaufgabe strukturiert. Sie unterscheiden sich jedoch in der Angabe des Dienstes. Einzelschrittaufgaben können nicht weiter unterteilt werden, wodurch ein Dienst eindeutig einer Einzelschrittaufgabe zugewiesen werden kann. Mehrschrittaufgaben referieren auf ein (Teil-) Verfahren anstatt eines Dienstes und enthalten keine Angaben zu den Zuständen möglicher Agenten. Eine Aufgabe kann somit in weitere Teilaufgaben aufgeteilt werden, die durch eine (Teil-)Verfahrensstruktur geordnet sind.

Definition 4.3. Die Datenstruktur zur Repräsentation einer *Mehrschrittaufgabe* ist ein 5-Tupel, bestehend aus dem:

- Mehrschrittaufgabenbezeichner / Indikator
- Aufgabensteller
- Aufgabenausführer
- Zugrunde liegendes (Teil-) Verfahren / Verfahrenskette
- Textuelle Aufgabenbeschreibung

Beispiel 4.3. Eine Mehrschrittaufgabe beschreibt einen komplexen Prozess, wie in Beispiel 4.2 der Ernteprozess. Zur Ausführung der Ernte eines Weizenfeldes muss der Weizen gemäht, das Stroh gewendet, geschwadet und anschließend gehäckselt werden, wobei gleichzeitig das zerkleinerte Stroh abtransportiert und in einem Offensilo gewalzt werden muss. Diese komplexe Aufgabe kann nicht mehr von einem Einzelnen ausgeführt werden, sondern ist nur durch Kooperation mit anderen Teilnehmern möglich.

4.2.3 Verfahrensstrukturen

Die Verfahrensstruktur organisiert eine Menge von Aufgaben in einem gerichteten Graphen. Struktur und Aufgaben bilden zusammen ein Verfahren, das durch konkrete Instanziierungen ausgeführt wird. Zur Bildung eines Verfahrens werden Aufgaben aufgrund ihrer Vor- und Nachbedingungen in sinnvolle Reihenfolgen angeordnet. Dabei werden Aufgaben in der Struktur mit einer *und*-Verknüpfung verbunden, wenn diese Aufgaben parallel und unbedingt ausgeführt werden müssen. Kann hingegen erst zur Ausführungszeit ermittelt werden, welche Aufgaben zur Bearbeitung anstehen, ist eine *oder*-Verknüpfung zu wählen, die auch eine parallele Ausführung mehrerer Aufgaben erlaubt. Ist es jedoch notwendig während der Ausführung zwischen mehreren Aufgaben eine exklusiv auszuwählen, werden die Aufgaben in der Verfahrensstruktur mit einem *exklusiven-oder* verbunden. Sequentielle Bearbeitungen von Aufgaben werden durch Kanten in der Struktur dargestellt (siehe Abbildung 4.4). Da ein Verfahren auch Element einer Struktur eines übergeordneten Verfahrens sein kann, ist ein rekursiver Aufbau komplexer Strukturen möglich.

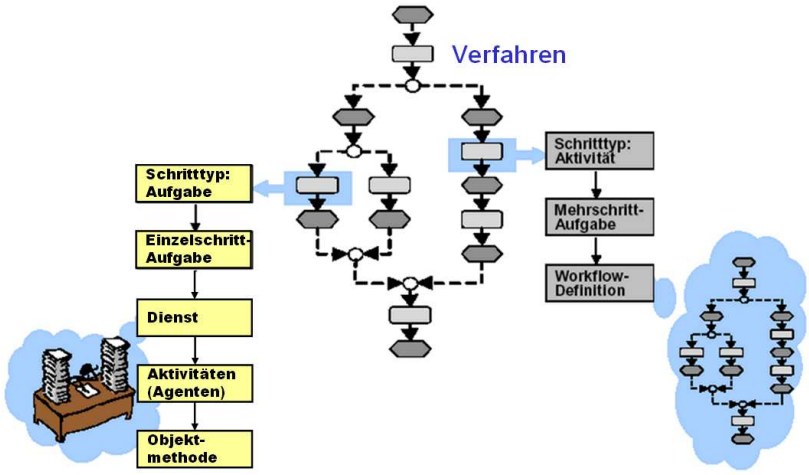

Abbildung 4.4: Aufbau einer Verfahrensstruktur

Definition 4.4. Die Datenstruktur zur Repräsentation einer *Verfahrensstruktur* besteht aus einem 5-Tupel:

- Verfahrensbezeichner
- Verfahrensanbieter
- Gerichteter Aufgabengraph
- Zustandsbeschreibung
- Textuelle Aufgabenbeschreibung

4.2.4 Dienstleister

Dienstleister verwalten und koordinieren ihre Ressourcen selbst und bieten Dienste an, die zur Bearbeitung von Aufgaben eingesetzt werden können. Dies reicht vom Ausführen einfacher Aufgaben bis zur Organisation komplexer Abläufe, bei denen mehrere Ressourcen in genau festgelegten Rollen eingesetzt werden. Für die korrekte Ausführung eines Dienstes (*Arbeitsschritt*) ist der Dienstleister eigenverantwortlich. Insbesondere bei Ereignissen, die dynamisch zur Ausführungszeit auftreten, wie z.B. Ausfälle von Ressourcen, ist es die Aufgabe des entsprechenden Dienstleisters durch geeignete Umplanungen seiner Ressourcen, oder durch Kooperation mit anderen Dienstleistern, eine Lösung des Problems zu finden.

Definition 4.5. Jeder *Dienstleister* wird im System durch folgendes 3-Tupel repräsentiert:

- Ressourcenliste der zu verwaltenden Ressourcen, repräsentiert durch Agenten

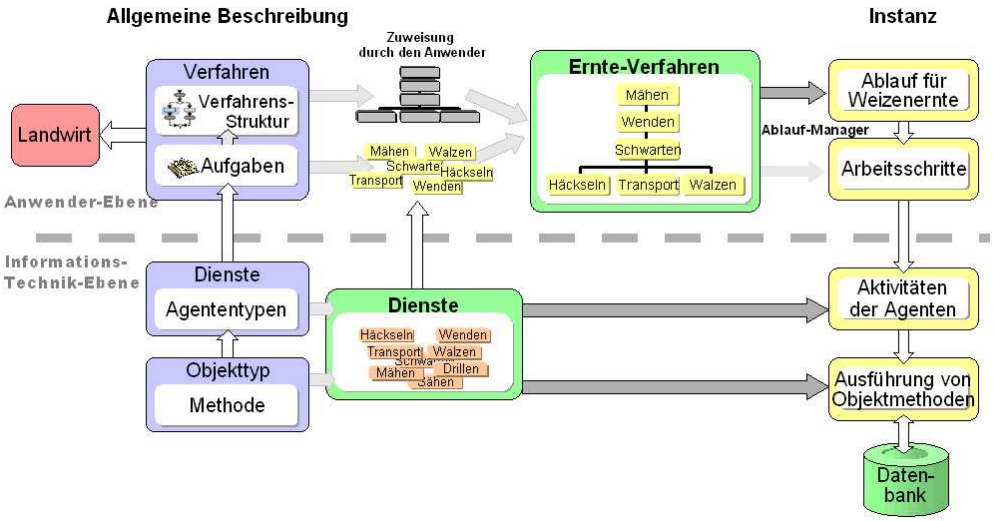

Abbildung 4.5: Beispiel einer Instanziierung aus der Landwirtschaftdomäne anhand der Weizenernte

- Aktivitätenliste, erbrachter und noch zu erfüllender Aufgaben
- Menge möglicher Dienste

Jede Ressource, repräsentiert durch einen Agenten, führt autonom ihren individuellen Einsatzplan, bestehend aus einer chronologisch sortierten Liste von Planschritten. Ein Planschritt beschreibt den Zustand eines Agenten zu einem bestimmten Zeitpunkt direkt vor der Ausführung einer Aktion. Über Sensorik und Aktoren ist es dem Agenten möglich, die reale Umwelt zu erfassen und bedingt zu steuern. Somit besitzt der Agent die Möglichkeit, auf eintretende Ereignisse reagieren zu können, indem er die Ausführbarkeit seines Plans prüft, gegebenenfalls eine Umplanung initiiert und falls notwendig die Störung an andere Agenten weiterleitet, um durch Kooperation mit anderen Agenten eine zulässige Umplanung zu finden.

Beispiel 4.4. Abbildung 4.5 zeigt die Anwendung der Ablaufmodellierung anhand der Weizenernte in der Landwirtschaftdomäne. Zunächst wird auf der informationstechnischen Ebene die Menge von möglichen Diensten ermittelt, die durch die Agenten zur Verfügung gestellt werden kann. Basierend auf dem Anwendungsszenario der Weizenernte werden die realen Abläufe unter Verwendung von Aufgaben und Verfahrensstrukturen zu komplexen Verfahren kombiniert und dabei mit den Diensten verknüpft. Die Definition eines Verfahrens auf der Anwender-Ebene gilt erst dann als abgeschlossen, wenn die realen Abläufe alle innerhalb der Verfahrensstruktur abgebildet sind und den Aufgaben entsprechende Dienste zugeordnet wurden, so dass die Erfüllung aller Aufgaben möglich ist. Anschließend wird mit der Ausführung der geplanten Abläufe begonnen. Dabei kontrolliert der Ablauf-Manager die korrekte Ausführung der Pläne und greift bei Bedarf ein, um eine Umplanung der Einsätze zu initiieren, bzw. die Anwender vor einem Scheitern des Plans zu warnen.

4.2.5 Ablauf-Manager

Ein Ablauf-Manager instanziiert Verfahren zu Abläufen und kontrolliert die korrekte Abarbeitung der Abläufe. Zur Modellierung von Verfahren steht dem Ablauf-Manager eine Menge von Aufgaben und Verfahrensstrukturen zur Verfügung, die er zu komplexen Verfahren kombinieren kann. Für die Organisation der Abläufe führt er die noch zu bearbeitenden Aufgaben und die bereits erfüllten Aufgaben in separaten Listen. Gespeichert werden diese Daten in einer Datenbank, die zur Protokollierung und Qualitätssicherung der durchgeführten und noch durchzuführenden Arbeiten dient.

Definition 4.6. Die Datenstruktur eines Objekts zur *Ablaufverwaltung* wird beschrieben durch folgendes 3-Tupel:

- Menge der möglichen Aufgaben, Verfahren und Dienste
- Aufgabenverwaltung (Instanziierung von Aufgaben)
  - Menge der noch zu bearbeitenden Aufgaben
  - Menge der abgeschlossenen Aufgaben

Beispiel 4.5. Ein Landwirt, der seine Arbeitsabläufe auf seinem Feld eigenständig verwaltet und die Koordination zwischen den einzelnen Arbeitsschritten vollzieht, übernimmt die Funktion eines Ablauf-Managers. Er gibt die Reihenfolge der Bearbeitungsschritte vor, kontrolliert deren Ausführung und modifiziert bei Bedarf den Ablauf.

4.3 Kriterien für den Einsatz von Ablauf-Manager-Agenten

Der konventionelle Ablauf von Geschäftsprozessen birgt eine Vielzahl von historisch gewachsenen Problemen und Schwachstellen in sich. Als Beispiele lassen sich folgende Problembereiche nennen:

- Lange Durchlaufzeiten durch hohe Liege- und Transportzeiten
- Mangelnde Prozesstransparenz verbunden mit hoher Arbeitsteilung
- Historisch gewachsene Aufgabenzuordnung
- Papierarchive ohne schnelle und sichere Zugriffsmöglichkeiten.

Einen Lösungsansatz stellt die Implementierung eines agentenbasierten Workflow-Management Systems, gekoppelt mit einer Umsetzung von Geschäftsprozessoptimierungen dar. Die Kernleistungen dabei sind:

- Automatisierung des Informations- und Prozessflusses
- Aktive Verknüpfung von Arbeitsschritten
- Flexible Implementierung organisatorischer Strukturen

- Repräsentation von Entitäten der realen Welt durch autonome Agenten
- Automatisierung der Organisation und Koordination von Arbeitsabläufen

Eine elektronische Vorgangsverarbeitung beschränkt sich dabei auf die Abbildung und Automatisierung strukturierter Abläufe, die

- eine Reihe von Aktivitäten umfassen,
- immer wieder in gleicher oder ähnlicher Form auftreten,
- oft mehrere Personen oder Abteilungen involvieren und
- einem starken Koordinationsbedarf unterliegen.

Ein Ablauf-Manager Agent unterstützt den Anwender auch bei der Verwaltung der verschiedenen Arbeitsabläufen zur Ausführungszeit, indem er im ständigen Kontakt zu den ausführenden Agenten steht, denen Aufgaben des Verfahrens zugewiesen wurden. Treten Probleme innerhalb des Ablaufs während der Ausführung auf, initiiert der Ablauf-Manager eigenständig Neu- bzw. Umplanungen. Gegebenenfalls müssen einzelne Aufgaben anderen Agenten zugewiesen werden, um die korrekte Bearbeitung eines Arbeitsablaufs zu gewährleisten. Der Ablauf-Manager wird durch ein Holon repräsentiert, das die Verwaltung der Abläufe transparent gegenüber den Anwendern hält. Des Weiteren wird dabei jede Instanz eines Ablaufs durch ein gesondertes Holon verwaltet. Dieser ist der alleinige Ansprechpartner bezüglich des ihm zugewiesen Ablaufs, woraus sich folgende Struktur der Agentengesellschaft (siehe Abbildung 4.6) ergibt.

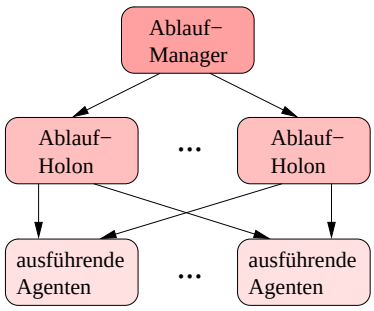

Abbildung 4.6: Holonische Agentenhierarchie zur Arbeitsablaufkoordination

In der Organisation von Versorgungsketten zur Bearbeitung von Arbeitsabläufen werden die lokale Bedarfs- und Einsatzplanung mit aktuell verfügbaren, partiell globalen Planungen entsprechender Koordinationszentren bedarfsgerecht und Just-in-Time aufeinander abgestimmt. Damit eine automatisierte Planung, die stark von den Ereignissen und äußeren Einflüssen abhängig ist, realisiert werden kann, gebe ich einen geeigneten Formalismus zur Beschreibung von Aufgaben, Dienste, Agenten und Holone im folgenden Kapitel 5 an. Darauf aufbauend werde ich im Kapitel 6 und Kapitel 7 Verfahren beschreiben, die zur Organisation und Koordination solcher hier beschriebener Arbeitsabläufen notwendig sind.

Kapitel 5

Vektorräume zur Repräsentation, Koordination und Kooperation von Multiagenten-Systemen

Die Projekte im Bereich Multiagenten-Systeme am DFKI führten meist zu einer domänenspezifischen Repräsentation der Agentengesellschaft. Dies hatte zur Folge, dass bei einem ähnlichen neuen Problem die existierenden Systeme nicht ohne einem erheblichen Programmieraufwand übernommen werden konnte. Um diese Situation zu verbessern, muss ein allgemeinerer und domänenunabhängiger Beschreibungsformalismus für Multiagentensysteme entwickelt werden. Es stellt sich die Frage, wie Agenten, Aufgaben und Dienste so repräsentiert werden können, dass eine klare formale Beschreibung existiert. Auch aus der in Kapitel 4 geschilderten Problematik der Organisation von Versorgungsketten ergibt sich die Forderung nach einer einheitlichen Beschreibung der Grundelemente eines Multiagenten-Systems. Die Beschreibungen der Elemente eines Multiagenten-Systems müssen dabei der Beschreibung der Domäne folgen, die durch eine Menge von charakterisierenden Attributen definiert wird. Nur wenn auch die Aufgaben und Dienste der Agenten die gleiche Beschreibung wie die Agenten selbst besitzen, ist es möglich Zustände zu beschreiben, die Agenten erfüllen müssen, um Dienste anbieten und Aufgaben basierend auf ihren Diensten bearbeiten zu können.

Kooperieren Agenten und verbinden sich beispielsweise zu einer neuen Struktur, reicht es nicht mehr aus, die Agenten durch einen beliebigen Vektor zu beschreiben. Es müssen Verknüpfungsregeln definiert werden. Um jedoch die Vielfalt der Vektorrechnung (Abstands-, Lagebestimmung, Prüfung von linearen Abhängigkeiten etc.) auch in Multiagenten-Systemen nutzen zu können, ist es daher notwendig eine Abbildung eines Multiagenten-Systems auf einen Vektorraum zu finden. Wenn es gelingt, durch eine geeignete Definition der Agenten und sinnvolle Verknüpfungen der Agenten untereinander einem Multiagenten-System Vektorraumcharakter zu verleihen, dann steht die gesamte Theorie der Vektorrechnung zur weiteren Bearbeitung zur Verfügung. Dazu gebe ich zuerst

eine geeignete Definition der Agenten, ihrer Dienste und Aufgaben an und zeige darauf aufbauend, dass eine Repräsentation eines Multiagenten-System durch Vektoren mit Vektorraumcharakter existiert. Die Abbildung von Multiagenten-Systemen auf Vektorräume kann jedoch nicht generell vollzogen werden. Es hängt vom jeweiligen Anwendungsfall und der Modellierung der Anwendungsdomäne ab, ob die Eigenschaften eines Vektorraums erfolgreich nachgewiesen werden können. Ist dies möglich, bietet diese Theorie manigfalltige Einsatzmöglichkeiten, so z.B. bei der Analyse von Relationen zwischen den Agenten, basierend auf einer Metrik für Matchmakingverfahren, Bestimmung alternativer Agenten zur Aufgabenbearbeitung über das Ähnlichkeitskriterium, oder bei der Ermittlung des Ressourcenbedarfs komplexer Aufgaben, für deren Bearbeitung unterschiedliche Agenten eingesetzt werden müssen, die mit Hilfe einer Linearkombination gelöst werden kann.

Ferner können Verfahren und Heuristiken aus der Computergrafik sinnvoll in diesem Kontext eingesetzt werden. Zum Beispiel können Algorithmen zur Kollisionsberechung eines grafischen Objekts mit der Topologie der Bodenkarte zum Einsatz kommen, um eine effiziente Auswahl von Agenten mit bestimmten Eigenschaften durchzuführen. Veranschaulicht entsprechen die Agenten einer Agentengesellschaft den Polygonen der Bodenkarte einer 3D-Welt, die durch Vektoren beschrieben ist und der gesuchte Agent dem grafischen Objekt. Für den Aufbau einer solchen Abbildung auf einen Vektorraum wurde ich zudem von der enormen Leistungssteigerung im Bereich der Computergrafik motiviert. Insbesondere, da bereits fotorealistische Darstellungen von menschlichen Gesichtern, detailgetreue Darstellung von Haaren etc., bereits in Echtzeit möglich sind. Die Repräsentation dieser 3D-Modelle geschieht ausschließlich durch Vektoren und überschreitet bereits heute die Grenze von einer Millionen Dreiecken, die pro Sekunde berechnet werden können. Dieses Beispiel verdeutlicht sehr genau die Mächtigkeit vektorieller Verfahren und gibt einen Ausblick auf die Mächtigkeit solcher Ansätze. Auch finden erste Bestrebungen statt [Dom], für nicht graphische vektorielle Berechnungen die GPUs (Graphic Processing Unit) der Grafikkarten zu nutzen, da diese für solche Berechnung hoch optimiert sind und diesbezüglich einen besser Performanz hinsichtlich herkömmlicher CPUs besitzen.

Dieses Kapitel bildet die Grundlage für die Entwicklung von Lernverfahren in Kapitel 6 und der Beschreibung und genauen Analyse von dynamischen Kooperationsverfahren in Kapitel 7.

5.1 Grundelemente von Multiagenten-Systemen

Die Grundelemente eines Multiagenten-System, die im Wesentlichen zur Planung, Koordination und Organisation von Verfahrensabläufen eingesetzt werden, sind:

- Aufgaben, die durch die Agenten bearbeitet werden,
- Dienste, die den Agenten entsprechend ihren Fähigkeiten zugeordnet werden und die Funktionalität eines Agenten beschreiben und

- Agenten, als autonom planende und ausführende Einheiten und Holone als Gruppenführer von Agenten.

Dienste dienen der Beschreibung der Bearbeitungsmethode eines Agenten. Welche Dienste zur Bearbeitung einer Aufgabe geeignet sind, wird bereits in der Domäne festgelegt. So ist es den Agenten aufgrund ihrer zugewiesenen Dienste möglich, zu prüfen welche Aufgaben sie bearbeiten können. Abbildung 5.1 zeigt eine mögliche Zuordnung von Diensten zu Aufgaben und Agenten. Dienste sind das Bindeglied zwischen Agenten und Aufgaben und enthalten zudem Informationen, in welchen Zuständen des Agenten sie ausgeführt werden können. Beispiel 5.1 beschreibt eine Anwendungsdomäne aus dem Bereich der Logistik, die sich in besonderer Weise für die Abbildung auf ein Multiagenten-System eignet. Die Beschreibung dieser Domäne basiert auf den drei genannten Grundelementen.

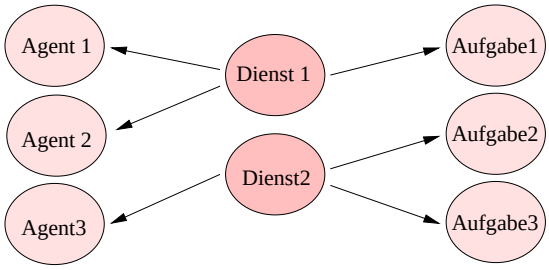

Abbildung 5.1: Zuweisungen von Dienstklassen zu Aufgaben und Agenten

Beispiel 5.1. Bei einem Transportunternehmen soll mit Hilfe eines Multiagenten-Systems die Planung flexibler gestaltet werden. Eine Spedition besitzt zwei Lastwagen (LKW_1 und LKW_2) von unterschiedlicher Bauart. Beide LKWs können Waren bis zu einem Gesamtgewicht von $7t$ *transportieren*. LKW_2 besitzt zudem zusätzlich einen Ladekran, so dass die Dienste *Laden* und *Entladen* ohne zusätzliche Ressourcen ausgeführt werden können.

Für die Organisation solcher Ressourcen, wie in Beispiel 5.1, eignet sich die Abbildung der Entitäten auf ein Multiagenten-System. Das Beispiel zeigt auch die Schwierigkeit von herkömmlichen Multiagenten-Systemen komplexe Ressourcenverbünde (wie z.B. LKW_2) zu beschreiben. Eine Möglichkeit, diese Problematik zu lösen, ist die Agentifizierung der atomaren Komponenten entsprechend ihren Diensten (hier LKW_2 mit den Diensten: Transport, Laden, Entladen) und diese zu einem Holon zusammenzufassen (siehe Kapitel 3.1). Die zugrunde liegende Idee hierbei ist, dass eine komplexe Entität durch einen Agentenverbund repräsentiert wird, wobei jeder Agent genau einen Dienst anbietet, der durch ein n-Tupel mit n-Attributen beschreibbar ist. Entsprechend den vorgegebenen Dienstklassen der Anwendungsdomäne wird einem Holon eine $n \times m$ Matrix zugeordnet, mit n = Anzahl der Agentenattribute und m = Index der möglichen Dienstklassen. Jede Spalte der Matrix steht für eine Dienstklasse und enthält die Attributwerte der Agenten bzgl. dieser Dienstklasse.

Damit eine solche Repräsentation möglich ist, muss auch die gesamte Anwendungsdomäne und all ihre atomaren Entitäten (Agenten, Aufgaben, Dienstklassen, etc.) durch n-Tupel beschrieben und die Entitätsbeschreibungen auf den $\mathbb{R}^n$ transformiert werden. $\mathcal{A}$ beschreibt dabei die Menge aller Agenten eines Multiagenten-Systems und $A, H \in \mathcal{A}$ steht für einen Agenten bzw. Holon dieses Multiagenten-Systems. Im Beispiel 5.1 werden die Entitäten der Anwendungsdomäne durch folgende Attribute charakterisiert:

a_1 Kosten pro Zeiteinheit

a_2 Verfügbarkeit (Ladekapazität)

a_3 Qualität

a_4 zulässige Gesamteinsatzzeit pro Tag

Beispiel 5.2. Aufbauend auf Beispiel 5.1 seien H_1 und H_2 zwei holonischen Agenten, die die Ressourcen LKW_1 und LKW_2 als holonische Agenten repräsentieren. Zu Beginn der Planung besitzt das Holon H_1 keine Aufgabe. Dem Holon H_2 wurde zum Zeitpunkt t_0 ebenfalls keine Aufgabe zugewiesen. Zum Zeitpunkt t_1 hingegen sollen Güter mit einem Gewicht von $2t$ transportiert werden. Wird eine weitere Transportaufgabe von $7t$ ausgeschrieben, kann das Holon H_1 jederzeit diese ausführen. Für Holon H_2 hingegen ist die Erfüllbarkeit dieser Aufgabe vom geforderten Ausführungszeitpunkt abhängig. So kann das Holon diese Aufgabe zum Zeitpunkt t_0 nur ausführen, wenn der Zeitraum $t_1 - t_0$ dafür ausreichend ist. Zum Zeitpunkt t_1 hingegen ist das Holon H_2 nicht in der Lage die vollständige Aufgabe allein zu bearbeiten und müsste daher mit H_1 eine Kooperation eingehen.

Die Beschreibung eines Agenten bzw. eines Holons muss daher zustandsabhängig dargestellt werden, d.h. die Erfassung der momentanen Attributwerte a_k sind nur für den Zeitpunkt t_0 des Erfassens gültig. Schreibweise: $a_k(t_0) \in A(t_0)$ mit $k = 1..n$ beschreibt die Attributwerte eines Agenten zum Zeitpunkt t_0. Zu einem späteren Zeitpunkt t_1 können sich die Werte bereits geändert haben, so dass die Beschreibung zum Zeitpunkt t_0 überholt sein kann. Die Attribute sind zeitabhängig und müssen ständig aktualisiert werden. Jedoch wird im Folgenden bei der Repräsentation eines Agenten auf den Zeitindex verzichtet und davon ausgegangen, dass die Attributwerte jedes Agenten aktuell sind.

5.1.1 Aufgaben

Die Zuordnung von Agenten auf n-Tupel und von Holonen auf $n \times m$ Matrizen hat auch Auswirkungen auf die Darstellung der Aufgaben, die von den Agenten bearbeiten werden sollen. Daraus ergibt sich, dass man den Begriff der „Aufgabe“ entsprechend einem Agenten definieren muss. Im obigen Beispiel 5.2 liegen Transportaufgaben vor, die durch Bedingungen bezüglich der Attribute gegeben sind. Zum Beispiel muss der ausführende Agent eine Restkapazität von mindestens $7t$ besitzen, um die Aufgabe annehmen zu können. Existieren mehrere Lösungen zur Bearbeitung einer Aufgabe, z.B. durch verschiedene Agenten mit unterschiedlichen Kosten, so muss die effektivste Lösung gewählt werden.

Aufgabenklasse	Beschreibung	Attribute	Randbedingungen
$AG^\circ{}_1$	Transport	ag_{11} = Kosten	$ag_{11} \leq 20$ EUR
		ag_{12} = Verfügbarkeit	$ag_{12} = 7t$
		ag_{13} = Qualität	$ag_{13} \geq 800$
		ag_{14} = Arbeitszeit	$ag_{14} = u$
$AG^\circ{}_2$	Beladen	ag_{21} = Kosten	$ag_{21} \leq 5$ EUR
		ag_{22} = Verfügbarkeit	$ag_{22} = u$
		ag_{23} = Qualität	$ag_{23} \geq 300$
		ag_{24} = Arbeitszeit	$t_k \leq ag_{24} \leq t_l$
$AG^\circ{}_3$	Entladen	ag_{31} = Kosten	$ag_{31} = 3$ EUR
		ag_{32} = Verfügbarkeit	$ag_{32} = 7t$
		ag_{33} = Qualität	$ag_{33} \geq 450$
		ag_{34} = Arbeitszeit	$t_m < ag_{34} < t_o$

Tabelle 5.1: Beispiele von Aufgabenklassen

Analog der Darstellung eines Agenten ist eine Aufgabe beschrieben durch ein n-Tupel der Domänen-Attribute. Aufgaben sind Instanzen von Aufgabenklassen, die je einen Aufgabentyp beschreiben. Jeder Aufgabenklasse ist genau ein Dienst zugeordnet (siehe Abbildung 5.1), der zur Bearbeitung einer Instanz dieser Klasse eingesetzt werden kann. Basierend auf Beispiel 5.2 können drei verschiedene Aufgabenklassen folgendermaßen charakterisiert werden (siehe Tabelle 5.1). Wird eine Aufgabe mit konkreten Werten festgelegt, geschieht dies auf Basis einer Aufgabenklasse. Attribute einer Aufgabenklasse, bei denen keine Bedingungen bzgl. der Ausprägung des Wertes gegeben werden können, sind im Sinne der entsprechenden Entität irrelevant für die Beschreibung und werden mit *k.A.* (*k*eine *A*ngaben) gekennzeichnet. Attributwerte, die erst zur Instanziierung einer Aufgabe bestimmbar sind, werden in der Klassendefinition einer Variablen u zugeordnet. $AG^\circ{}_1 = (\leq 20; = 7; \geq 800; u)$ wäre eine mögliche Darstellung der Aufgabeklasse $AG^\circ{}_1$. Auch die Instanziierung einer Aufgabe erzeugt durch die Randbedingungen eine Gruppe von Einzelvektoren, die die gegebenen Bedingungen erfüllen. Entsprechend für die Aufgabenklassen $AG^\circ{}_2$ und $AG^\circ{}_3$. Eine Aufgabenklasse wird damit folgendermaßen definiert:

Definition 5.1. Jeder Aufgabenklasse eines Multiagenten-Systems wird ein n-Tupel $AG^\circ{}_i = (ag^\circ{}_{i1}, ..., ag^\circ{}_{in})$ zugeordnet, wobei die Komponenten $ag^\circ{}_{ik}$ den Attributen unter Berücksichtigung der Randbedingungen der Aufgabeklasse entsprechen, mit $i = [1..m]$ Aufgabenindex und $k = [1..n]$ Attributindex.

Eine Abbildung der Attribute einer Aufgabe ist wie folgt definiert:

Definition 5.2. Sei α eine Abbildung der Attributmenge einer Aufgabe AG auf $\mathbb{R}^n$ mit $\alpha : AG \rightarrow \mathbb{R}^n$ und n der Mächtigkeit der Anwendungsdomäne. Gilt $(ag_k; val_k) \in \alpha$ für $k = [1..n]$ und $val_k \subset \mathbb{R}$, dann wird ein Attribut ag_k einer Aufgabe AG auf ein Intervall val_k durch die Abbildungsvorschrift $\alpha : ag_k \mapsto val_k$ abgebildet. Für val_k schreibt man auch $\alpha(ag_k)$, also $val_k = \alpha(ag_k)$. Für die Bild- oder Wertemenge von AG bei der Abbildung α schreibt man:
$\alpha(AG) := \{\alpha_k(ag_k) \mid \forall ag_k \in AG\}$.

Die Struktur einer Aufgabe muss dabei der des entsprechenden Dienstes

Dienstklasse	Beschreibung	Attribute	Randbedingungen
$D^\circ{}_1$	Transport	d_{11} = Kosten	$d_{11} = u$
		d_{12} = Verfügbarkeit	$d_{12} \leq 7t$
		d_{13} = Qualität	$d_{13} = u$
		d_{14} = Arbeitszeit	$d_{14} = u$
$D^\circ{}_2$	Beladen	d_{21} = Kosten	$d_{21} = 4$ EUR
		d_{22} = Verfügbarkeit	$d_{22} = u$
		d_{23} = Qualität	$d_{23} = u$
		d_{24} = Arbeitszeit	$d_{24} = u$
$D^\circ{}_3$	Entladen	d_{31} = Kosten	$d_{31} = 3$ EUR
		d_{32} = Verfügbarkeit	$d_{32} = u$
		d_{33} = Qualität	$d_{33} = u$
		d_{34} = Arbeitszeit	$d_{34} = u$

Tabelle 5.2: Beispiele von Dienstklassen

folgen, damit die Erfüllbarkeit überprüfbar ist und die vektoriellen Relationen wie Teilmengen, Mengenaddition u.ä. anwendbar sind.

5.1.2 Dienste

Im Sinne der Problemstellung sind die Aufgaben bestimmten Agenten bzw. Holonen (bei komplexen Aufgabenstrukturen) zuzuordnen: $Aufgabe \leftrightarrow Agent$. Eine konkrete Aufgabe AG_i kann nicht von jedem Agenten bzw. Holon gelöst werden, wie z.B. die Ladeaufgabe aus Beispiel 5.2. Das macht die Einführung des Begriffs *Dienst* erforderlich. Unter einem Dienst D eines Agenten versteht man eine Tätigkeit, die er ausführen kann. In ihrer Gesamtheit beschreiben Dienste das Leistungsspektrum eines Agenten bzw. Holons.

Analog einer Aufgabe sind Dienste Instanzen von Dienstklassen und werden durch die Domäne beschrieben. Anhand der Zuordnung von Dienstklassen zu Agenten, kann die Einsatzmöglichkeit jedes Agenten eindeutig beschrieben werden. Durch Einsetzen der aktuellen Attributwerte eines Agenten in die Dienstbeschreibung bzw. Dienstklasse wird ein Dienst abgeleitet und instanziiert. Die Aufgabenklassen und Dienstklassen haben die gleichen Strukturen, wodurch sie vergleichbar werden. Eine Dienstklasse ist daher folgendermaßen definiert:

Definition 5.3. Eine Dienstklasse ist eine Menge von Diensten gleichen Typs. Alle Dienste einer Dienstklasse beschreiben die gleich Funktionalität, die durch zugeordnete Agenten ausgeführt werden. Jeder Dienstklasse wird eindeutig ein n-Tupel $D^\circ{}_i = (d^\circ{}_{i1}, ..., d^\circ{}_{in})$ zugeordnet, wobei die Komponenten $d^\circ{}_{ik}$ der Attributen der Dienstklasse entsprechen, mit $i = 1..m$ Dienstindex und $k = 1..n$ Attributindex, basierend auf der Domänenbeschreibung.

Analog einer Aufgabe definieren die Randbedingungen eines Dienstes genau, unter welchen Umständen ein Dienst ausgeführt werden kann. Es ist jedoch nicht immer möglich, konkrete Werte für jedes der Attribute einer Dienstbeschreibung anzugeben, so dass Bedingungen angegeben werden müssen, die z.B. ein Intervall beschreiben, in dem der Wert des Dienstattributs sich befinden muss.

Definition 5.4. Sei $D|_{A_k}$ ein Dienst, der von einem Agenten A_k angeboten wird. Sei α eine Abbildung der Attributmenge eines Dienstes D unter Einhaltung der Randbedingungen auf $\mathbb{R}^n$ mit $\alpha : D \rightarrow \mathbb{R}^n$ und n der Mächtigkeit der Anwendungsdomäne. Gilt $(d_k; val_k) \in \alpha$ für $k = [1..n]$ und $val_k \subset \mathbb{R}$, dann wird ein Attribut d_k eines Dienstes D auf ein Intervall val_k durch die Abbildungsvorschrift $\alpha : d_k \mapsto val_k$ abgebildet. Für val_k schreibt man auch $\alpha(d_k)$, also $val_k = \alpha(d_k)$. Für die Bild- oder Wertemenge von D bei der Abbildung α schreibt man: $\alpha(D) := \{\alpha(d_k) \mid \forall d_k \in D\}$.

Jedem Attribut von D_i wird somit ein Intervall aus $\mathbb{R}$ durch eine Abbildung zugeordnet. Das n-Tupel D_i ist somit Repräsentant einer Menge von n-Tupeln in Anhängigkeit der Randbedingungen. Die Instanziierung eines Dienstes hängt dabei von dem aktuellen Zustand des Agenten ab und ist somit zeitlich variabel.

Bemerkung. Ein Dienst beschreibt die Ausführung einer Aufgabe durch einen Agenten oder Holon und eine Dienstklasse beschreibt eine Funktionalität der Agenten. Ein Unterschied zwischen Agenten und Holonen (holonische Agenten) liegt in der Anzahl der Dienstklassen, die einer solchen Entität zugewiesen ist. Einem Agenten ist genau eine Dienstklasse zugeordnet ($\alpha : A \mapsto Dienstklasse$). Einem Holon, bestehend aus mindestens einem Agenten, ist somit mindestens eine Dienstklasse zugeordnet. Da ein Agent genau einen Diensttyp ausführen kann, kann ein Holon bestehend aus n Agenten maximal n verschiedene Dienstklassen besitzen, falls nicht mindestens zwei dieser Agenten den gleichen Diensttyp anbieten. Holone dienen der Bündelung der Dienste eines Agentenverbunds.

5.1.3 Agenten

Ein Agent kann genau einen Aufgabentyp bearbeiten, weshalb ihm eine entsprechende Dienstklasse zugeordnet wird. Ähnlich den Attributen sind auch die Dienstbeschreibungen durch die eingangs eingeführte Struktur der Domäne festgelegt und die Agenten müssen dieser Beschreibungen folgen, wenn sie Dienste eines definierten Typs anbieten. Bei der Darstellung eines Agenten, wird nicht der vollständige Agent (inkl. seines Wissen und seiner Pläne) auf einen Vektor reduziert, sondern nur die situationsunabhängige, ihn bzgl. der Domäne beschreibenden Attribute, wie in Tabelle 5.3 beispielhaft skizziert ist.

Die Agenten können durch eine Matrix entsprechend obiger Tabelle dargestellt werden, die genau eine Spalte mit Werten enthält (Index der Dienstklasse). Die restlichen Spalten enthalten keine Angaben bzgl. der Attributausprägungen. So kann die Darstellung eines Agenten auf ein n-Tupel reduziert werden, woraus Definition 5.5 folgt:

Definition 5.5. Jedem Agenten A_i bezüglich einer Dienstklasse $D^\circ{}_j$ wird ein n-Tupel, darstellbar durch einen Vektor des $\mathbb{R}^n$, $A_i|_{D^\circ{}_j} = (a_{i1}|_{D^\circ{}_j}, ..., a_{in}|_{D^\circ{}_j})$ zugeordnet, wobei die Komponenten a_{ik} den Attributen des Agenten entsprechen, mit i dem Agentenindex und $k = 1..n$ dem Attributindex.

Definition 5.6. Sei α eine Abbildung der Attributmenge eines Agenten A_i auf $\mathbb{R}^n$ mit $\alpha : A_i \rightarrow \mathbb{R}^n$ und n der Mächtigkeit der Domäne. Gilt $(a_{ik}; val_{ik}) \in \alpha$ für $a_{ik} \in A_i$ und $val_{ik} \in \mathbb{R}$, dann wird ein Attribut a_{ik} eines Agenten A_i auf einen

Agent	Dienst-klasse	Beschreibung	Attribute	Randbedingungen
$A_1 = LKW_1$	$D^\circ{}_1$	Transport	a_{111} = Kosten = 25 EUR a_{112} = Verfügbarkeit = 7t a_{113} = Qualität = 1000 a_{114} = Einsatzzeit = 8Std	$d_{11} = u$ $a_{112} \leq d_{12} = 7t$ $d_{13} = u$ $d_{14} = u$
	$D^\circ{}_2$	Beladen	a_{121} = Kosten = k.A. a_{122} = Verfügbarkeit = k.A. a_{123} = Qualität = k.A. a_{124} = Arbeitszeit = k.A.	$a_{121} = d_{21} = 4$ EUR $d_{22} = u$ $d_{23} = u$ $d_{24} = u$
	$D^\circ{}_3$	Entladen	a_{131} = Kosten = k.A. a_{132} = Verfügbarkeit = k.A. a_{133} = Qualität = k.A. a_{134} = Arbeitszeit = k.A.	$a_{131} = d_{31} = 3$ EUR $d_{32} = u$ $d_{33} = u$ $d_{34} = u$
⋮	⋮	⋮	⋮	⋮
$A_4 = Krahn$	$D^\circ{}_1$	Transport	a_{411} = Kosten = k.A. a_{412} = Verfügbarkeit = k.A. a_{413} = Qualität = k.A. a_{414} = Einsatzzeit = k.A.	$d_{11} = u$ $a_{412} \leq d_{12} = 7t$ $d_{13} = u$ $d_{14} = u$
	$D^\circ{}_2$	Beladen	a_{421} = Kosten = k.A. a_{422} = Verfügbarkeit = k.A. a_{423} = Qualität = k.A. a_{424} = Arbeitszeit = k.A.	$a_{421} = d_{21} = 4$ EUR $d_{22} = u$ $d_{23} = u$ $d_{24} = u$
	$D^\circ{}_3$	Entladen	a_{431} = Kosten= 2 EUR a_{432} = Verfügbarkeit = u a_{433} = Qualität= 500 a_{434} = Arbeitszeit = u	$a_{431} = d_{31} = 3$ EUR $d_{32} = u$ $d_{33} = u$ $d_{34} = u$

Tabelle 5.3: Beispiele von Agentenmatrizen

Wert val_{ik} durch die Abbildungsvorschrift $\alpha : a_{ik} \mapsto val_{ik}$ abgebildet. Für val_{ik} schreibt man auch $\alpha(a_{ik})$, also $val_{ik} = \alpha(a_{ik})$. Für die Bild- oder Wertemenge von A_i bei der Abbildung α schreibt man: $\alpha(A_i) := \{\alpha(a_{ik}) \mid \forall a_{ik} \in A_i\}$.

5.1.4 Holone

Ein Holon setzt sich aus mehreren Agenten oder anderen holonischen Agenten rekursiv zusammen. Als Verbund ist ein Holon in der Lage, die Dienstklassen seiner Subagenten zu verwenden, entsprechend den Parametern der jeweiligen Agenten. Dies führt zu einer Beschreibung der Holonen durch eine Matrix, in der jede Zeile für ein Attribut der Domäne steht und jede Spalte die Bedingungen einer Dienstklasse seiner Subagenten beschreibt. Im Fall, dass ein Holon einen bestimmten Diensttyp nicht anbieten kann, können zu den Attributen der Dienstklasse keine Angaben gemacht werden, so dass den Attributen des Symbol *k.A.* zugewiesen wird. Ein Holon ist durch eine $n \times m$ Matrix darstellbar, mit n Attributen und m Dienstklassen (siehe Tabelle 5.4).

Beispiel 5.3. Im Beispiel 5.2 sind $H_1, H_2 \in \mathcal{A}$ zwei LKW-Holone, die sich aus insgesamt vier Agenten zusammensetzen. Diese Agenten bieten Dienste an, die durch die jeweiligen Holone koordiniert werden. So kann ein Holon, stellvertretend für die ihm untergeordneten Agenten, Dienste anbieten, die dann von den

jeweiligen Agenten ausgeführt werden. Beide Holone können einen „Transport"-Dienst anbieten, ausgeführt durch Agenten A_1 und A_2. Das Holon H_2 besteht aus zwei weiteren Agenten, so dass es zusätzlich noch zwei weitere Dienste anbieten kann, das „Laden" und „Entladen", die von den Agenten A_3 und A_4 realisiert werden. H_1 besteht aus nur einem Agenten A_1, H_2 aus einem Zusammenschluss von den drei Agenten A_2, A_3 und A_4. Beschrieben werden die Dienstklassen der Agenten durch die 4-Tupel bzgl. der Beispieldomäne.

Holon	Dienst-klasse	Beschreibung	Attribute	Randbedingungen
$H_1 = LKW_1$	$D^\circ{}_1 \rightarrow \{A_1\}$	Transport	a_{111} = Kosten =25 EUR a_{112} = Verfügbarkeit =7t a_{113} = Qualität = 1000 a_{114} = Einsatzzeit = 8Std	$d_{11} = u$ $a_{112} \leq d_{12} = 7t$ $d_{13} = u$ $d_{14} = u$
	$D^\circ{}_2 \rightarrow \emptyset$	Beladen	a_{121} = Kosten = k.A. a_{122} = Verfügbarkeit = k.A. a_{123} = Qualität = k.A. a_{124} = Arbeitszeit = k.A.	$a_{121} = d_{21} = 4$ EUR $d_{22} = u$ $d_{23} = u$ $d_{24} = u$
	$D^\circ{}_3 \rightarrow \emptyset$	Entladen	a_{131} = Kosten = k.A. a_{132} = Verfügbarkeit = k.A. a_{133} = Qualität = k.A. a_{134} = Arbeitszeit = k.A.	$a_{131} = d_{31} = 3$ EUR $d_{32} = u$ $d_{33} = u$ $d_{34} = u$
$H_2 = LKW_2$	$D^\circ{}_1 \rightarrow \{A_2\}$	Transport	a_{211} = Kosten =15 EUR a_{212} = Verfügbarkeit = 7t a_{213} = Qualität = 750 a_{214} = Einsatzzeit = 8Std	$d_{11} = u$ $a_{212} \leq d_{12} = 7t$ $d_{13} = u$ $d_{14} = u$
	$D^\circ{}_2 \rightarrow \{A_3\}$	Beladen	a_{221} = Kosten=7 EUR a_{222} = Verfügbarkeit = u a_{223} = Qualität = 310 a_{224} = Arbeitszeit = u	$a_{221} = d_{21} = 4$ EUR $d_{22} = u$ $d_{23} = u$ $d_{24} = u$
	$D^\circ{}_3 \rightarrow \{A_4\}$	Entladen	a_{231} = Kosten=2 EUR a_{232} = Verfügbarkeit = u a_{233} = Qualität = 500 a_{234} = Arbeitszeit = u	$a_{131} = d_{31} = 3$ EUR $d_{32} = u$ $d_{33} = u$ $d_{34} = u$

Tabelle 5.4: Beispiele von Holonenmatrizen

Es ergibt sich damit für das LKW_1-Holon folgende Holonenmatrix

$$H_1 = \begin{pmatrix} 25 & k.A. & k.A. \\ 7 & k.A. & k.A. \\ 1000 & k.A. & k.A. \\ 8 & k.A. & k.A. \end{pmatrix} \text{ analog für } H_2 = \begin{pmatrix} 15 & 7 & 2 \\ 7 & u & u \\ 750 & 310. & 500. \\ 8 & u & u \end{pmatrix}$$

Sollen zwei Agenten zusammengefasst werden, so kann dies nur unter Zuhilfenahme eines Holons geschehen, das die Agenten als eine Einheit repräsentiert. Agenten sind in der Holonenmatrix durch einen Spaltenvektor dargestellt. Im Beispiel 5.2 erzeugt also das Holon H_1 den Agenten A_1:

$$A_1|_{Transport} = \begin{pmatrix} 25 \\ 7 \\ 1000 \\ 8 \end{pmatrix}$$

Analog können die Agenten des Holons H_2 durch folgende Spaltenvektoren dargestellt werden.

$$A_2|_{Transport} = \begin{pmatrix} 15 \\ 7 \\ 750 \\ 8 \end{pmatrix} \; A_3|_{Beladen} = \begin{pmatrix} 7 \\ u \\ 310 \\ u \end{pmatrix} \; A_4|_{Entladen} = \begin{pmatrix} 2 \\ u \\ 500 \\ u \end{pmatrix}$$

Aus diesen Betrachtungen ergibt sich folgende Zuordnung:

Definition 5.7. Jedem Holon H_i eines Multiagenten-Systems wird eine $n \times m$-Matrix $H_i = \begin{pmatrix} a_{i11} & \cdots & a_{i1m} \\ \vdots & & \vdots \\ a_{in1} & \cdots & a_{inm} \end{pmatrix}$ zugeordnet, wobei die Komponenten a_{ikl} den Attributen der Agenten bzw. Holonen entsprechen, mit i Holonenindex, $k = 1..n$ Attributindex und $l = 1..m$ Dienstindex.

Die Extraktion eines Agentendienstvektors aus der Matrix eines Holons kann durch eine geeignete Matrizenmultiplikation mit einem Zielvektor (z.B. $\vec{z} = (0, 0, 1, 0, ..0)$ für den 3.Dienst) gebildet werden. Der resultierende Vektor beschreibt dann einen Agenten, der einen Dienst entsprechend der ausgewählten Dienstklasse anbieten kann, oder sogar eine Menge von Agenten, die alle die gleiche Dienstklasse besitzen.

Bemerkung. Wird ein Dienst von einem Agenten A_i oder einem Holon H_i ausgeführt, schreibt man dafür $A_i|_{D_i}$ bzw. $H_i|_{D_i}$. Für die Attribute des Agenten entsprechend $a_{ik}|_{D_i}$.

5.1.5 Aufgabenausschreibung

Steht eine Aufgabe zur Bearbeitung an, so müssen Agenten gefunden werden, die diese Aufgabe ausführen können. Dazu werden die Vektoren der relevanten Agenten untersucht. Sind zur Bearbeitung einer Aufgabe mehrere Agenten notwendig, so muss für jeden benötigten Dienst die relevanten Agenten auf ihre Einsatzfähigkeit gesondert untersucht werden.

Im obigen Beispiel 5.2 besteht eine Lösung des Transportproblems darin, eine Konstellation von Agenten zu finden, die die aufgeführten Aufgaben unter gewissen Randbedingungen der Aufgaben und eingesetzten Dienste lösen. Im obigen Beispiel lautet also eine Lösung der Aufgabe: Ableisten der Dienste $D_2 \rightarrow D_1 \rightarrow D_3$. Dieser geordnete Graph kann jedoch auf verschiedene Weise erfüllt werden. Holon H_2 kann zum Zeitpunkt t_0 diese Aufgabe noch allein lösen, denn zu diesem Zeitpunkt besitzt der Agent $A_2|_{Transport}$ ausreichende Transportkapazität, um den *Transport*-Dienst eigenständig auszuführen und die Agenten $A_3|_{Laden}$ und $A_4|_{Entladen}$ sind mit einem Ladekran ausgestattet, so dass das Holon die Dienste D_2 und D_3 auch ausführen kann. Da diese drei Agenten zu einem Holon zusammengefasst sind, kann das Holon H_2 diese Dienstsequenz eigenständig ohne weitere Hilfe von anderen Agenten bearbeiten. Zum Zeitpunkt t_1 ist die alleinige Ausführung jedoch nicht mehr möglich, da die Kapazität von $A_2|_{Transport}$ nicht ausreicht. Hierbei muss H_2 mit H_1 kooperieren, um gemeinsam die Aufgabe zu lösen.

Beispiel 5.4. Seien A_1 und A_2 die Agenten aus Beispiel 5.3. AG_5 bis AG_8 sind weitere Transport-Aufgaben, die von diesen beiden Agenten ausgeführt werden sollen. Die Agenten befinden sich in ihrem Ausgangszustand und verfügen so über vollständige Ressourcen zur Bearbeitung dieser Aufgaben. In Abbildung 5.2 sind die Werteintervalle der Attribute *Kosten* und *Qualität* der Agenten und Aufgaben graphisch dargestellt. Die Größe und Position der grafischen Elemente richtet sich nach den Attributwerten, bzw. den zulässigen Intervallen, vorgegeben durch die Randbedingungen der Aufgaben und Dienste der Agenten. Aufgabe AG_5 liegt in der Schnittmengen der Attributwerte beider Agenten A_1 und A_2 und kann so von beiden Agenten eigenständig bearbeitet werden. AG_6 hingegen kann nicht von A_2 allein bearbeitet werden, da die Intervalle der Attribute von A_2 außerhalb des gültigen Wertebereichs der Aufgabe liegen. A_1 kann die Aufgabe jedoch vollständig allein oder auch in Kooperation mit A_2 bearbeiten. AG_7 kann unter Umständen von A_2 allein bearbeitet werden, was jedoch erst zur Ausführungszeit bestimmt werden kann. So ist es gegebenenfalls notwendig, nachträglich eine Kooperation zwischen A_1 und A_2 zur Bearbeitung dieser Aufgabe zu bilden, oder A_1 übernimmt die alleinige Bearbeitung dieser Aufgabe. Aufgabe AG_8 kann von keinem Agenten allein gelöst werden. Nur durch die Bildung eines Holon kann diese Aufgabe bearbeitet werden. So kann durch die Kooperation beider Agenten Kosten bis zu 40 EUR pro Stunde und eine Gesamtqualität von 1750 Einheiten erzielt werden, die zur Bearbeitung der Aufgabe AG_8 ausreichen. Die Kooperation auf Aufgabenebene zwischen Agenten kann so durch ein Holon beschrieben werden.

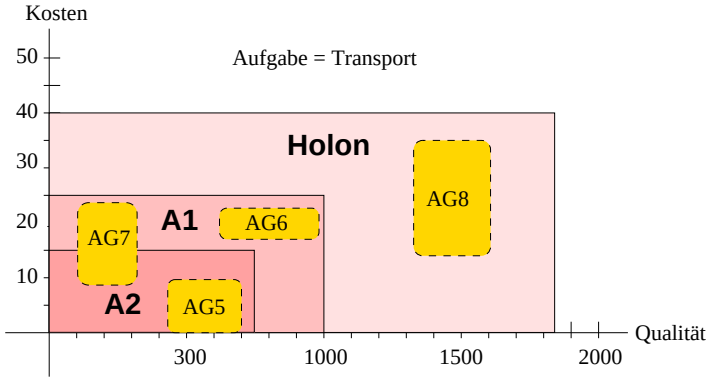

Abbildung 5.2: Interpretation der Attributwerte von Aufgaben und Agenten

Bemerkung. Damit eine eindeutige Zuweisung der Bedeutungen zu den Attributen möglich ist, müssen die Domänenattribute unabhängig voneinander definieren sein. Nur so kann eine redundanzfreie und effiziente Analyse ermöglicht werden. Existiert eine Abhängigkeit zwischen mehreren Attributen untereinander, so ist der Eigenschaftsraum überbestimmt und deren Bedeutung nicht mehr eineindeutig.

5.1.6 Plandarstellung

Ein Plan ist eine geordnete Menge von Planschritten. Jeder Planschritt steht für die Ausführung eines Dienstes und protokolliert so die Attributwerte des Agent zum Zeitpunkt t_k, des Ausführungsbeginns der geforderten k-ten Maßnahmen. Ein Planschritt gibt also die *situationsabhängige* Beschreibung eines Agenten $A(t)$ zu einem bestimmten Zeitpunkt wieder. Der beschriebene Zustand in einem Planschritt hat solange Gültigkeit, bis zu dem Zeitpunkt t_{k+1} des nachfolgendem Planschritts. Die Planschritte stehen so in Abhängigkeit zueinander und bilden chronologisch sortiert den Ablaufplan eines Agenten. Der Plan beinhaltet so zu jedem Zeitpunkt die aktuelle Beschreibung des Agenten. Der Zugriff auf einen Planschritt geschieht durch Angabe des Zeitindex t bezüglich eines Agent $A_i(t) = (a_1(t), ..., a_n(t)) =$ Beschreibender Vektor des Agenten A_i zum Zeitpunkt t. Der Plan eines Holons enthält hingegen keine Planschritte, sondern den zeitlichen Ablauf der zu erbringenden Dienste und der dazugehörigen Agenten, die er koordiniert.

5.2 Agentenvektoren

Die Theorie der Vektorräume ermöglicht formale Aussagen über die jeweiligen abstrakten Lagebeziehungen zwischen Objekten. Zur Abbildung der Elemente des Multiagenten-Systems auf den Vektorraum der reellen Zahlen wird jedem Attribut der Domäne als Wertebereich eine Dimension des Vektorraums zugewiesen und die Elemente durch Vektoren, basierend auf den Attributen der Domäne, dargestellt. Jeder Vektor, analog der elementargeometrischen Interpretation, ist dann ein Repräsentant einer Klasse von Elementen des gleichen Typs. Im folgenden wird diskutiert, ob eine solche Abbildung von Attributen auf Vektoren existiert und welche Bedeutung die Vektorrechung für Multiagenten-Systeme hat.

Beispiel 5.5. Während der Planung soll der Agent A_2 aus Beispiel 5.3 seine Kontrolle an ein übergeordnetes Holon H_2 abgeben, ohne dass der komplette Plan übergeben wird. A_2 übermittelt lediglich den Vektor eines hypothetischen Planschritts an das übergeordnete Holon. Die Bedeutung ist dabei folgende: Agent A_2 erstellt zunächst einen hypothetischen, aber noch unspezifizierten Dienst und trägt diesen in seinen eigenen Plan ein. Durch diesen hypothetischen Dienst kann der Agent A_2 den maximalen Wertbereich vorgeben, den H_2 für eine übergeordnete Planung nutzen kann. So ist es dem Holon H_2 ohne ständiges Rückfragen mit A_2 möglich, Einsätze für A_2 zu planen. Im Anschluss der Planung werden dann die Aufgabendetails an A_2 zurück übertragen.

Bemerkung. Diese Vorgehensweise ist insbesondere bei der Holonenbildung von besonderer Bedeutung, wenn Agenten Teile ihrer Autonomie abgeben und sich zu einem Verbund zusammenschließen. So kann dies durch entsprechende hypothetische Dienste geschehen, die die Subagenten dem Superagenten zur Verfügung stellen, der dann bei Bedarf auf die angebotenen Ressourcen zugreifen kann (solange das Angebot der hypothetischen Dienste Gültigkeit besitzt).

Für die Vergleichbarkeit der Eigenschaften von Agenten ist es notwendig, dass zu jeder Eigenschaft je ein Wertebereich angegeben wird und dass ein Anfangswert (Ausgangswert) existiert, so dass der Vektorraum zentriert werden kann. Wählt man einen Punkt $O \in P$ in einem n-dimensionalen Punkteraum, also einen gemeinsamen, festen Anfangspunkt, so gibt es nach der Definition eines affinen Punktraumes zu jedem Punkt $A \in P$ genau einen Agentenvektor $\vec{a} = \vec{OA}$ im Vektorraum V. Dadurch ist der Punktraum P zentriert und eine absolute Aussage beim Vergleich zweier Agenten bzw. deren Dienste ist möglich. Der Agentenvektor $\vec{a}$ ist ein Repräsentant der Attribute eines Agenten mit $\vec{a} := \vec{OA}$, also das Bild von A. $\vec{a}$ ist der Ortsvektor von A bezüglich O. Analog für die Dienste eines Agenten. Das Multiagenten-System kann so durch die Festlegung eines fixen Ausgangspunkts normiert werden.

5.2.1 Interpretation der Agentenbeschreibung

Bei der Abbildung der Agenten-Attribute und für weitere Analysen ist es wichtig, dass die Attribute auf konkrete Werte bzw. auf definierte Wertebereiche abgebildet werden. Um dies zu ermöglichen, ist es notwendig für jedes Attribut eine geeignete Interpretation der Abbildung zu finden, so dass der Wertebereich der Attribute durch die reellen Zahlen dargestellt werden kann. Entsprechend dem Beispiel 5.4 ist die Interpretation der Abbildung der ersten beiden Attribute offensichtlich. Kosten und Volumen können direkt auf die reellen Zahlen abgebildet werden. Das vierte Attribut hingegen lässt mehrere Abbildungen zu. Je nach Anwendung, kann bei einer Vereinigung von Agenten eine sequentielle Bearbeitung vorliegen, so dass die Werte dieses Attributs addiert werden können und das resultierende Ergebnis den gesamten Zeitbedarf des Verbunds beschreibt. Ist jedoch eine parallele Bearbeitung vorgesehen, so kann z.B. der Mittelwert eine brauchbare Beschreibung der Bearbeitungszeit liefern. Um den Mittelwert unabhängig der Eintrittsreihenfolge berechnen zu können, muss die Abbildungsfunktion erweitert werden. Bei der Zusammenstellung von Agenten zu einem Verbund werden bereits die Werte dieses Attributs addiert, jedoch erst bei Anfragen interpretiert. Im Fall der Mittelwertberechnung wird der Summenwert durch die Anzahl der Verbundsmitglieder dividiert, deren Attributwert ungleich dem neutralem Element „Null“ des Wertebereichs ist. Ist noch kein Agent dem Verbund beigetreten, besteht der Verbund so aus nur einem Agenten. Dabei ist diese Vorgehensweise offensichtlich gültig. Ähnlich verhält es sich bei der Interpretation der Qualität.

Die Interpretation der Attributwerte wird erst dann durchgeführt, wenn der Wert des Attributs benötigt wird. Bei der Modellierung der Qualität eines Agenten bzw. eines Verbunds von Agenten im Allgemeinen werden keine obere bzw. untere Schranke dieses Wertebereichs angegeben, so dass der Wertebereich von $-\infty$ bis ∞ reicht. In einigen Anwendungsfällen kann es jedoch notwenig sein, diesen Wertebereich zu begrenzen, um genaue Aussagen treffen zu können, wann ein Verbund die höchste Qualitätsstufe erreicht hat. Jedoch ist dann eine weitere Steigerung nicht mehr möglich und eine Differenzierung der Verbünde höchster Stufe nicht mehr möglich. Die Bewertung, welche Qualität ein Agent besitzt, ist ein subjektiver Eindruck, der nur schwer durch ein Computersystem simuliert

Attribut-Wert	-1000	-600	-300	-100	0	100	300	600	1000
Stufe	-4	-3	-2	-1	0	1	2	3	4

Tabelle 5.5: Segmentierung eines Attributs in Stufen

werden kann. Jedoch sollte die Hinzunahme eines *schwach* bewerteten Agenten zu einem bereits *stark* bewerteten Verbund, die Bewertung des Verbunds nur gering, die Hinzunahme eines *stark* bewerteten zu einem *schwachen* Verbund jedoch stark beeinflussen. Kann über einen Agenten keine Aussage bzgl. seiner Qualität getroffen werden, so wird ihm das neutrale Element des entsprechenden Wertebereichs zugewiesen. Mit steigender Qualifikation eines Agenten steigt sein Wert. Einen negativen Wert erhalten Agenten, die schlecht qualifiziert sind und deren Dienstergebnisse als mangelhaft bewertet werden. Auch hier gilt, je höher der Betrag des Wertes ist, desto stärker ist seine Ausprägung.
Eine Möglichkeit die Granularität solcher Attribute einzuschränken ist, feste Intervalle vorzugeben und die Agenten in *Stufen* je Attribut einzuteilen. Diese Standardisierung birgt den Vorteil, dass die Bewertungen verschiedener Agenten vergleichbar werden. Damit die Qualitätswerte zweier Agenten in Relation gesetzt werden können, muss eine Funktion analog dem Attribut „Arbeitszeit“ eingesetzt werden, die eine Interpretation des Wertes liefert. Um eine Einteilung der Agenten in *Qualitätsstufen* zu erreichen, die schwache und starke Ausprägungen berücksichtigt, sollten die Werteintervalle pro Stufe nicht konstant sein, sondern sollten sich mit fortschreitender Stufe erhöhen. Die Wahl der Intervallgrenzen beruht auf empirische Erfahrungswerte und ist dementsprechend stark von der Anwendungsdomäne abhängig. Eine geeignete Progression könnte z.B. sein, dass das folgende Intervall um 100 Punkte größer ist, wie das vorherige. Mit Hilfe der folgenden Formel können beispielsweise die unteren Intervallgrenzen (bzgl. des Betrags des Attributwerts) einer Stufe berechnet werden:

$$Stufe_i = \begin{cases} 0 \text{ , falls } i = 0 \\ 100 \text{ ,falls } i = 1 \\ -100 \text{ ,falls } i = -1 \\ i \cdot (i+1) \cdot 50 \text{ ,falls } i > 0 \\ i \cdot (i-1) \cdot -50 \text{ ,falls } i < 0 \end{cases}$$

Zum Erreichen der 1. Stufe (siehe Tabelle 5.5) sind mindestens 100 Punkte notwendig sind. Stufe 0 ist definiert durch das Intervall von -99 bis 99. Diese Formel erfüllt also die geforderten Bedingungen an die Auswirkung der Hinzunahme von schwachen Agenten zu einem starken Verbund. So wird z.B. ein Agent A_1 mit der Qualitätsstufe 16 ([13600;15299]) kaum durch einen Agenten A_2 mit einer Qualitätsstufe 2 ([300; 599]) beeinflusst, da der Agent A_2 maximal 599 Punkte zur Erhöhung der Qualitätsstufe beitragen kann, jedoch das Intervall von A_1 1699 Punkte umfasst. Das heißt besitzt der Agent A_1 einen Qualitätswert zwischen 13600 und 14700, so bewirkt die Hinzunahme kein Steigerung der Stufe. Sonst beträgt der Anstieg höchstens eine Stufe, ist jedoch unwahrscheinlich. Umgekehrt nimmt Agent A_2 A_1 zu seinem Verbund, ist ein wesentlich höherer Anstieg zu erwarten. Analog verhält sich die Beeinflussung

durch Agenten geringerer Qualität mit negativen Werten. Die Progression bei der Intervallsvergrößerung bei zunehmender Stufe basiert auf Erfahrungswerten und muss so individuell für jedes Anwendungsszenario gewählt werden. Eine Wahl der Progression (hier z.B. 100) kann in einem Szenario gut geeignet sein und anderen Anwendungsfällen zu schlechten Bewertungen führen. Es gilt jedoch zu beachten, dass die Einteilung der Stufenintervalle zur Laufzeit fixiert ist, sonst wäre eine vergleichbare Bewertung der Entitäten des Multiagenten-Systems durch die Agenten nicht mehr möglich. Für die Agenten aus Beispiel 5.3 wird eine Interpretation des Dienstes „Transport“ folgendermaßen beschrieben werden:

($\mathcal{I}$) Interpretation eines Transportdienstes durch einen Verbund von Agent V mit $\forall A \in V$ gilt: $A \in \mathcal{A}$:

$$Transport|_V = \begin{cases} Transport|_{v_1} - & \text{Gesamtosten des Dienstes „Transport“} \\ Transport|_{v_2} - & \text{Verfügbarkeit (Lademeter) des Verbunds} \\ Transport|_{v_3} - & \text{Zuordnung der Summe der Einzelqualitätswerte der Verbundagenten zu Qualitätsstufen} \\ Transport|_{v_4} - & \text{Summe der Einzelzeiten (seq. Bearb.) Summe der Einzelzeiten dividiert durch Anzahl von Verbundsmitgliederung mit } v_4 \neq 0 \text{ (parallele Bearb.)} \end{cases}$$

5.2.2 Agentenaddition

Die klassische Vektoraddition ist bei der Addition zweier Agentenvektoren begrenzt einsetzbar. Aufgrund der unterschiedlichen Bedeutungen der einzelnen Dimensionen des Vektorraums ist es nicht möglich, generell die arithmetische Summe für alle Dimensionen gleichermaßen zu verwenden. Beispiel 5.4 zeigt, dass für die ersten beiden Attribute die arithmetische Addition sinnvoll eingesetzt werden kann. Jedoch bei der dritten und vierten Eigenschaft ist eine modifizierte Interpretation notwendig. Bei der vierten Eigenschaft „Arbeitszeit“ existiert sogar eine Abhängig von der Aufgabenbeschreibung. Bei der Addition von Agenten zu einem Holon müssen zwei Fälle unterschieden werden. Zum einem, wenn sich zwei Agenten mit *gleicher* Dienstklasse vereinen, wodurch eine Addition der Attributwerte und Vereinigung der Randbedingungen durchgeführt werden muss, und zum anderen wenn sich zwei Agenten mit *unterschiedlichen* Dienstklassen vereinen. Da die Addition zu einem Holon führt, das durch eine Matrix dargestellt wird, ist der zweite Fall trivial, indem die Vektoren der beiden Agenten an die der Dienstklassen entsprechenden Spalten gestellt werden. Der erste Fall ist hierbei ein Sonderfall, bei dem das Holon nach der Vereinigung anstatt durch eine Matrix weiterhin auch durch einen Vektor dargestellt werden kann, da das Holon auch nach der Vereinigung nur eine Dienstklasse besitzt. Zur Addition zweier Agenten, bei der mindestens einer der beiden Agenten ein holonischer Agent ist, wird die Spalte i der Holonenmatrix[1] mit dem Vektor des

[1] i ist der Dienstklassenindex des zu vereinenden Agenten.

Agenten vektoriell addiert.

$$H \oplus A' = \begin{pmatrix} a_{11} & \dots & a_{1i} & \dots & a_{1k} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \dots & a_{ni} & \dots & a_{nk} \end{pmatrix} \oplus \begin{pmatrix} a'_{1i} \\ \vdots \\ a'_{ni} \end{pmatrix} = \begin{pmatrix} a_{11} & \dots & a_{1i} + a'_{1i} & \dots & a_{1k} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \dots & a_{ni} + a'_{ni} & \dots & a_{nk} \end{pmatrix}$$

Vereinen sich zwei Holone zu einem Superholon, so kann die resultierende Matrix des Superholons durch Addition der beiden Holonenmatrizen gebildet werden. Dieser neue holonische Agent kann bei Bedarf auch wieder in seine Einzelagenten zerfallen, wobei die entsprechenden Vektoren der austretenden Agenten von der jeweiligen Spalte der Holonenmatrix abgezogen werden.

$$H \oplus H' = \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nk} \end{pmatrix} \oplus \begin{pmatrix} a'_{11} & \dots & a'_{1k} \\ \vdots & & \vdots \\ a'_{n1} & \dots & a'_{nk} \end{pmatrix} = \begin{pmatrix} a_{11} + a'_{11} & \dots & a_{1k} + a'_{1k} \\ \vdots & & \vdots \\ a_{n1} + a'_{n1} & \dots & a_{nk} + a'_{nk} \end{pmatrix}$$

Das heißt, bei der Vereinigung zweier Holonen werden die Dienste der jeweiligen Subagenten zusammengefasst und das Resultat ist ein neues Holon mit vereinten Fähigkeiten. Das Problem der Vereinigung der Agenteneigenschaften kann so auf das Problem der Addition zweier Agentenvektoren reduziert werden, da bei einem Holon die Vereinigung spaltenweise vollzogen wird. Jede Spalte wird dabei gesondert betrachtet. Für die Addition zweier Agentenvektoren muss daher eine innere Verknüpfung definiert werden, so dass zwei Agentenvektoren $\vec{a}$ und $\vec{b}$ eindeutig einem dritten Agentenvektor $\vec{c} = \vec{a} \oplus \vec{b}$ zugewiesen werden können. Die Verknüpfung zweier Agenten zu einen Verbund wird dabei zum einem durch geeignete Rechenvorschriften $\oplus$ und zum anderen durch eine nachfolgende Interpretation der resultierenden Werte erreicht.

Beispiel 5.6. Seien A_1 und A_2 zwei Agenten aus Beispiel 5.4. Durch folgende innere Verknüpfung wird eine Vektoraddition $\oplus$: $\vec{a} \oplus \vec{b} = \vec{c}$ definiert:

(A) Addition zweier Agentenvektoren bezüglich der Domäne aus Beispiel 5.3

$$\oplus \rightarrow \begin{cases} c_1 = a_1 + b_1 \\ c_2 = a_2 + b_2 \\ c_3 = a_3 + b_3 \\ c_4 = \begin{cases} a_4 + b_4 & \text{,bei sequentieller Bearbeitung} \\ \frac{a_4 + b_4}{k} \;\Big|\; \text{,mit } k \text{ der Anzahl der Agenten, deren Attributwert} \neq 0 \text{ ist - bei paralleler Bearbeitung} \end{cases} \end{cases}$$

5.2.3 Agenten-S-Multiplikation

Bei der Ermittlung der benötigten Ressourcen zur Bearbeitung einer gestellten Aufgabe, kann es in Fällen, bei denen die geforderte Menge einer Ressource nicht durch einen einzigen Agenten gedeckt werden kann, notwendig sein, mehrere Agenten dieses Typs an diese Aufgabe zu binden. Am Beispiel einer Spedition lässt sich dieser Sachverhalt einfach zeigen. Die Spedition besitzt eine Menge von LKWs mit gleichen Eigenschaften (Ladevolumen, Fahrzeit, etc.). Sollen nun Güter transportiert werden, die das Fassungsvermögen eines LKWs

überschreiten, so müssen weitere eingesetzt werden. Mit Hilfe der Agenten S-Multiplikation kann der Faktor der benötigten Agenten eines Typs ermittelt werden, die zur vollständigen Bearbeitung der Aufgabe eingesetzt werden müssen. Analog der Vektoraddition muss die S-Multiplikation auch für den Einsatz in der Multiagenten-Theorie angepasst werden. Zum Beispiel muss entsprechend dem Beispiel 5.6, das Ergebnis einer Multiplikation des Attributs c_4 bei einer parallelen Bearbeitung anders behandelt werden, als bei einer sequentiellen. Auch hier gilt, dass nach der Verknüpfung die Ergebnisse interpretiert werden müssen. Zur Interpretation wird die gleiche Funktion $\mathcal{I}$ verwendet, wie auch bei der Vektoraddition.

Bemerkung. Analog der Addition von Agentenvektoren, werden auch bei der Agenten S-Multiplikation die Attribute der Agenten für jede Dienstklasse gesondert betrachtet. So wird bei der S-Multiplikation eines Holons jede Spalte der Holonenmatrix getrennt behandelt, wodurch die hier beschriebene Agenten-S-Multiplikation angewandt werden kann.

Beispiel 5.7. Analog dem Beispiel 5.6 der Vektoraddition für Agenten muss die Verknüpfungsvorschrift angepasst werden, damit die ursprünglichen Bedeutungen der Eigenschaften auch nach dem Anwenden des Operators erhalten bleiben. Eine S-Multiplikation für die im Beispiel 5.3 skizzierten Agenten, wird hier exemplarisch durch folgende äußere Verknüpfungsvorschrift definiert:

$$\odot \rightarrow \begin{cases} c_1 = s \cdot a_1, \text{ elementargeometrische S-Multiplikation} \\ c_2 = s \cdot a_2 \\ c_3 = s \cdot a_3 \\ c_4 = \begin{cases} s \cdot a_4, \text{ bei sequentieller Bearbeitung} \\ a_4 \text{ bei paralleler Bearbeitung} \end{cases} \end{cases}$$

5.2.4 Besondere Agenten

Neutraler Agent

Basierend auf der Definition A.13 ist ein neutraler Agent, ein Agent ohne Dienstklasse, d.h. sein Verhalten gegenüber einer Anfrage können keine Angaben gemacht werden. Eine Verknüpfung mit dem neutralen Agenten hat also keine Auswirkung auf die restlichen Agenten. Einem neutralen Agenten ist somit in allen Attributen das neutrale Element der entsprechenden Dimension zugewiesen.

Beispiel 5.8. Entsprechend der Agenten-Vektoraddition aus Beispiel 5.6 ist der neutrale Agent wie folgt definiert:

$$N \rightarrow \begin{cases} n_1 = 0 \\ n_2 = 0 \\ n_3 = 0 \\ n_4 = 0 \end{cases}$$

Bei der S-Multiplikation für Agenten aus Beispiel 5.7, ist das neutrale Element n die 1, denn: $n \odot A = A$.

Inverser Agent

Inverse Agenten sind Agenten für die gilt $A^{-1} \oplus A = N$. Dabei sind alle Werte der Attribute A^{-1} invers zu den Attributwerten von A. Jedoch ist diese Aussage allgemein nicht immer zutreffend, weshalb nicht jedes Multiagenten-System eine Gruppe bildet. Die Eigenschaften und Verknüpfungsvorschriften müssen daher entsprechend definiert werden. Betrachtet man die Agentendefinition aus Beispiel 5.6, so ist es möglich einen inversen Agenten anzugeben.

Beispiel 5.9. Zwei Agenten A und A^{-1} sind invers zu einander, wenn die Attributwerte sich gegenseitig aufheben, so dass der neutrale Agent das Ergebnis der Verknüpfung ist: $A \oplus A^{-1} = N$. Sei A ein Agent aus Beispiel 5.3. Ein inverser Agent A^{-1} hat dann folgende Attributwerte:

$$A^{-1} = \begin{cases} a_1^{-1} = -a_1 = -25 \\ a_2^{-1} = -a_2 = -7 \\ a_3^{-1} = -a_3 = -1000 \\ a_4^{-1} = -a_4 = -8 \end{cases}$$

Anschaulich bedeutet A^{-1}: Statt Kosten zu verursachen, bringt der Agent ein Guthaben von $25, -EUR$, jedoch hat er einen Mehrbedarf an Transportkapazität von 7 Tonnen, die er selbst nicht erbringen kann. Dieser Agent gilt als unverlässlich, weshalb seine Qualität einen negativen Wert von -1000 hat. Er erspart den Verbund Arbeitszeit von 8 Stunden, da er z.B. zu einem früheren Zeitpunkt in Vorleistung getreten ist und daher einen Arbeitszeitpuffer besitzt. Führt man beide Agenten in einem Verbund zusammen und verteilt die Aufgaben beider Agenten auf diese Gruppe neu, so neutralisieren sich ihre Eigenschaften bezüglich der Problemstellung und das Ergebnis ist der neutrale Agent. Dies trifft sowohl für eine sequentielle Bearbeitung, wie auch für die parallele Bearbeitungsweise zu.

5.3 Der Agenten-Vektorraum

5.3.1 Mulitagenten-Systeme als Vektorraum

Definition 5.8. Wenn eine Menge V von Agentenvektoren zusammen mit einer inneren Verknüpfung $\oplus$ eine kommutative Gruppe gemäß der Definition A.13 und einen Körper $(K, \oplus, \cdot)$ zusammen mit einer äußeren Verknüpfung $\odot$ von V gemäß Definition A.16 bildet, so heißt diese Menge ein *Vektorraum* über K (geschrieben $(V, K, \oplus, \odot)$ oder einfach V), wenn für alle Elemente in K und V folgende Axiome erfüllt sind: $\forall r_1, r_2 \in \mathbb{R}$ und $\forall \vec{a}, \vec{b} \in V$ gilt:

(V_1) Assoziativgesetz: $r_1 \odot (r_2 \odot \vec{a}) = (r_1 \cdot r_2) \odot \vec{a}$

(V_2) S-Distributivgesetz: $(r_1 + r_2) \odot \vec{a} = (r_1 \odot \vec{a}) \oplus (r_2 \odot \vec{a})$
V-Distributivgesetz: $r_1 \odot (\vec{a} \oplus \vec{b}) = (r_1 \odot \vec{a}) \oplus (r_1 \odot \vec{b})$

(V_3) Unitäres Gesetz: $1 \odot \vec{a} = \vec{a}$

Der Nachweis der Vektorraumeigenschaften eines Multiagenten-Systems erfolgt in drei Schritte:

1. Nachweis der Gruppeneigenschaften

2. Nachweis der Körpereigenschaften

3. Nachweis der Vektorraumeigenschaften

zu 1) Der allgemeine Beweis des (G_1) Assoziativgesetzes und des (G_5) Kommutativgesetzes ist trivial, da die Agenten-Vektoraddition zurückgeht auf die komponentenweise arithmetische Addition, bei der diese Gesetze erfüllt sind. Analog trifft diese Aussage auch für das Kommutativgesetzt zu. Die Existenz eines neutralen Elements (G_2) ist bereits im vorherigen Abschnitt 5.2.4 gezeigt worden. Auch dass ein inverser Agent existiert wurde in Abschnitt 5.2.4 dargestellt. Auch das letzte ausstehende Axiom (G_4) ist durch die Definition der Agentenaddition (siehe Abschnitt 5.2.2), die für jede Komponente eine arithmetische Addition durchführt, offensichtlich erfüllt. Aus der Gültigkeit dieser Axiome ergibt sich, dass ein Multiagenten-System eine kommutative Gruppe bildet bezüglich der Verknüpfung $\oplus$.

zu 2) Der Nachweis der Axiome (K_1) bis (K_4) verläuft analog dem Nachweis der Gruppeneigenschaften, da die Operationen mit Agentenvektoren auf die komponentenweise arithmetische Addition und Multiplikation zurückgeführt werden können. Somit besitzt ein Multiagenten-System auch Körpercharakter.

zu 3) Weiterhin erfüllt die in Abschnitt 5.2.3 eingeführte Agenten S-Multiplikation die Axiome (V_1), (V_2) und (V_3), da auch hier die Beweise wie auch zuvor, auf die arithmetischen Operationen bezüglich jeder einzelnen Komponente zurückgehen. □

Bemerkung. Nicht jedes Multiagenten-System erfüllt die Bedingungen eines Agenten-Vektorraums. Dies hängt insbesondere von der Modellierung der Anwendungsdomäne ab, ob z.B. ein neutraler Agent oder inverse Agenten existiert, bzw. ob die Axiome V_1, V_2 oder V_3 erfüllt werden können. Im folgendem werden jedoch Multiagenten-Systeme betrachtet, die sich als Vektorraum darstellen lassen.

Im Allgemeinen verwendet man für die Verknüpfung $\oplus$ und $+$ dasselbe Zeichen, nämlich $+$; analog wird in der Schreibweise der Verknüpfungen $\odot$ und $\cdot$ kein Unterschied gemacht, indem man bei beiden das Verknüpfungszeichen weglässt.

Folgerung: Für alle $k \in K$ und für alle $\vec{v} \in V$ gilt:

a) $k\vec{v} = \vec{o} \Leftrightarrow k = 0 \vee \vec{v} = \vec{o}$

b) $k(-\vec{v}) = (-k)\vec{v} = -(k\vec{v})$

5.3.2 Untervektorräume eines Multiagenten-Systems

Untervektorräume sind zunächst von geringer Bedeutung für die Anwendung innerhalb einer Anwendungsdomäne, da bei Operationen bezüglich der Entitäten der Domäne deren vollständige Beschreibung durch alle Domänenattribute notwendig ist. Jedoch kann es aufgrund sehr umfangreicher Domänenbeschreibungen zu einer Vielzahl von notwendigen Attributen kommen. Soll nun ein Vergleich zweier Agenten bezüglich einer vorgegebenen Aufgabe durchgeführt werden, bei der die meisten Attribute nicht relevant sind, dann werden diese Attribute auf Null (neutrales Element) gesetzt. So kann eine Einschränkung des Vektorraums sinnvoll erfolgen.

Definition 5.9. Sei V ein Vektorraum über einem Körper K und U eine Teilmenge von V. Ist U bezüglich der auf V definierten Vektoraddition und der S-Multiplikation selbst ein Vektorraum über K, so heißt U ein *Untervektorraum* bzw. Unterraum von V.

Zum Nachweise der Untervektorraumeigenschaft genügt analog der algebraischen Vektorrechnung folgendes einfachere Kriterium:

Satz 5.1. Sei U eine Teilmenge eines Vektorraums V. U ist genau dann ein Untervektorraum von V, wenn gilt:

a) U ist nicht leer

b) $\vec{u_1} + \vec{u_2} \in U \ \forall \vec{u_1}, \vec{u_2} \in U$

c) $k\vec{u} \in U \ \forall k \in K, \forall \vec{u} \in U$

Bemerkung. Für die Softwareentwicklung und die Wiederverwendbarkeit von existierenden Softwarepaketen können Untervektorräume sinnvoll eingesetzt werden, beispielsweise zur Untersuchung, welche der existierenden Komponenten wiederverwendet werden können. Bildet so z.B. die Domäne des neu zu entwickelnden Systems einen Untervektorraum zu einer bereits bestehenden Domäne, so können die existierenden Agenten ohne Modifikation wiederverwendet werden.

Beispiel 5.10. Analog dem Beispiel 5.7 der S-Multiplikation für Agenten, kann die S-Multiplikation für einen Untervektorraum U definiert werden. Sei also für eine Untersuchung, welcher Agent eine Transportaufgabe am günstigsten ausführen kann, lediglich die Kosten von Relevanz, so ist es nicht notwendig alle Attribute des Vektorraums zu untersuchen. In diesem Fall genügt die Untersuchung eines Attributs. Da zur Planung unter Umständen komplexere Berechnungen notwendig sind, kann es von Vorteil sein, das Planungsproblem auf einen Untervektorraum U zu beschränken. Eine S-Multiplikation, basierend auf der im genannten Beispiel definierten S-Multiplikation, wird hier exemplarisch durch folgende Verknüpfungsvorschrift definiert:

$$\odot \rightarrow \left\{ \ c_1 = s \cdot a_1, \text{ elementargeometrische S-Multiplikation } \text{ mit } c_1, a_1, s \in \mathbb{R} \right.$$

5.3.3 Linearkombination von Agentenvektoren

Die Suche nach einer Lösung einer Aufgabe besteht darin, aus den existierenden Diensten eine Konstellation zu finden, deren resultierender Vektor den Aufgabenvektor beschreibt. Zur Erfüllung einer Aufgabe, die nicht durch einen einzelnen Dienst ausgeführt werden kann, müssen mehrere Dienste von Agenten zusammengestellt werden, damit eine vollständige Aufgabenausführung möglich ist. Dabei kann ein Dienst mehrmals angewendet, bzw. verschiedene Dienste in unterschiedlicher Anzahl kombiniert werden. Zur Koordination der Dienste schließen sich die relevanten Agenten zu einem Holon unter zu Hilfenahme der Agenten-Vektoraddition 5.2.2 und der Agenten S-Multiplikation 5.2.3 zusammen. So lässt sich eine Linearkombination gemäß der algebraischen Linearkombination A.4.7 auch für Agenten und ihre Dienste definieren. Das Ergebnis einer Linearkombination von Agenten ist ein „resultierender Agent“ oder auch Holon, der die Fähigkeiten des Verbunds repräsentieren kann und die Dienste verwaltet.

Definition 5.10. Sei V ein Vektorraum über einem Körper K; $\vec{a_1}, \vec{a_2}, ..., \vec{a_n} \in V$, $k_1, k_2, ..., k_n \in K$. Der Vektor $k_1\vec{a_1} + k_2\vec{a_2} + ... + k_n\vec{a_n}$ heißt dann eine *Linearkombination* der Agentenvektoren $\vec{a_1}, \vec{a_2}, ..., \vec{a_n}$.

Erzeugendensystem

Von einem Erzeugendensystem bei Multiagenten-Systemen spricht man, wenn von Agenten Dienste angeboten werden, so dass zu allen möglichen Aufgaben ihrer Domäne Linearkombinationen dieser Dienste existieren.

Definition 5.11. Eine Menge $W := \{\vec{a}_1, ..., \vec{a}_n\} \subseteq V$ heißt ein Erzeugendensystem des Vektorraums V, wenn jeder Vektor aus V Linearkombination der Vektoren $\vec{a}_1, ..., \vec{a}_n$ ist.

Lineare Abhängigkeit und Unabhängigkeit

Definition 5.12. Sei V ein Vektorraum über einem Körper K. Die Vektoren $\vec{a}_1, ..., \vec{a}_n \in V$ heißen *linear abhängig*, wenn es Skalare $k_1, ..., k_n \in K$ gibt, die nicht alle den Wert 0 haben, so dass gilt: $k_1\vec{a}_1 + ... + k_n\vec{a}_n = \vec{o}$.

Veranschaulicht bedeutet dies: Existiert eine lineare Abhängigkeit zwischen Agenten eines Verbundes, ist dies ein Indiz dafür, dass der Verbund überbestimmt ist und redundante Agenten innerhalb des Verbunds existieren bzw. Dienste ausgeführt werden, deren Ergebnis bereits durch Kombination anderer Dienste erreicht wird.

Satz 5.2. Die Vektoren $\vec{a}_1, ..., \vec{a}_n \in V (n \geq 2)$ sind genau dann linear abhängig, wenn sich (mindestens) einer von ihnen als Linearkombination der anderen darstellen lässt.

Definition 5.13. Sei V ein Vektorraum über einem Körper K. Die Vektoren $\vec{a}_1, ..., \vec{a}_n \in V$ heißen *linear unabhängig*, wenn sie nicht linear abhängig sind.

Die lineare Unabhängigkeit der Vektoren $\vec{a}_1, ..., \vec{a}_n$ ist somit gleichbedeutend mit der Aussage: $k_1\vec{a}_1 + ...k_n\vec{a}_n = \vec{o}$ ist nur lösbar durch die triviale Lösung $k_1 = k_2 = ... = k_n = 0$. Sind die Agentenvektoren linear unabhängig, so sind die Verbünde redundanzfrei.

Basis eines Agenten-Vektorraums

Definition 5.14. V sei ein Agenten-Vektorraum über einem Körper K. Eine Teilmenge $B := \{\vec{b}_1, ..., \vec{b}_n\}$ von V heißt *Basis* des Agenten-Vektorraums V, wenn die Vektoren der Agenten, bzw. ihrer Dienste $\vec{b}_1, ..., \vec{b}_n$ linear unabhängig sind und ein Erzeugendensystem bilden, d.h. eine Basis ist ein Erzeugendensystem mit *minimaler* Mächtigkeit.

Definition 5.15. Die Mächtigkeit n der Basis eines Agenten-Vektorraums V heißt die *Dimension* des Vektorraums. n entspricht dabei der Anzahl von unterschiedlichen Attributen, die dem Multiagenten-System zugrunde liegt. Schreibweise: $dim(V) = n$.

5.4 Relationen zwischen den Grundelementen von Multiagenten-Systemen

5.4.1 Normierung von Agenten, Diensten und Aufgaben

Beim Vergleich zweier Agenten-Vektoren auf Ähnlichkeit bzw. Kongruenz in Bezug auf eine gegebene Aufgabe, kann Definition 5.17 verwendet werden. Dabei sind die Dienste als Abbildungen der Agenten auf einen Vektorraum zu betrachten, d.h. $H : |\alpha(\vec{XY})| = |\vec{XY}|$ entspricht nun $H : |D(\vec{A}_i)| = k|\vec{AG}|$. Existiert also eine ähnliche Abbildung H des Dienstes eines Agenten bezüglich einer Aufgabe, so ist ein Vergleich möglich. Liegt sogar eine Kongruenz vor, dann entspricht der Dienst des Agenten genau den Anforderungen der Aufgabe. Normen stellen dabei die Grundlagen zur Berechnung der 'Länge' eines Vektors dar, die zur Untersuchung der Ähnlichkeit notwendig sind. Vergleicht man nun eine Anzahl von Agentendiensten bezüglich ihrer Tauglichkeit zur Bearbeitung einer Aufgabe miteinander, so muss zu jedem Dienst der Ähnlichkeitsfaktor k bestimmt werden. Anschließend wird der Agentendienst ausgewählt, dessen Ähnlichkeitsfaktor der 1 am Nächsten ist, d.h. seine relative Abweichung (Abstand) ist im Idealfall Null, wodurch eine 100%ige Übereinstimmung existiert. Man spricht dann von einem kongruenten Dienst. Für die Bestimmung der Ähnlichkeit muss zunächst eine geeignete Norm gefunden werden, die die entsprechende Definition A.28 erfüllt. Aufbauend auf dieser Norm wird eine Metrik definiert, die eine Funktion liefert, die den *Abstand* zweier Vektoren bezüglich einer Vorgabe berechnet. Mit dieser Metrik ist es dann möglich die Ähnlichkeit bzw. Kongruenz zweier zu vergleichender Objekte eines Multiagenten-Systems in Relation zu einer Vorgabe zu bestimmen.

Eine geeignete Norm stellt das gewichtete Mittel dar, das aus standardisierten Vektoren einen Wert ermittelt, der die gesamtheitliche Bewertung eines

Vektors bezüglich eines Präferenzvektors g darstellt. Die Normierung eines Vektors (z.B. ein Agentendienst) bezüglich einem gegebenen Vorgabevektors (z.B. eine Aufgabe), wird in zwei Schritten durchgeführt:

1. Standardisierung der Attributwerte eines Agentendienste bezüglich den Vorgaben durch eine Aufgabe
2. Berechung des Ähnlichkeitsfaktors durch ein gewichtetes Mittel

Standardisierung

Jeder Dimension des Agentenvektorraums ist ein eigener Wertebereich zugeordnet, so dass bei der Berechnung des gewichteten Mittels eine Standardisierung zwingend notwendig ist. Dadurch wird z.B. vermieden, dass allein durch die unterschiedliche Wahl der Einheitsgrößen (z.B. Volumen in *Tonnen* und Kosten in *Euro-Cent*) die Abweichungen bezüglich den Vorgaben (z.B. 1 Tonne und 10.000 Euro-Cent) gewichtet werden. Daher müssen die Absolutwerte zuerst auf einen, für alle Attribute gleichen, und damit vergleichbaren Wertebereich projiziert werden. Zur Ähnlichkeitsbestimmung eines Vektors zu einem gegebenen Zielvektor, muss für jedes Attribut eine geeignete Funktion ausgewählt werden, die die Bedingungen der Aufgabenstellung erfüllt. Gegeben sei ein Agentenvektor $\vec{a} \in V$ und ein Zielvektor $\vec{a_v} \in V$ der Aufgabenstellung, der die Vorgaben v an den Agentenvektor enthält. Dabei können folgende Bedingungen an den gesuchten Vektor bzgl. des Zielvektors aus der Aufgabenstellung gestellt werden:

1. $a_i \mapsto min$ (minimaler Wert aller untersuchten Agentenvektoren bzgl. a_i)
2. $a_i \geq a_{i,v}$ (Vorgabe)
3. $a_i \mapsto max$ (maximaler Wert aller untersuchten Agentenvektoren bzgl. a_i)
4. $a_i \leq a_{i,v}$ (Vorgabe)
5. $a_i = a_{i,v}$
6. $a_{i,v1} \leq a_i \leq a_{i,v2}$ (innerhalb des Intervalls $[a_{i,v1}; a_{i,v2}]$)
7. $a_i \leq a_{i,v1} \vee a_{i,v2} \leq a_i$ (außerhalb des Intervalls $[a_{i,v1}; a_{i,v2}]$)
8. a_i enthält keine Bedingungen, sondern muss lediglich in der Attributmenge enthalten sein.

In Anlehnung an die Boolesche Logik wurden die Funktionen zur Anpassung der Attributwerte an einen Vorgabewert der Aufgabenstellung so gewählt, dass die Attributwerte auf des geschlossene Intervall von 0 bis 1 abbildet werden; $f : a \mapsto [0; 1]$. $a = 1$ bedeutet vollständige Übereinstimmung mit der Vorgabe durch die gestellte Aufgabe und $a = 0$ liegt dann vor, wenn keine Übereinstimmung existiert. Das Ziel dieser Anpassung ist eine Bewertung von Attributen von bzgl. einer Vorgabe. Dazu sind boolesche Bewertungen jedoch zu restriktiv, um eine gewertete Liste für die Fähigkeiten der Agenten bezüglich den Vorgaben einer Aufgabe zu erstellen. So sind Funktionen notwendig, die Werte berechnen,

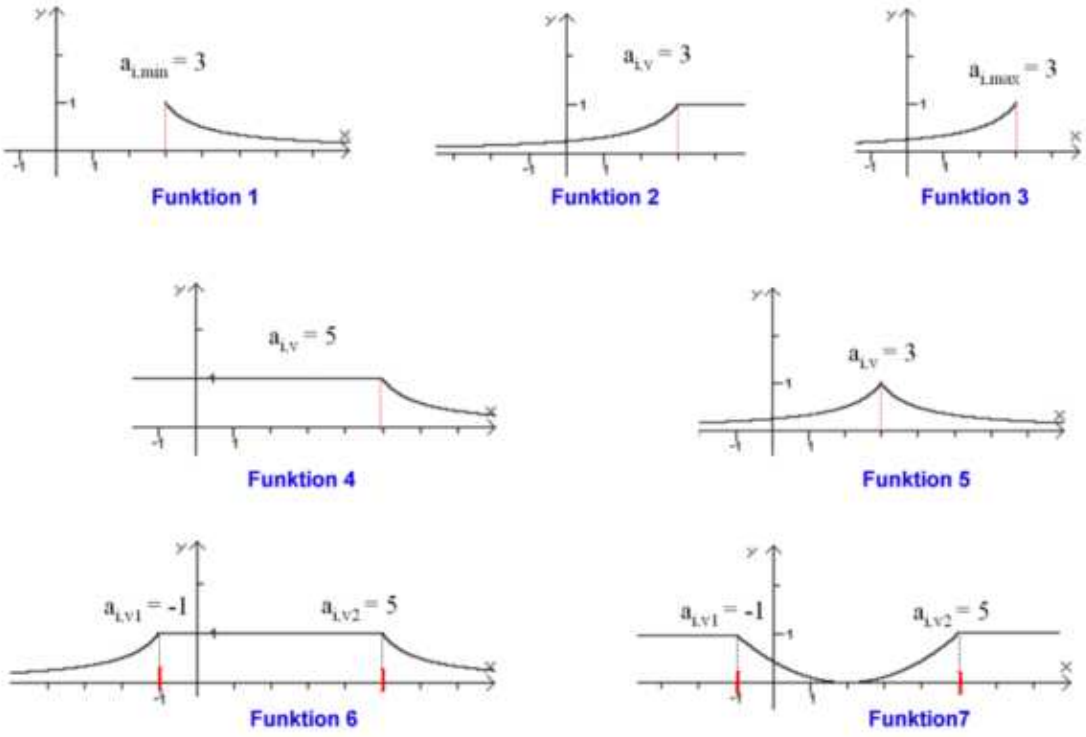

Abbildung 5.3: Funktionen zur Abbildung der Attributwerte auf das Intervall $[0; 1]$

die die *Ähnlichkeit* zu den Vorgabewerten angeben. Eine Möglichkeit ist die Verwendung linearer Funktionen, jedoch stellt sich hierbei die Schwierigkeit, eine geeignete Steigung zu finden, so dass nicht bei den meisten Vergleichen 0 oder 1 als Ergebnis geliefert wird. Stattdessen ist es sinnvoller eine Funktion zu wählen, die sich bei einer großen Abweichung der Null nähert und bei geringer Abweichung vom Vorgabewert sich stärker von der Eins entfernt, als dies eine lineare Funktion realisiert. Dies führt zu einer genaueren Differenzierung der Agenten, deren Attributwerte sehr genau den Vorgaben entsprechen. So steigt die Differenzierung bei den bestmöglichen Agentendiensten, wo hingegen sie bei den restlichen Agentendiensten abnimmt. Dies ist sinnvoll, da nur ein bzw. eine geringe Anzahl von Agentendiensten ausgewählt werden soll. Abbildung 5.3 veranschaulicht den Verlauf der nachfolgenden Funktionen graphisch:

1. $a^\circ_i \mapsto \frac{1}{|a_i - a_{i,min}|+1}$ (Abweichung vom minimalem Wert $a_{i,min}$)

2. $a^\circ_i \mapsto \begin{cases} 1 \text{ ,wenn } a_i \geq a_{i,v} \\ \frac{1}{a_{i,v} - a_i + 1} \text{ , sonst} \end{cases}$ (Abweichung von einem Mindestwert)

3. $a^\circ_i \mapsto \frac{1}{|a_{i,max} - a_i|+1}$ (Abweichung vom maximalem Wert $a_{i,max}$)

4. $a^\circ{}_i \mapsto \begin{cases} 1 \text{ ,wenn } a_i \leq a_{i,v} \\ \frac{1}{a_i - a_{i,v}+1} \text{ , sonst} \end{cases}$ (Abweichung von einem Höchstwert)

5. $a^\circ{}_i \mapsto 1 - \frac{1}{|a_i - a_{i,v}|+1}$ (Abweichung von einem vorgegebenen Wert $a_{i,v}$)

6. $a^\circ{}_i \mapsto \begin{cases} 1 \text{ ,wenn } a_{i,v1} \leq a_i \leq a_{i,v2} \\ \frac{1}{|a_i - a_{i,v1}|+1} \text{ ,wenn } a_i < a_{i,v1} \\ \frac{1}{|a_i - a_{i,v2}|+1} \text{ ,wenn } a_i > a_{i,v2} \end{cases}$ (Abweichung von einem vorgegebenen Intervall)

7. $a^\circ{}_i \mapsto \begin{cases} 1 \text{ ,wenn } a_i \leq a_{i,v1} \vee a_{i,v2} \leq a_i \\ (\frac{a^\circ{}_i - m}{d})^2 \text{ mit } d = \frac{a_{i,v2} - a_{i,v1}}{2},\ m = a_{i,v1} + d \text{ , sonst} \end{cases}$ (Abweichung von einem ausgenommenen Intervall)

8. $a^\circ{}_i \mapsto \begin{cases} 0 \text{ ,wenn } a_i = 0 \\ 1 \text{ ,sonst} \end{cases}$

Bemerkung. Handelt es sich bei dem zu vergleichenden Agenten um den *neutralen Agenten*, dann werden alle standardisierten Attributwert $a^\circ{}_i$ auf Null gesetzt, so dass das Ergebnis der Norm Null beträgt. Analog werden auch beim *neutralen Vorgabevektor* alle standardisierten Attributwerte auf Null gesetzt. Jedoch ist dieser Fall nur von geringer Bedeutung, da eine Untersuchung ohne Randbedingungen keine Gewichtung erzeugen würde, d.h. eine Analyse mit neutralem Vorgabevektor ist kaum von Nutzen.

Gewichtetes Mittel

Mit Hilfe der beschriebenen Funktionen lassen sich die Attribute der Agenten in Relation zu seinen Vorgaben setzen, so dass die Ergebnisse direkt miteinander vergleichbar sind. Jedoch ist es gerade bei sehr vielen zu untersuchenden Attributen wünschenswert, wenn eine Abbildung $||A^\circ|| \rightarrow r \in \mathbb{R}$ existiert, die die Attribute der Domäne auf einen Wert abbilden kann. Dabei muss für zwei Agenten A_1 und A_2, die mit solch einer Abbildung auf die Werte r_1 und r_2 abgebildet werden, gelten: $r_1 < r_2 \Leftrightarrow$ „A_2 eignet sich besser als A_1“. Diese Anforderungen werden durch das gewichtete Mittel erfüllt. Um die Wichtigkeit der Attribute untereinander in Relation setzen zu können, werden die Attribute durch je einen Faktor g_i gewichtet. $g_i \in [0;1]$. Somit wird jedem $A \in \mathcal{A}$ eine reelle Zahl zugeordnet:

$$||A^\circ|| = \frac{g_1 \cdot |a^\circ{}_1| + g_2 \cdot |a^\circ{}_2| + ... + g_n \cdot |a^\circ{}_n|}{g_1 + g_2 + ... + g_n}$$

Je größer das gewichtete Mittel eines Agenten bezüglich einer Aufgabenstellung ist, desto besser erfüllt er die Anforderungen der Aufgabenbeschreibung. Da über dem neutralen Agenten keine Aussage getroffen werden kann ($||A^\circ|| = 0$), erhält er auch die geringste Bewertung aller Agenten. Der höchstmögliche Wert, den ein Agent bei voller Übereinstimmung mit den Anforderungen erzielen kann, beträgt eins. Dass es sich bei dem gewichteten Mittel um eine Norm handelt,

kann durch den Nachweis der Bedingungen N_1, N_2, N_3 aus Definition A.28 erbracht werden.

(N_1) $||\vec{x}|| \geq 0, ||\vec{x}|| = 0 \Rightarrow \vec{x} = \vec{0}$

(N_2) $||s \cdot \vec{x}|| = |s| \cdot ||\vec{x}||$

(N_3) $||\vec{x} + \vec{y}|| \leq ||\vec{x}|| + ||\vec{y}||$

Satz 5.3. $||A||$ ist eine Norm.

Beweis. *(Norm)*

1. *N_1 ist erfüllt, da durch die Standardisierung alle $a^\circ{}_i$ auf den Wertebereich $[0;1]$ abgebildet werden und beim gewichteten Mittel alle g_i aus dem Definitionsbereichs $[0;1]$ zu wählen sind. Auch gilt $||\vec{x}|| = 0 \Rightarrow \vec{x} = \vec{0}$, da $||\vec{x}||$ nur dann den Wert Null annehmen kann, wenn das neutrale Element eingesetzt wird. Sonst nähern sich die Werte lediglich der Null, erreichen sie jedoch nie.* □

2. *$||A^\circ||$ erfüllt N_2, da bereits $||A||$ N_2 erfüllt, denn:*

$$\begin{aligned} &||s \cdot A^\circ|| = \\ &= \frac{g_1 \cdot |s \cdot a^\circ{}_1| + ... + g_n \cdot |s \cdot a^\circ{}_n|}{g_1 + ... + g_n} = \\ &= \frac{|s| \cdot (g_1 \cdot |a^\circ{}_1|) + ... + |s| \cdot (g_n \cdot |a^\circ{}_n|)}{g_1 + ... + g_n} = \\ &= |s| \cdot (\frac{g_1 \cdot |a^\circ{}_1| + ... + g_n \cdot |a^\circ{}_n|}{g_1 + ... + g_n}) = \\ &= |s| \cdot ||A^\circ|| \end{aligned}$$

□

3. *N_3 ist offensichtlich auch erfüllt, da der Wertebereich aller Attribute a_i nach der Standardisierung auf das positive Intervall $[0;1]$ abgebildet werden. Somit gilt $a^\circ{}_i \geq 0$ für alle Attribute eines Agenten und ferner gilt:*

$$\begin{aligned} &||A^\circ + B^\circ|| = \\ &= \frac{g_1 \cdot |a^\circ{}_1 + b^\circ{}_1| ... + g_n \cdot |a^\circ{}_n + b^\circ{}_n|}{g_1 + ... + g_n} \leq \\ &\leq \frac{g_1 \cdot |a^\circ{}_1| + |b^\circ{}_1| < ... + g_n \cdot |a^\circ{}_n| + |b^\circ{}_n|}{g_1 + ... + g_n} = \\ &= \frac{g_1 \cdot |a^\circ{}_1| + ... + g_n \cdot |a^\circ{}_n|}{g_1 + ... + g_n} + \frac{g_1 \cdot |b^\circ{}_1| + ... + g_n \cdot |b^\circ{}_n|}{g_1 + ... + g_n} = \\ &= ||A^\circ|| + ||B^\circ|| \end{aligned}$$

Somit ist N_3 erfüllt und $||A^\circ||$ eine Norm.

□

q.e.d.

Beispiel 5.11. Seien $A_1, A_2 \in \mathcal{A}$ zwei Agenten aus Beispiel 5.3. Betrachtet man das vierte Attribut (die Ausführungszeit), so besagt das Axiom N_3, dass die Ausführungszeit des Holons geringer (höchstens gleich) der Summe der Ausführungszeiten der einzelnen Agenten ist. Dies gilt jedoch nur, wenn ausschließlich die Ausführungszeit untersucht werden soll und die Ausprägungen der restlichen Attribute unbedeutend sind. Bei der Möglichkeit die Aufgaben parallel ausführen zu können, benötigt ein Verbund nur 25 Minuten. Hingegen benötigt eine Menge von Agenten, die sich nicht zu einem Holon zusammengeschlossen haben, unter Umständen sogar 50 Minuten, wenn aufgrund der fehlenden Koordination keine parallelen Arbeiten möglich sind und alle Bearbeitungsschritte sequentiell ausgeführt werden müssen.

Bemerkung. Zu beachten gilt, dass aus $||A_1{}^\circ|| < ||A_2{}^\circ||$ nicht unbedingt folgt: $a^\circ{}_{1i} < a^\circ{}_{2i}$ für alle i ! Die Norm beschreibt hier einen Agenten in der Gesamtheit seiner Attribute, weshalb sich Aussagen wie im Beispiel 5.11 nicht verallgemeinern lassen. Konkretere Aussagen können erst mit Hilfe einer Metrik gemacht werden, wie in den folgenden Abschnitten beschrieben wird. Beim Vergleich zweier Holone miteinander werden diese Spaltenweise verglichen, analog dem Vergleich zweier Agenten je Spalte.

5.4.2 Einführung einer Metrik auf dem Raum der Agentenvektoren

Für die Bestimmung eines Agenten, der zu einer vorgegebenen Aufgabe am besten geeignet ist, ist es erforderlich eine Metrik auf den Raum der Agentenvektoren einzuführen. Insbesondere wenn die Domäne mehr als ein Attribut enthält, spielt die unterschiedliche Gewichtung der einzelnen Attribute eine entscheidende Rolle. So kann die Auswertung individuell der aktuellen Situation und dem Verhalten der jeweiligen Agenten angepasst werden, die eine Aufgabe ausschreiben. Mit Hilfe einer Metrik wird es nun möglich die Angebote von verschiedenen Agenten zu bewerten und entsprechend der Gewichtung so den geeignetsten Agenten für eine gestellte Aufgabe auszuwählen. Beispiel 5.12 zeigt die Bedeutung der Gewichtung bei der Ermittlung des geeignetsten Agenten bezüglich einer gestellten Transportaufgabe.

Beispiel 5.12. Ein Chemiefabrik stellt zwei unterschiedliche Chemikalien her, die nach der Produktion zum Kunden transportiert werden müssen. Dazu wird je ein LKW für den Transport einer der beiden Chemikalien eingesetzt. Die zweite Chemikalie ist dabei wesentlich gefährlicher zu transportieren als die erste, weshalb die Qualität der LKW-Züge von größerer Bedeutung ist als z.B. die Kosten. Die beiden Transportaufgaben haben die gleichen Randbedingungen, es soll der günstigste LKW gefunden werden, der mindestens 7 Tonnen Kapazität besitzen, der Qualitätsindex soll ein Mindestwert von 10 besitzen und eine Stunde steht zur Bearbeitung zur Verfügung. Die beiden Aufgaben werden durch den Vektor $AG|_{Transport} = (min, \geq 7, \geq 10, 1)$ beschrieben. Seien A_1, A_2 die zwei Agenten aus Beispiel 5.3, die je einen Transportdienst anbieten. A_1 wird durch den Vek-

tor $\vec{a}_1 = \begin{pmatrix} 50 \\ 7 \\ 8 \\ 1 \end{pmatrix}$ und A_2 durch den Vektor $\vec{a}_2 = \begin{pmatrix} 120 \\ 10 \\ 10 \\ \frac{3}{4} \end{pmatrix}$ beschrieben. Nach der Standardisierung, ergeben sich folgende angepasste Vektoren für A_1 und A_2: $\vec{a}^\circ{}_1 = \begin{pmatrix} 1 \\ 1 \\ \frac{1}{3} \\ 1 \end{pmatrix}$ und $\vec{a}^\circ{}_2 = \begin{pmatrix} \frac{1}{71} \\ 1 \\ 1 \\ \frac{4}{5} \end{pmatrix}$. Nach einer gleichmäßigen Gewichtung von $g_1 = (1, 1, 1, 1)$ für den Transport der ersten Chemikalie besitzt Agent A_1 einen Wert von 0.833 und für A_2 ergibt sich ein Wert von 0.704. Agent A_1 ist somit für die Bearbeitung der ersten Transportaufgabe zu wählen. Hingegen wird aufgrund der gefährlichen Fracht bei der zweiten Transportaufgabe die Gewichtung folgendermaßen gewählt $g_2 = (0.5, 1, 1, 0.5)$, so ergibt sich für A_1 ein Wert von 0.778 und für A_2 ein Wert von 0.802. Da die Qualität der Aufgabenausführung hier wesentlich stärker gewichtet ist als z.B. die Kosten, ist unter den gleichen Voraussetzungen Agent A_2 für den Transport der gefährlichen Chemikalie zu wählen.

Dieses Beispiel veranschaulicht die Wichtigkeit einer Norm, die als Basis zur Bildung einer Metrik dient. Mit Hilfe der Metrik wird der *Abstand*, die *Distanz* $d(A_1, A_2)$ zwischen zwei Agenten A_1 und A_2 aus $\mathcal{A}$ definiert. Auf der Menge der Agenten $\mathcal{A}$ sei jedem Agentenpaar $A_1, A_2 \in \mathcal{A}$ eine reelle Zahl $d(A_1, A_2)$ zugeordnet, so dass für beliebige Agenten aus $\mathcal{A}$ die folgenden Eigenschaften eines *metrischen Raums* erfüllt sind:

(M_1) *Nichtnegativität* $d(A_1, A_2) \geq 0$ und $d(A_1, A_2) = 0 \Leftrightarrow A_1 = A_2$

(M_2) *Symmetrie* $d(A_1, A_2) = d(A_2, A_1)$

(M_3) *Dreiecksungleichung* $d(A_1, A_3) \leq d(A_1, A_2) + d(A_2, A_3)$

Eine Funktion $d : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$, die die Eigenschaften M_1, M_2 und M_3 erfüllt heißt *Metrik*, *Distanz* oder *Abstand* auf der Menge $\mathcal{A}$, und das Paar $\mathcal{A} = (\mathcal{A}, d)$ heißt *metrischer Raum* eines Multiagenten- Systems.

Bemerkung. Die Axiome (M_1) bis (M_3) bedeuten anschaulich folgendes: Der Abstand zweier normierter Agentenvektoren ist positiv, während der Abstand eines normierten Agentenvektors von sich selbst verschwindet; der Abstand eines normierten Agentenvektors A_1 zu A_2 ist ebenso groß wie umgekehrt der Abstand des normierten Agentenvektors A_2 zu A_1. Deshalb kann bei der Betrachtung des Abstands zweier Agenten zueinander, die Beachtung der Reihenfolge vernachlässigt werden; in Bezug auf Agenten bedeutet veranschaulicht die Dreiecksungleichung, dass z.B. auf der Suche nach *ähnlichen Agenten* der direkte Vergleich zweier Agenten nie ein schlechteres Resultat ergibt, als der Vergleich einer Reihe von Subsumptionen anderer Agenten bis hin zu den ursprünglich zu vergleichenden Agenten. Für die Suche nach *einem* Agenten muss so nicht die Potenzmenge aller möglichen Agentensubsumptionen untersucht werden, sondern nur die Agentenpaare, bei denen einer der beiden Agenten der Ausgangsagent ist.

Definition 5.16. $d(A_1, A_2) = ||A^\circ{}_1 - A^\circ{}_2||$, mit einer Aufgabe AG als Vorgabe bezüglich der Relativierung, heißt die *Distanz* d zwischen zwei Agenten A_1 und A_2.

Satz 5.4. d ist eine Metrik auf der Menge $\mathcal{A}$ aller Agenten eines Multiagenten-Systems unter Verwendung der Norm $||A^\circ||$ aus Satz 5.3.

Beweis (Metrik)**.** *Zum Beweis, dass unter Verwendung der Norm* $||A^\circ||$*, der Agentenvektorraum eine Metrik besitzt, müssen die Axiome* (M_1) *bis* (M_3) *erfüllt sein.*

1. *Die Nichtnegativität ergibt sich aus der Definition der Norm. Der zweite Teil dieses Axioms ist auch erfüllt, da diese Norm nur dann den Wert Null annimmt, wenn der neutrale Agent normiert wird. Der neutrale Agent ergibt sich aus der Differenz jedoch nur dann, wenn gilt:* $A_1 = A_2 \Leftrightarrow A_1 - A_2 = \vec{o}$*, also beide Agenten identisch sind.* □

2. *Die Symmetrie ist erfüllt, da:*

$$d(A, B) =$$

$$= ||A^\circ - B^\circ|| =$$

$$= \frac{g_1 \cdot |a^\circ{}_1 - b^\circ{}_1| + ... + g_n \cdot |a^\circ{}_n - b^\circ{}_n|}{g_1 + ... + g_n} =$$

$$= \frac{g_1 \cdot |b^\circ{}_1 - a^\circ{}_1| + ... + g_n \cdot |b^\circ{}_n - a^\circ{}_n|}{g_1 + ... + g_n} =$$

$$= ||B^\circ - A^\circ|| =$$

$$= d(B, A)$$

□

3. *Die Dreiecksungleichung ist auch erfüllt, da:*

$$d(A, C) =$$

$$||A^\circ - C^\circ|| =$$

$$||A^\circ - B^\circ + B^\circ - C^\circ|| \leq$$

$$\leq ||A^\circ - B^\circ|| + ||B^\circ - C^\circ|| =$$

$$= d(A, B) + d(B, C)$$

□

Damit bildet $d(A, B) = ||A^\circ - B^\circ||$ *eine Metrik auf den Agentenvektorraum.*

Bemerkung. Aussage M_3 veranschaulicht, dass die Effizienz eines Zusammenschluss von Agenten zu einem Holon mindestens genauso hoch ist wie die Bewertung der Agenten im Fall einer unabhängigen Einzelbearbeitung ohne Bildung eines Holons. Betrachtet man nur die für die Bearbeitung der Aufgabe notwendigen Attribute, so liefert die Metrik eine hoch signifikante Aussage bezüglich dieser Attribute. M_3 führt dann zu der Aussage: wenn die Möglichkeit zur Bildung eines Holons gegeben ist, dann sollte sie auch genutzt werden, jedoch nicht unter allen Umständen (Beschränkung auf die relevanten Attribute).

5.4.3 Ähnliche und kongruente Agenten

Die Überprüfungen zweier Agenten auf Ähnlichkeit ist beispielsweise zur Ermittlung von Agenten von Bedeutung, die einen Agenten gleichen Typs innerhalb eines Verbunds ersetzen können. Dazu müssen die Ausprägungen der Attribute jedes relevanten Agenten mit dem auftretenden Agenten verglichen und die Abweichungen geeignet bewertet werden. Auch bei der Bestimmung, welcher Agent am besten eine Aufgabenbeschreibung erfüllen kann, ist die Bestimmung der Ähnlichkeit von Diensten relevanter Agenten mit der Aufgabebeschreibung notwendig.

Definition von ähnlichen und kongruenten Abbildungen

Unter den affinen Abbildungen eines *metrischen* Punktraums P in sich (vgl. Kapitel 5.8) sind diejenigen von besonderem Interesse, bei denen alle Abstände erhalten bleiben oder alle Abstände mit ein und derselben positiven reellen Zahl multipliziert werden. Es sind dies die Kongruenz- und Ähnlichkeitsabbildungen.

Definition 5.17. P sei ein euklidischer Punktraum. Eine affine Abbildung $H : P \to P$ heißt eine *ähnliche Abbildung* oder eine *Ähnlichkeit*, wenn es eine reelle Zahl $k > 0$ gibt, so dass für alle $X, Y \in P$ und $k \in \mathbb{R}$ gilt:

$$|H(X)\vec{H}(Y)| = k|\vec{XY}|$$

k heißt *Ähnlichkeits- oder Steckungsfaktor*. H heißt eine *kongruente Abbildung* oder eine Kongruenz, wenn $k = 1$ ist.

Aus dieser Definition folgt für die durch H induzierte lineare Abbildung $\alpha : V \to V$ mit $\alpha(\vec{XY}) = \vec{H(X)H(Y)}$, bei einer ähnlichen Abbildung H : $|\alpha(\vec{XY})| = k|\vec{XY}|$ und bei einer kongruenten Abbildung $H : |\alpha(\vec{XY})| = |\vec{XY}|$. Es gilt also:

Satz 5.5. Jede ähnliche Abbildung mit dem Ähnlichkeitsfaktor k induziert das k-fache einer isometrischen linearen Abbildung, jede kongruente Abbildung eine isometrische lineare Abbildung.

5.4.4 Äquivalenz von Agenten

Abbildung 5.4 zeigt den Wertebereich der Attribute Kosten und Qualität der beiden Agenten aus Beispiel 5.4 und eines weiteren Transportagenten A_5, der

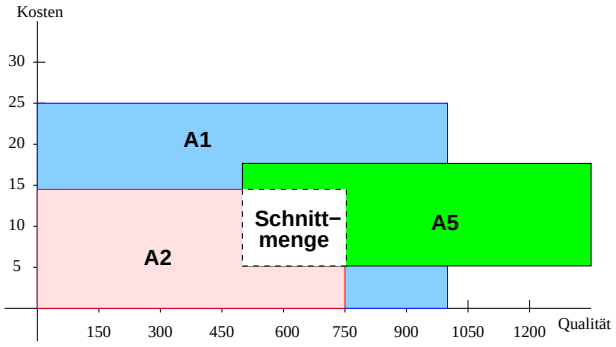

Abbildung 5.4: Schnittmenge gemeinsamer Attribute zweier Agenten

Transporte zu mindestens 6 EUR und maximal 17 EUR anbieten kann. Ein solcher Agent existiert z.B. dann, wenn die Fixkosten bereits 6 EUR pro Stunde betragen. Soll eine Aufgabe bearbeitet werden, deren Attributwerte innerhalb der Schnittmenge der Agenten-Attribute liegt, so können alle drei Agenten diese Aufgabe gleichermaßen bearbeiten. Ein Agent kann somit willkürlich zur Ausführung der Aufgabe ausgewählt werden. Die unterschiedlichen Agenten sind unter den Vorgaben einer gegebenen Aufgabe gleichwertig, also äquivalent.

Somit ergibt sich:

Definition 5.18. Zwei Agenten A_1 und A_2 sind bezüglich einer Dienstklasse D genau dann äquivalent, wenn eine Abbildung α existiert mit $\alpha : A_i|_D \rightarrow \mathbb{R}^n$ mit $i \in \{1, 2\}$, so dass alle entsprechenden Attribute der beiden Agenten unter Berücksichtigung der individuellen Randbedingungen bezüglich ihrer Dienste jeweils auf den gleichen Wert bzw. Wertebereich bezüglich einer Aufgabe abgebildet werden. Dann gilt: $\alpha(a_{1i}) = \alpha(a_{2i})\ \forall a_{1i} \in A_1 \wedge a_{2i} \in A_2$.

Bemerkung. Wird zur Aufgabenbearbeitung ein Dienst gesucht, müssen die Beschreibungen der Agentendienste mit der Aufgabenbeschreibung verglichen werden. Dazu kann auch Definition 5.18 eingesetzt werden, da Agenten, Aufgaben und Dienste die gleiche Struktur aufweisen und daher austauschbar sind. Zwei Entitäten gleichen Typs können so auf Gleichheit geprüft werden (z.B. ob zwei Agenten gleichwertig zur Bearbeitung einer gegebenen Aufgabe sind). Bei einem Vergleich zweier Entitäten werden dann nur die Attribute der Aufgabe untersucht, denen Wertebereiche bzw. konkrete Werte zugeordnet sind.

5.4.5 Umkehrung der Zuordnung Agent-Aufgabe

Durch die Aufgabenbeschreibung sind die Werte relevanter Attribute gegeben. Zum Lösen der Aufgabe werden Dienste von Agenten gesucht, die der Aufgabenbeschreibung am besten (im Idealfall genau) entsprechen. Im Gegensatz zu den vorherigen Betrachtungen ändert sich hierbei die Perspektive von der des Agenten, der prüft, welche Aufgaben er bearbeiten kann, zu der Sicht eines informierten Koordinators, der aus einer Menge von Agenten bzw. Diensten diejenigen bestimmen muss, die am besten für die Aufgabenbearbeitung geeignet

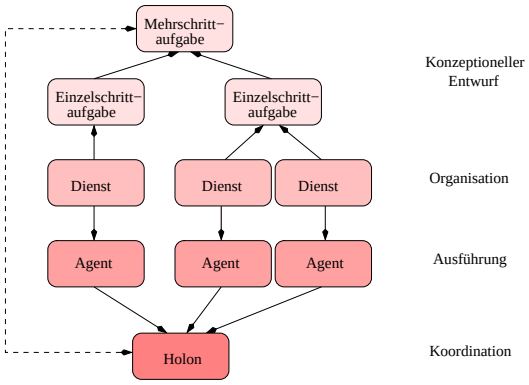

Abbildung 5.5: Relationen zwischen Aufgaben, Diensten und Agenten

sind. Zur Ermittlung von möglichen Diensten zur Bearbeitung einer gegebenen Aufgabe bedarf es also einer Abbildung α', die eine umgekehrte Abbildung gegenüber α vornimmt und den Wertebereich von α auf dessen Definitionsbereich abbildet: $\alpha' : AG \rightarrow D$

Solche Abbildungen sind auch notwendig für eine Zusammenstellung von Agenten, die einen Dienst gleichen Typs anbieten können. Die Suche nach einer Lösung einer Aufgabe gliedert sich dabei in zwei Phasen.

1. Ermittlung geeigneter Dienstklassen zur Bearbeitung einer Aufgabe

2. Ermittlung geeigneter Agenten, die die ermittelten Dienstklassen besitzen

In diesem Sinne sind umgekehrte Abbildungen notwendig, da bei der Initialisierung eines Multiagenten-Systems in umgekehrter Reihenfolge vorgegangen wird und den Agenten je eine Dienstklasse ($\alpha : A \rightarrow D^{\circ}$) und Aufgaben Dienstklassen ($\beta : D^{\circ} \rightarrow AG^{\circ}$) zur Bearbeitung zugeordnet werden. Abbildung 5.5 veranschaulicht die Relationen zwischen den beschriebenen Elementen der Domäne. Die Aufgaben können so in zwei Gruppen aufgeteilt werden:

1. Einzelschrittaufgaben, unteilbare und direkt ausführbare Aufgabe.

2. Mehrschrittaufgaben, die sich aus verschiedenen Einzelschrittaufgaben zusammensetzen.

Der Unterschied zwischen diesen beiden Aufgabentypen ist, dass Mehrschrittaufgaben dem konzeptionellen Entwurf eines Problems dienen und erst auf Basis der Einzelschrittaufgabe durch Dienste bearbeitet werden können. Zusammengefasst bilden alle Agenten, die zur Bearbeitung der Mehrschrittaufgabe eingesetzt werden ein Holon, das die Ausführung und Bearbeitung einer Mehrschrittaufgabe durchführen kann. Bei der Analyse, welche Agenten zur Lösung einer Aufgabe eingesetzt werden können, werden die Attribute der Agenten bzgl. der Dienstklasse zu den jeweiligen Zeiten miteinander verglichen.

5.4.6 Agenten-Gruppen

Eine Untersuchung der mathematischen Gruppeneigenschaften zeigt, dass hierbei ebenso eine sinnvolle Abbildung dieser Theorie auf ein Multiagenten-System vollzogen werden kann. Der Nachweis, dass ein Multiagenten-System eine Gruppe bildet, führt zu einer formalen Beschreibung der Eigenschaften einer Gruppe von Agenten - dem Holon.

Definition 5.19. Eine Struktur von Agenten $(G, \Box)$, die aus einer nichtleeren Menge G zusammen mit einer inneren Verknüpfung $\Box$ besteht, heißt eine *Agenten-Gruppe*, wenn folgende Axiome erfüllt sind:

(G_1) *Assoziativgesetz* für alle Agenten $A_1, A_2, A_3 \in G$ gilt: $(A_1 \Box A_2) \Box A_3 = A_1 \Box (A_2 \Box A_3)$

(G_2) *Existenz eines neutralen Agenten* Es gibt ein Agent $n \in G$, so dass $\forall A \in G$ gilt: $A \Box n = n \Box A = a$

(G_3) *Existenz von inversen Agenten* Zu jedem Agent $A \in G$ gibt es ein Agent $A^{-1} \in G$, so dass gilt: $A \Box A^{-1} = A^{-1} \Box A = n$

Gilt zudem noch

(G_4) *Kommutativgesetz* $A_1 \Box A_2 = A_2 \Box A_1 \ \forall A_1, A_2 \in G$, so heißt eine solche Gruppe eine *kommutative* Agenten-Gruppe.

Bemerkung. Wegen der Assoziativität können Klammern ganz weggelassen werden, also $(A_1 \Box A_2) \Box A_3 = A_1 \Box (A_2 \Box A_3) = A_1 \Box A_2 \Box A_3$. Wie bereits in Abschnitt 5.3 gezeigt wurde, bilden die Agenten zusammen mit einer inneren Verknüpfung $\oplus$ eine kommutative Agenten-Gruppe.

Satz 5.6. In jeder Agenten-Gruppe gibt es genau ein neutrales Element.

Satz 5.7. In jeder Agenten-Gruppe gibt es zu jedem Element A genau ein inverses Element A^{-1}.

Satz 5.8. In jeder Agenten-Gruppe sind die Gleichungen $A_1 \Box X = A_2$ und $Y \Box A_1 = A_2$ für die unbekannten $X, Y \in \mathcal{A}$ eindeutig lösbar.

5.5 Zusammenfassung

In diesem Kapitel habe ich einen mathematischen Formalismus aufgebaut, der es ermöglicht Agenten durch Vektoren zu repräsentieren. Durch die Definition spezieller Verknüpfungsvorschriften zwischen Agenten und der Definition

besonderer Agententypen konnte einem Multiagenten-System[2] Vektorraumcharakter nachgewiesen werden. So ist es beispielsweise in Multiagenten-Systemen mit Vektorraumcharakter möglich, die Agentenvektoren einer Menge von Agenten (Holon) zu einem neuen Vektor zusammenzufassen, unabhängig deren Reihenfolge. Dies ist eine wichtige Eigenschaft, denn durch die nebenläufige Aufgabenbearbeitung durch die Agenten können im Voraus oftmals keine genauen Aussagen getroffen werden, in welcher Reihenfolge die Zwischenergebnisse zustande kommen. Handelt es sich jedoch bei dem Multiagenten-System um einen Vektorraum, ist z.B. aufgrund der Assoziativität die Bearbeitungsreihenfolge unwichtig. Diese Eigenschaft eines Vektorraums hat auch Auswirkung auf die Formierung eines Holons. Veranschaulicht bewirkt die Assoziativität eines Multiagenten-System-Vektorraums, dass die Eintrittsreihenfolge der Agenten in ein Holon keine Auswirkung auf dessen Funktionalität hat. Für die Ermittlung wie viele Agenten notwendig sind, eine Aufgabe vollständig zu bearbeiten, kann dies zum Beispiel mit Hilfe einer Linearkombination berechnet werden. Diese zwei Beispiele veranschaulichen bereits den Nutzen der Vektorraumeigenschaften. Wird zusätzlich auf dem Vektorraum eine Metrik definiert, können zudem Relationen zwischen den Agenten formal beschrieben werden, wie z.B. der Abstand zwischen einem Agenten und einer Aufgabenbeschreibung. Dies ist notwendig, um den Agenten zu bestimmen, der eine Aufgabe am besten bearbeiten kann. Speziell die Bestimmung von Ähnlichkeiten zwischen Agenten anhand einer Abstandsberechnung ist in den folgenden Kapiteln von besonderer Bedeutung. Der Abstand kann auch zur Bestimmung der Ähnlichkeit zwischen Agenten, Diensten und Aufgaben eingesetzt werden, um z.B. den Agenten zu bestimmen, welcher am besten eine Aufgabenbeschreibung erfüllen kann. Ist der Abstand zweier Agent gleich Null, bilden diese Agenten eine Äquivalenzklasse. Ein Agent kann somit willkürlich zur Ausführung der Aufgabe aus der entsprechenden Klasse ausgewählt werden, ohne dass alle Agenten erneut analysiert werden müssen. Die unterschiedlichen Agenten sind unter den Vorgaben einer gegebenen Aufgabe gleichwertig, also äquivalent. Durch die Abbildung eines Multiagenten-System auf einen Vektorraum kann so die Vielfalt der Vektorrechnung (Abstands-, Lagebestimmung, Prüfung von linearen Abhängigkeiten etc.) auch in Multiagenten-Systemen genutzt werden. Ferner lassen sich durch Ableitung der algebraischen Strukturen auf ein Multiagenten-System zudem klare Aussagen über die Struktur von Holonen machen.

[2]Nicht jedes Multiagenten-System ist ad hoc auf einen Vektorraum abbildbar, was den Einsatz von Verfahren ausschließt, die einen Vektorraum voraussetzen.

Kapitel 6

Dynamische Kooperation in holonische Multiagenten-Systemen

Die Kooperationen zwischen Agenten und Gruppen von Agenten ist ein wesentlicher Mechanismus in Multiagenten-Systemen um komplexe Probleme zu lösen (siehe Kapitel 2.1.2). Unter Berücksichtigung einer sich dynamisch verändernden Umwelt ist die Bildung von Kooperationen besonders schwierig. Agenten müssen dabei mit vagem und unvollständigem Wissen umgehen können und Agentengruppen, die zur Bearbeitung von komplexen Aufgaben gebildet wurden, können zerfallen, bevor die zugrundeliegenden Aufgaben erfüllt sind. Gruppenstrukturen müssen daher ständig überprüft und der aktuellen Situation angepasst werden.

6.1 Agentengruppen

Während sich Kooperationen bilden bestehen bereits Abhängigkeiten zwischen den Agenten, so dass der exakte Nutzen eines Agenten für eine Gruppe ad hoc nicht zu bestimmen ist. Die Berechnung des Nutzens eines Agenten kann nur dann korrekt vollzogen werden, wenn jeder Agent seinen Plan veröffentlicht. Ein zentrales Problem bei der Bildung von Agentengruppen ist die Nutzenbestimmung eines Agenten gegenüber einer Gruppe. Die Nutzenbestimmung ist ein wesentliches Unterscheidungsmerkmal zur Klassifikation von Gruppenformierungsverfahren. Zum einen existieren *Koalitionen*, die in der Regel auf spieltheoretischen Ansätzen zur Nutzenbestimmung beruhen, zum anderen die *Teambildung*, die auf spieltheoretische Ansätze verzichtet und in direkten Verhandlungen zwischen den Agenten den Nutzen und die Kooperationsbedingungen aushandelt. Im Folgenden werden beide Ansätze kurz vorgestellt und miteinander verglichen.

6.1.1 Rationale Koalitionen

1890 definierte Meyer [Mey90] den Begriff der Koalition folgendermaßen: *'Der Begriff Koalition stammt aus dem lateinischen und bedeutet Verbindung, Ver-*

bündung, namentlich im politischen Leben die Verbindung einzelner Parteien oder einzelner Staaten miteinander zu einem bestimmten Zweck. Von besonderer Wichtigkeit ist das Koalitionsrecht, d.h. das Recht der freien Vereinigung zur Besserung der Lage der Teilnehmer, soweit dies nicht erworbene Rechte Dritter verletzt oder das Gesamtwohl schädigt.'

Das deutsche Gesetz definiert eine Koalition auf Basis einer Vereinigung wie folgt:

- Eine *Vereinigung* besteht aus mehreren Personen, die einen Zusammenschluss auf *längere* Zeit zu einem gemeinsamen Zweck bilden - organisierte Willensbildung.
- Eine *Koalition* ist eine Vereinigung zur Wahrung und Förderung der Arbeits- und Wirtschaftsbedingungen (Unmittelbare Drittwirkung aus Art. 9 III GG) [Art].

Allen drei Definitionen ist gemein, dass eine Koalition gebildet werden soll, um in der Gemeinschaft stärker zu sein. Der Nutzen eines jeden Teilnehmers wird gesteigert, d.h. dass sich die Mitglieder einer Koalition, wie auch die neuen Mitglieder nach dem Beitritt, nicht schlechter stellen, als vor ihrem Zusammenschluss. Die Bildung von Koalitionen basiert in der Regel auf spieltheoretischen Ansätzen [SSJ98], [HW98] und kann unter dem Begriff *Koalitionsspiele* zusammengefasst werden. Osborne und Rubinstein [OR99] charakterisieren eine Koalition durch zwei Eigenschaften:

- Eine grundlegende Vorgehensweise dieses Modells ist die Zusammenstellung von Aktionen, die durch Gruppen von Spielern (so genannte Koalitionen), unabhängig von den restlichen Spielern, angenommen und ausgeführt werden können. Ein Ergebnis eines Koalitionsspiels ist eine Spezifikation / Beschreibung einer Koalitionsstruktur, die die ausgewählte Aktionsmenge dann ausführen muss.
- Die andere Eigenschaft von Koalitionsspielen ist die Berücksichtigung des Prefärenzenprofils der Spieler bei der Berechnung möglicher Ausgänge des Spiels. Folglich ist die Wahl der Aktionen durch eine Koalition abhängig von den Wünschen und Bedürfnissen der Koalitionsteilnehmer.

Nach [CS95] können die Modelle der Koalitionsformierung in zwei Gruppen, gemäß dem Prinzip 'bellum omnium contra omnes', aufgeteilt werden: in von der Spieltheorie bevorzugten *utility-based-models* [LR57], [Axe84], und *complementary-based-models*. Im Letzten vereinigen sich Agenten mit komplementären Eigenschaften, um eine Nutzensteigerung jedes Agenten der Gruppe zu erwirken. Als Gemeinschaft führen sie ihre Aufgaben aus und können so ihre Ziele erreichen, was ohne Teilnahme an der Koalition als einzelner Agent nicht möglich wäre. Noch heute folgen die meisten Methoden und Protokolle zur Bildung einer Koalition aus rationalen Agenten dem *nutzenbasierten* Ansatz. Sie setzen auf einem abgeleiteten Konzept der kooperativen Spieltheorie, Volkswirtschaftslehre und Operations Research, auf. Allgemein besteht die Koalitionsformierung aus zwei Aktivitäten:

1. Berechnung von Koalitionsstrukturen (Koordination). Dabei wird die Agentengesellschaft in Gruppen, so genannten *Koalitionen*, aufgeteilt, so dass die Summe der Einzelnutzen aller Agenten durch die gemeinschaftliche Bearbeitung der Aufgaben maximiert wird.

2. Verteilung des Profits auf die Mitglieder jeder Koalition durch eine zentrale Instanz.

Diese Aktivitäten sind oftmals ineinander verschachtelt und voneinander abhängig. Während der Koalitionsformierung behält jeder Agent seine eigenen Ziele und das Ziel der Koalition wird dem Koalitionsführer zugewiesen. Die Agenten einer Koalition teilen also kein gemeinsames Ziel. Eine Koalition ist somit loser Verbund von Agenten. Eine umfassende Klassifikation von relevanten Arbeiten im Bereich der Koalitionsformierung wird in [Kra97], [VFS01] gegeben.

Koalitionen zählen neben den so genannten Teams zu den wichtigsten Ansätzen bei der Kooperation von Agenten [ZR94], [Ket94], [SL97], [SK98], [SK99]. In Multiagenten-Systemen, bestehend aus eigennützigen Agenten, wird ein Agent nur dann einer Koalition beitreten, wenn sich sein Nutzen dabei vergrößert. Hierbei und bei der Festlegung der Koalitionsstruktur ist die Berechnung der Nutzenverteilung von großer Wichtigkeit. Spieltheoretische Verfahren zur Koalitionsformierung können dazu eingesetzt werden. Arbeiten in der Spieltheorie [Rap70], [Shi95], [Voh95] beschreiben, welche Koalitionen sich bei einem *N-Person* Spiel unter welchen Vorgaben bilden und wie die Spieler den Gewinn der Kooperation untereinander aufteilen. Das Koalitionsformierungsproblem wird unter Zuhilfenahme verschiedener Stabilitätsbegriffe gelöst. Einige der wichtigsten Definitionen zur Stabilität bietet der *Core*, der *Shapley-Wert* und der *Kernel* [KR84]. Jeder dieser Stabilitätsbegriffe ist motiviert von unterschiedlichen Methoden zur Bewertung der relativen Stärke der teilnehmenden Agenten. Jedoch werden besondere Fragestellungen von Mulitagenten-Systemen von der Spieltheorie nicht berücksichtigt, wie z.B. Kommunikationskosten, begrenzte Rechenzeit, etc.

In [SK99] betrachten Shehory und Kraus eine Koalitionsformierung von eigennützigen Agenten zur Erfüllung von Zielen. Beides, die Koalitionsstruktur und die Verteilung des Gewinns, werden hierbei berücksichtig. Eine Fülle von Anytime-Algorithmen zur Koalitionsformierung wurde bereits entwickelt, die z.B. den Kernel-Stabilitätsbegriff erfüllen. Die Eigenschaften dieser Verfahren wurden mit Hilfe von Simulationen erforscht. Als Ergebnis dieser Untersuchungen zeigte sich, dass der Nutzen der Agenten innerhalb einer angemessenen Zeitspanne zunimmt und dass mit einer höheren Anzahl von Koalitionen der individuellen Nutzen der Agenten steigt. Klusch und Shehory [KS96] wenden diese Erkenntnis bei der Formierung von Koalitionen unter Informationsagenten an. Sandholm et al. untersuchten in [SLA+99] das Verhalten der Qualität von Koalitionsstrukturen in Abhängigkeit der Größe des Suchraums möglicher Koalitionsstrukturen. Sie zeigten, wie untere Schranken bezüglich der untersuchten Strukturen in Abhängigkeit der Qualität angegeben werden können. Sandholm und Leser entwickeln in [SL95] ein Modell zur Koalitionsformierung für begrenzt rationale Agenten und präsentieren eine allgemeine Klassifikation von Koalitionsspielen. Sie konzentrieren sich dabei auf das Problem der Nutzenberechnung

einer Koalition. In ihrem Modell hängt der Wert von der zur Verfügung stehenden Rechenzeit der Agenten ab. Zlotkin und Rosenschein [ZR94] untersuchen das Problem der Nutzenverteilung in *subadditiven* und*zielgerichteten* Anwendungsdomänen. Sie betrachten nur die große Koalition, bei der alle Agenten einer einzigen Koalition angehören, und entwickeln einen linearen Algorithmus, der jedem Agenten einen Nutzen garantiert, der gleich seinem eigenen Shapley-Wert ist. Ketchpel [Ket94] präsentiert einen Mechanismus zur Nutzenverteilung für ähnliche Szenarien, bei denen es jedoch unsicher ist, welchen Gewinn eine Koalition erzielt. Das strategische Verhandlungsmodell kann zur Lösung des Nutzenverteilungsproblems eingesetzt werden, bei dem die Agenten die große Koalition bilden.

Die meisten Methoden zur Bildung von stabilen Koalitionen sind jedoch nur in statischen und vollständig informierten Umgebungen einsetzbar. Erschwert wird dies, wenn die Umgebungen nicht stabil sind und weder sicheres und noch vollständiges Wissen den Agenten zur Verfügung steht. Das Problem, unter diesen Umständen Koalitionen zu bilden, wird als *dynamic coalition formation (DCF)* bezeichnet. Zur Koordination von Kooperationen stellt die Spieltheorie eine Sammlung von analytischen Werkzeugen zu Verfügung. Die Grundannahmen dabei sind, dass die Entscheidungsträger fest definierten Spielregeln unterliegen und dass sie zur Entscheidungsfindung ihr Wissen bzw. ihre Abschätzung bezüglich des Verhaltens der anderen Mitstreiter einbeziehen. Beispielsweise stellt das *Nash-Gleichgewicht* eine Standardlösung in der Spieltheorie dar und wird auch zur Analyse von politischen Auseinandersetzungen verwendet [Nas50]. Insbesondere, wenn für die Spieler keine Möglichkeit bindender, einklagbarer Vereinbarungen besteht, muss die Lösung so gestaltet sein, dass es im Eigeninteresse aller Beteiligten liegt, sich daran zu halten.

John Nash (1951) formalisierte so ein Lösungskonzept, das sich als grundlegend für die Diskussion *konsistenter Lösungen* von nicht-kooperativen Spielen erwiesen hat. Das Nash-Gleichgewicht fordert, dass jeder Spieler eine Strategie wählt, die bei gegebener Strategie des anderen die höchste Auszahlung sichert. Ausgehend von einem Nash-Gleichgewicht besteht für keinen Spieler ein Anreiz, von seiner Gleichgewichtsstrategie abzuweichen. Damit werden die Erwartungen über das Verhalten der Mitspieler bestätigt [HI01]. Betrachtet man das Gefangenendilemma, stellt sich heraus, dass das gefundene Gleichgewicht in dominanten Strategien auch ein Nash-Gleichgewicht ist. Dies ist ersichtlich, denn ein Gleichgewicht in dominanten Strategien ist per Definition eine Situation, in der alle Spieler die Strategie gewählt haben, die ihnen, unabhängig von der Wahl des Gegners, den besseren Ertrag von den zur Verfügung stehenden Strategiekombinationen bringt. Demzufolge ist es für keinen Spieler profitabel seine Strategie zu ändern, wenn der andere seine Strategie beibehält. Dies deckt sich mit der Definition des Nash-Gleichgewichtes [OR99]. Das Nash-Gleichgewicht ist jedoch in dynamischen und instabilen Umgebungen nicht anwendbar, da die Teilnehmer hierbei nicht immer in der Lage sind, die Strategie zu verwenden, die ihren Gewinn maximiert. Dies ist z.B. dann der Fall, wenn ein Teilnehmer unbeabsichtigt aufgrund von technischen Problemen (z.B. Netzwerkfehler, o.ä.) aus einer Verhandlung ausscheidet. Bedingt durch die Randbedingungen einer DCF Anwendungsdomäne existiert per se weder eine rationale noch eine optimale

Verteilung des Gewinns. Etablierte Verfahren, wie z.B. das Nash-Gleichgewicht, können daher nicht angewendet werden. Beispielsweise ist die Verwendung des *Kernels* nur dann möglich, wenn alle Agenten ihre Aktionen rational planen (können) und auch in der Lage sind, ihre Aktionen so auszuführen, wie sie es beabsichtigen [Klu96]. Diese Annahme ist in einer dynamischen Umgebung aufgrund willkürlicher Störungen nicht möglich, weshalb z.B. der Kern und Verfahren, basierend auf dem Kernel, nicht einsetzbar sind.

Eine weitere Möglichkeit der Nutzenberechnung besteht in der vollständigen Abbildung der Aktionen und der Beschreibung eines Übergangsmodells vom aktuellen Zustand in einen Zustand, der als Resultat der entsprechenden Aktion eintritt. Es ist dann möglich, alle Reaktionsvarianten im Voraus zu bestimmen und zu bewerten. Anhand zuvor festgelegter Optimierungskriterien können nun die Koalitionsstrukturen berechnet werden, die das gegebene Problem optimal lösen. In einer stochastischen offenen Umgebung mit einem bekannten Übergangsmodell ist dieses Problem zur Berechnung optimaler Koalitionsstrukturen bekannt als *'Markov decision problem'*. Ein Übergangsmodell weist jedem Übergang eines Ausgangszustands in einen Folgezustand einen Wert aus einer Menge von Wahrscheinlichkeiten zu. Einem Agenten wird so eine Sequenz $\mathcal{X}_t$ von Werten zugewiesen, wobei sich jeder Wert aus dem vorherigen Wert des aktuellen Zustands ergibt. $\mathcal{P}(\mathcal{X}_t|\mathcal{X}_{t-1})$ ist bekannt als *state evolution model* oder auch *Markov Chain* [RN95]. Es ist jedoch in einer offenen und dynamischen Welt nicht möglich, eine solche vollständige Abbildung der Zustände und ihrer Auswirkungen exakt zu ermitteln, weshalb dieses Verfahren nicht eingesetzt werden kann. Ein alternativer Ansatz zur dynamischen Formierung von Agentengruppen bietet die Teambildung.

6.1.2 Planbasierte Teams

Kirchner u. Pink [KP93] definieren Team als *'eine Gruppe von Menschen, in der alle von guter Zusammenarbeit abhängen. Jeder Einzelne kann nur dann den größtmöglichen Erfolg, sowohl persönlich wie auch in der Sache erzielen, wenn diese Zusammenarbeit gelingt'*.

Nach Katzenbach [Kat93] ist *'ein Team eine Anzahl von Personen mit einander ergänzenden Fähigkeiten, die sich alle für ein gemeinsames Ziel einsetzen, sich nach gemeinsam entwickelten Arbeitsregeln richten und gemeinsam Verantwortung für ihre Arbeit tragen'*.

Haug [Hau98] versteht unter einem Team *'eine Gruppe von Mitarbeitern, die für einen geschlossenen Arbeitsgang verantwortlich ist und die das Ergebnis ihrer Arbeit als Produkt oder Dienstleistung an einen internen oder externen Empfänger liefert'*.

Kirchner, Pink, Katzenbach und Haug beziehen ihre Definition auf Menschen, die gemeinsam *eine* Aufgabenstellung bearbeiten, diese Definitionen lassen sich jedoch direkt auf Multiagenten-Systeme übertragen. Trotz der unterschiedlichen Definitionen lassen sich gemeinsame Merkmale herausarbeiten:

Die Arbeiten von Wiendieck [Wie93], Sycara et al. [GS01] und Tambe et al. [SPR+03] heben folgende fünf gemeinsame Merkmale eines Teams hervor:

1. **kleine, funktional gegliederte/organisierte Arbeitsgruppe**
2. **gemeinsame Zielsetzung,**
3. **intensive wechselseitige Beziehungen,**
4. **ein ausgeprägter Gemeinschaftsgeist und**
5. **ein starker Gruppenzusammenhalt unter den Team-Mitgliedern.**

Ein Team zeichnet sich insbesondere durch ein gemeinsames Ziel aus [SRG99]. Jedes Mitglied ist von der Erfüllbarkeit des Ziels mindestens solange überzeugt, wie es dem Team angehört. Hält ein Mitglied das Ziel nicht mehr für erreichbar, so wird es das Team verlassen, d.h. alle Mitglieder eines Teams haben die gleichen Intensionen bzgl. des gemeinsamen Ziels [CL91] [Jen95].Ist das gemeinsame Ziel erreicht, löst sich das Team wieder auf. Die Ausrichtung eines Teams ist also Ziel, bzw. aufgabenorientiert [SK96]. Ausschlaggebend für eine erfolgreiche Arbeit im Team sind zudem die weitreichenden Selbstbestimmungsmöglichkeiten. Das Team ist autonom bei der Umsetzung von Konzepten und Maßnahmen und löst seine Probleme unabhängig von anderen Teams. Notwendige Umstrukturierungen werden durch den Teamführer veranlasst. Bezeichnend ist auch ein entsprechendes Maß an Eigenverantwortung und Selbstbestimmung, welche Aktionen ein Agent ausführt und welchen Teams er beitritt. Gute Kommunikation und kontinuierlicher Informationsaustausch sind im Team ebenso Grundbedingungen. Alle Teams basieren auf Regeln, die die Zusammenarbeit und das Miteinander in der Gruppe bestimmen. Aktive Beteiligung, basierend auf der Selbstbestimmung jedes Teammitglieds am Gruppenprozess, ist ein Grundmerkmal der Teamarbeit und Bedingung für eine erfolgreiche Problemlösung. Abschließend besteht ein gut funktionierendes Team aus dem Zusammenwirken unterschiedlicher Ideen, Persönlichkeiten, Erfahrungen und Arbeitsweisen, mit dem Ziel, effektive Leistungen zu erbringen [Car98], [GS01].

Vergleich zwischen Teams und Koalitionen

Teams sind den Koalitionen zwar in den Merkmalen sehr ähnlich, jedoch sind die zugrundeliegenden Verfahren zu ihrer Bildung sehr verschieden. Während bei der Koalitionsformierung spieltheoretische Ansätze für die Koordination Verwendung finden, wird bei der Teambildung das Problem zur Bildung von Agentengruppen in dynamischen Umgebungen als dezentrale Suche betrachtet [GL03]. Durch die Verwendung spezieller Suchverfahren, die auf einem variablen Suchraum operieren können, ist es möglich, in dynamischen Umgebungen eine Lösung des Kooperationsproblems zu finden. Eine Schwierigkeit bei der Gruppenbildung in dynamischen Umgebungen ist, dass es in der Regel nicht möglich ist, vollständiges Wissen zu erlangen. Dies ist jedoch für die meisten spieltheoretischen Methoden Grundvoraussetzung zur Nutzenbestimmung der Agenten. Die dieser Arbeit zugrundeliegende Annahme, dass Agenten jederzeit

der Gesellschaft beitreten, bzw. sie verlassen können, beschreibt gerade solche dynamischen Effekte, die den Einsatz von Koalitionen erschweren. Klusch et al.[BKS03] schlagen einen neuen Ansatz zur Koalitionsformierung vor, der die Problematik hinsichtlich von dynamischen Umgebungen[1] herkömmlicher Koalitionsformierungsalgorithmen löst. Diese Lösung setzt Fuzzy Koalitionswerte ein und ermöglicht den Agenten, stabile Koalitionen zu bilden. Dazu werden Konzepte der Fuzzy-Mengentheorie mit dem spieltheoretischen Stabilitätskriterium des Kernels kombiniert, um ein neues Konzept des *Fuzzy-Kernels* einzuführen. Aufbauend auf dem Kernel-basierten Koalitionsformierungsalgorithmus KCA[Klu96] führen die Autoren einen erweiterten Algorithmus KCA-F ein, bei dem der Fuzzy-Kernel eingesetzt wird.

Spieltheoretische Ansätze lassen typischerweise keine Überlappungen in den Koalitionsstrukturen zu. Jedoch gibt es auch hier spezielle Verfahren, die diese Ansätze erweitern, so dass überlappende Koalitionen dennoch möglich sind [SK96]. Die Definition von Holonen schließt eine Überlappung nicht aus. Ein Holon (bestehend aus mehreren Agenten) ist zudem in der Lage an mehreren Gruppen von Agenten gleichzeitig teilzunehmen. Die allgemeine Sichtweise von Teams sieht generell keine Einschränkung der Mitgliedschaft der Agenten vor, so dass hinsichtlich der Überlappungseigenschaft *alle* Teambildungsverfahren einsetzbar sind. Mit Hilfe der Spieltheorie können mittel- und langfristige Strategien der Teilnehmer analysiert und prognostiziert werden. Koalitionsformierungsalgorithmen kommen daher in Domänen zum Einsatz, bei denen solche Abschätzungen möglich und notwendig sind. Dies setzt jedoch voraus, dass sich die Umwelt abschätzbar dynamisch verhält. Ist das Verhalten der autonomen Agenten hingegen nicht vorhersehbar, eignen sich Teambildungsverfahren besser. Teams werden gebildet, um einzelne komplexe Aufgaben gemeinschaftlich zu lösen. Koalitionen hingegen sind von längerem Bestand und werden in der Regel nicht nur für die Bearbeitung einer einzigen Aufgabe gebildet, sondern überdauern unter Umständen die Bearbeitung mehrerer komplexer Aufgaben. Teams werden primär zur Erfüllung eines Ziels gebildet, d.h. pro (Sub-)Ziel wird ein (Sub-)Team gebildet. Koalitionen werden hingegen zur Steigerung des monetären Gesamtnutzens unabhängig der Aufgaben gebildet [BD03].

Tabelle 6.1 zeigt eine Gegenüberstellung der Eigenschaften eines Teams und einer Koalition, die für die Bildung von Agentengruppen in dynamischen Umgebungen von Bedeutung sind. Als wichtigstes Unterscheidungsmerkmal zwischen Teams und Koalitionen gilt die Handhabung der Ziele und Pläne der Agenten. Alle Agenten eines Teams folgen einem gemeinsamen Ziel, bei Koalitionen verfolgt jeder Agent seine eigenen Ziele. Ein rationaler Agent tritt daher einer Koalition nur dann bei, wenn sein Nutzen/Profit dabei steigt. Ein weiteres Unterscheidungsmerkmal zwischen Koalitionen und Teams ist, dass der Schwerpunkt bei der Koalitionsformierung auf der Nutzenbestimmung und bei der Teambildung auf dem koordinativen Prozess der Aufgabenverteilung liegt. Tabelle 6.2 zeigt einen Vergleich unterschiedlicher Ansätze und Definitionen von Koalitio-

[1] Allgemeine spieltheoretische Verfahren basieren auf der Annahme, dass die Werte von Koalitionen bekannt sind und zum Zeitpunkt der Koalitionsverhandlungen stabil und sicher sind, was in dynamischen Umgebungen so nicht gewährleistet werden kann.

	Team	Koalition
Zweck	Bearbeitung einer Aufgabe	allgemeine Nutzensteigerung
Ziel	gemeinsames Ziel	eigene Ziele
Pläne	zum Teil gemeinsame Pläne	getrennte Pläne
Gruppenstruktur	fest / hierarchisch	loser Verbund
Autonomie der Subagenten zur Bildung	hoch	niedrig
Autonomie der Subagenten zur Ausführung	niedrig	hoch
Fluktuation der Mitglieder	hoch	niedrig, wegen stabilen Koalitionen
Verfahren zur Nutzenberechnung	Suche	Spieltheorie
Überlappende Gruppenstrukturen	generell ja	nur spezielle Verfahren
Dauer der Zusammenarbeit	kurzfristig	längerfristig

Tabelle 6.1: Vergleich der Unterschiede zwischen Team und Koalition

nen und Teams. Dabei wird deutlich, dass die Trennung der beiden Begriffe nur sehr schwer möglich ist und in einigen Punkten, wie z.B. der 'Lebensdauer' einer Gruppe, keine einheitliche Vorstellung existiert.

6.1.3 Strukturierte Holone

Holonische Multiagenten-Systeme beschreiben eine flexible Strukturierungsmethode zur Bildung von Agentengruppen, die besonders für den Einsatz in dynamischen Anwendungsszenarien geeignet sind. Im Gegensatz zu den beiden vorgestellten Kooperationsmethoden, Koalitions- und Teambildung, gibt das Holonenparadigma Richtlinien zum Design eines Multiagenten-Systems vor, die Wahl der Kooperationsmechanismen bleibt dem Systemdesigner frei überlassen. Gerber [Ger99b] und Vierke [Vie00] geben in der Definition eines holonischen Agenten sechs Charakteristika eines Holon an:

1. **Eingeschränkte Autonomie:** Agenten, die sich zu einem Holon zusammenschließen, handeln als eine Einheit mit festen Hierarchien. Die Autonomie jedes Teilnehmers wird dabei eingeschränkt, um als eine Einheit auftreten zu können.

2. **Gemeinsames zielgerichtetes Verhalten:** 1) Das Holon hat ein eigenes Ziel, das die Grundlage für seine Bildung ist. 2) Die Sub-Agenten eines Holons verfolgen weiterhin ihre eigenen Ziele, von denen mindestens ein Ziel dem gemeinsamen Ziel des Holons entspricht.

Ansätze:	[GL03]	[SPR+03]	[GS01]	[SYIV02]	[OR99]	[Kra01]
Suche	Team	Team	Team			
Spieltheorie	Koalition				Koalition	Koalition
Bearbeitung einer Aufgabe	Team	Team	Team			
allgemeine Nutzensteigerung	Koalition				Koalition	Koalition
gemeinsames Ziel/Pläne	Koalition	Team	Team	Team		
eigene Ziele/Pläne	Team			Koalition	Koalition	Koalition
feste Hierarchie	Team		Team	Team		
loser Verbund	Koalition			Koalition	Koalition	
Dauer: kurz	Team			Koalition		
Dauer: langfristig	Koalition			Team		Koalition
Gruppe endet mit der Umformierung	Team		Team			
Aus-/Eintritt zu jeder Zeit	Koalition					

Tabelle 6.2: Vergleich unterschiedlicher Ansätze zur Agentengruppenbildung

3. **Steigerung der Fähigkeiten:** Die Fähigkeiten der Agenten fügen sich bei einem Holon zusammen und ermöglichen einem Holon auf einer abstrakten Ebene der Aufgaben komplexere Dienste anbieten zu können.

4. **Glauben:** Die Agenten eines Holons behalten ihre eigene Repräsentation der Umwelt. Jedoch kann dieses Wissen bei Bedarf explizit durch den Holonenführer gesammelt und dargestellt werden.

5. **Begrenzte Rationalität:** Ein Holon kontrolliert seine Ressourcen, um ein begrenztes rationales Verhalten zu erlangen. Die Ressourcenverwaltung der Sub-Agenten wird vom Holon überwacht, das Richtlinien an seine Sub-Agenten für die lokale Ressourcenverwaltung vergibt.

6. **Kommunikation:** Sie ist eine elementare Eigenschaft von Agenten, die auch innerhalb des Holons weiterhin besteht. So beruht die Organisation eines Holons auf einer intensiven Kommunikation.

Ein Vergleich von Holonen mit den Eigenschaften von Teams und Koalitionen, wie sie in Tabelle 6.1 durchgeführt wurde, zeigt, dass eine Vielzahl der Eigenschaften, die Teams und Koalitionen unterscheiden, von Holonen vereint werden. Der *Kooperationszweck* kann sowohl die Bearbeitung einer einzelnen Aufgabe sein, wie die allgemeine Nutzensteigerung der Gruppe. Damit zusammenhängend ist die *Dauer* der Zusammenarbeit in einem Holon variabel und reicht von kurzfristigen Kooperation bis hin zu langfristigen Vereinigungen. Die Definition von Holonen lässt verschiedene Gruppenstrukturen zu, die den Agenten eines Holons während der Holonenbildung und zur Ausführungszeit des Holons mehr oder weniger *Autonomie* zugesteht, so dass zur Holonenformierung beliebige Mechanismen der Gruppenbildung verwendet werden können. Entsprechend

Eigenschaften eines Teams	⟶	Eigenschaften eines Holons
1. Kleine, funktional gegliederte organisierte Arbeitsgruppen		1. Eingeschränkte Autonomie und 5. Begrenzte Rationalität
2. Gemeinsame Zielsetzung		2. Gemeinsames zielgerichtetes Verhalten
3. Intensive wechselseitige Beziehungen		6. Kommunikation
4. Ausgeprägter Gemeinschaftsgeist		4. Glauben
5. Starker Gruppenzusammenhalt und feste Hierarchien		1. Eingeschränkte Autonomie und 3. Steigerung der Fähigkeiten

Tabelle 6.3: Abbildung der Teameigenschaften auf die Eigenschaften eines Holons

der Struktur eines Holons, können die Agenten zu einem neuen Agenten fest verschmelzen, wodurch das Holon die *Pläne* der Subagenten komplett verwaltet und somit gemeinsame Pläne entstehen. Oder das Holon besteht aus einem losen Verbund von kooperierenden Agenten, die ihre Pläne getrennt verwalten. Hinsichtlich dieser Eigenschaften können sowohl Team-, wie auch Koalitionsformierungsmechanismen zur Bildung eines Holons eingesetzt werden. Betrachtet man die restlichen Merkmale der in Tabelle 6.1 diskutierten Eigenschaften, so zeichnet sich eine Tendenz ab, im Allgemeinen Teamformierungsverfahren zur Holonenformierung einzusetzen, wobei jedoch in manchen Anwendungsdomänen auch Koalitionsformierungsverfahren alternativ sinnvoll eingesetzt werden können.

Als Grundlage für die **Holonenbildung** habe ich den Einsatz von **Teambildungsverfahren** gewählt, da insbesondere die hierarchischen Teamstruktur denen eines Holons sehr ähnlich ist, die generelle Überlappung der Teams die rekursive Definition eines Holons sicherstellt und die Behandlung von gemeinsamen Zielen bei Teams eher dem gemeinsamen zielgerichteten Verhalten der Mitglieder eines Holons entspricht als bei Koalitionen.

Abbildung der Teamdefinition auf die Holonendefinition

Ein Vergleich der Definitionen von Teams und Holonen (siehe Tabelle 6.3) verdeutlich die zuvor getroffene Aussage, dass Teams (und Koalitionen) als Spezialisierung von Holonen angesehen werden können und eine Abbildung von $Team \rightarrow Holon$ existiert. Merkmal 1 der Teamdefinition beschreibt die interne Kontrollstruktur eines Teams hinsichtlich den Fähigkeiten und Ressourcen der Teilnehmer. Ähnlich dem Team können Agenten zu kleinen funktionalen Gruppen mit einer festen Hierarchie innerhalb eines Holons zusammengefasst werden. Die Agenten des Holons benötigen dabei nur eine begrenzte Rationalität, was

aus der funktionalen Zusammenfassung hervorgeht. Dabei behalten die Mitglieder eines Teams, wie auch die eines Holons, die Autonomie über die Kontrolle ihrer Ressourcen. Die gemeinsame Zielsetzung ist in beiden Definitionen gleichermaßen enthalten. Die intensiven wechselseitigen Beziehungen werden durch eine ausgeprägte Kommunikation unter den Sub-Agenten eines Holons realisiert, wie sie auch den Teams zugrunde liegt. Der Gemeinschaftsgeist eines Teams zeichnet sich durch den Austausch von Information zwischen den Mitgliedern des Teams aus, analog der Forderung der internen Verteilung des Wissens auf die Mitglieder eines Holons. So führt der starke Gruppenzusammenhalt unter den Team-Mitgliedern auch beim Holon zu einer Steigerung der Fähigkeiten der Gruppe, so dass neue zusätzliche Dienste von der Agentengruppe angeboten werden können.

6.2 Grundlagen von dynamischen Kooperationen

Die Agenten sammeln Wissen über die Welt aus vergangenen Verhandlungen und Kooperationen und erhalten mit der Zeit vages Wissen über die Fähigkeiten der anderen Agenten. Mit diesem Wissen können Agenten gezielt für die Verhandlungen ausgewählt werden. In der Regel müssen so nicht alle n Agenten der Agentengesellschaft angeschrieben werden, bis ein Agent gefunden wird, der die Aufgabe bearbeiten kann. Besitzt der ausschreibende Agent hingegen kein Wissen über die Umwelt, muss er alle n Agenten anschreiben (vgl. Contract-Net Protokoll), um die beste Lösung zu finden. Dabei werden auch die Agenten angeschrieben, die aufgrund ihrer Fähigkeiten die Aufgabe nicht bearbeiten können. Verfügt der ausschreibende Agent hingegen über sehr exaktes Wissen, ist es möglich den besten Agenten im Voraus zu identifizieren. Im günstigsten Fall muss nur ein Agent für die Zuweisung einer Aufgabe angeschrieben werden. Agenten mit (vagen) Wissen können das Kommunikationsaufkommen um die Anzahl l der Agenten reduzieren, die aufgrund ihrer Fähigkeiten die Aufgaben nie bearbeiten können. In einer offenen und dynamischen Umgebung sind jedoch die Pläne der Agenten nicht (vollständig) bekannt, so dass der Holonenführer zwar eine Vorauswahl geeigneter Agenten treffen kann, jedoch im ungünstigsten Fall dennoch alle $n-l=k$ Agenten anschreiben muss, bzw. die Suche wird nach k erfolglosen Versuchen abgebrochen.

6.2.1 Wahl der Holonenpartner

Wurde eine Auswahl von relevanten Agenten getroffen, bestehen mehrere Möglichkeiten die Agenten zu kontaktieren. Der Agent kann eine *aktive* Rolle übernehmen und die relevanten Agenten eigenständig benachrichtigen. Dabei sind an den Verhandlungen ausschließlich die Agenten beteiligt, die vom Holonenführer ausgewählt wurden. Der Erfolg und die Qualität des Holons sind maßgeblich vom Wissen des Ausschreibenden abhängig. Eine andere Möglichkeit ist z.B. die Verwendung von *Matchmakern* [Klu01], die die Angebote und Nachfragen von Agenten sammeln und geeignete Teilnehmer vermitteln. Aufgrund der Existenz von detaillierteren Informationen eines Matchmakers über die Bedürfnisse der Agentengesellschaft können bessere Verhandlungsergebnisse erzielt werden,

Selektor	optimale Partner	Freiheitsgrad	Datensicherheit	zentraler Ansatz
Agent/Holon	abhängig vom Wissen der Agenten	freie Partnerwahl	maximal	nein
Matchmaker	optimal auf der Menge der registrierten Agenten	beschränkt die Auswahl des Matchmakers	abhängig vom Matchmaker	ja
Blackboard	abhängig von der Zugriffsfrequenz der Agenten	freie Partnerwahl	keine	ja

Tabelle 6.4: Vergleich verschiedener Vorgehensweisen bei der Selektion von potentiellen Holonenmitglieder

jedoch auf Kosten der Robustheit und Geheimhalten von Informationen. Die Agenten müssen dem Matchmaker vertrauen, denn auf die Wahl der Kontakte und Weitergabe der zuvor abgegebenen Informationen haben sie keinen Einfluss mehr. Soll stattdessen den Agenten maximale Freiheit bei der Wahl ihrer Verhandlungspartner und der Informationspreisgabe gewährleistet werden, so kann z.B. ein *black-board* Mechanismus eingesetzt werden. Diese Methode bietet den Agenten in der Wahl der Verhandlungspartner maximale Freiheiten, jedoch unter vollständigem Verlust der Geheimhaltung von Informationen. Die Preisgabe von Informationen ist in einigen Fällen unerwünscht, da über den Inhalt und die Dauer der 'Aushänge' Rückschlüsse über die Strategien der Ausschreiber getroffen werden können [SV98], [SK99]. Tabelle 6.4 zeigt eine Gegenüberstellung der genannten Vorgehensweisen auf.

6.2.2 Ausschreibungs- und Verhandlungsmechanismen

'Verhandlung' ist ein vager Begriff, der eine Vielzahl von Mechanismen zur Interaktion zwischen unterschiedlichen Parteien zusammenfasst. Obwohl Verhandlungen sehr häufig in Multiagenten-Systemen eingesetzt werden, ist es dennoch schwierig eine formale Definition zu geben, die sinnvolle Unterscheidungen zwischen einzelnen Verhandlungstypen beschreibt. Die wesentliche Frage, die Wissenschaftler 1998 auf dem *International Workshop on Multiagenten-Systems (IWMAS'98)* erörterten, ist, welche Interaktionen als 'Verhandlung' bezeichnet werden und welche nicht. Im nicht-technischen Sinne wird der Begriff 'Verhandlung' häufig in wirtschaftlichen und politischen Szenarien verwendet. Die klassischen Beispiele wirtschaftlicher Verhandlungen haben den Abschluss eines Vertrags zwischen den verhandelnden Parteien zum Ziel. Verhandlungsmechanismen sind charakterisiert durch die *Vielfalt* der Teilnehmer. Gewöhnlicherweise sollen *viele Teilnehmer* in Verhandlungen angesprochen werden. Die Koordination des Flugverkehrs zum Beispiel erfordert die Lösung aller Bedürfnisse der Flugzeuge innerhalb einer Flugzone. Das Design eines Produkts erfordert z.B. das Einverständnis vieler Abteilungen, wie Marketing, Service und Produktion. Verhandlungen werden häufig eingesetzt, um die Bedürfnisse *mehrere Objekte/Parteien* zu vereinen. Häufig sind *mehrere Verhandlungsrunden* notwendig,

um ein einziges Problem zu lösen. Die klassischen iterativen Auktionen (English- und Dutch-Auction) fallen in diese Kategorie, im Gegensatz zu den 'closed-bid procurement' Mechanismen, die nur einen einzigen Zyklus erlauben (wie z.B. First-Price Sealed-Bid Auction). Verhandlungen zeichnen sich durch den Austausch von Geboten über mehrere Zyklen aus. So sind nicht alle Auktionstypen Verhandlungen, wie z.B. die First-Price Sealed-Bid Auction, die als Ausschreibungsmechanismus typisiert werden kann. Van Parunak [Par98] definiert drei Eigenschaften einer Verhandlungstechnik:

1. Verhandlungstechniken unterscheiden sich in dem Grad des 'allgemeinen Wissens' über die vermuteten Zusammenhänge. Verhandlungen können in zwei Klassen aufgeteilt werden: Kooperative und konkurrierende Verhandlungen [LSM97]. Kooperative Verhandlungen unterscheiden sich von konkurrierenden darin, dass hierbei alle Parteien nach Abschluss der Verhandlungen einen Mehrwert erzielen. Kooperative Verhandlungen, wie z.B. beim Produktdesign, sind in der Regel so strukturiert, um das allgemeine Wissen unter den Verhandlungsteilnehmern zu steigern und somit eine Lösung des Verhandlungsproblems zu finden. Kooperative Verhandlungen können zur Koordination mehreren Parteien eingesetzt werden. In konkurrierenden Verhandlungen hingegen sind die Teilnehmer bestrebt, den Informationsaustausch so gering wie möglich zu halten. Gutman und Maes geben in [GM98] eine detaillierte Untersuchung der Unterschiede zwischen konkurrierenden und kooperativen Verhandlungen.

2. Verhandlungstechniken unterscheiden sich in der Reaktion auf Seiteneffekte. Ein Designentwurf wird evtl. durch einen anderen (bei Missfallen) während einer laufenden Verhandlung ersetzt, ohne den Verhandlungsablauf an sich zu verändern. Handelt es sich beispielsweise um eine Friedensverhandlung, so können bereits wenige falsch gewählte Äußerungen zum Scheitern der Verhandlung führen.

3. Abschnitte verschiedener Verhandlungen können voneinander abhängig bzw. von Aktionen im weiteren Sinn vernetzt sein. Eine Störung in der Umgebung eines Agenten kann dazu führen, dass Verhandlungen erneut durchgeführt werden müssen.

Es existieren sehr viele verschiedene Ansätze, Verhandlungen auszuführen. Drei davon werden im Folgenden vorgestellt, die sich für die Umsetzung in einem Multiagenten-System eignen.

Multi-Attributverhandlungen

Wie in dem Designbeispiel eines Produktes deutlich wird, hängt der Verhandlungsablauf oftmals nicht nur vom zu verhandelnden Preis ab, sondern von einer Menge unterschiedlicher Verhandlungsattribute. In einer Multi-Attributverhandlung, wie es Jonker und Treur [JT01] vorschlagen, hat ein Gebot die Form eines gewichteten Mittels, das ein Repräsentant für die Ausprägungen der Attribute ist. Handelt es sich beispielsweise um eine Verhandlung zu einem Autoverkauf und die relevanten Verhandlungsattribute sind *CD-Player,*

Extralautsprecher, Klimaanlage und der *Preis* des Autos mit allen gewünschten Extras, dann besteht ein Gebot aus einem Identifikator, der den CD-Player beinhaltet, der beschreibt welche Extralautsprecher, Klimaanlage und wie der Preis des Gebots ist. Um die Gebote der anderen Teilnehmer abschätzen zu können, ist es notwendig, eine Evaluationsmethode einzusetzen. Die Evaluation findet auf zwei Ebenen statt: auf der Attributebene (*attribute evaluation*), und auf der Ebene des Gebots als Ganzes (*overall bid utilitiy*). Unter Berücksichtigung dieser Aufteilung können folgende Charakteristika einer Multi-Attributverhandlung formuliert werden:

1. Explizites *reasoning* über die Verhandlungsstrategie und Koordination des Verhandlungsprozesses
2. Die Evaluation eines Gebots umfasst die Analyse der einzelnen Attribute und der overall bid utilitiy
3. Zur Planung eines neuen Gebots werden beide Entscheidungsebenen, overall bid utilitiy Ebene und die Attributebene, separat berücksichtigt.

Das Verhandlungsmodell von Jonker und Treur ist eine modulare Struktur, die innerhalb der Kooperations-Management Komponente des *Generic Agenten Model* [BJT00] eingesetzt wird. Das Verfahren ist in fünf aufeinander folgende Schritte unterteilt:

- Für jede Verhandlungsrunde findet zuerst eine Evaluation der Attributausprägungen der Gebote aus den vorherigen Runden statt.
- Anschließend werden die Ergebnisse der Evaluation in die übergeordneten Bewertungen (overall utilities) der vorherigen Gebote aggrigiert.
- Daraus wird eine Zielbewertung (target utility) ermittelt, die beschreibt, welche Konzessionen beim nächsten Gebot bezüglich des Gebotwerts über allen Attribute gemacht werden muss, um die Verhandlungen weiterführen zu können.
- Um das nächste Gebot, basierend auf der zuvor berechneten Zielbewertung, zu bestimmen, werden mögliche Verteilungsszenarien bestimmt, die beschreiben, welche Werte die einzelnen Attribute annehmen können, um in der übergeordneten Bewertung die Zielbewertung zu erreichen.
- Zum Schluss wird aus der Menge von Verteilungsszenarien, dasjenige ausgewählt, dessen Attributwertausprägungen am nahesten den Attributwerten der Zielbewertung sind. Dieses Verteilungsszenario bildet dann das neue Gebot.

Um das eben beschriebene Verfahren zu realisieren, besteht die Kooperations-Management Komponente eines Agenten aus fünf Modulen (siehe Abbildung 6.1): Koordination der Verhandlung, Evaluation der Verhandlungsattribute, Bestimmung der Zielbewertung, Planung der Verteilungsszenarien und Planung der Attributwerte des Gebots. Mit Hilfe der Koordinationskomponente

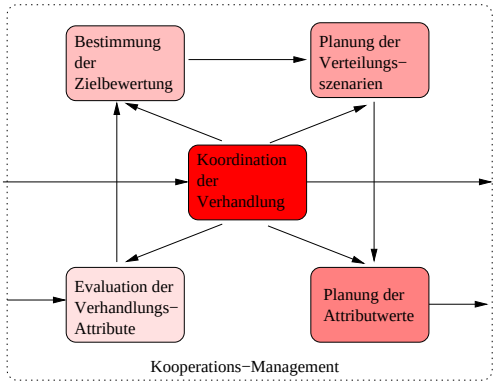

Abbildung 6.1: Komponentenbasiertes Multi-Attributverhandlungs-Modell

ist es möglich, direkten Einfluss auf die restlich vier Komponenten auszuüben. So verteilt beispielsweise die Koordinationskomponente als zentrale Instanz des Multi-Attributverhandlungs-Modells die eingegangenen Informationen der letzten Verhandlungsrunde an das Evaluationsmodul oder variiert die Toleranzen zwischen dem eigenen Verhandlungsziel und der aktuellen Notwendigkeit, davon abzuweichen, um die Verhandlungen weiterführen zu können. Die Verhandlung kann so während der Ausführung den aktuellen Ereignissen und Strategien der Verhandlungspartner angepasst werden. Das hier beschriebene Modell von Jonker und Treur ist besonders für direkte Verhandlungen (insbesondere bei bilateralen Verhandlungen) geeignet. Soll jedoch eine große Anzahl von Parteien an einer Verhandlung teilnehmen, können solche aufwendigen Verhandlungsmechanismen nicht mehr eingesetzt werden. Stattdessen werden Auktionsmechanismen zur Organisation der Verhandlungen verwendet, die genaue Vorschriften für den Austausch von Geboten, den zeitlichen Ablauf der Verhandlungen und Kriterien für die Auswahl von Auktionsgewinnern vorgeben. Das Verfahren von Jonker und Treur kann bei Auktionen für die Gebotsbestimmung eines Agenten zusätzlich eingesetzt werden.

Auktionen

Auktionen sind spezielle Verhandlungsmechanismen, die unabhängig von der Anzahl der Verhandlungsteilnehmer durchgeführt werden können. Im Unterschied zu multilateralen Verhandlungen tritt in Auktionen der Anbieter nicht direkt mit den Käufern in Kontakt, um die Verhandlungen auszuführen. Hierbei ist ein Auktionator zwischengeschaltet, der die Ausschreibung des Angebots übernimmt, die Auktion ausführt und dabei die Angebote entsprechend dem Auktionsverfahren sammelt, die Auktion nach Beendigung auswertet und den Anbieter mit dem 'Gewinner' der Auktion in Kontakt bringt. *'An auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis from the market participants.'* [MM87] Unter den Auktionen wird unterschieden zwischen *single sided auctions* und *double sided auctions*. Bei den single sided auctions handelt es sich um einen Auktionstyp, bei dem aus-

schließlich die Mitbieter Gebote abgeben. Der Begriff der double sided auctions beschreibt ein wesentliches Charakteristikum dieses Verhandlungsmechanismuses: Beide Marktseiten besitzen symmetrische Handlungsmöglichkeiten, indem Nachfrager Kauf-Offerten und Anbieter Verkaufs-Offerten vornehmen. In der Praxis werden double sided auctions besonders auf Börsenmärkten eingesetzt und haben sich dort als effiziente Verhandlungsmechanismen erwiesen. Einen guten Überblick über dieses Anwendungsgebiet bieten die Arbeiten von [Fri93], [BS98] und [Dom93]. Im Folgenden wird ein Überblick über mögliche Auktionstypen gegeben, beginnend mit den single sided auctions:

- single sided auctions
  - *English Auction:* Aufsteigende Auktion, bei der sich alle Bieter zum gleichem Zeitpunkt an einem gemeinsamen Ort befinden, um Gebote abzugeben. Der Auktionator bittet dabei um ständig steigende Gebote bis ein Höchstgebot gefunden wird, das von keinem mehr überboten wird, oder ein Zeitlimit zum Ende der Auktion abgelaufen ist. Der Gewinner erhält die angebotene Ware zu dem Preis, den er geboten hat [BSS96].
  - *First-Price Sealed-Bid Auction:* Bieter geben für jedes angebotene Objekt genau ein verdecktes Gebot ab. Die geheimen Gebote werden gesammelt und zu einem zuvor festgelegten Zeitpunkt, an dem die Auktion endet, geöffnet und miteinander verglichen. Der Bieter mit dem höchsten Gebot gewinnt die Auktion und bezahlt den Preis seines Gebots [BSS96].
  - *Vickrey Auction:* Dieser Auktionsmechanismus unterscheidet sich nur gering von der First-Price Sealed-Bid Auction. Der Unterschied besteht darin, dass der Meistbietende den Zuschlag zum Preis des nächstbesten Gebots erhält. Die Vickrey Auction, oder auch Second-Price Sealed-Bid Auction, liefert die gleichen Erlöse wie die English Auction, wenn man annimmt, dass der Meistbieter bei einer English Auction nur einen geringfügigen Zuschlag auf das zweitbeste Gebot zahlen muss. Die Bieter müssen im Falle der Second-price sealed-bid Auction aber nicht über die Angebote der Mitbieter informiert sein. Der Vorteil dieses Verfahrens besteht demnach im wirtschaftlichen Einsatz von Information, die ein Bieter zur Erstellung eines optimalen Gebots benötigt. Es genügt, dass ein Bieter nur seine eigenen Preisvorstellungen kennt, nicht jedoch die der anderen Bieter. Deswegen müssen sich die an der Auktion Teilnehmenden weder zeitlich noch örtlich verabreden [RS81] [Bic99] [Vic61].
  - *Dutch Auction:* Kontinuierlich absteigende Auktion, bei dem der Anbieter ein Startgebot vorgibt, das in fest definiert Zeitintervallen um einen festen Betrag reduziert wird. Gewinner ist der Bieter, der sich als erster zu dem aktuellen Gebot meldet. Er muss dann den Preis des aktuellen Gebots des Auktionators bezahlen [BSS96].
  - *Matrix Auction:* Matrix-Auktionen [GSV99b] [GSW98] können im Gegensatz zu den bisher genannten Auktionstypen eingesetzt wer-

den, um gleichzeitig mehrere Objekte innerhalb einer Auktion anzubieten. Der Auktionator bietet in einer Matrix-Auktion k Objekte den Bietern an, die ihrerseits für jede mögliche Zusammenstellung der k Objekte ein Gebot berechnen und diese dem Auktionator melden. Aufgrund der gesammelten Daten ermittelt der Auktionator die optimale Verteilung der Objekte auf die Agenten nach individuellen Parametern. So kann z.B. nach dem Gewinn für den Anbieter optimiert werden oder eine Verteilung gesucht werden, bei der möglichst viele Objekte verkauft werden oder die Wünsche der Bieter bestmöglich erfüllt werden. Der zu zahlende Preis entspricht dem zweithöchsten Gebot.

  - *Reverse Auction:* In einer Reverse Auction wird kein Gegenstand angeboten, bei dem die Gebote den Preis für diesen Gegenstand beschreiben, sondern ein Anbieter möchte einen Gegenstand erwerben und die Bieter geben Gebote ab, die den Preis enthalten, zu dem sie bereit sind, ihre Ware zu verkaufen. Der Gewinner erhält seinen Preis vom Anbieter und der Anbieter erhält die Ware vom Bieter.

- double sided auctions

  - *Continuous Double Auction:* Beide, Anbieter und Bieter, geben Gebote ab, die dann vom höchsten zum niedrigsten Gebot sortiert werden, um die Profile für Angebot und Nachfrage zu erstellen. Die Auktion endet und der Handel wird initiiert, sobald ein Gebot eines Anbieters mit dem Gebot eines Bieters übereinstimmt. Der Bieter zahlt den Preis seines Angebots [Bic99].
  - *Periodic Double Auction:* Ähnlich geben bei einer Continuous Double Auction Anbieter und Bieter Gebote bezüglich eines Verhandlungsobjekts ab. Jedoch ist die Auktion zeitlich begrenzt. Nach Ablauf der Auktionsfrist, wird der Bieter mit dem höchsten Gebot und der Anbieter mit dem niedrigsten Angebot zusammen geführt [Bic99].
  - *Multi-Attribute Auction:* In Gegensatz zu den bisher genannten Auktionen ist es in Multi-Attribute Auctions möglich, um mehr als nur den Preis zu verhandeln. So können beispielsweise der Preis, Menge und Anlieferungszeitraum einer Ware verhandelt werden. Für die Bieter ist es hierbei umso wichtiger den genauen Nutzen bezüglich des Angebots in allen Verhandlungsattribute zu bestimmen und mit dem Anbieter zu kommunizieren. Durch ein entsprechendes Feedback des Anbieters sind die Bieter in der Lage ihre Gebote gegenüber den anderen Bietern anzupassen [Bic99].

Auktionsmechanismen eignen sich besonders für Verhandlungen, an denen mehrere Parteien (Anbieter und Bieter) teilnehmen. Beam und Segev bieten anhand einer Analyse von 100 Internetauktionshäusern einen Überblick über die aktuelle Entwicklung. Als Ergebnis dieser Untersuchung [BS98] ergibt sich: Mehr als die Hälfte der beobachteten Auktionsplätze bieten Auktionen von Business-to-Consumer an, ein viertel der Sites bieten Consumer-to-Consumer Auktionen.

Business-to-Business Auktionen stehen mit 6% an der letzten Stelle. Als Auktionsgüter kommen meist einfach zu versendende, materielle Güter in Frage, gefolgt von immateriellen Gütern wie Software und Dienstleistungen.

Bemerkung. Vergleicht man die beiden Auktionstypen First-Price Sealed-Bid und die Second-Price Sealed-Bid mit dem in Multiagenten-Systemen stark verbreiteten Contract-Net Protokoll, so ist das zugrunde liegende Verfahren dasselbe. Lediglich die konkrete Bestimmung der Kosten und das Ende des Verfahrens variieren. In allen drei Fällen handelt es sich um so genannte *one-shot* Verfahren, bei denen die Abgabe der Gebote zeitlich nicht von den zuvor abgegebenen Geboten der anderen Teilnehmer abhängig ist.

Holonische Auktion

Ein Spezialfall der Auktionen sind holonisch strukturierte Auktionshäuser [GR01a]. Ziel dieser Erweiterung herkömmlicher Auktionshäuser ist es, mit Hilfe von holonischen Agenten die Koordination zwischen mehreren Auktionen zum Erwerb eines Gegenstands, bzw. mehrerer Gegenständen auf verschiedenen Auktionen zu synchronisieren, insbesondere wenn der Erwerb eines Gegenstands von dem Erfolg in anderen Auktionen abhängig ist.

6.2.3 Eigenschaften von dynamischen Kooperationen

Agenten, die auf dynamisch auftretende Ereignisse reagieren sollen, müssen in der Lage sein, Um- und Neuplanungen durchführen zu können. Dazu müssen sie ihre Aktionen erneut untereinander in den Planabschnitten synchronisieren, welche Modifikationen in den Überlappungen der Agentenplänen ausführbar sind, ohne dass die Einzelpläne der Agent verworfen werden müssen. Jeder Agent entscheidet individuell, wie er auf Ereignisse reagiert, die sich auf die Ausführung seines Plans auswirken:

- Stornieren aller betroffenen Aufgaben, die aufgrund der Veränderungen nicht mehr korrekt ausgeführt werden können.
- Akzeptieren der Auswirkungen, falls eine Umplanung möglich ist.
- Verschieben der Ausführung der betroffenen Aufgaben zu einem späteren Zeitpunkt und weitere Änderungen abwarten. So können in einigen Fällen weitere Ereignisse eintreten, die das Problem zu einem späteren Zeitpunkt lösbar machen.
- Weiterleiten der nicht bearbeitbaren Aufgaben an andere Agenten.
- Erzeugung neuer Unteraufgaben, die die durch die Modifikationen neu entstandenen Probleme lösen können, wodurch der Agent seine ursprünglichen Aufgaben wieder korrekt bearbeiten kann.

Bemerkung. Das Verzögern der Ausführung einer Aufgabe kann jedoch in einigen Fällen dazu führen, dass Aufgaben nicht bearbeitet werden und scheitern,

obwohl durch einen Aufgabentausch eine Lösung des Problems möglich gewesen wäre. Beispielsweise seinen zwei Agenten A und B Repräsentanten zweier LKWs. Agent A führt zwei Transportaufgaben T_1 und T_2 hintereinander aus. Agent B besitzt zur Zeit noch keine Aufgabe. Bevor A mit der Bearbeitung von T_1 beginnt, erhält er eine weitere Aufgabe T_3. Dieser neue Transport soll jedoch zum gleichen Zeitpunkt stattfinden wie T_2. T_3 führt zudem zu einem wesentlich höheren Gewinn als die Ausführung von T_2, jedoch ist es bei dieser Aufgabe T_3 unsicher, ob sie nicht kurz vor Beginn der Bearbeitung storniert wird. T_2 hingegen gilt als sicher. Agent A hat nun mehrere Möglichkeiten auf die Anfrage zu reagieren, ob er den Transport T_3 führen wird. Agent A muss sich zwischen der Aufgabe T_2 und T_3 entscheiden und eine Aufgabe annehmen und die andere ablehnen bzw. an einen anderen Agenten abtreten. Eine Alternative dazu ist: er behält beide Aufgaben, obwohl er in der aktuellen Situation an einer Aufgabe scheitern wird. Der Agent behält beide Aufgaben und wartet bis er mit der Bearbeitung einer der beiden Aufgaben anfangen muss. Unter Umständen scheitert so eine der Aufgaben, da sie auch nicht mehr den anderen Agenten übertragen werden kann. Jedoch besteht auch die Möglichkeit, dass in diesem Beispiel die Aufgabe T_3 vor Bearbeitungsbeginn wieder storniert wird. So kann Agent A problemlos Aufgabe T_2 ausführen. Hätte er sich direkt bei Aufgabenausschreibung entscheiden müssen, hätte er evtl. Aufgabe T_3 angenommen und T_2 abgetreten, was letztlich seinen Gewinn deutlich reduziert hätte. Das heißt, diese Strategie maximiert zwar den einzelnen Gewinn der Agenten, jedoch scheitern dabei Aufgaben, die durch rechtzeitige Ausschreibung an andere Agenten hätten ausgeführt werden können. Eine alternative Lösung wäre die Erzeugung einer Unteraufgabe T_3', für die Agent A der Besitzer ist und deren Ziel die Bearbeitung von T_3 ist. Agent A kann die Aufgabe an Agent B übergeben und gegebenenfalls vor Beginn entscheiden, ob er die Aufgabe dennoch selbst ausführt. Dabei wird der initiierende Agent seinerseits zu einem Holon (zur Verwaltung und Koordination seiner neu erzeugten Aufgaben) erweitert. Somit enthält das ursprüngliche Holon nicht nur Sub-Agenten, sondern auch Sub-Holone, was durch die rekursive Definition eines Holons möglich ist. Problem bei dieser Vorgehensweise ist die Regelung für die Auszahlung bzw. Entschädigung für den Profitausfall von Agent B.

Störanfällige Entitäten

In dynamischen Umgebungen können jederzeit Ereignisse auftreten, die Auswirkungen auf die Holonenbildung haben können. Einflüsse wirken sich dann auf eine der folgenden drei *Risikogruppen* aus:

- *Aufgaben:* Die Menge der Aufgaben, Ziele und damit verbunden, die Pläne der Agenten können sich jederzeit ändern. Störungen werden dabei bewirkt durch Veränderungen der Aufgabenbeschreibung, den Nutzen und die Kosten der Ausführung einer Aufgabe in Bezug auf die jeweilige Situation des Agenten. Dies erfordert von einem Agenten die Fähigkeit, umplanen zu können, so dass das Ziel trotz der Störungen weiterhin erfüllt werden kann. Beim Umplanen spielen Faktoren wie die Wiederverwendbarkeit der Pläne und der geforderte Grad der Vollständigkeit eine wichtige Rolle.

Echtzeitanforderungen bezüglich der Planung erschweren zusätzlich das Problem.

- *Agenten:* Agenten verlassen oder treten einer Agentengesellschaft bei, bzw. einige Agenten verbergen sich für einen unbestimmten Zeitraum gegenüber der Agentengesellschaft.
- *Informationen:* Informationen und Daten ändern sich, oder werden manipuliert, entweder gezielt durch den Absender der Nachricht oder durch Fehler während der Übertragung durch das Netzwerk. Fernen können sich die Anforderungen der Anwender fortwährend ändern. Ebenso können die Netzwerkverbindungen der Agenten unsicher sein.

Unter den beschriebenen Vorgaben einer dynamischen Umgebung besteht eine weitere Problematik der Holonenbildung darin, die Strukturen der Holone über die Zeit möglichst stabil zu halten. Zum einem bieten instabile Systeme kaum Planungssicherheit für die Anwender und zum anderen sinkt bei häufigem Umstrukturieren der Holone das Vertrauen und das Engagement gegenüber den Holonenpartnern. Dazu sind neue adaptive Mechanismen zur Bildung von Holonen mit unsicherem und beschränktem Wissen notwendig. Es gilt dabei die Balance zwischen Effizienz während der Holonenformierung und Qualität der gebildeten Holonen zu finden, um die Reaktionsgeschwindigkeit des Systems auf einem akzeptablen Niveau zu halten.

Handlungsfreiraum von autonomen, eigennützigen Agenten

Zur Entwicklung eines *Dynamic Holon Formation* (DHF) Algorithmus ist es notwendig neben der beschriebenen Umgebung und den Risikogruppen, auch den Handlungsfreiraum der eigenständig handelnden Agenten festzulegen. Annahmen über die Umwelt sind im Allgemeinen unvollständig, bzw. werden durch eintretende Ereignisse falsifiziert. Dies tritt besonders in Szenarien auf, in denen viele autonome eigenständige Agenten handeln und sich die Umgebung schnell ändert. Daher ist es von besonderer Bedeutung, dass die Agenten autonom und eigennützig agieren, jedoch auf der Basis von zuvor festgelegten Regeln, die beschreiben, welche Aktionen zulässig sind. Im Weiteren gehe ich von einem Multiagenten-System aus, bei dem die Agenten

- mit unsicheren und unvollständigen Informationen umgehen müssen
- die Möglichkeit besitzen, jederzeit ein Holon zu verlassen auch wenn sie ihre Aufgaben bezüglich dieses Holons noch nicht erfüllt haben
- zu jedem Zeitpunkt Verhandlungen abbrechen können
- möglichst auf zentrale Vermittlungsdiensten verzichten, um Manipulation von Nachrichten durch Dritte zu verhindern. Die Agenten kommunizieren dann vorzugsweise direkt miteinander.

Bemerkung. Dezentrale Koordination verursacht unter Umständen einen wesentlich höheren Kommunikationsaufwand, so dass diese Tatsache bei der Entwicklung eines Holonenformierungsalgorithmus berücksichtigt werden muss, um

die Kommunikationslast auf einem akzeptablen und performanten Maß zu halten.

6.3 Stabilität von dynamischen Kooperationen

Für die Bewertung von holonischen Multiagenten-Systemen in dynamischen Umgebungen ist es notwendig, den Begriff der Stabilität neu zu definieren. Eine Untersuchung zeigte, dass in vielen naturwissenschaftlichen Disziplinen der Begriff der *Stabilität* in ähnlicher Weise verwendet wird. Bei der Holonenbildung fehlt jedoch gänzlich eine solche Definition, wann ein System als stabil angesehen werden kann. In Anlehnung an interdisziplinären Definitionen dieses Begriffs (z.B. in der Elektrotechnik bei einem gedämpft schwingenden System, oder in der Politik) definiere ich die Stabilität von Holonen in diesem Kapitel anhand von unterschiedlichen Störfällen. Ein Vergleich der Definitionen aus verschiedenen Bereichen zeigt wiederkehrende Eigenschaften. Stabilität charakterisiert die Reaktion eines dynamischen Systems auf äußere Einflüsse (Stell- oder Störgrößen sowie Anfangswerte). So beschreibt beispielsweise die Stabilität in einem Ökosystems die Fähigkeit, dem Einfluss von Störungen zu widerstehen oder nach einer Störung wieder zum ursprünglichen Zustand zurückzukehren. Stabilität basiert dabei auf der Fähigkeit zur Selbstregulation. Beim Stabilitätsverhalten geht es um die Art und Weise wie das gesamte System auf äußere Einflüsse (Störungen) reagiert. Diese äußeren Einflüsse können in zwei Kategorien unterteilt werden:

1. Anfangsbedingungen oder Anfangsstörungen, die nur zum Zeitpunkt $t = 0$ auf das System einwirken
2. Eingangsgrößen, die kontinuierlich (also für $t > 0$) wirksam sind.

In dieser Betrachtungsweise eines dynamischen Systems als Übertragungsglied, das kontinuierlich Störungen erzeugt, ist es nicht möglich, die Kategorie der Anfangsstörungen zu berücksichtigen. Es ist deshalb naheliegend, die Reaktion eines dynamischen Systems auf Eingangsgrößen zur Stabilitätsdefinition heranzuziehen. Dabei ist diese Definition nicht nur für Regelkreise sondern auch für beliebige lineare zeitinvariante Übertragungssysteme formulierbar. Bevor der Begriff *Stabilität* definiert werden kann, müssen zunächst die Bedeutungen einiger Begriffe geklärt werden. Eine Störung eines Multiagenten-Systems beruht auf inneren und äußeren Störungen, wie sie in Abschnitt 7.1 näher beschrieben werden. Diese Einflüsse haben direkte Auswirkung auf die Ausführbarkeit der Pläne einzelner Agenten. Da in holonischen Multiagenten-Systemen die Pläne der Agenten im Allgemeinen voneinander abhängig sind, bewirkt eine Störung in dem Plan eines Agenten eine kaskadierende Störung in den Plänen einer Menge von Agenten, so dass die Ausführung einer Reihe von Aufgaben durch die aktuell existierenden Holonenstrukturen nicht mehr möglich ist. Als Folge müssen einige Holone umstrukturiert werden, um die korrekte Bearbeitung wieder sicherzustellen.

Beispiel 6.1. Seien zwei Agenten A Repräsentant eines LKW-Zugs und B einer LKW-Zugmaschine. Die Agenten haben sich zur Bearbeitung einer Aufgabe,

die sie beide nicht allein ausführen können, zu einem Holon zusammengeschlossen. Bei der Aufgabe handelt es sich um einen Transport von Hamburg nach München. Da beide Agenten sehr ausgelastet sind, wurde eine Übergabe des Anhängers in Köln vereinbart. So fährt A den Hänger von Hamburg nach Köln und übergibt ihn dort an Agent B, der ihn nach München weiterfährt. Das heißt die beiden Agenten haben eine starke Abhängigkeit zueinander in ihren Plänen. Erfährt nun einer dieser Agenten eine Störung in seinem Plan, die auch diese Transportaufgabe beeinflusst, so hat dies auch direkte Auswirkung (kaskadierender Effekt) auf den anderen Agenten. Beide müssen sich dann bei der Umplanung koordinieren, um einen anderen Treffpunkt bzw. Zeitpunkt für die Übergabe zu vereinbaren. Unter Umständen muss sogar das Holon umstrukturiert werden und zusätzliche Agenten aufgenommen werden.

Haben sich Holone gebildet, so dass alle gestellten Aufgaben zu bestmöglichen Bedingungen bearbeitet werden können, besteht für die Holonen keine Notwendigkeit zur Umstrukturierung mehr. Ein holonisches Multiagenten-System kehrt dann in einen *Ruhezustand* zurück. Im spieltheoretischen Sinne erreicht ein solches System ein Gleichgewicht. In diesem Zustand finden keine Umstrukturierungen der Holone mehr statt, und das System gilt wieder als stabil. Die Stabilität eines Systems wird durch seine Antwort auf eine Störung bestimmt: So verharrt ein stabiles Multiagenten-System solange in seinem Ruhezustand, bis es durch innere Störungen der Agenten selbst, oder von außen angeregt wird. Letzteres wird durch eine Änderung der Ausgangssituation (z.B. Modifikationen der Aufgabenstellung) oder Störung der Umgebung, z.B. Kommunikationsprobleme, ausgelöst. Sind alle Anregungen abgeklungen, kehrt das System in einen Ruhezustand zurück, in der alle Rekonfigurationen der Holone beendet sind[2]. Sonst führt bereits eine Störung zu ständigen Umstrukturierungen der Holone und das System kehrt nicht in einen Ruhezustand zurück. In Abhängigkeit der Störungsart werden im Folgenden verschiedene Stabilitätsbegriffe definiert.

6.3.1 Asymptotische Stabilität

Definition 6.1. Ein Multiagenten-System ist stabil, wenn es nach einer Störung mit wachsendem t gegen unendlich wieder in eine Ruhelage zurückkehrt (*asymptotische Stabilität*).

Beispiel 6.2. Abbildung 6.2 zeigt die Auswirkung einer Störung auf ein Multiagenten-System, das sich bis zum Zeitpunkt t' in einem Ruhezustand befindet (mit $u(t)$ der Anzahl von Rekonfigurationen der Holone pro Zeiteinheit). Zum Zeitpunkt t' tritt eine Störung auf, die einen sprunghaften Anstieg der Umstrukturierungen in der Agentengesellschaft bewirkt. Konvergiert die Anzahl der Umstrukturierungsmaßnahmen mit größer werdendem t wieder gegen Null, so ist das System stabil (siehe Definition 6.1). Kann hingegen kein t angegeben werden, zum dem sich das System wieder in einer Ruhelage befindet, so spricht man von einem *instabilen System*. Im ungünstigsten Fall findet eine

[2]Unter der Voraussetzung dass es einen stabilen Zustand gibt.

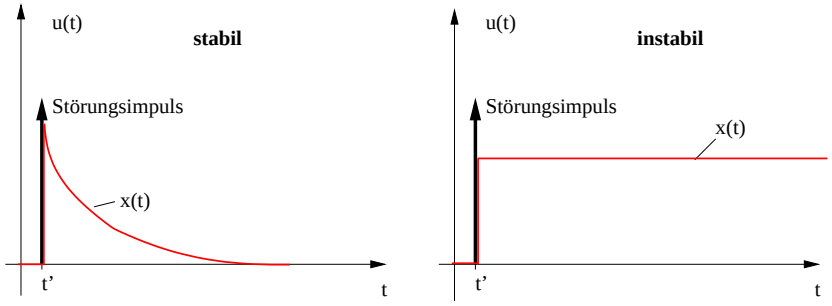

Abbildung 6.2: Asymptotische Stabilität als Reaktionen des Systems nach einer Störung

Resonanzkatastrophe statt, wobei die ursprüngliche Störung durch die Abhängigkeiten der Agenten untereinander weitere Störungen verursacht, so dass die Anzahl der Rekonfigurationen stetig zunimmt, bis schließlich kein Agent mehr in der Lage ist, überhaupt eine Aufgabe auszuführen, d.h. das System blockiert die Ausführung der Planung vollständig, bis von außen eingegriffen wird.

Alternativ kann man die Stabilität eines Systems an der Systemantwort auf eine begrenzte Menge von Störungen innerhalb eines Zeitintervalls untersuchen.

6.3.2 *Bounded Input Bounded Output* - Stabilität

Definition 6.2. Ein System ist stabil, wenn jede begrenzte Störung mit wachsendem t gegen unendlich zu einer begrenzten Änderung der Holonenstrukturen führt. Eine Änderung ist begrenzt, wenn nur eine beschränkte Anzahl der Agenten in ständigen Rekonfigurationen involviert sind, so dass ein ϵ mit 'Anzahl der restrukturierenden Agenten' $\leq \epsilon$ als obere Schranke angegeben werden kann. Diese Stabilitätsdefinition beschreibt die so genannte BIBO (*Bounded Input Bounded Output*)-Stabilität.

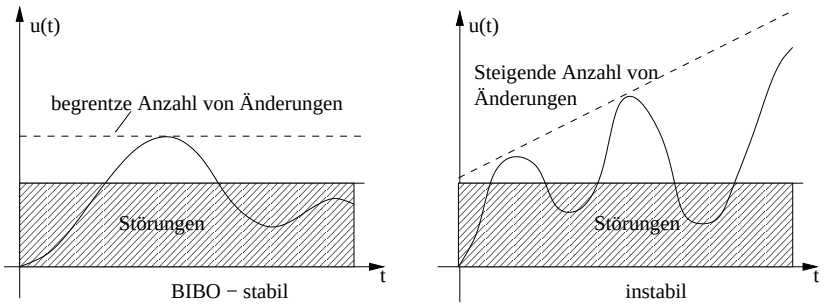

Abbildung 6.3: Sprungantworten stabiler und instabiler Systeme

Abbildung 6.3 zeigt zwei Fälle wie ein holonisches Multiagenten-System auf kontinuierliche Störungen (schraffierter Bereich) reagieren kann. Verhält sich

das System BIBO-stabil, entsprechend der Definition 6.2, so kann eine obere Schranke angegeben werden, die die maximale Anzahl von Rekonfigurationen der holonischen Strukturen pro Zeiteinheit nicht überschreitet. Kann keine solche Schranke angegeben werden, ist das System nicht stabil. Aus der Kombination der beiden Definitionen 6.1 und 6.2 folgen als Konsequenz zwei weitere Definitionen von Stabilität:

6.3.3 *Single Input Bounded Output* - Stabilität

Definition 6.3. Ein System ist stabil, wenn es nach *einer* Störung mit wachsendem t gegen unendlich zu einer begrenzten Änderung der Holonenstrukturen führt, so dass ein ϵ mit 'Anzahl der restrukturierenden Agenten' $\leq \epsilon$ als obere Schranke angegeben werden kann. Diese Stabilitätsdefinition wird als SIBO (*Single Input Bounded Output*)-Stabilität bezeichnet.

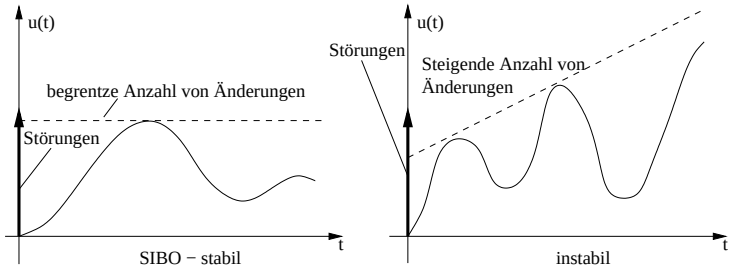

Abbildung 6.4: Bounded Output Verhalten nach einer einzigen Störung

Nach Auftreten einer einzigen Störung in einer Phase, in der das System sich in einem Ruhezustand befindet, kann sich das System asymptotisch stabil verhalten, oder instabil werden (siehe Beispiel 6.2). Betrachtet man jedoch ein System mit sehr vielen Abhängigkeiten zwischen einzelnen Planschritten der Agenten, so macht es dennoch Sinn von Stabilität zu sprechen, auch wenn unter Umständen eine Ruhelage nie erreicht wird. Analog der Definition 6.2 wird hier eine schwächere Form des Stabilitätsbegriffs verwendet. Ein System ist SIBO-stabil, wenn eine obere Schranke für die Anzahl der Umstrukturierungsversuche pro Zeiteinheit angegeben werden kann. Anschaulich bedeutet dies, dass ein Großteil der Pläne der Agenten stabil ist und nur ein geringer, vernachlässigbarer Teil einem steten Wandel unterliegt.

Bemerkung. Wird das System kontinuierlich gestört, ist es nicht möglich asymptotische Stabilität zu erreichen, da sich im Allgemeinen noch keine stabile Holonen gebildet haben, bevor die nächste Störung eintritt. Es existieren also folgende drei Definitionen der Stabilität in einem dynamischen, holonischen Multiagenten-System:

- Asymptotische Stabilität
- Bounded Input Bounded Output-Stabilität
- Single Input Bounded Output-Stabilität

Wie schnell ein solches System auf eine Störung reagiert und neue Holonenstrukturen bildet, hängt im Wesentlichen davon ab, wie gut das Wissen jedes Agenten über die Umwelt ist. Je genauer die Prognosen/Abschätzungen der Eigenschaften der Agenten sind, desto präziser können potentielle Agenten für die Bearbeitung einer Aufgabe ausgewählt werden und somit unnötige Verhandlungen vermieden werden. Insbesondere werden so weniger destabilisierende Umstrukturierungsversuche vorgenommen, die aufgrund falscher Einschätzungen durchgeführt, jedoch mit realistischen Annahmen nie begonnen würden.

6.4 Lernen in dynamischen Kooperationen

6.4.1 Prognose

Um Stabilität zu erreichen ist es notwendig, möglichst viele Störungsquellen im Voraus zu vermeiden, wie z.B. durch eine geeignete Wahl von Agenten zur Bildungen von Holonen. Durch eine Auswertung der gesammelten Informationen über Agenten und die Umwelt, kann so das wahrscheinliche Verhalten von Agenten bzw. das Auftreten von Ereignissen prognostiziert werden. Diese Prognosen dienen dann als Grundlage für die Planung der Aufgabenverteilung. Prognosen schätzen zukünftige Trends ab, indem das Verhalten der Teilnehmer unter gegebenen Bedingungen und Einbeziehung von vorherigen Erfahrungen abgeschätzt wird. Eine Prognose stellt eine hypothetische Aussage über zukünftige Ereignisse und die Annahme von zukünftigen Werten von Variablen dar, beruhend auf Beobachtungen aus der Vergangenheit und auf theoretisch fundierten objektiven Verfahren. Grundlage von Prognosen ist eine allgemeine Stabilitätshypothese, die besagt, dass gewisse Grundstrukturen in der Vergangenheit und Zukunft unverändert wirken. Gabler gibt in [Gab97] drei Definitionen von Prognoseverfahren an:

Definition 6.4 (Direkte/indirekte Prognose)**.** Direkte Prognose liegt vor, wenn Werte einer Variablen ausschließlich aus Werten derselben Variablen aus der Vergangenheit heraus prognostiziert werden. Bei indirekter Prognose wird der Wirkungszusammenhang zwischen verschiedenen Variablen in die Prognose einer Variablen eingebaut; hierbei muss allerdings letztlich wieder auf direkte Prognose zurückgegriffen werden.

Definition 6.5 (Qualitative/quantitative Prognose)**.** Bei qualitativer Prognose werden nur Art und Richtung der Entwicklung bestimmt, bei quantitativer Prognose auch das Ausmaß.

Definition 6.6 (Indikator-Prognose)**.** Indikatoren werden zur Prognose von Entwicklungen herangezogen. Indikatoren können, müssen aber nicht, in kausaler Beziehung zu der zu prognostizierenden Variable stehen. Sie lassen sich unterteilen in vorauseilende, koindizierende und nacheilende Indikatoren.

Prognoseverfahren

Die Wahl eines Prognoseverfahrens hängt stark von dem Zeitraum ab, für die die Prognose Gültigkeit besitzen soll. So wird z.B. für kurzfristige Prognosen

(Zeitraum ein Jahr) in erster Linie das direkte Verfahren angewendet, vor allem Zeitreihen-Prognose mittels gleitender Durchschnitte oder exponentiellem Glätten. Für die mittelfristige Prognose findet die Methode der kleinsten Quadrate zur Fortrechnung des Trends oder auch, etwa bei Marktprognosen, die Prognose mittels Wachstumsfunktionen Anwendung. Beim Vorhandensein von saisonalen Komponenten erfolgt die Prognose des Trends auf der Grundlage von Vergangenheitswerten, die einer Trendbereinigung unterworfen werden. Für die Prognose des Zukunftswertes wird die Saisonkomponente auf eine geeignete Art hinzugerechnet. Indirekte Prognosen erfolgen meist mit Hilfe der Regressionsanalyse. Grundsätzlich unterschieden werden:

- *Quantitative Prognoseverfahren:* diese basieren auf mathematischen Verfahren (z.B. Trend, Indikatorprognose, exponentielles Glätten)

- *Qualitative Prognoseverfahren:* Basieren auf Erfahrung, Kenntnissen und Fingerspitzengefühl; angewandt beim Fehlen quantitativer Daten (z.B. Expertenbefragung, Szenario-Technik)

Die Prognose, wie sie zuvor beschrieben wurde, ist kein echter Planungs- oder Entscheidungsprozess, da sie 'nur' versucht, die Zukunft so genau wie möglich vorauszusagen. Sie betrachtet nicht die eigene Einflussnahme. Soweit erscheint die Prognose als ein einfaches und zuverlässiges Werkzeug, das exakte Zahlen für die Planung liefert. Jedoch können Prognosen aufgrund der komplexen Abhängigkeiten und Störeinflüssen auf die Umwelt keine exakten Vorhersagen für die Zukunft liefern. Nahmias [Nah97] formulierte daher folgende These: Prognosen haben eines gemeinsam: *sie sind normalerweise falsch.* Die einzelnen Prognoseverfahren und die ihnen zugrunde liegenden Prognosemodelle unterscheiden sich vor allem darin, wie falsch eine Prognose ist. Daher enthält jeder Planungsschritt, der auf solche Prognosen basiert, ein gewisses Maß an Unsicherheit. Die Differenz zwischen den prognostizierten Werten und den tatsächlichen Zahlen beeinflusst hier z.B. die Qualität der zu bildenden Holonen. Da es sich nur um Schätzungswerte handelt, muss während der Planung mit Toleranzen gerechnet werden. Doch wenn viele Agenten Informationen zur Berechnung der Prognose liefern, kann es zu polarisierenden Meinungen kommen, die aufgrund der großen Menge von Bewertungen sich gegenseitig aufheben. Im Folgenden werden die Charakteristiken und Verfahren der meist verwendeten Prognosemethoden beschrieben. Der erste Teil stellt die Prognosetechniken vor, die für Zeitreihenmodelle verwendet werden.

Statistische Prognose-Techniken

Prognose-Methoden wurden seit den 50er Jahren des 20. Jahrhunderts für Geschäftsprognosen und gleichzeitig für ökonomische Zwecke (z. B. Arbeitslosenzahlen etc.) entwickelt. Jede dieser Methoden versucht, die vergangenen Kennzahlen in die Prognose für zukünftige Zahlen einzubringen. Es existieren zwei prinzipielle Verfahren, die auf unterschiedlichen Grundlagen basieren: Die *Zeitreihenanalyse* und *Kausalmodelle*. Die so genannte Zeitreihenanalyse geht

davon aus, dass die Bewertungen einem spezifischen Muster folgen. Daher besteht die Aufgabe einer Prognose-Methode darin, dieses Muster durch vergangene Beobachtungen zu schätzen. Die Prognosen können dann durch dieses geschätzte Muster errechnet werden. Der Vorteil dieser Methoden ist, dass sie nur vergangene Beobachtungen der Nachfrage benötigen. Das zweite Verfahren zur statistischen Prognose sind Kausalmodelle. Sie gehen davon aus, dass die Nachfrage durch einige bekannte Faktoren bestimmt wird. Zum Beispiel hängt die Nachfrage nach Eis von der Temperatur eines bestimmten Tages ab. Daher ist die Temperatur ein so genannter Hauptindikator für die Nachfrage nach Eis. Wenn genügend Beobachtungen der Nachfrage und der Temperatur vorhanden sind, kann das zugrunde liegende Modell geschätzt werden.

Gleitender Durchschnitt und Glättungsmethoden

Um zufällige Störterme in den formalen Lösungsansätzen zu vermeiden, kommt den genauen Schätzungen der Parameter eine entscheidende Bedeutung zu. Zusätzlich können sich die Parameter über die Zeit ändern. Daher müssen gegenwärtige und vergangene Beobachtungen geeignet zusammengeführt werden. Je größer der Umfang der Zeitreihe, desto besser gelingt es, zufällige Schwankungen zu eliminieren.

Einfacher gleitender Durchschnitt Der einfache gleitende Durchschnitt wird im Allgemeinem für Produkte verwendet, die eine gleich bleibende Nachfrage vorweisen. Die Parameter-Schätzung für die Höhe der konstanten Nachfrage $\hat{a}$ wird berechnet durch Bildung des Durchschnitts der letzten n Beobachtungen. Dieser Parameter dient als Prognose für alle zukünftigen Perioden, da die Prognose $\hat{x}_t$ unabhängig von der Zeit ist. Die Genauigkeit der Prognose wird mit der Länge n der betrachteten Zeitreihe zunehmen, da die zufälligen Abweichungen weniger Gewicht erhalten. Falls sich die Umgebung bzw. das Nachfrageobjekt ändert, erstellt diese Methode keine brauchbaren Ergebnisse mehr. Außerdem gehen die Informationen aus vergangenen Beobachtungen bei dieser Prozedur verloren.

Exponentielles Glätten Dieser Informationsverlust wird bei Verwendung des exponentiellen Glättens vermieden, da es allen beobachteten Daten unterschiedliche Gewichte zuweist und sie in die Prognose aufnimmt. Das Gewicht für die Beobachtungen nimmt in Richtung Vergangenheit exponentiell ab. Daher können sich die Prognosewerte eventuellen Veränderungen des Nachfragemusters besser anpassen. Für den Fall der gleich bleibenden Nachfrage x_t wird die Prognose für die Periode $t+1$ gemäß der folgenden Formel exponentielle Glättung erster Ordnung berechnet. Der Parameter α ist ein Glättungsparameter, der mit Werten von 0 bis 1 belegt werden kann:

$$\hat{x}_{t+1}^{(1)} = \alpha \cdot x_t + \alpha \cdot (1-\alpha) \cdot x_{t-1} + \alpha \cdot (1-\alpha)^2 \cdot x_{t-2} + ... =$$

$$= \sum_{i=0}^{t} \alpha \cdot (1-\alpha)^i \cdot x_{t-i}$$

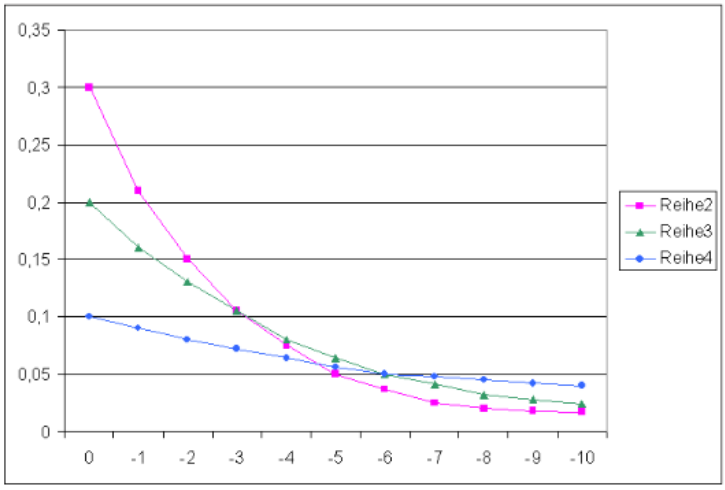

Abbildung 6.5: Exponentielle Abnahme der Gewichtung bei unterschiedliche Parametern

Tempelmeier [Tem99] empfiehlt, Werte zwischen 0.1 und 0.3 zu verwenden (siehe Abbildung 6.5). Je größer α ist, umso stärker ist der Einfluss des neuesten Beobachtungswertes der Nachfragezeitreihe auf den Prognosewert. Die exponentielle Glättung ist allerdings nur dann ein geeignetes Verfahren zur kurzfristigen Nachfrageprognose, wenn die Beobachtungswerte um einen im Zeitablauf konstanten Mittelwert schwanken. Für $\alpha = 0.2$ werden die Gewichte in Tabelle 6.5 benutzt, wenn die Prognose für Periode 1 erstellt werden soll.

Periode	0	-1	-2	-3	-4	...
Gewicht	0.2	0.16	0.13	0.10	0.08	...

Tabelle 6.5: Gewicht einer Beobachtung

Darüber hinaus kann diese Formel vereinfacht werden. In der Datenbank muss so nur die letzte Prognose des letzten Nachfragewerts gespeichert werden.

$$\hat{x}_{t+1} = \hat{y}_t^{(1)} = \alpha \cdot y_t + (1 - \alpha) \cdot y_{t-1}^{(1)}$$

Der Prognosewert $\hat{x}_{t+1}$ für die Periode $t+1$ entspricht der geglätteten Nachfrage erster Ordnung $y_t^{(1)}$ in t. Er wird aus der tatsächlichen Beobachtung y_t in Periode t und der geglätteten Nachfrage erster Ordnung $y_{t-1}^{(1)}$ in Periode $t-1$ (entspricht wiederum der Prognose für Periode t) berechnet. Prognosen bieten auf Basis von gesammelten Erfahrungen eine Möglichkeit das Verhalten von Teilnehmern in der Zukunft vorherzusagen. Das Ziel, die Zukunft auch nur ansatzweise berechen zu können, findet sich in vielen Forschungsrichtungen wieder. Wie zuvor dargestellt stellen Prognosen dazu eine Sammlung von Verfahren zur Verfügung. Analog existieren stochastische Verfahren, um das Eintreten von Ereignissen in der Zukunft, unter Umständen in Abhängigkeit von anderen zuvor

eingetretenen Ereignissen, abzuschätzen. Nachfolgender Abschnitt zeigt alternativ zu den Prognosen, wie mit Hilfe von stochastischen Verfahren gesammelte Informationen ausgewertet werden können.

6.4.2 Stochastische Bewertung

Neben den statistischen Verfahren zur Erstellung von Prognosen über das Verhalten und die Eigenschaften von Agenten, ist es auch möglich, die Bewertungen der Agenten mit stochastischen Methoden abzuschätzen. Im Kapitel 5 habe ich gezeigt, wie Agenten durch Vektoren dargestellt und Agenten zu einem Holon auf Basis der Vektorrechnung zusammengefasst werden können. Es wurde gezeigt, wie aus einer Auswahl von Agenten mit Hilfe einer Norm diejenigen bestimmt werden können, die am besten geeignet sind, eine gegebene Aufgabe zu lösen. Diese Berechnungen basieren jedoch auf der Voraussetzung, dass vollständige Informationen über die relevanten Agenten vorliegen. Im diesem Idealfall kennt ein Koordinator die speziellen Ausprägungen der Attribute jedes einzelnen Agenten. In der Anwendungsdomäne dieser Arbeit ist diese Voraussetzung jedoch nicht gegeben. Die Umwelt ist im ständigen Wandel und die Agenten sind nicht notwendigerweise benevolent, so dass die Ausprägungen der einzelnen Agentenvektoren einem Koordinator oder Kopf eines Holons nicht direkt zugänglich sind. Die Agenten und Holone müssen also mit unvollständigem Wissen umgehen können und sollen zu akzeptablen Lösungen in einer annehmbaren Zeitspanne gelangen, wie z.B. bei der Bildung von stabilen Holonen. Um die Schwankungen in den Informationen über eine Entität der Domäne zu relativieren und die Erkenntnisse der Vergangenheit auch zu berücksichtigen, müssen lernende Verfahren zum Einsatz kommen. Ziel hierbei ist es, so genannte Ausreißer aus der Menge der eingehenden Informationen herauszufiltern. Informationen, die den mehrheitlichen Informationen entsprechen, können dabei stärker gewichtet werden. Der Mittelwert wäre hierbei ungeeignet, da eine Information mit großer Abweichung vom Mittelwert, diesen zu stark beeinflussen würde. Dies ist insbesondere bei einer geringen Anzahl von eingegangen Informationen der Fall.

Gesucht wird also eine Methode, die bei der Bestimmung des Durchschnittwerts berücksichtigt, wie groß die Streuung der eingegangenen Informationswerte vom Erwartungswert ist. Es soll so der Trend ermittelt werden, dem die eingehenden Informationen folgen. Je häufiger ein so genannter *Ausreißer* auftritt, desto stärker sollen die vorausgegangenen ähnlichen Ausreißer im Nachhinein bewertet werden, so dass schließlich der erwartete Wert sich dem Wert der Ausreißer annähert. Ein Maß dafür ist die Varianz bzw. die Standardabweichung, mit deren Hilfe eine geeignete Gewichtung der Daten erfolgen kann. In Anwendungsszenarien, in denen keine vollständigen und sichere Informationen den Agenten vorliegen, müssen die Agenten (und Holone) so Informationen eigenständig sammeln und bewerten. Dieses Wissen stellt dann die Basis für weitere Verhandlungen und Auswahl von Kooperationspartnern dar. Das heißt, je genauer die Agenten die Realität erfassen und vorherberechnen können, desto erfolgreicher können die Aufgaben gelöst werden. Im Folgenden gebe ich ein Verfahren an, wie mit Hilfe von stochastischen Methoden die Eigenschaften

von Agenten anhand einer Trainingsmenge erlernt werden können. In Anhang B werden dazu die stochastischen Grundlagen [3] beschrieben.

Lernen durch Agentenbewertungen

Wie am Anfang dieses Kapitels beschrieben, ist es für die Bildung von Holonen notwendig, dass die Agenten für die anstehenden Verhandlungen Wissen über die anderen Agenten der Agentengesellschaft besitzen. Zum einen ist dieses Wissen für die Auswahl relevanter Agenten notwendig, zum anderen dient es der Abschätzung, welche Auswirkungen die Hinzunahme eines weiteren Agenten zu einem Holon hat. Entsprechend den Voraussetzungen einer dynamischen Umgebung, besitzen die Agenten kein vollständiges Wissen über die Welt, sondern müssen die Eigenschaften und Fähigkeiten der Entitäten mit der Zeit „lernen". Mitchell [Mit97] definiert maschinelles Lernen als:

Definition 6.7. Einem Computer-Programm wird die Fähigkeit *Lernen* zugesprochen, wenn aufgrund einer Erfahrung E, bezüglich einer Menge von Aufgaben T und einem Performance-Maß P, die Performance der Aufgaben in T (bestimmt durch P) mit der neuen Erfahrung E steigt.

Den Aufgaben T entsprechen hier die Menge der Attribute, die bezüglich einer Aufgabenklasse untersucht werden sollen. Die Erfahrung E macht ein Agent sobald er eine neue Bewertung über einen anderen Agenten erhält. Das Performance-Maß P steht für den Prozentsatz der erfolgreichen Verhandlungen, die indirekt als Maß für die Güte der Bewertungen angesehen werden können. Denn nur durch eine realitätsnahe interne Repräsentation der Agenten, können die Agenten ausgewählt werden, die sehr wahrscheinlich die Aufgaben zu den geforderten Bedingungen erfüllen können, was sich in erfolgreichen Verhandlungen niederschlägt. Den Agenten stehen dazu verschiedene Informationsquellen zur Verfügung, wie z.B. Verhandlungsergebnisse, angebotene Dienste und zentrale Agenteninformationsdienste. Jedoch kann nicht davon ausgegangen werden, dass eine dieser Informationsquellen vollständige und korrekte Daten zur Verfügung stellt. Ein Agent muss daher möglichst viele Daten von verschiedenen Quellen sammeln und nach eigenen Kriterien gewichtet bewerten. Ziel eines jeden Agenten ist es, eine interne Repräsentation der anderen Agenten aufzubauen. Als Ergebnis einer Auswertung der gesammelten Daten für jeden bekannten Agenten liefert diese Wissensbasis einen beschreibenden Vektor. Im Folgenden werden zwei Ansätze zur Realisierung einer solchen Wissensbasis diskutiert, die im weiteren Verlauf dieser Arbeit zum Einsatz kommen.

1. Berechnung der Erwartungswerte bezüglich der Attribute eines Agenten, oder

2. Erstellen von *Statistiken*, im Speziellen: Berechnen des arithmetischen Mittels mit einer Gewichtung der gesammelten Daten in Abhängigkeit ihrer Standardabweichung und Aufnahmezeitpunkt.

[3] zur Bestimmung eines gewichteten Mittels in Abhängigkeit der Varianz der eingegangen Informationen und deren Alter.

Bemerkung. Beide Ansätze der Modellierung einer Wissensbasis haben den Nachteil, dass bei einem geringen Informationsfluss oder bei dynamischen Veränderungen der Eigenschaften eines Agenten, die Abweichungen der eigenen Bewertungen von den realen Werten unter Umständen sehr groß sein können.

Definition 6.8. Sei A ein Agent in einem n dimensionalen Vektorraum, über den Daten gesammelt wurden und mit den nachfolgenden Verfahren ausgewertet werden. Im Folgenden wird stellvertretend das Attribut a_i aus der Menge aller Attribute eines Agentenvektors $A = \{a_1, ..., a_n\}$ verwendet. Die Mächtigkeit der Agentengesellschaft wird durch die Variable m beschrieben, d.h. maximal m Agenten können pro Zeitintervall eine Bewertung bezüglich des Agenten A abgeben. So werden pro informierenden Agent je ein Bewertungsvektor bezüglich A zum Zeitpunkt t gesammelt. $A_j(t)$ gibt dabei den Beschreibungsvektor an, abgegeben von Agent j zum Zeitpunkt t an. Da nicht zu jedem Zeitpunkt jeder Agent eine Bewertung abgeben muss, gilt: $0 \leq j \leq m$ (da ein Agent auch eine Bewertung über sich selbst absenden kann) mit k der Anzahl der abgegebenen Bewertungen pro Zeitintervall t. Eine Bewertung des j-ten Agenten wird geschrieben als: $A_j(t) = \begin{pmatrix} a_{1j}(t) \\ \vdots \\ a_{ij}(t) \\ \vdots \\ a_{nj}(t) \end{pmatrix}$

Bevor weitere Berechnungen zur Bestimmung des Erwartungswerts eines Agentenvektors oder eines gewichteten Mittels der Attribute des zu untersuchenden Agenten durchgeführt werden können, wird zunächst aus den k gesammelten Bewertungen $A_j(t)$ eines Zeitintervalls t ein gemittelter Agentenvektor $\hat{A}(t) = \begin{pmatrix} \hat{a}_1(t) \\ \vdots \\ \hat{a}_i(t) \\ \vdots \\ \hat{a}_n(t) \end{pmatrix}$ berechnet.

Definition 6.9. $\hat{A}(t)$ heißt *gemittelter Agentenvektor*, mit

$$\hat{a}_i(t) = \tfrac{1}{k} \sum_{j=1}^{k} a_{ij}(t).$$

Dieser Vektor $\hat{A}(t)$ beschreibt das Meinungsbild von k Agenten der Agentengesellschaft zum Zeitpunkt t. Dabei ist auch von Interesse, wie groß die Streuung der Bewertungen von einem solchen "Mittelwertvektor" sind. So existiert zu $\hat{A}(t)$ ein so genannter *Bewertungsvektor* $V(t) = \begin{pmatrix} v_1(t) \\ \vdots \\ v_i(t) \\ \vdots \\ v_n(t) \end{pmatrix}$ mit:

Definition 6.10.

$$v_i(t) = \frac{1}{k} \sum_{j=1}^{k} (\hat{a}_i(t) - a_{ij}(t))^2$$

$V(t)$ beschreibt die Varianz des Bewertungsvektors $\hat{A}(t)$.

Erwartungswertmethode

Der erste Ansatz baut auf den Annahmen von Satz B.3 auf, indem sehr häufig Informationen über einen Agenten gesammelt werden. So kann über einen größeren Zeitraum betrachtet, eine zuverlässige Aussage über die Attributwerte des Agenten getroffen werden. Die Wissensbasis dient so zum Aufbau einer Verteilungsfunktion, die als Grundlage für die Berechung des Erwartungswerts eingesetzt wird. So ist es möglich mit

$$E(\hat{a}_i) = \sum_{t=1}^{k} \hat{a}_i(t) \cdot h(\hat{a}_i(t))$$

die Attributwerte eines Agenten zu bestimmen. Dieser Ansatz beinhaltet jedoch eine Schwierigkeit: Da es sich bei den Wertebereichen der Attribute generell um die Menge der reellen Zahlen handelt ($\hat{a}_i \in \mathbb{R}$), ist es unmöglich, eine Verteilungsfunktion $h(\hat{a}_i)$ für alle möglichen $\hat{a}_i$ anzugeben. Eine mögliche Lösung ist es den Wertebereich in Intervalle aufzuteilen, z.B. in zwei Intervalle $\hat{a}_i > 5$ und $\hat{a}_i \leq 5$. So kann die Komplexität deutlich verringert werden. Durch eine Abbildung der Attributwerte auf Intervalle mit einer Funktion $f : \hat{a}_i \mapsto x_i$, deren Wertebereich der Zufallsvariablen X entspricht, kann für jedes $x_i \in X$ ein Erwartungswert bestimmt werden:

$$E(X) = \sum_{i=1}^{n} x_i \cdot P(X = x_i)$$

Die Genauigkeit und Komplexität hängt dann direkt von der Granularität der Intervalle ab. So führt eine höhere Genauigkeit (kleinere Intervalle) zwangsläufig zu einer höheren Komplexität, beispielsweise zu einem linearen Anstiegt wie in Abbildung 6.6 dargestellt.

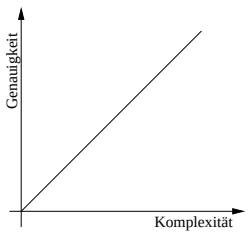

Abbildung 6.6: Verhältnis zwischen Genauigkeit und Datenkomplexität

Gewichtetes Mittel mit Glättung

Der zweite Ansatz versucht nicht die Wahrscheinlichkeit zu bestimmen, mit der ein Attribut eines Agenten einen bestimmten Wert annimmt, sondern führt eine Statistik der gesammelten Daten und berechnet hierauf das arithmetische Mittel. Da die Vergleichbarkeit zweier solcher Werte unter Umständen nicht möglich ist (siehe Beispiel B.3), muss zusätzlich ein Maß für die Kontinuität der gesammelten Daten angegeben werden. Die Varianz ist hierfür ein geeignetes Maß. Der letzte Schritt des zweiten Ansatzes führt eine Glättung der in Definition 6.9 bestimmten Attributwerte für ein gegebenes Zeitinterwal $[t_1; t_r]$ durch. Da aufgrund der dynamischen Umgebung Informationen schnell veralten können, muss dies entsprechend in der Glättung berücksichtigt werden. Es wird daher davon ausgegangen, dass mit zunehmendem Alter der Informationen deren Wichtigkeit sinkt und neue Informationen besser die aktuellen Werte der Agenten beschreiben. Dies führt zu einer Bewertung der Information in Abhängigkeit der Zeit mit Hilfe einer exponential Funktion $e^{-(t_r - t_l)}$. Je älter die Informationen sind, desto geringer ist ihr Gewicht bei der Ermittlung des zu schätzenden Attributwertes. In einem zweiten Schritt wird die Varianz jedes Attributs bei der Ermittlung dessen Mittelwerts durch einen zweiten Faktor $e^{-v_i(t_l)}$ berücksichtigt. Der Zeitindex fließt als Differenz zur aktuellen Zeit, wie auch die zugehörige Varianz in die Glättung mit ein. $\tilde{A} = (\tilde{a}_1; ...; \tilde{a}_i; ...; \tilde{a}_n)$ beschreibt den Agent A zum Zeitpunkt t_r und wird folgendermaßen für alle $1 \leq i \leq n$ berechnet:

$$\tilde{a}_i = \frac{\sum_{l=1}^{r} \hat{a}_i(t_l) \cdot e^{-(t_r - t_l)} \cdot e^{-v_i(t_l)}}{\sum_{l=1}^{r} e^{-(t_r - t_l)} \cdot e^{-v_i(t_l)}}$$

Das Resultat $\tilde{a}_i$ beschreibt nach der Analyse der gesammelten Daten einen zeitlich gewichteten, über die Menge der Daten pro Zeiteinheit und pro Agent gleichmäßig gewerteten Attributwert. Mit steigender Anzahl von Agenten, die Informationen senden, sinkt die Einflussnahme eines einzelnen Agenten, so dass so genannte *Ausreißer* schwächer berücksichtigt werden. Alte Informationen, deren Relevanz im Allgemeinen in dynamischen Umgebungen mit fortschreitender Zeit abnimmt (vgl. Abschnitt 6.4.1), werden durch den Faktor $e^{-\triangle t}$ entsprechend abgeschwächt und nähern sich dem Wert 0. Betrachtet man den ursprünglichen Bewertungsvektor, so wird dieser nach einigen Zeitintervallen näherungsweise auf den Nullvektor abgebildet. Somit verschwindet diese Information aus der Summe des resultierenden Vektors $\tilde{A}$, der das Ergebnis der Analyse ist. Der Faktor $e^{-v_i(t_l)}$ berücksichtigt die Varianz der eingegangenen Daten. So werden Informationen mit einer hohen Streuung weniger stark gewichtet, wie eine Datenserie, deren Varianz gering ist. D.h. gelegentliche Ausreißer fließen nur zu einem sehr geringen Anteil in das gewichtete Mittel von $\tilde{a}_i$ ein. Um die Menge der gesammelten Daten zu begrenzen, brauchen lediglich die Datensätze gespeichert zu werden, für die gilt: $e^{-\triangle t} > \epsilon$, denn alle Datensätze mit $e^{-\triangle t} < \epsilon$ werden so gering gewichtet, dass sie kaum noch Einfluss auf das Endergebnis haben und somit vernachlässigt werden können.

Bemerkung. Der Vektor $\tilde{A}$ repräsentiert eine Abbildung des Agenten A in der Wissensbasis und beschreibt eine individuelle Einschätzung dieses Agenten durch den bewertenden Agenten auf Grundlage der gesammelten Informationen

und Fremdbewertungen. Diese Bewertung ist zeitunabhängig, weshalb sie ohne Anpassung für jede nachfolgende Anfrage verwendet werden kann.

Bemerkung. Um die die zeitliche Abschwächung der Daten und den Einfluss der Varianz gezielt steuern zu können, werden zusätzlich noch zwei weitere Variablen w_1 und w_2 eingeführt. Sie bieten die Möglichkeit eine individuelle Gewichtung dieser beiden genannten Einflüsse durchzuführen. Somit ergibt sich folgen Formel für die Berechnung der Glättung eines Attributs $\tilde{a}_i$:

$$\tilde{a}_i = \frac{\sum_{l=1}^{r} \hat{a}_i(t_l) \cdot e^{-(t_r - t_l)\cdot w_1} \cdot e^{-v_i(t_l)\cdot w_2}}{\sum_{l=1}^{r} e^{-(t_r - t_l)\cdot w_1} \cdot e^{-v_i(t_l)\cdot w_2}}$$

6.4.3 Support Vector Machine

In den letzten Jahren entwickelte sich die Forschung im Bereich der Kernel basierten Lernverfahren sehr schnell, wie zum Beispiel *Support Vector Machines* (SVM) [BGV92], [CV95b], [SBS99], [SPS+01]. Diese Forschungsarbeiten haben nicht nur praktische Relevanz für Klassifikations- und Regressionsprobleme gezeigt, sondern auch in unbeaufsichtigtem Lernen (*unsupervised lerning*) [SSM99], [SMB+99]. Erfolgreiche Anwendungen fanden Kernel-basierte Algorithmen, z.B. in den Bereichen der optischen Mustererkennung, Objekterkennung, Texterkennung, Vorhersage von Zeitserien, Genanalyse und weitere. All diesen Anwendung ist eines gemein, der Mustervergleich eines Zielobjekts mit einer Menge von Testobjekten. SVM beschreibt ein spezielles Verfahren zur Einteilung (*clustering*) einer Menge von Objekten in verschiedene Klassen. Die Objekte werden dabei durch Vektoren beschrieben und miteinander verglichen. Erst durch den Nachweis der Vektorraumeigenschaft eines Multiagenten-Systems können holonische Agenten (also Gruppen von Agenten) direkt mit einzelnen Agenten anhand ihrer Vektoren verglichen und klassifiziert werden. Die Annahme der Existenz nichtbenevolenter Agenten führt dazu, dass den Klassifikationen durch die Agenten selbst nicht vertraut werden kann. Die Informationen über die bekannten Agenten müssen daher eigenständig durch die Agenten überprüft werden, um den geeignetsten Agenten für die Bearbeitung einer Aufgabe zu finden. Es bietet sich der Einsatz von SVM-Algorithmen an, um effizient eine Vorauswahl von Agenten zu treffen.

Durch den Einsatz von SVMs kann die Agentengesellschaft entsprechend bestimmter Aufgabentypen klassifiziert werden. Zur Definition der Aufgabentypen müssen charakteristische Eigenschaften zuvor festgelegt werden. Diese Charakterisierung ist stark von der Anwendungsdomäne abhängig und muss individuell von Fall zu Fall festgelegt werden. Zur Beschreibung eines Aufgabentyps genügt es eine Untermenge der Aufgabenattribute zu betrachten, für die Werteintervalle festgelegt werden. Die Ausprägungen der restlichen Aufgabenattribute werden nicht berücksichtigt. Basierend auf dem erlernten Wissen eines Agenten wird mit Hilfe eines SVM-Algorithmus für jeden Aufgabentyp eine Hyperebene bestimmt, beschrieben durch einen Normalenvektor w und Bias b. Diese Hyperebene teilt die Agentengesellschaft in zwei Gruppen auf, die die Anforderungen des Aufgabentyps erfüllen und in die, die diese nicht erfüllen. Erst nachdem eine Agentenklasse bestimmt wurde, müssen die Mitglieder

dieser Klasse hinsichtlich der Aufgabe genau analysiert werden. Durch diese Vorgehensweise reduziert sich der Analyseaufwand, indem nur noch eine Teilmenge der Agenten betrachtet werden muss, was bei geeigneter Aufgabentypisierung zu einem deutlichen Performanzgewinn führen kann.

SVM Verfahren

Das SVM Verfahren, das hier vorgestellt wird, basiert auf den Arbeiten von [MMR+01], [SGB+02], [SVMb] und beschreibt eine Standardvorgehensweise, wie sie auch in weiteren Quellen zu finden ist. SVM eignen sich besonders, um linear separierbare Kategorisierungsprobleme zu lösen. Gegeben ist eine endliche Trainingsmenge (Lernaufgabe)

$$\mathcal{L} = \{(\vec{x}_m, d_m) | m = 1, ..., M\}$$

mit Trainingsvektor $\vec{x}_m \in \mathbb{R}^n$, dem zugehörigen Kennzeichner des Trainingsergebnisses $d_m \in \{+1, -1\}$ und M der Anzahl von Testvektoren, welche durch eine Hyperebene $H = H(\vec{w}, b)$ der Form

$$\vec{w}^T \vec{x}_m + b = 0$$

klassifiziert werden. Im 2-dimensionalen Beispiel in Abbildung 6.7 wird dies durch eine Gerade symbolisiert. $\vec{w}$ stellt dabei einen Gewichtsvektor dar. Eine *optimale Hyperebene* für eine Trainingsmenge $\mathcal{L}$ ist eine $\mathcal{L}$ trennende Hyperebene $H^* = H(w^*, b^*)$ mit maximalem Wert $u_{\mathcal{L}}(w^*, b^*)$. Für eine optimale Hyperebene H für $\mathcal{L}$ müssen also *positive Instanzen* $\vec{x}_m$ mit $\vec{w}^T \vec{x}_m + b = +1$ und *negative Instanzen* $\vec{x}_m$ mit $\vec{w}^T \vec{x}_m + b = -1$ existieren. Solche Instanzen mit größter Nähe zu H heißen *Support-Vektoren* (von H bezüglich $\mathcal{L}$). Unter dieser Voraussetzung bildet sich so eine möglichst breite Grenze $u_{\mathcal{L}}$.

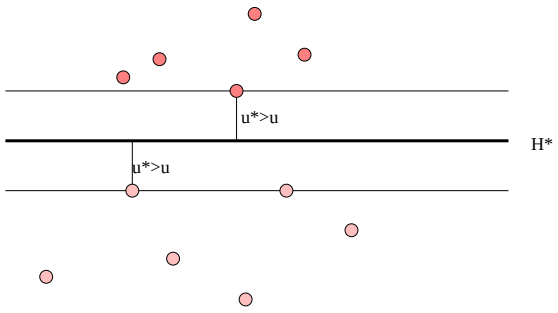

Abbildung 6.7: Optimale Hyperebene H^* für eine Trainingsmenge $\mathcal{L}$

Durch Formulierung des folgenden Optimierungsproblems kann eine optimale Hyperebene für $\mathcal{L}$ gefunden werden: Finde $\vec{w} \in \mathbb{R}^n$ und $b \in \mathbb{R}$ mit minimalem Wert $\frac{1}{2}\vec{w}^T\vec{w}$ unter den Randbedingungen

$$\vec{w}^T \vec{x}_m + b = \begin{cases} \geq +1 \text{ falls } d_m = +1 \\ \leq -1 \text{ falls } d_m = -1 \end{cases}$$

Minimalität von $\frac{1}{2}\vec{w}^T\vec{w}$ entspricht der Maximalität von $\frac{2}{||\vec{w}||}$, also der Trennbreite. Die zu minimierende Funktion $\frac{1}{2}\vec{w}^T\vec{w}$ ist eine streng konvexe Funktion von $\mathbb{R}^n$ in $\mathbb{R}$. Die Randbedingungen des Problems sind lineare Randbedingungen, die für $m = 1, ..., M$ einheitlich zusammengefasst werden kann:

$$d_m \cdot (\vec{w}^T\vec{x} + b) - 1 \geq 0$$

Das Optimierungsproblem hat ein eindeutiges Minimum w, da die zu minimierende Funktion und die Randbedingungen konvex sind (ist jedoch numerisch kompliziert). Durch Umformungen und Anwendung von *Lagrange*-Multiplikatoren kann das primale Optimierungsproblem in ein duales Optimierungsproblem umgewandelt werden:

Maximiere

$$\mathcal{Q}(\alpha) = \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{M} \alpha_m \cdot \alpha_n \cdot d_m \cdot d_n \cdot ((\vec{x}_m)^T \vec{x}_n)$$

unter den Randbedingungen:

$$\sum_{m=1}^{M} \alpha_m \cdot d_m = 0 \text{ mit } \alpha_1, ..., \alpha_m \leq 0$$

Der Gewichtungsvektor wird also aus der Linearkombination der Trainingsvektoren berechnet, wobei nur jene Vektoren $\vec{x}_i$ einen Beitrag liefern, deren α_i ungleich 0 ist. Sind die optimalen Werte α für $\mathcal{Q}(\alpha)$ ermittelt, so ergeben sich die Gewichte $\vec{w}$ durch:

$$\vec{w} = \sum_{m=1}^{M} \alpha_m \cdot d_m \cdot \vec{x}_m$$

Der Bias b wird dann wie folgt berechnet: Man wählt einen beliebigen Support-Vektor $\vec{x}_s$, der eine positive Instanz bildet und setzt die beiden Vektoren $\vec{x}_s$ und $\vec{w}$ in

$$\vec{w}^T\vec{x}_s + b = 1$$

ein. Daraus ergibt sich b.

Bemerkung. Die Trainingsvektoren $\vec{x}_m$ gehen in die Formel für $\mathcal{Q}(\alpha)$ nur in Form der paarweisen Skalarprodukte $(\vec{x}_m^T\vec{x}_n)$ ein. Diese Vorgehensweise eignet sich jedoch nur, falls die Trainingsmenge $\mathcal{L}$ linear trennbar ist.

Einbettung des Eingaberaums in einen Merkmalsraum

Kann die Trainingsmenge nicht linear durch eine Hyperebene getrennt werden, müssen die Trainingsvektoren durch eine Abbildung $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N$ in einen (erheblich) größer dimensionierten Vektorraum abgebildet werden. Dabei heißt $\mathbb{R}^n$ der *Eingaberaum (Input Space)* und $\mathbb{R}^N$ der *Merkmalsraum (Feature Space)*. Statt des ursprünglichen Lernproblems

$$\mathcal{L} = \{(\vec{x}_m, d_m) | m = 1, ..., M\}$$

wird nun von einem in den Merkmalsaum eingebettete Problem

$$\phi(\mathcal{L}) = \{(\phi(\vec{x}_m), d_m) | m = 1, ..., M\}$$

ausgegangen. Das duale Optimierungsproblem für $\phi(\mathcal{L})$ lautet nun: Maximiere

$$\mathcal{Q}^{\phi}(\alpha) = \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{M} \alpha_m \cdot \alpha_n \cdot d_m \cdot d_n \cdot ((\phi(\vec{x}_m))^T \phi(\vec{x}_n))$$

unter den Randbedingungen:

$$\sum_{m=1}^{M} \alpha_m \cdot d_m = 0 \text{ mit } \alpha_1, ..., \alpha_m \leq 0$$

Dabei müssen nicht die Merkmalsvektoren $\phi(\vec{x}_m)$ berechnet werden, sondern nur die paarweisen Skalarprodukte $\phi(\vec{x}^m)^T \phi(\vec{x_n})$. Um die Funktion ϕ nicht berechnen zu müssen, wird der so genannte *Kernel-Trick* angewandt, bei dem das Skalarprodukt durch eine Kernelfunktion ersetzt wird. Allgemein heißt eine Funktion $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ eine *Kernel-Funktion*, falls es einen mit einem Skalarprodukt $<>$ versehenen Vektorraum H (Hilberraum) und eine Einbettung $\phi : \mathbb{R}^n \to H$ gibt, so dass für alle $\vec{x}, \vec{y} \in \mathbb{R}^n$ gilt:

$$K(\vec{x}, \vec{y}) =< \phi(\vec{x}), \phi(\vec{y}) >$$

Ist dabei $\phi_a : \mathbb{R}^n \to \mathbb{R}^N$ die Abbildung, die zu einem Vektor x den Vektor aller Produkte von jeweils genau a Attributen (Komponenten des Vektorraums) von $\vec{x}$ bildet, so lautet die zugehörige Kernel-Funktion:

$$K_a(\vec{x}, \vec{y}) =< \phi_a(\vec{x}), \phi_a(\vec{y}) >= (\vec{x}^T \vec{y})^a = (\sum_{i=1}^{n} x_i y_i)^a$$

Allgemeine Vorgehensweise

Zusammenfassend kann die Klassifizierung für ein nicht-lineares Lernproblem $\mathcal{L} = \{(\vec{x}_m, d_m)/m = 1, ..., M\}$ mit einer Einbettung $\phi : \mathbb{R}^n \to \mathbb{R}^N$ und einer Kernel-Funktion $K^{\phi}(x, y) = \phi(x)^T \phi(y)$ folgend gelöst werden: Maximiere (quadratisches Optimierungsproblem)

$$\mathcal{Q}^{\phi}(\alpha) = \sum_{m=1}^{M} \alpha_m - \frac{1}{2} \sum_{m=1}^{M} \sum_{n=1}^{M} \alpha_m \cdot \alpha_n \cdot d_m \cdot d_n \cdot K^{\phi}(\vec{x}_m, \vec{x}_n)$$

unter den Randbedingungen:

$$\sum_{m=1}^{M} \alpha_m \cdot d_m = 0 \text{ mit } \alpha_1, ..., \alpha_m \leq 0$$

Wurden durch die Optimierung die α-Werte bestimmt, kann durch folgende Formel der Bias bestimmt werden:

$$b = \frac{1}{|I|} \sum_{i=1}^{|I|} (d_i - \sum_{j=1}^{n} d_j \cdot \alpha_j \cdot K^{\phi}(x_i, x_j))$$

I enthält dabei alle Indizes der Support-Vektoren ($\alpha_i > 0$) und $|I|$ ist die Anzahl von Support-Vektoren. Bei Eingabe eines neuen Vektor x ist also zu entscheiden, ob gilt:

$$f(x) = sgn(\sum_{m=1}^{M} \alpha_m \cdot d_m \cdot K^{\phi}(x_m, x) + b) > 0$$

Ist diese Bedingung erfüllt, so konnte gezeigt werden, dass der Eingabevektor x eine positive Instanz ist und dem Testmuster entspricht. SVM lassen sich so gut bei Problemen mit sehr vielen Attributen einsetzen, da die Berechnungen nicht auf den Attributen, sondern auf Abstandsberechnungen beruhen. Voraussetzung dazu ist, dass auf dem Vektorraum eine Metrik definiert ist, wie z.B. in Kapitel 5.4.2 am Beispiel der Abbildung eines Multiagenten-Systems auf einen Vektorraum.

Optimierungstechniken

Um das SVM-Problem zu lösen, muss das *(convex) quadratic promming (QP)* Problem gelöst werden. Dazu kann die Formel für $\mathcal{Q}(\alpha)$ folgend umgeschrieben werden:

$$\mathcal{Q}(\alpha) = e^T \alpha - \frac{1}{2} \alpha^T \hat{K} \alpha$$

mit $\hat{K}$ ist eine Matrix $\hat{K}_{ij} = y_i y_j k(x_i, x_j)$ und e dem Einheitsvektor. Aufgrund der konvexen Eigenschaft der Kernel-Funktion ist jedes lokale Maximum zugleich auch ein globales Maximum. Jedoch können mehrere Maxima existieren, in Bezug auf die α_i-Werte des α-Vektors, was zu unterschiedlichen Auswirkungen der Testläufe führen kann. Zur dieser Problematik der quadratischen Optimierung existiert eine Vielzahl von Lösungsvarianten, jedoch eignen sich nicht alle Algorithmen, insbesondere wenn die Dimension des Vektorraums sehr hoch ist und eine große Menge von Testvektoren existiert. Dazu müssen besonders effiziente Verfahren eingesetzt werden, wie sie beispielweise in [SBS99], [SS01] und [OFG96] beschrieben werden. Dabei sind drei unterschiedliche Lösungsansätze erwähnenswert:

- *Chunking* Ausnutzung des Fakts, dass bei einer optimalen Lösung die meisten $\alpha_i = 0$ sind. So kann die Matrix $\hat{K}$ minimiert werden, ohne die Lösung zu verändern.

- *Decomposition:* Aufteilung des Problems in kleinere Probleme. Jedoch konvergieren solche Verfahren nur sehr langsam, weshalb zusätzlich Heuristiken eingesetzt werden müssen.
- *Sequential Minimal Optimization (SMO):* Ähnliche Vorgehensweise wie bei der Dekomposition, jedoch werden nur Probleme mit einer Zweiermatrix betrachtet, wodurch kein quadratischer Optimierer eingesetzt werden muss [Pla99].

Es existiert bereits eine Menge von implementierten Optimierungsalgorithmen, die teilweise frei verfügbar im Internet zum download bereit stehen [SVMa]. Diese Umsetzungen reichen von einfachen MATLAB Implementierungen, bis hin zu hochentwickelten Programmen in C, C++ und FORTRAN [CL02b].

6.5 Zusammenfassung

In diesem Kapitel habe ich eine Charakterisierung von Koalitionen und Teams gegeben und dabei die Unterschiede beider Ansätze bzgl. einer dynamischen Umgebung gezeigt. Aufgrund der Dynamik einer sich verändernden Umwelt von Agenten müssen diese autonom handeln, um ihre Ziele zu erreichen. Es wird in dieser Arbeit ein Verfahren zur Holonenbildung entwickelt und eingesetzt, um den Agenten bei der Bildung von Gruppen maximale Autonomie zu gewähren. Als Folge der Annahme dynamischer, nicht-benevolenter Umgebungen, wie sie hier betrachtet werden, wurde der Begriff der Stabilität zur Bildung von Holonen entsprechend definiert. Erst durch den Einsatz von Lernmechanismen ist möglich Stabilität in dynamischen Umgebungen zu erreichen. Dazu habe ich drei Lernverfahren exemplarisch vorgestellt, bzw. entwickelt, die von statistischen, über stochastischen bis hin zu kernel-basierten Lernen reichen. Die Anwendung dieser Verfahren bei der Formierung von Holonen wird erst möglich durch den Nachweis der Vektorraumeigenschaften eines Multiagenten-Systems aus Kapitel 5. Durch den formalen Nachweis dieser Eigenschaften ist es möglich eine Verknüpfungsvorschrift zur Bildung eines Holonenvektors anzugeben, sodass ein Vergleich der Summation von Agentenvektoren (Holon) mit einzelnen Agentenvektoren möglich ist. Im folgenden Kapitel werden die Erkenntnisse und Verfahren dieses und der vorherigen Kapitel eingesetzt, um ein neues Verfahren zu entwickeln, welches das Problem der dynamischen Formierung von Holonen löst.

Kapitel 7

Simulationsbasierte Verhandlung für dynamische Kooperationen

Multiagenten-Systeme eignen sich besonders gut für die autonome Bearbeitung komplexer Aufgaben, die von einem einzelnen Agenten nicht mehr gelöst werden können. Agenten müssen dazu miteinander kooperieren und sollten sich zu Holonen formieren können. Je komplexer die Aufgabe und damit die Holonenstruktur ist, desto höher steigt die Fehleranfälligkeit des Holons. Sie können zerfallen, bevor alle in einer Mehrschrittaufgabe enthaltenen Aufgaben vollständig bearbeitet sind. Diese Tatsache hat Konsequenzen für die Mitglieder eines Holons und ihre Planausführungen, so dass es unter Umständen trotz Umplanungsversuche nicht gelingt, das gemeinsame Ziel zu erreichen. Agenten müssen dann aufgenommen oder gegebenenfalls aus der Holonenstruktur entfernt werden. Falls das Holon während der Teilnahme an einer Auktion oder anderen Verhandlungen zerbricht, ist es im Allgemeinen notwendig, in kürzester Zeit Ersatz zu finden. Die Kooperationsmöglichkeiten sind hierbei meist stark beschränkt, wie z.B. bei Auktionen, bei denen es im Allgemeinen nicht erlaubt ist während einer laufenden Verhandlung mit anderen Teilnehmern Kontakt aufzunehmen. In einer solchen Situation muss der Holonenführer auf Grund seines Wissens über die Agenten das Risiko abwägen, ob nach Beendigung der Verhandlungen eine Chance zur Umstrukturierung besteht, ohne zuvor mit den anderen Agenten zu kommunizieren. Ist das Risiko zu hoch, dass die nachträglichen Verhandlungen scheitern, so bleibt letztlich nur die Möglichkeit, aus den Verhandlungen auszutreten. Sonst führt der Holonenführer die Verhandlungen mit einer modifizierten, hypothetischen Holonenstruktur weiter, die ihm fortan als Entscheidungsgrundlage dient. Damit solche Abwägungen möglich sind, müssen die Agenten ihr Wissen über die Agentengesellschaft ständig aktualisieren und über diesem Wissen lernen. Für die weiteren Betrachtungen wird von dynamischen Umgebungen ausgegangen, die folgende Eigenschaften besitzen:

- Informationen sind unvollständig,
- Informationen sind unsicher bzw. fehlerhaft und
- Agenten verlassen die Agentengesellschaft bzw. treten ihr jederzeit bei.

7.1 Problematik dynamischer Holonenbildung

Aus den Forschungsbereichen der dynamischen Formierung von Koalitionen, Teams und Holonen existiert eine Vielzahl von Methoden und Schemata zur Bildung von Kooperationen in benevolenten, dynamischen und verteilten Umgebungen. Basis aller Verfahren sind Verhandlungen. Unabhängig davon, ob die Verhandlungen unmittelbar durch die Agenten in bilateralen Verhandlungen ausgeführt werden oder ob eine zentrale Instanz existiert, die die Strukturen der Holone vorgibt, müssen bei der Entwicklung von Verfahren folgende Verhandlungsparameter berücksichtigt werden:

- *Aufgabenverwaltung:* Die Bearbeitung der Aufgaben durch einen Agenten kann auf zwei Arten erfolgen: a) der Agent arbeitet seine Aufgaben sequentiell ab, oder b) die Aufgaben werden parallel bearbeitet. Im Weiteren wird von Agenten ausgegangen, die Aufgaben sequentiell bearbeiten (siehe Kapitel 5).

- *Gewinnverteilung:* Die Verteilung des Gewinns auf die Mitglieder des Holons, ist ein Maß für die Performanz der Agenten eines Holons.

- *Auszahlung (Bedingungen):* Der Gewinn kann auf drei verschiedene Arten auf die Agenten eines Holons verteilt werden:

  - Die Mitglieder erhalten den Teil des Gewinns, entsprechend der zuvor verhandelten und festgelegten Gewinnverteilung.
  - Es existiert kein Gewinn der ausgezahlt wird, sondern die Agenten haben ihren Nutzen aufgrund der Kooperationen an sich, beispielsweise durch Einsparung von Zeit, Ressourcen etc., die durch die Zusammenarbeit mit anderen Agenten in einem Holon möglich sind.
  - Die Agenten erhalten einen sozialen Profit in dem Sinne, dass Agenten ihre Ressourcen kostenfrei den Agenten zur Verfügung stellen, denen Ressourcen fehlen unter der Annahme, dass zu einem späteren Zeitpunkt sie selbst Ressourcen kostenfrei zur Verfügung gestellt bekommen.

- *Auszahlung (Zeit):* Die Auszahlung des Gewinns erfolgt vor oder nach Ausführung der Bearbeitung von Aufgaben durch den Holonenführer. Dabei kann die Auszahlung *synchron* (alle Mitglieder des Holons erhalten zum selben Zeitpunkt ihren Anteil), oder *asynchron* (jeder Agent eines Holons erhält direkt nach Erfüllung seiner Aufgaben seine Auszahlung) durchgeführt werden. Der Auszahlungszeitpunkt wirkt sich stark auf das Verhalten der Agenten aus. Erhalten die Agenten ihre Auszahlung vor der Ausführung, bietet dies beispielsweise einen Anreiz die Aufgaben gegebenenfalls nicht bzw. nicht vollständig zu bearbeiten (siehe 6.2.3).

Diese Parameter sind entweder Teil des Verhandlungsgegenstands bei direkten Verhandlungen der Agenten untereinander, oder sie werden durch den Verhandlungsführer im Voraus festgelegt. Verhandlungen sind ein wesentlicher

Bestandteil zur Bildung von Holonen. Jedoch wird die Umsetzung der geplanten Holonenstrukturen in dynamischen Umgebungen fortwährend gestört. Die Agenten müssen daher auf die unterschiedlichsten Störungen und Ereignissen reagieren. Diese Störungen können in interne und externe Störeffekte aufgeteilt werden können:

- *Interne Störeffekte:* Indem Agenten autonom handeln und zu jedem Zeitpunkt vollständige Kontrolle über ihre Aktionen behalten, sind sie jederzeit in der Lage ein Holon aus eigenem Willen zu verlassen. Diese geschieht aufgrund agenteninterner Entscheidungen, basierend auf eigenem Wissen, Annahmen, Ziele und Bedürfnissen.

- *Externe Störeffekte:* Das Zerbrechen eines Holons kann neben den internen Effekten auch bedingt sein durch externe Effekte, die von *außen* (der Umgebung) auf das Holon und die Agenten einwirken. Externe Effekte beschreiben Einflüsse auf die unmittelbare Umgebung der Sub-Agenten eines Holons. Die Einflüsse haben dabei keinen direkten Zusammenhang zu den Aktionen der Agenten. Sie schränken jedoch die Agenten in ihrem Handeln so ein, dass einige Ziele der Agenten nicht mehr erreicht werden können und dadurch das Holon instabil wird. Beispielsweise zwingt eine Änderung der Aufgabenbeschreibung durch Dritte den Holonenführer zu Umstrukturierungsmaßnahmen. Diese äußeren Störeinflüsse können in zwei Unterklassen aufgeteilt werden:

  - Änderungen in der *Spezifikation* einer Aufgabe, eines Ziels, oder der Eigenschaften eines Agenten. Sie kann den Holonenführer zu Restrukturierungsmaßnahmen des Holons und der Aufgabenverteilung veranlassen.
  - *Veränderungen* in der Umgebung der Agenten führen dazu, dass zuvor getroffene Entscheidungen nicht mehr durchführbar sind. Solche Störungen bewirken das Austreten von Agenten ohne eigenen Willen, beispielsweise durch plötzlich auftretende Probleme in der technischen Kommunikation.

  Die Klasse der externen Effekte fasst alle Störungen zusammen, die einen Agenten veranlassen aus einem Holon auszuscheren, ohne dass dies vom Agenten geplant ist.

Mit Hilfe der internen und externen Effekten können die Ursachen einer Störung beschrieben werden. Eine genauere Analyse der Störfaktoren führt zur Definition von Unsicherheitsklassen. V.Ermolayev et al. [EBT01] schlägt 4 unterschiedliche Unsicherheitsklassen vor, die alle dem Verhalten eines Agenten, also den internen Effekten, zugeschrieben werden können:

a) Vages Wissen über die *Fähigkeiten* eines Agenten.

b) *Kapazität* an freien Ressourcen eines Agenten, abhängig von der jeweiligen Aufgabensituation.

c) *Bedarf* eines Agenten an den zu verhandelnden Objekten.

d) *Qualität* eines Agenten, ermittelt aus der Differenz zwischen den Soll-Anforderungen einer Aufgabe und dem Ist-Zustand der Ausführung durch den Agenten.

Untersucht man die externen Effekte, so können weitere vier Unsicherheitsklassen identifiziert werden, die sich auf die Bildung von Holonen auswirken können:

e) *Störung* der Kommunikation (z.B. Netzwerkproblem).

f) *Änderung* der Aufgabenbeschreibung durch Dritte.

g) *Ausfall* von Agenten während der Aufgabenbearbeitung oder bereits in der Verhandlungsphase.

h) *Eintritt* neuer Agenten in die Gesellschaft zu einem beliebigen Zeitpunkt.

Ein Verfahren zur Bildung von Holonen in dynamischen Umgebungen muss also mit den hier definierten Unsicherheitsklassen umgehen können.

7.1.1 Dynamic-Resource-Allocation Algorithmen

Zur Lösung der Problematik in dynamischen Systemen gibt es außerhalb der Gruppenbildung Verfahren, die in modifizierter Version zur Holonenformierung eingesetzt werden können. Eine der Holonenbildung verwandte Thematik ist die dynamische Allokation von Ressourcen zur Laufzeit, indem die Dienste der Agenten als Ressourcen angesehen werden, die temporär für verschiedene Holone reserviert werden. Bocheck und Chang beschreiben in [BC98] ein Video-On-Demand-System, das dynamisch die notwendige Bandbreite in einem Netzwerk reserviert, entsprechend dem Bedarf, der durch die aktuell betrachteten Filmsequenzen entsteht. Die Architektur dieses Systems ist jedoch auf einer zentralen Komponente aufgebaut, wodurch die Fehleranfälligkeit stark zunimmt und hinsichtlich Sicherheit Risiken birgt. Ebenfalls enthält das System von Gibney und Jennings [GJ98] zur dynamischen Ressourcenverteilung einen zentralen Mediator, der die Ressourcenverhandlung der Agenten koordiniert und somit eine Ausfallquelle für das gesamte System birgt. Zudem wird hierbei, wie auch im Allgemeinen bei anderen Dynamic-Resource-Allocation Algorithmen [ESR+00], [CMM97], [MSS02] von einer benevolenten Agentengesellschaft ausgegangen. Dabei werden Probleme dynamischer Umgebungen, wie sie in dieser Arbeit betrachtet werden, ausgeblendet, wie z.B. der uneingeschränkte Austausch von Informationen und deren Korrektheit. Durch die Annahme einer benevolenten geschlossenen Gesellschaft entfällt nicht nur der Aspekt der Datensicherheit im Hinblick auf Weitergabe an Dritte, ohne dass Rückschlüsse auf diese möglich sind, sondern auch die Fragestellung, wie mit unsicheren Daten umzugehen ist. Insbesondere führt diese Problematik dazu, dass die Eigenschaften bzw. das Verhalten von Agenten/Ressourcen erlernt werden müssen. Dies ist ein zentraler Aspekt von dynamischen Systemen, der bei der dynamischen Ressourcenverteilung im Allgemeinen nicht berücksichtigt wird. Verfahren aus dem Bereich der *dynamic resource allocation* können daher nicht direkt für die hier beschriebene Problematik der dynamischen Holonenbildung eingesetzt werden.

7.1.2 Zweiphasen Verhandlungsprotokoll

Um das Problem der dynamischen Bildung von Holonen lösen zu können, müssen neben Verhandlungen, wie sie z.B. während der Ressourcenallokation eingesetzt werden, auch Lernverfahren integrativ eingesetzt werden. Zusammenfassend kann ein Verfahren zur Bildung von Holonen in zwei Phasen unterteilt werden, die nebenläufig ausgeführt werden, jedoch von einander abhängig sind (siehe Abbildung 7.1). Phase eins dient dem Aufbau von Wissen über die Umwelt des Agenten und Phase zwei befasst sich mit dem Kooperations- und Koordinationsprozess zur und während der Ausführung von Aufgaben. Diese Betrachtungen führen zu nachfolgendem simulationsbasierten Schema zur Bildung von Holonen.

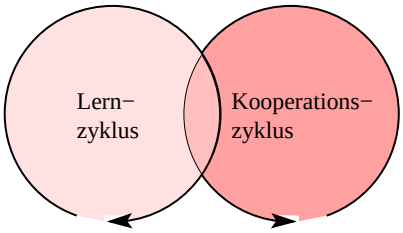

Abbildung 7.1: Zweiphasen Verhandlungsprotokoll

7.2 DHF-S: Dynamische Bildung von Holonen

In einem ersten Ansatz habe ich ein DCF Schema entwickelt, das eine Vorgehensweise beschreibt, mit der es möglich ist, in einer dynamischen Umgebung Gruppen von autonomen, zum Teil nicht-benevolenten Agenten zu bilden. Das '*simulation based dynamic coalition formation*' ($DCF-S$) Schema beschreibt eine grobe Vorgehensweise, mit dessen Hilfe es möglich ist, spezielle Schemata und Algorithmen entsprechend der Anwendungsdomäne daraus zu entwickeln. Dieses Schema habe ich zunächst als Grundlage für die Koalitionsformierung entwickelt und anschließend zur Formierung von Holonen erweitert. Aus dem allgemeinen DCF-S Schema habe ich dann ein Schema abgeleitet, das speziell zur Bildung von Holonen in einer dynamischen Umgebung eingesetzt werden kann. Unter Einsatz eines '*simulation based dynamic holon formation*' ($DHF-S$) Algorithmus ist es den Agenten dann möglich, effizient auf die Veränderungen ihrer Ziele und Umgebung reagieren zu können. Im Folgenden wird das DHF-S Schema vorgestellt und anschließend in Abschnitt 7.3 anhand von vier Algorithmen exemplarisch instanziiert, die abschließenden im Kapitel 8 in der Praxis evaluiert werden.

7.2.1 Umgebung

Ein Holonenformierungsalgorithmus ist notwendig in Szenarien, in denen die Agenten kontinuierlich neue Aufgaben erhalten, wobei die alleinige Ausführung

nicht bei jeder Aufgabe möglich ist und der Agent dazu die Unterstützung anderer Agenten benötigt. Den Agenten ist es dabei zu jeder Zeit möglich, ein Holon zu führen, einem Holon beizutreten oder es zu verlassen. Bei dem Entscheidungsprozess eines Agenten müssen Nutzenberechnungen auf vagen Informationen durchgeführt werden. Durch Lernen aus dem Verhalten anderer Agenten, ist es einem Agenten möglich, Rückschlüsse aus der Vergangenheit zu ziehen und somit die Wahl potentieller Mitglieder einzuschränken und größtmögliche Erfolgschancen zu garantieren. Um den Agenten eine Möglichkeit zu geben, eine allgemeine Einschätzung der restlichen Agenten zu erhalten, wird ein so genannter *World-Utilitiy-Agent* (WUA) eingeführt. Dieser Agent stellt jedem Agenten globale Informationen auf Anfrage zur Verfügung. Die Agenten haben somit zusätzlich die Möglichkeit, eine Bewertung der Gesellschaft bezüglich der angefragten Agenten zu erhalten. Zudem erfahren die Agenten direkt beim WUA welche Agenten dem System beigetreten bzw. ausgeschieden sind. Die Agenten müssen nicht notwendigerweise die Dienste des WUA nutzen, sondern können sich ausschließlich auf ihre eigenen Einschätzungen stützen. Damit der WUA allgemeine Bewertungen von Agenten durchführen kann, senden die Agenten in regelmäßigen Abständen Informationen über ihren eigenen Status und Einschätzungen über andere Agenten an den WUA, der diese Informationen geeignet bewertet. Information über einen Agenten enthalten z.B. Aussagen über die Verlässlichkeit und Qualität der Aufgabenausführung. Der Einsatz von WUAs ist jedoch optional und dient lediglich der Performanzsteigerung.

7.2.2 Schema

Das Schema zur Bildung von Holonen in dynamischen Umgebungen löst das DHF-Problem. Im DHF-S Schema simuliert und verhandelt jeder Kopf eines Holons mit potentiellen Partnern neue Holonenstrukturen. Bei der Wahl von potentiellen Mitgliedern muss ein sinnvolles Maß zwischen dem Risiko eines Fehlschlags und dem Nutzen für das Holon abgewogen werden. Das Schema ist in drei Phasen unterteilt:

1. *Initialphase*
   Das Schema startet mit der Initialphase zum Finden eines hypothetischen Holons, das in der folgenden Phase durch konkrete Verhandlungen mit den Agenten realisiert wird. Für die Simulation des hypothetischen Holons sind zwei Schritte notwendig:

   (a) *Vorbereitungsphase*
   Die zugewiesene Mehrschrittaufgabe bestimmt die Aufgabenmenge, die durch Kooperationen mit anderen Agenten bearbeitet werden müssen. Um im weiteren Verlauf eine Auswahl an potentiellen Partnern treffen zu können, aktualisiert der Holonenführer in regelmäßigen Abständen sein Wissen über Umwelt und Agenten durch Anfragen beim WUA. Die Wissensbasis des Holons enthält (partielle) Informationen über Problemlösungsverfahren anderer Agenten, wie auch die Bewertungen der vergangenen Zusammenarbeiten bezüglich Ausfallsicherheit, Vertrauen, etc. Die Informationen können jedoch

unvollständig und unzutreffend sein. Daher müssen zur Bestimmung der potentiellen Partner zusätzlich Lernverfahren eingesetzt werden, um die fehlenden Informationen realistisch abzuschätzen. Diese Funktionalität wird in der Methode *Update_ Agent_ Information()* zusammengefasst, die eine aktualisierte Agentenliste bezüglich der angegebenen Mehrschrittaufgabe (MAG) zurückgibt.

Pseudo-Code 7.1 (Vorbereitungsphase 1a)**.**

$Actionlist|_{MAG} := BestActionlist|_{MAG} := \emptyset; penalties := 0;$
$Agentlist|_{MAG}$:=Update_Agent_Information(MAG);
$HypHolon|_{MAG} := H|_{MAG}$;
Update_MAG;

(b) *Simulation*
Der Kopf des Holons simuliert die Bildung eines modifizierten Holons aufgrund seines Wissens. Dabei tritt er während der Simulation mit keinem Agenten in direkten Kontakt. Verschiedene Strukturen des Holons werden, ohne Bindungen mit den Agenten einzugehen, auf ihre Qualität getestet, d.h. die Simulation hat keine Auswirkungen auf die Agenten. Die Simulationsphase besteht aus folgenden Schritten:

i. Zu Beginn legt der Holonenführer die Anzahl der Simulationsrunden in Abhängigkeit der zur Verfügung stehenden Zeit fest.

ii. In jeder Simulationsrunde wird aus der Liste der möglichen Aktionen (noop - keine Aktion ausführen, add_agent und remove_agent) eine randomisiert ausgewählt und entsprechend des Operators ein Agent selektiert, auf dem diese Operation ausgeführt wird. Soll ein Agent dem Holon hinzugefügt werden, so wird mit Hilfe der Funktion *SelectAgent_ MaxValue(Agentlist)* eine noch nicht zugewiesene Aufgabe der Mehrschrittaufgabe ausgewählt. Anschließend wird ein Agent aus der Liste aller Agenten, der *Agentlist*, ausgewählt, der diese Aufgabe lösen kann. Liegt die Bewertung des Holons und des Agenten über einer festgelegten Schranke, so wird dieser Agent dem hypothetischen Holon zugefügt und die Aufgabe aus der Liste der noch offenen Aufgaben gelöscht. Damit zu einem späteren Zeitpunkt das hypothetische Holon durch Verhandlungen realisiert werden kann, werden alle Aktionen einer Simulationsphase in einer *Aktionsliste* protokolliert. Analoge Vorgehensweise bei der Durchführung der Aktion *remove_ agent*. Die Funktion *SelectAgent_ MinValue(HypHolon)* wählt einen Agenten aus dem hypothetischen Holon aus, das minimalem Nutzen für das Holon besitzt. Ist der Nutzen höher als den Verlust an Reputation durch eine mögliche schlechte Bewertung dieses Agenten, so wird die Aktion ausgeführt und die entsprechenden Aufgaben der Liste, der noch offenen Aufgaben,

wieder hinzugefügt.

iii. Nach Ende der Simulationsrunden wird das so gefundene hypothetische Holon mit einer vorherigen Lösung verglichen. Ist die Bewertung des neu zusammengestellten Holons besser, wird dieses fortan als beste Holonenstruktur gespeichert, so dass bei einem Abbruch der Simulationsphase diese als Resultat verwendet werden kann.

iv. Im letzten Schritt der Simulationsphase wird überprüft, ob die beste hypothetische Holonenstruktur eine deutliche Verbesserung gegenüber der aktuellen Holonenstruktur verspricht. Ist dies der Fall, wird die Simulationsphase unter der Voraussetzung abgebrochen, dass sich die Umgebung währenddessen nicht verändert hat. Anschließend wird zu *Phase 2* des Schemas gewechselt, der Umsetzung der hypothetischen Holonenstruktur. Sonst wiederholt sich das Schema mit *Phase 1*, bis eine signifikante Verbesserung gegenüber der aktuellen Situation gefunden wird.

Pseudo-Code 7.2 (Simulationphase 1b)**.**

```
for [i := 1 to MaxSimulationRounds] do
    op := Random({noop, add_agent, remove_agent});
    if [op == add_agent] then
        (A, AG, costs):= SelectAgent_MaxValue(Agentlist|_MAG);
        Actionlist|_MAG := Actionlist|_MAG + (A, add, costs);
        HypHolon|_MAG := HypHolon|_MAG ∪ {A};
        MAG := MAG \ {AG};
    if [op == remove_agent] then
        (A, AG, costs):= SelectAgent_MinValue(HypHolon|_MAG);
        if [Value(HypHolon|_MAG \ {A}) ≻ RepMalus(A, HypHolon|_MAG)] then
            Actionlist|_MAG := Actionlist|_MAG + (A, remove, costs);
            HypHolon|_MAG := HypHolon|_MAG \ {A};
            MAG := MAG ∪ {AG};
            penalties := penalties+ RepMalus(A, HypHolon|_MAG);
if [Value(Actionlist|_MAG) > Value(BestActionlist|_MAG) AND penalties < OwnPenalty] then
        BestActionlist|_MAG := Actionlist|_MAG;
if [(Value(BestActionlist|_MAG) >> v(H|_MAG) AND Events(BestActionlist|_MAG) = ∅)] then
        stop Simulationsphase;
```

2. *Verhandlung*
   Aufbauend auf der in *Phase 1b* gefundenen hypothetischen Holonenstruktur, wird in dieser Phase versucht, die nötigen Modifikationen an dem existierenden Holon hinsichtlich der hypothetischen Struktur durch Neuverhandlungen zu realisieren. Scheitert eine der geplanten Verhandlungen zur Realisierung des hypothetischen Holons, oder verändert sich die Umwelt, wird diese Phase abgebrochen, da die Voraussetzungen der Simulationsphase nicht mehr gegeben sind. Bei einem Abbruch, wie auch bei einer erfolgreichen Durchführung der Verhandlungen, gilt fortan das resultierende Holon als die aktuelle Holonenstruktur.

Pseudo-Code 7.3 (Verhandlungphase 2).

```
halt := false;
foreach[(A, op, costs) in BestActionlist|MAG] do
    try
        if [op == add] then
            if [BilateralNegotiation(A, costs) == successful] then
                MAG := MAG \ {AG};
                H|MAG := H|MAG∪ {A};
        if [op == remove] then
            MAG := MAG \ {AG};
            H|MAG := H|MAG\ {A};
            Payment(A);
    catch(Events(BestActionlist|MAG) <> ∅)
        StopNegotiation;
        Goto (3);
```

3. *Evaluation*
   Der Kopf des Holons analysiert die Ergebnisse der durchgeführten Verhandlungen und kombiniert diese Erkenntnisse mit dem eigenen Wissen. Die überarbeitete Bewertung kann vom Holonenführer dem WUA mitgeteilt werden. Abschließend kontrolliert der Holonenführer die Verteilung der Auszahlungen bzw. die Verteilung der Ressourcen.

Pseudo-Code 7.4 (Evaluierungsphase 3).

```
EvalRes:= Evaluate(Agentlist|MAG);
if desired then Send(EvalRes, WUA);
```

Nach Abschluss der Evaluierungsphase wiederholt sich das Schema mit *Phase 1* solange, bis alle Aufgaben der Mehrschrittaufgabe erfolgreich ausgeführt wurden. Für das Schema von Bedeutung sind neben den bereits vorgestellten Funktionen noch weitere Funktionen und Variable. Das Drei-Tupel $(A, AG, costs)$, das von den beiden Funktionen *SelectAgent_ MaxValue()* und *SelectAgent_ MinValue()* als Ergebnis zurückgegeben wird, beschreibt einen durch die Funktionen selektierten Agenten A, eine Aufgabe AG, die diesem Agenten zugewiesen bzw. entnommen wird und die Kosten *costs*, die dem Holon durch Hinzunehmen oder Entfernen dieses Agenten entstehen. Das Ablegen der Kosten in der *Actionlist* dient zur Qualitätskontrolle während den Verhandlungen. Insbesondere für die Ausführung der Funktion *BilateralNegotiation()* ist es für das Holon notwendig zu wissen, unter welchen Bedingungen eine Verhandlung erfolgreich ist und wann die Verhandlungen abgebrochen werden müssen, wie z.B. Annahmen über ein Zeitintervall zur Aufgabenausführung. Liegen die Attributwerte außerhalb der Toleranzen der Simulation, muss die Verhandlungsphase abgebrochen werden, da die Simulationsergebnisse unter anderen Bedingungen

erstellt wurden. Sonst bestünde die Gefahr, dass beispielsweise die Gesamtkosten für das Holon die Einnahmen durch die Bearbeitung der Aufgaben übersteigen. Bevor mit weiteren Verhandlungen fortgefahren werden kann, müssen zunächst erneute Simulationen ausgeführt werden, Zur Nutzenbestimmung wird eine Funktion *Value()* eingesetzt. Mit deren Hilfe können verschiedene Holonenstrukturen verglichen werden. Zur Erfassung von Veränderungen in der Umwelt wird eine Funktion eingesetzt, die gegebenenfalls einen Neustart der Simulationsphase initiiert, bzw. die Verhandlungsphase abbricht, falls sich die Voraussetzungen hinsichtlich der Simulationsergebnisse geändert haben. Ein Abbruch der Verhandlungsphase führt auch zu einem Neustart der Simulationsphase, in deren Vorbereitungsphase die aktuelle Situation erfasst und Veränderungen wahrgenommen werden. So basieren alle folgenden Berechnungen auf aktuellen und gültigen Randbedingungen. Im Folgenden wird eine Übersicht des Schemas im Pseudo-Code 7.5 gegeben.

Bemerkung. Resultate der Value-Funktion können nur dann miteinander verglichen werden, wenn sich die Ergebnisse auf die gleiche Mehrschrittaufgabe beziehen und sich die Umgebung währenddessen nicht verändert hat. Wird diese Funktion von verschiedenen Agenten ausgeführt, können die Ergebnisse nicht miteinander verglichen werden. Jeder Agent, der diese Funktion verwendet, erstellt eine individuelle Gewichtung der Aufgaben- und Agentenattribute. Somit würde ein Vergleich zweier Holone unter verschiedenen Randbedingungen durchgeführt, wodurch keine Aussagen über das Vergleichsergebnis getroffen werden könnten.

Pseudo-Code 7.5 (DHF-S Schema).

```
foreach MAG do concurrently, until all tasks are fulfilled
    1) Initialisation
    while [not halt] do
        1a) Preparation
        Actionlist|_MAG := BestActionlist|_MAG := ∅; penalties := 0;
        Agentlist|_MAG:=Update_Agent_Information(MAG);
        HypHolon|_MAG := H|_MAG;
        Update_MAG;
        1b) Simulation
        for [i := 1 to MaxSimulationRounds] do
            op := Random({noop, add_agent, remove_agent});
            if [op == add_agent] then
                (A, AG, costs):= SelectAgent_MaxValue(Agentlist|_MAG);
                Actionlist|_MAG := Actionlist|_MAG + (A, AG, add, costs);
                HypHolon|_MAG := HypHolon|_MAG ∪ {A};
                MAG := MAG \ {AG};
            if [op == remove_agent] then
                (A, AG, costs):= SelectAgent_MinValue(HypHolon|_MAG);
                if [Value(HypHolon|_MAG \ {A}) ≻ RepMalus(A, HypHolon|_MAG)] then
                    Actionlist|_MAG := Actionlist|_MAG + (A, AG, remove, costs);
                    HypHolon|_MAG := HypHolon|_MAG \ {A};
                    MAG := MAG ∪ {AG};
                    penalties := penalties+ RepMalus(A, HypHolon|_MAG);

        if [Value(Actionlist|_MAG) > Value(BestActionlist|_MAG) AND penalties < OwnPenalty] then
                BestActionlist|_MAG := Actionlist|_MAG;
        if [(Value(BestActionlist|_MAG) >> v(H|_MAG) AND Events(BestActionlist|_MAG) = ∅)] then
                halt := true;
    end-while

    2) Negotiation
    halt := false;
    foreach[(A, op, costs) in BestActionlist|_MAG] do
        try
            if [op == add] then
                if [BilateralNegotiation(A, AG, costs) == successful] then
                    MAG := MAG \ {AG};
                    H|_MAG := H|_MAG∪ {A};
            if [op == remove] then
                MAG := MAG \ {AG};
                H|_MAG := H|_MAG\ {A};
                Payment(A);
        catch(Events(BestActionlist|_MAG) <> ∅)
            StopNegotiation;
            Goto (3);

    3) Evaluation
    EvalRes:= Evaluate(Agentlist|_MAG);
    [if desired then Send(EvalRes, WUA);]
```

7.2.3 Eigenschaften

Der Umgang mit unvollständigen und vagen Informationen ist Ziel der Entwicklung dieses Schemas. Das hier vorgestellte Verfahren zur Holonenformierung bildet unter Umständen sub-optimale Lösungen aufgrund des eingeschränkten Suchraums. Durch andauernde Verhandlungen wird kontinuierlich versucht, die Qualität der Lösung zu verbessern. Alle Agenten behalten zu jedem Zeitpunkt ihre Autonomie und kontrollieren selbstständig ihre Ressourcen und Aktionen. Dabei müssen die Agenten miteinander kooperieren, um die Mehrschrittaufgaben zu lösen, die ein einzelner Agent nicht lösen könnte. Erhält ein Holon eine Mehrschrittaufgabe, führt er zunächst Verhandlungssimulationen mit den erlernten Profilen der anderen Agenten durch, ohne mit ihnen direkt zu kommunizieren. Wird eine akzeptable Lösung gefunden, versucht dieser Agent mit den entsprechenden Agenten der simulierten Lösung verbindliche Verträge auszuhandeln. Scheitert eine dieser Verhandlungen, ändert sich die Umgebung oder ein Agent verlässt das Holon vor Beendigung seiner Aufgaben, führt der Holonenkopf erneut Simulationen durch, bis eine neue hypothetische Lösung gefunden wird. Diese versucht er anschließend durch bindende Verhandlungen zu realisieren. Analog versuchen die Agenten ihre eigene Situation zu verbessern. Wurde durch die Simulation eine signifikante Verbesserung gefunden, wird erneut verhandelt, bzw. einzelne Agenten verlassen das Holon. Als Basis für die Simulation baut jeder Agent eine Wissensbasis auf, in der die Attribute der anderen Agenten abgebildet werden. Der Vergleich der Vor- und Nachteile des Schemas zeigt, dass das Schema wesentliche Probleme der Holonenbildung in dynamischen Umgebungen lösen kann, jedoch nicht ohne Zugeständnisse an Performanz und Vollständigkeit. Die Vorteile sind mit $\oplus$ und die Nachteile mit $\ominus$ gekennzeichnet und entsprechend der behandelten Probleme aufgeführt.

1. Zentrale Instanz

$\oplus$ Das Schema benötigt keine zentrale Instanz zur Bildung von Holonen und vermeidet so einen Engpass bzgl. Kommunikation, Berechnungskomplexität und Reaktionszeit.

$\ominus$ Durch Vermeiden einer zentralen Instanz mit vollständigem Wissen kann unter Umständen eine existierende optimale Lösung nicht gefunden werden.

2. Autonomie

$\oplus$ Autonomie der Agenten bezüglich ihrer Aktionen bleibt erhalten. Die Agenten entscheiden eigenständig, welchen Holonen sie beitreten oder verlassen.

$\ominus$ Das Vermeiden einer weisungsbefugten Einheit zur Wahrung der Autonomie der Agenten verhindert gegebenenfalls die Möglichkeit, notwendige Veränderungen zur Steigerung der globalen Effizienz durchzuführen.

3. Wissen

⊕ Agenten behalten die eigenständige Verwaltung ihres Wissens. Zur Bildung von Holonen muss die Wissensbasis nicht notwendigerweise veröffentlich werden. Der Agent bestimmt selbst zu welchem Grad er Informationen weitergibt.

4. Qualität

⊕ Durch Einschränkung der Simulationsphase auf eine beschränkte Anzahl von maximal k Agenten, die zur Holonenbildung untersucht werden, bleibt der Algorithmus auch in Szenarien mit einer sehr hohen Anzahl von Agenten performant.

⊖ Die Qualität der Lösungen hängt stark von der Anzahl der untersuchten Agenten und der Anzahl von Simulationszyklen (siehe *for*-Schleife der Simulationsphase) ab. Ist die Anzahl der Zyklen zu gering, findet der Algorithmus unter Umständen keine Lösung trotz ihrer möglichen Existenz. Die Zyklenanzahl wird daher von den Agenten individuell angepasst und ist dabei direkt proportional zur Komplexität des möglichen Suchraums. Die Länge einer Simulationsrunde kann so in Abhängigkeit der zur Verfügung stehenden Planungszeit gewählt werden, bzw. vergrößert werden, um eine bessere Lösung zu finden.

5. Effektivität

⊕ Zur Anpassung der Holonenstruktur müssen nicht alle Holone neu berechnet werden. Es genügt die Partner der betroffenen Holone auszutauschen, die zum Erreichen der Ziele (unter Berücksichtigung der Veränderungen) notwenig sind. Im Idealfall muss nur eine Modifikation am Holon durchgeführt werden, wo hingegen bei herkömmlichen Ansätzen oft die kompletten Gruppenstrukturen neu berechnet und gebildet werden müssen. Die Strukturen der Holone bleiben so im Allgemeinen stabiler, da nur wenige Agenten von den Veränderungen betroffen sind.

6. Anwendbarkeit

⊕ Der Algorithmus ist sowohl für den Kopf eines Holons, wie auch für die Teilnehmer anwendbar. Die Agenten können jederzeit den Nutzen ihres Holons (als Kopf oder Teilnehmer) individuell ermitteln und durch gezieltes Hinzunehmen oder Ausschließen von Subagenten, bzw. durch Ein- oder Austreten aus dem Holon aktiv auf Veränderungen der Umwelt reagieren.

7. Lernen

⊕ Einsatz von Lernverfahren ermöglicht den Agenten eine realistische Abbildung der Agentengesellschaft individuell vorzunehmen, die als Grundlage für Verhandlungen und Strategien dient. Dadurch wird eine realistischere Prognose über das Verhalten der anderen Agenten möglich.

⊖ Für den Einsatz von Lernverfahren ist eine Vorlaufzeit notwendig, bis die Einschätzungen den realen Werten nahe kommt. Ein neu erzeugter Agent wird daher zu Beginn mehr Fehlentscheidungen bzw. Fehleinschätzungen machen. Um den Lernprozess zu beschleunigen, kann jeder Agent Informationen vom WUA über die anderen Agenten anfordern.

8. Aktualisierung

⊕ Aufgrund der ständigen Suche nach einer signifikanten Verbesserung in der aktuellen Struktur des Holons (Simulationsphase), werden Veränderungen in der Agentengesellschaft automatisch durch Aktualisierung der Wissensbasis der Agenten berücksichtigt.

⊖ Aussagen bezüglich der Qualität der Lösungen zu einem vorgegebenen Zeitpunkt lassen sich nur abschätzen, analog bei Angaben, wann und ob eine optimale Lösung gefunden wird. Hierbei sind keine exakten Laufzeitangaben möglich.

9. Lösungsfindung

⊕ Die andauernde Simulationsphase ist notwendig, um eine Lösung für eine gegebene Aufgabe *AG* zu finden. Wenn zum aktuellen Zeitpunkt nicht genügend Agenten zum Lösen der Aufgabe existieren, so bricht der Algorithmus nicht ab, sondern analysiert weiter. Auf diese Weise findet er möglicherweise zu einem späteren Zeitpunkt neue Agenten, die zur Lösung des Problems beitragen.

10. Manipulation

⊕ Der Einsatz von WUA ermöglicht den Agenten, schneller Veränderungen in der Gesellschaft zu erkennen. Dies betrifft das Ein- und Austreten von Agenten in die Gesellschaft und die Bewertung ihrer Attribute durch die Gesellschaft. Der WUA ermittelt eine globale, allgemeine Bewertung der Agenten auf der Basis von lokalen, individuellen Einschätzungen. Durch den Einsatz von WUAs können negative Bewertungen aufgrund von Bestrafungen schneller verbreitet werden, wodurch das Vermeiden von Strafen einen höheren Stellenwert erlangt. Als Konsequenz werden die Holone stabiler.

⊖ Der Einsatz von WUA beinhaltet die Gefahr der zentralen Manipulation der *öffentlichen Meinung*. Dies kann jedoch entschärft werden, wenn von jedem Agenten z.B. nur die aktuellste Aussage gewertet wird. Somit sind Mehrfachäußerungen in kurzer Zeit wirkungslos. Um eine signifikante Veränderung zu bewirken, bedarf es somit vieler Agenten, die eine ähnliche Aussage machen.

Zusammenfassend überwiegen die hier vorgestellten Vorteile gegenüber den Nachteilen. Die Ursachen dieser Nachteile beruhen im Wesentlichen auf den Vorgaben der dynamischen Umgebung. Annahmen und präzise Angaben können nicht gemacht werden, wodurch Nachteile entstehen, die in dynamischen

Umgebungen zum Teil unvermeidlich und allgemeiner Natur sind. Zum Beispiel können Qualitätsaussagen und Laufzeitangaben in dynamischen Umgebungen nur approximativ angegeben werden.

7.2.4 Terminierung

Als Grundlage für den Beweis der Terminierung von Algorithmen, die auf diesem Schema aufbauen, wird vorab gezeigt, dass das Schema unter der Voraussetzung terminiert, dass die eingesetzten Funktionen terminieren[1]:

- *Phase 1a* führt Aktualisierungen der eigenen Wissensbasis durch, wobei die Anzahl der Aktualisierungsschritten nach oben durch die Anzahl der Agenten begrenzt ist.

- *Phase 1b* ist durch einen Rundenbegrenzer begrenzt. Phase 1b wiederholt sich solange, bis alle Aufgaben der MAG vollständig abgearbeitet sind. Existiert zunächst kein Holon, wird in der Simulationsphase zumindest ein hypothetisches Holon gefunden, das einen Teil der MAG lösen kann, sofern die Wissensbasis mindestens die Beschreibungen eines Agenten enthält, die zur Aufgabenbearbeitung in Frage kommen. Enthält die Wissensbasis keinen Agenten für mindestens eine Aufgabe der MAG, kann keine vollständige Lösung gefunden werden.

- Die Initialisierungsphase wird angehalten sobald eine bessere Lösung als die existierende gefunden wurde. Dies trifft bei dem ersten Durchlauf der Simulationsphase immer zu, da jede Lösung besser ist, als das leere Holon, außer die Wissensbasis enthält zu keiner Aufgabe der MAG einen Agenten, der geeignet ist, diese zu lösen. Dann wiederholt sich die erste Phase solange, bis entweder der Holonenführer Kenntnis von geeigneten Agenten durch die Aktualisierung seiner Wissensbasis erhält, oder die Bearbeitung der MAG ist nicht mehr möglich, da zum Beispiel die Zeit abgelaufen ist, in der die MAG bearbeitet werden muss. Die Simulationsphase endet, da keine offenen Aufgaben mehr existieren.

- In *Phase 2* werden Verhandlungen ausgeführt. Dabei handelt es sich um eine begrenzte Anzahl von Agenten, die angeschrieben werden. Eine obere Schranke für die Anzahl von Verhandlungen ist durch die Anzahl der möglichen Runden in der Simulationsphase gegeben. Scheitert eine Verhandlung, wird diese Phase beendet. Sonst werden alle Aktionen ausgeführt und diese Phase endet anschließend.

- *Phase 3* bewertet die Agenten nachträglich, mit denen erfolgreiche Verhandlungen durchgeführt wurden. So ist die maximale Anzahl von Agenten ebenfalls durch den Rundenbegrenzer in der Simulationsphase gegeben, so dass die Bewertung nach endlicher Zeit beendet wird.

[1] Algorithmen basierend auf diesem Schema terminieren also prinzipiell. Für den vollständigen Nachweis der Terminierung eines DHF-S Algorithmus genügt es dann die Terminierung dieser Funktionen (siehe Kapitel 7.3) im Detail zu zeigen.

Diese Phasen wiederholen sich solange, bis alle Aufgaben entweder korrekt abgearbeitet wurden, oder die Zeit für die Bearbeitung überschritten und somit die Aufgaben gelöscht wurden und das Verfahren terminiert.

□

7.3 DHF-S Algorithmen

Das DHF-S Schema beschreibt eine Klasse von Algorithmen, die im Wesentlichen aus zwei Teilen bestehen, zum Einen die Simulation von möglichen Holonenstrukturen und zum Anderen die Verhandlungsphase zur Realisierung der zuvor generierten hypothetischen Holonenstrukturen. Im Folgenden werden sieben Funktionen des Schemas definiert, so dass durch deren konkreten Instanziierung aus dem DHF-S Schema ein Algorithmus entsteht:

- Phase 1a

  F1 'Update_Agent_Information(MAG)'
  F2 'Update_MAG'

- Phase 1b

  F3 'SelectAgent_MaxValue($Agentlist|_{MAG}$)'
  F4 'SelectAgent_MinValue($HypHolon|_{MAG}$)'
  F5 'Value('$HypHolon|_{MAG}$)' und Value('$Actionlist|_{MAG}$)'

- Phase 2

  F6 'BilateralNegotiation(A, costs)'

- Phase 3

  F7 'Evaluate($Agentlist|_{MAG}$)'

Als Grundlage für die Beschreibung der Algorithmen werden die Agenten, Dienste und Aufgaben wie im Kapitel 5 durch Vektoren und Holonen als Matrizen dargestellt. Eine Instanziierung dieser Funktionen wird nachfolgend anhand von vier DHF-S Algorithmen demonstriert.

7.3.1 Algorithmus 1 - *Random*$^-$ DHF-S

Random$^-$ DHF-S ist ein vollständig randomisierter Algorithmus, bei dem auf die Verwendung einer expliziten Wissensbasis verzichtet wird. Nur die Adressen der bekannten Agenten werden gespeichert und die Selektion von Agenten wird daher in der Simulationsphase zufällig getroffen. Ziel ist es, neue hypothetische Holonenstrukturen zu finden, bei denen mindestens ebenso viele Aufgaben bearbeitet werden, wie in der existierenden Struktur. Durch den Verzicht einer expliziten Repräsentation der Agenteneigenschaften in der Wissensbasis müssen nicht alle Funktionen des Schemas implementiert werden, bzw. enthalten einen leeren Funktionsrumpf.

F1 - Aktualisierung der Wissensbasis

Dieser DHF-S Algorithmus wählt zufällig unter den aufgabenrelevanten Agenten eine geeignete Menge zur Holonenbildung aus. Daher ist bei diesem Algorithmus keine explizite Repräsentation der Agentengesellschaft in der Wissensbasis notwendig, weshalb der Funktionsrumpf von F1 leer ist.

Funktionsrumpf 1. (F1)

```
Update_Agent_Information(MAG)
{
    empty
}
```

Terminierung Da der Rumpf dieser Funktion leer ist, terminiert diese Funktion.

Laufzeit Der Aufruf dieser Funktion wird in konstanter Zeit $O(1)$ ausgeführt.

F2 - Aktualisierung der Aufgabenzustände

Die Aktualisierung der Aufgabenzustände ist ein wesentlicher Bestandteil aller Algorithmen, die anhand des Schemas implementiert werden. Sie dient der Erkennung, welche dynamische Ereignisse Auswirkung auf die jeweilige Planung der Agenten haben. Die Funktion 'Update_MAG' fordert alle Agenten auf, den Bearbeitungsstatus der zugewiesen Aufgaben zurückzusenden. Nach Erhalt aller Antworten werden diese ausgewertet und Aufgaben gegebenenfalls erneut als unbearbeitet, bzw. als nicht zugewiesen markiert.

Funktionsrumpf 2. (F2)

```
Update_MAG
{
    foreach(Task in MAG)
        Status := Get_Status(Task, Aufgabe.Receiver);
        if (Task.Receiver <> null)
            Update_MAG_Status(Task, Status);
}
```

Terminierung Die Anzahl der for-Schleifen ist durch die endliche Anzahl von Aufgaben einer MAG begrenzt. Bei Anfragen eines Agenten an einen anderen Agenten kann es zu einem Deadlock kommen, falls ein Agent nicht antwortet. Um dies zu vermeiden, werden in der allgemeinen Kommunikationsinfrastruktur eines Multiagenten-Systems Timeout-Mechanismen standardmäßig eingesetzt. Die Anfrage wird nach einem definierten Zeitintervall abgebrochen, falls nicht alle Agenten zuvor geantwortet haben. Als Resultat erhält der anfragende Agent die bis zu diesem Zeitpunkt eingegangen Antworten. Durch diese Vorgehensweisen, können Blockaden durch das Kommunikationsfehlverhalten der Agenten aufgelöst werden. Die Funktion *Get_ Status(Task, Receiver)* fordert vom Empfänger den aktuellen Status einer Aufgabe an. Aufgrund des

Timeout-Mechanimus terminiert diese Funktion nach endlicher Zeit. Die Funktion *Update_MAG_Status(Task, Status)* ersetzt die bisherigen Task-Attribute durch neue Werte, basierend auf dem aktuellen Status der Aufgabe. Die Anzahl von Attributen ist begrenzt, weshalb diese Funktion terminiert. Jeder Schritt der Funktion F2 terminiert, so terminiert auch die Funktion F2 selbst.

Laufzeit Zur Ausführung von F2 wird der Status aller n Aufgaben abgefragt, was $O(n)$ Schritte benötigt (mit n der Anzahl aller Aufgaben der gegebenen MAG). Der Zeitbedarf der *Preperation* Phase 1a) beträgt $O(n)$.

Agentenselektion

Der *Random*$^-$ DHF-S-Ansatz setzt einen Randomisierer zur Auswahl von Agenten in der Simulationsphase ein. Der daraus resultierenden Aufbau der Funktionen 'SelectAgent_MaxValue()' und 'SelectAgent_MinValue()' setzt sich wie folgt zusammen:

Auswahl von Agenten mit maximalem Nutzen (F3) Die Funktion 'SelectAgent_MaxValue($Agentlist|_{MAG}$)' wählt zufällig eine Aufgabe aus den noch zu bearbeitenden / offenen Aufgaben der MAG aus und wählt anschließend zufällig einen Agenten aus. Die Kosten, die der Agent durch Bearbeitung dieser Aufgabe erzeugt, kann nicht abgeschätzt werden, da kein Wissen über die Fähigkeiten des Agenten existiert. Die Kosten werden daher auf Null gesetzt und für die weiteren Betrachtungen nicht berücksichtigt.

Funktionsrumpf 3. (F3)

```
SelectAgent_MaxValue(Agentlist|MAG)
{
    Task := null;
    Agent := null
    MAG' := MAG;
    while(Task=null AND MAG<>∅)
        Task := Random(MAG');
        if (Task.Receiver <> null)
            Task := null;
            MAG' := MAG' - Task;
    if (Task<>null)
        Agent = Random(Agentlist|MAG)
    return {Agent,Task,0}
}
```

Terminierung Die Funktion *Random()* greift bei herkömmlichen Programmiersprachen auf eine vom Betriebssystem bereitgestellte Funktion zurück, die einen pseudo-zufälligen Wert berechnet und zurückgibt. Hier wird davon ausgegangen, dass solche Basisfunktionen in konstanter Zeit $O(1)$ ausgeführt werden und immer terminieren. Die Funktion *Random(List)* gibt als Ergebnis das Element der Liste mit dem zufällig ermittelten Index zurück. Die Anzahl der Durchläufe der while-Schleife ist durch die Anzahl der Aufgaben der MAG begrenzt, denn mit jedem Durchgang wird MAG' um ein Element verkleinert, falls

die ausgewählte Aufgabe bereits einem Agenten zugewiesen ist. Ist die Aufgabe keinem Agenten zugewiesen, oder ist MAG' leer, bricht die while-Schleife ab. Die Funktion F3 terminiert somit.

Laufzeit Bis eine noch nicht zugewiesene Aufgabe gefunden wird, müssen im ungünstigsten Fall alle Aufgaben der MAG analysiert werden. Die Analyse geschieht in konstanter Zeit, so dass für die while-Schleife und somit die komplette Funktion $O(n)$ Zeit benötigt wird.

Auswahl von Agenten mit minimalem Nutzen (F4) Die Funktion 'SelectAgent_MinValue($Agentlist|_{MAG}$)' wählt zufällig aus der Aufgabenliste eine Aufgabe aus, der bereits ein Agent zugewiesen wurde. Analog F3 kann keine Nutzenabschätzung der Entnahme des Agenten aus dem Holon für das Holon gegeben werden, so dass die Kosten auf Null gesetzt werden. So besteht das zurückgegebene 3-Tupel aus dem Agenten, der die ausgewählte Aufgabe bearbeitet, der ausgewählten Aufgabe und den Kosten Null.

Funktionsrumpf 4. (F4)

```
SelectAgent_MinValue(Agentlist|_MAG)
{
    Task := null;
    Agent := null
    MAG' := MAG;
    while(Task=null AND MAG<>∅)
        Task := Random(MAG');
        if (Task.Receiver = null)
            Task := null;
            MAG' := MAG' - Task;
    if (Task<>null)
        Agent = Task.Receiver
    return {Agent,Task,0}
}
```

Terminierung Die Funktion F4 terminiert, wegen gleicher Struktur wie F3.

Laufzeit Analog der Funktion F3 benötigt die Ausführung dieser Funktion F4 $O(n)$ Zeit.

F5 - Nutzenbestimmung eines Holons

Mit Hilfe der Funktionen 'Value($HypHolon|_{MAG}$)' und 'Value('$Actionlist|_{MAG}$)' wird die Anzahl der noch offenen Aufgaben einer MAG bestimmt und als Wert zurückgeliefert.

Funktionsrumpf 5. (F5)

```
Value(HypHolon|_MAG) {
    i:=0;
    foreach(Task in MAG)
        if(Task.Receiver <> null)
            i++;
    return i;
}
```

Terminierung Die for-Schleife ist durch die Anzahl der Aufgaben einer MAG begrenzt, weshalb die Funktion F5 terminiert.

Laufzeit Die Anzahl von Aufgaben einer MAG dominiert die Laufzeit dieser Funktion. F5 kann daher in $O(n)$ ausgeführt werden.

F6 - Bilaterale Verhandlungen

Für die Durchführung von Verhandlungen wird ein Contract-Net Protokoll eingesetzt, bei dem lediglich ein *Bieter* vom *Manager* existiert.

Funktionsrumpf 6. (F6)

```
BilateralNegotiation(Agent, Task, Costs)
{
    Bid := CN_Announce(Agent, Task);
    Wait(Bid);
    if (Bid.Costs <= Costs)
        CN_Answer(Agent,Task,grand);
        Success := true;
    else
        CN_Answer(Agent,Task,refuse);
        Success := false;
    return Success;
}
```

Terminierung Die Funktion *CN_ Announce(Agent, Task)* schreibt eine Menge von Agenten[2] an, mit der Aufforderung ein Gebot für eine Aufgabe abzugeben. Nach der Ausschreibung ist die Funktion beendet. Anschließend wird auf die zurückgesendeten Gebote gewartet, bis alle Gebote eingegangen sind, oder der Timeout-Mechanismus der *Wait()*-Funktion das Warten beendet. *CN_ Answer(Agent,Task,Result)* benachrichtig einen Agenten über den Erfolg seines Gebots und wird beendet nachdem die Nachricht abgeschickt wurde. Somit terminiert die gesamte Funktion F6.

Laufzeit Die Funktion F6 benötigt konstante Laufzeit $O(1)$, da immer nur ein Agent angeschrieben wird.

F7 - Agentenbewertung

Der Funktionsrumpf ist leer, da die Agenten kein Wissen bezüglich ihres Verhandlungspartners besitzen.

Funktionsrumpf 7. (F7)

```
Evaluate(Agentlist|MAG)
{
    empty
}
```

Terminierung Die Funktion terminiert direkt nach dem Aufruf.

[2] Hier ist die Menge einelementig.

Laufzeit Auf die Evaluation der Agent wird in diesem Algorithmus verzichtet. Die Laufzeit für die Evaluierung beträgt somit $O(1)$.

Terminierung

Für den Beweis der Terminierung des vollständigen Algorithmus wurde eben gezeigt, dass die Funktionen F1 bis F7 terminieren. Da zuvor gezeigt wurde, dass das DHF-S Schema unter der Annahme der Terminierung der Funktion F1 bis F7 terminiert, folgt die Terminierung des gesamten Algorithmus.

□

Gesamtlaufzeit

Exakte Angaben über die Laufzeit des Algorithmus können nicht gegeben werden, da die Ausführung von den dynamischen Einflüssen der Umgebung abhängig ist. In den folgenden Laufzeitanalysen dieser Arbeit wird daher der durchschnittlich zu erwartende Zeitbedarf ermittelt. Um den dynamischen Effekten entgegen zu wirken, ist das Schema so ausgelegt, dass es zum einen ständig läuft und dabei die Holonenstruktur den aktuellen Einflüssen anpasst. Zum anderen dient die Simulationsphase auch der schrittweisen Optimierung der Holonenstruktur, in Anlehnung an das *simulated trading* Optimierungsverfahren[3]. Der Zeitbedarf der *Initialisierungsphase* 1a) beträgt $O(n)$ und der *Simulation* Phase 1b) beträgt $O(n)$ für einen Simulationslauf, da die Anzahl der Simulationsrunden des Algorithmus durch die Konstante c (*MaxSimulationRounds* [4]) begrenzt ist. Die Länge der *for*-Schleife in Phase 2 ist durch die Anzahl von Operationen in der *BestActionList* beschränkt. Eine obere Schranke für die Größe dieser Liste ist durch die Konstante c (*MaxSimulationRounds*) gegeben. Scheitert eine Verhandlung, wird diese Phase abgebrochen, wodurch sich die Anzahl der *for*-Schleifen reduziert. Ebenso reduziert sich diese Anzahl, falls unvorhergesehene Ereignisse eintreten, die sich auf die Verhandlungsphase auswirken. Die Konstante c ist dabei eine obere Schranke für die Anzahl durchgeführter Verhandlungen. Bei den Verhandlungen muss zwischen zwei Fällen unterschieden werden:

1. *Einfügen eines Agenten in die Holonenstruktur.* Die Laufzeit ist von dem zugrundeliegenden Verhandlungsprotokoll abhängig. In diesem Algorithmus wird ein Contract-Net Protokoll mit nur einem Bieter ausgeführt, was in $O(1)$ realisiert werden kann.

2. *Entbinden eines Agenten von einer Aufgabe.* Je nachdem, ob eine Bestätigung der Entbindung erforderlich ist, sind für diesen Vorgang ein bis zwei Kommunikationsschritte notwendig, so dass diese Aktion in konstanter Zeit $O(1)$ ausgeführt werden kann.

[3] Je mehr Zeit zur Verfügung steht, desto stärker kann die Holonenstruktur optimiert werden.

[4] Die maximale Anzahl von Simulationsrunden wird zu Beginn des Algorithmus einmalig festgelegt und ist für die folgenden Durchläufe konstant.

Es ergibt sich eine Gesamtlaufzeit für die Verhandlungsphase im ungünstigsten Fall von $O(c \cdot c_{Verh}) = O(1)$. Aufgrund des Verzichts einer Wissensbasis findet keine Evaluation in Phase 3 statt, so dass diese Phase in $O(1)$ beendet wird. Die Laufzeit des Schemas wird im Wesentlichen von der Simulationsphase dominiert. Im günstigsten Fall benötigt das Verfahren eine Initialisierungsrunde, um die beste Holonenstruktur zu finden[5]. Die Gesamtlaufzeit beträgt:

$$O_{Initialisierung}(n) = \begin{cases} 1 \cdot \left(\begin{array}{l} O_{Update\_Agent\_Information()}(1) + \\ O_{Update\_MAG()}(n) \end{array} \right) \\ + \\ c \cdot \left(\begin{array}{l} O_{SelectAgent\_MaxValue()}(n) + \\ O_{SelectAgent\_MinValue()}(n) + \\ O_{Value()}(n) \end{array} \right) \end{cases}$$

$$O_{Verhandlung}(1) = O_{BilateralNegotiation()}(1)$$

$$O_{Evaluierung}(1) = O_{Evaluate()}(1)$$

$$\Rightarrow O_{Gesamt}(n) = O_{Initialisierung}(n) + O_{Verhandlung}(1) + O_{Evaluierung}(1)$$

Dies gilt jedoch nur für den günstigsten Fall, falls nur eine Simulationsrunde notwendig ist, eine vollständige Lösung zu finden, in der jeder Aufgabe ein Agent zugewiesen wird. Die Gesamtlaufzeit hängt jedoch von zwei Faktoren ab:

1. dem durchschnittlichen Füllgrad[6] einer MAG nach Beendigung einer Simulationsrunde,
2. der Wahrscheinlichkeit für den vorzeitigen Abbruch der Verhandlungsphase aufgrund falsch ausgewählter Agenten in der Simulationsphase, oder einer Störungen der Umwelt.

Über den zweiten Faktor können keine direkten Aussage gemacht werden, weshalb das Verhalten der Algorithmen anhand von Testläufen analysiert werden soll. Entscheidend für die tatsächliche Laufzeit ist, wie schnell eine vollständige Verteilung der Aufgaben auf die Agenten gefunden werden kann. Mit Hilfe von stochastischen Verfahren kann ein Maß dafür angegeben werden, im welchen Verhältnis die Anzahl von Verhandlungen zwischen den unterschiedlichen Algorithmen steht. Dies führt zu folgender Betrachtung:

Die Randomisierung während der Simulationsphase kann als Bernoulli-Experiment betrachtet werden (siehe Anhang B.9). Die ständige Wiederholung

[5] Dies ist möglich, da das Verfahren randomisiert eine Holonenstruktur aufbaut. Enthält zudem die Wissensbasis in diesem Moment sichere Informationen über die am geeignetsten Agenten, kann innerhalb einer Initialisierungsrunde die optimale Holonenstruktur gefunden werden.

[6] Der Füllgrad beschreibt die Anzahl von Aufgaben der MAG, denen in der Simulationsphase ein Agent zugewiesen wurde.

des Algorithmus entspricht dann einer Bernoulli-Kette (wiederholtes Werfen einer Münze). Hierbei ergibt sich die Fragstellung, wie viele Runden des Algorithmus notwendig sind, um mit einer gewissen (z.B. 99%igen) Wahrscheinlichkeit allen Aufgaben einer MAG je einen Agenten zu zuweisen. Eine Aufgabe wird zufällig mit der Wahrscheinlichkeit $p_1 = \frac{1}{n}$ ausgewählt. Anschließend wird ein Agent mit einer Trefferwahrscheinlichkeit vom $p_2 = \frac{1}{m}$ bestimmt, der in der Lage ist die Aufgabe am besten zu lösen. Zur Bestimmung der Wahrscheinlichkeit, welcher Füllgrad nach einem Durchlauf erreicht werden kann, ist nur der 'add'-Operator von Bedeutung, der mit einer zuvor festgelegten Wahrscheinlichkeit p_3 eintritt. Ein Bernoulli-Experiment hat die Wahrscheinlichkeit $p = \frac{1}{nm} \cdot p_3$. Eine Bernoulli-Kette liefert dann die Anzahl von Runden, um eine vorgegebene Trefferwahrscheinlichkeit (jeder Aufgabe ist genau einem Agenten zugeordnet) zu erreichen.

$$P(\{\omega\}) := p^k(1-p)^{n-k}$$

mit ω ist ein n-Tupel mit genau k Einsen. Hier wird die Länge n des Tupels ω gesucht, bei dem das Tupel mindestens eine Eins enthält und die Wahrscheinlichkeit für dieses Ereignis bei 99% liegt. Für die Berechnung von n betrachtet man das Gegenereignis. Sei ω' ein n-Tupel mit keiner Eins. So folgt:

$$P(\{\omega\}) := 1 - P(\{\omega'\}) = 1 - (1-p)^n > 0.99$$

$$0.01 > (1-p)^n$$

$$lg(0.01) > n \cdot lg(1-p)$$

$$\text{da } 0 \geq p \geq 1 \Rightarrow lg(1-p) < 0$$

$$\frac{lg(0.01)}{lg(1-p)} < n$$

Die Anzahl der Durchläufe, bis eine vollständige Zuweisung der MAG möglich ist, beträgt im Allgemeinen für das Schema $n > \frac{lg(0.01)}{lg(1-p)}$. p ist also umgekehrt abhängig zu n, so dass $p = p_1 \cdot p_2 \cdot p_3 = \frac{1}{nm} \cdot p_3$ möglichst groß sein soll um n klein zu halten. p_1 und p_3 sind fest vorgegeben und ausschließlich p_2 kann durch die Wahl des Selektionsverfahrens der Funktionen F3 und F4 beeinflusst werden. Durch den jeweiligen Algorithmus wird schließlich p bestimmt. Für den *Random*$^-$ DHF-S Algorithmus ist $p_2 = \frac{1}{m}$. Die Zahl n bietet so auch ein Maß für die Anzahl von Verhandlungen. Je kleiner n desto geringer die Anzahl von notwendigen realen Verhandlungen, da n die zu erwartende Anzahl von hypothetischen Holonen wiederspiegelt und somit ein Richtwert für die Anzahl von Verhandlungen ist.

7.3.2 Algorithmus 2 - *Random*$^+$ DHF-S

Die Wahrscheinlichkeit, dass Verhandlungen scheitern, steigt durch die zufällige Wahl von Agenten, da auch Agenten ausgewählt werden, die aufgrund ihrer Eigenschaften nicht in der Lage sind, die Bedingungen der Aufgaben zu erfüllen. Es wird dabei angenommen, dass trotz der steigenden Verhandlungskosten, ab einer bestimmten Größe der Agentengesellschaft die Kosten für die Verwaltung der Agenteninformationen größer sind als die Ersparnisse durch die Reduktion der durchgeführten Verhandlungen.

	Zeitpunkt t_0	...	Zeitpunkt t_n
Agent A_1	$\vec{A}_1(t_0)$		$\vec{A}_1(t_n)$
$\vdots$			
Agent A_k	$\vec{A}_k(t_0)$		$\vec{A}_k(t_n)$

Tabelle 7.1: Agenten-Informationsverwaltung in der Tabelle *AgentEval*

F1 - Aktualisierung der Agenteninformationen

Die Funktion 'Update_Agent_Information(MAG)' dient der Aktualisierung der Informationen über die relevanten Agenten hinsichtlich einer gegebenen Mehrschrittaufgabe (MAG). Dazu wird vom WUA eine aktuelle Liste der Agentenvektoren angefordert. Die gesammelten Vektoren der Agenten werden in einer speziellen Tabelle *AgentEval* chronologisch abgelegt. So entsteht eine Matrix, in der jede Zeile für einen Agenten steht und jede Spalte den Zeitpunkt entspricht, an dem die Daten gesammelt wurden (siehe Tabelle 7.1). Diese Tabelle dient dann als Grundlage für die Berechnung der gewichteten Mittel mit Glättung für jeden Agenten über alle gespeicherten Zeitpunkte. Entsprechend dem stochastischen Lernverfahren, das in Kapitel 6.4.2 entwickelt wurde, kann zu jedem Attribut eines Agentenvektors die Varianz berechnet werden, die für die Berechnung notwendig ist. Für die Anzahl der gespeicherten Zeitpunkte kann wiederum eine obere Schranke angegeben werden, um den Berechnungsaufwand zu beschränken. Eine solche Begrenzung ist insbesondere sinnvoll, da die Gewichtung von Informationen mit der Zeit exponentiell mit $e^{-(t_r - t_l)}$ abnimmt. Die Gewichtung alter Daten sinkt gegenüber neuer Daten, so dass sie nach einem definierten Zeitraum aus der Tabelle gelöscht werden können, ohne das Ergebnis der Berechnung zu verändern. Nachdem die Berechnungen abgeschlossen sind, hält der Agent in der Liste 'Agentlist' zu jedem Agenten einen Vektor, der ihn beschreibt. Diese Liste ist nach den Aufgabenklassen der MAG sortiert und dient als Grundlage für weitere Anfragen an die Agentenwissensbasis bezüglich der Bewertung anderer Agenten.

Funktionsrumpf 8. (F1)

```
Update_Agent_Information(MAG)
{
    NewAgentList := Get_Agent_Info_From_WUA(MAG);
    if (NewAgentList <> null)
        AgentEval := AgentEval + NewAgentList;
        if (AgentEval.Lenght > MaxLenght)
            Remove_Oldest_Entry(AgentEval);
        Agentlist := ComputeVariance(AgentEval, MAG);
}
```

Terminierung Die Funktion *Get_Agent_Info_From_WUA(MAG)* fordert vom WUA eine aktuelle Bewertung aller Agenten an, die für die Bearbeitung der MAG notwendig sind. Wie für alle Agentenkommunikationen existiert auch

hier ein Timeout-Mechanismus, so dass diese Funktion terminiert. Die Funktion *Remove_ Oldest_ Entry(AgentEval)* entfernt den ältesten Eintrag einer Liste und ist anschließend beendet. Die Funktion *ComputeVariance(AgentEval, MAG)* berechnet entsprechend dem Verfahren aus Kapitel 6.4.2 ein gewichtetes Mittel. Die Anzahl von Eingabevektoren ist begrenzt und im Kern der Funktion werden endliche Summationen und einfache Rechenoperationen ausgeführt, wodurch diese Funktion terminiert, so dass die Funktion F1 insgesamt terminiert.

Laufzeit Die Aktualisierung der Agenteninformation bezüglich einer MAG mit n Aufgaben benötigt maximal m Schritte pro Aufgabentyp, mit m der Anzahl aller Agenten. Im ungünstigsten Fall enthält eine MAG mit n Aufgaben n verschiedene Aufgabentypen, denen wiederum jeweils alle Agenten zugeordnet sein können. Die Berechnung des gewichteten Mittels mit Glättung geschieht in drei Schritten:

1. Bestimmung des gemittelten Agentenvektors mit l Attributen zum Zeitpunkt t über den Aussagen der m Agenten $\Rightarrow l \cdot m$ Schritte.
2. Berechnung der Varianz aller l Attribute des gemittelten Agentenvektors zum Zeitpunkt $t \Rightarrow l \cdot m$ Schritte.
3. Ermittlung des geglätteten Vektors über die Anzahl k der Aufzeichnungszeitpunkte $\Rightarrow l \cdot k$ Schritte.

Zusammenfassend führt dies zu $2m+k$ Schritten pro Attribut und $l \cdot (2m+k)$ pro Aufgabentyp. Diese Werte können als konstant für den Algorithmus betrachtet werden, da l und k zwar abhängig von dem Anwendungsfall, jedoch für ein Szenario fest sind. Es ergibt sich daraus eine maximale Laufzeit von $O(nm)$ für n Aufgabentypen. Für die Aktualisierung der Aufgabenstati durch die Funktion F2 (analog dem ersten Algorithmus) werden zusätzlich n Abfragen benötigt. So ergibt sich ein Zeitbedarf der *Preperation* Phase von:

$$O(nm) + O(n) = O(nm + n) = O(n \cdot (m + 1)) = O(nm)$$

Agentenselektion

Für die Selektion von Agenten in der Simulationsphase werden zwei Funktionen eingesetzt, die zur Auswahl von Agenten dienen. Im $Random^+$ DHF-S-Algorithmus werden die Agenten zufällig ausgewählt. Durch diese Vorgehensweise werden die Berechnungskosten für den Algorithmus unter der Annahme gesenkt, dass die Kosten von zusätzlichen Verhandlungen (die aufgrund der zum Teil schlechteren Agentenwahl notwendig sind) geringer sind, als die Pflege- und Analysekosten einer Wissensbasis. Die Funktionen 'SelectAgent_MaxValue()' und 'SelectAgent_MinValue()' liefern nach Ausführung ein 3-Tupel als Ergebnis. Das Tupel enthält den ausgewählten Agenten A, bzw. eine leere Menge, falls die Auswahl eines Agenten nicht möglich ist, die dazugehörige Aufgabe AG, die dem Agenten zugewiesen wird bzw. entnommen und den Nutzen *cost*, der durch die Zuweisung, bzw. durch die Entnahme der Aufgabe AG dem Holon zugerechnet werden kann.

Auswahl von Agenten mit maximalem Nutzen (F3) Die Funktion 'SelectAgent_MaxValue($Agentlist|_{MAG}$)' dient der Auswahl eines Agenten mit maximalem Nutzen für das Holon für eine noch nicht zugewiesene Aufgabe aus der Aufgabenmenge der MAG. Beim $Random^+$ DHF-S-Ansatz wird darauf verzichtet, eine aufwendige Analyse durchzuführen. Stattdessen wird für jede offene Aufgabe ein Agent zufällig aus der Liste der aufgabenrelevanten Agenten ausgewählt. Bei dieser Vorgehensweise wird angenommen, dass

- bei einer hinreichend großen Anzahl von Agenten genügend Agenten existieren, die die entsprechende Aufgabe bearbeiten können,
- die Kollektion der Daten und deren Analyse mehr Zeit beansprucht, als der zusätzliche Zeitbedarf der höher zu erwartenden Anzahl von Verhandlungen.

Es wird kein Agent ausgewählt, wenn bereits alle Aufgaben der MAG Agent zugewiesen sind und somit gilt: $A = \emptyset$. Die Existenz einer Wissensbasis ermöglicht im Gegensatz zum ersten Algorithmus, dass die realen Kosten abgeschätzt werden können, was zu einer Verbesserung der hypothetischen Holone und zu einer Reduzierung der Anzahl von Verhandlungen führt.

Terminierung Die Funktion ist analog der Funktion F3 des ersten Algorithmus aufgebaut, mit dem Unterschied, dass die Funktion um eine weitere while-Schleife erweitert wurde. Die zusätzliche while-Schleife dient der Suche nach einem Agenten, der, basierend auf dem Wissen des Agenten, relevant für die Aufgabenbearbeitung ist. Für die Analyse, ob die Funktion F3 terminiert, muss daher nur die zweite while-Schleife untersucht werden. Für den Rest wurde bereits die Terminierung gezeigt. Die zweite while-Schleife wird beendet, falls ein Agent gefunden wird (Agent <> null), oder die Kopie der Agentlist leer ist. Die Funktion *Can_ Agent_ Perform_ Task(Agent, Task)* ermittelt für jedes Attribut, ob das Werteintervall des Agenten größer gleich dem der Aufgabe ist. Ist dies der Fall liefert die Funktion 'true' zurück, sonst 'false' und terminiert spätestens nach 'Anzahl der Attribute' Schritten. Insgesamt terminiert so die Funktion F3.

Laufzeit In der Funktion F3 wird im ersten Schritt ein Agent zufällig ausgewählt. Anschließend wird der Vektor des Agenten aus der Wissensbasis gesucht und geprüft, ob die Aufgabenausführung durch diesen Agenten möglich ist. Das Durchsuchen der Wissensbasis kann in logarithmischer Zeit $O(log(m))$ vollzogen werden und der Vergleich der Attribute bei konstanter Attributanzahl l in konstanter Zeit $O(l) = O(1)$, so dass die zweite while-Schleife maximal $O(m \cdot log(m))$ Zeit benötigt. Kombiniert mit der ersten while-Schleife, die maximal n Durchläufe macht, ergibt sich eine Laufzeit von $O(nm \cdot log(m))$.

Funktionsrumpf 9. (F3)

```
SelectAgent_MaxValue(Agentlist|_MAG)
{
    Task := null;
    Agent := null
    MAG' := MAG;
    AgentFound = false;
    while(Task=null AND MAG<>∅) % Searching for an unhandled task
        Task := Random(MAG');
        if (Task.Receiver <> null)
            Task := null;
            MAG' := MAG' - Task;
        if (Task<>null)
            Agentlist' := Agentlist;
            while(Agent=null AND Agentlist'<>∅) % Searching for a relevant agent
                Agent = Random(Agentlist|_MAG)
                if (NOT Can_Agent_Perform_Task(Agent, Task))
                    Agentlist' := Agentlist' - Agent;
                    Agent := null;
            if (Agent=null)
                Task:=null;
    return {Agent,Task,0}
}
```

Auswahl von Agenten mit minimalem Nutzen (F4) Ziel der Funktion 'SelectAgent_MinValue($HypHolon|_{MAG}$)' ist die Auswahl eines Agenten aus der Menge der bekannten Agenten, der bei den nachfolgenden Analysen aus dem Holon entfernt wird. Der ausgewählte Agent besitzt dabei bereits mindestens eine Aufgabe der MAG. Wird immer der Agent mit dem niedrigsten Nutzen ausgewählt, so besteht die Gefahr, dass auf der Suche nach bessern Lösungen der Algorithmus in lokalen Maxima 'festhängt' und bei erneuten Aufrufen dieser Funktion nur Agenten mit geringen Unterschieden in ihren Eigenschaften austauscht. Agenten mit einem hohen Nutzen verbleiben in der Holonenstruktur solange es Agenten mit einem niedrigeren Nutzen gibt und werden so nie ausgewählt. Zwar kann durch diese Vorgehensweise gewährleistet werden, dass eine neu gefundene Lösung nicht wesentlich schlechter als die vorherige ist, jedoch kann unter Umständen die optimale Holonenstruktur mit dem größtmöglichen Nutzen nie gefunden werden. Dieses Problem entsteht, da hoch bewertete Agenten unter Umständen niemals ausgetauscht werden. Um dies zu verhindern, wird in dieser Funktion zufällig ein Agent aus der Menge von Agenten ausgewählt, denen eine Aufgabe aus der MAG zugewiesen wurde. Durch Entnahme eines Agenten aus dem Holon entstehen umgekehrt Kosten für das Holon, die mit Hilfe des Wissens über die anderen Agenten abgeschätzt werden kann.

Terminierung Die Funktion F4 terminiert, da identisch mit Funktion F4 des ersten Algorithmus.

Laufzeit Die Funktion F4 benötigt $O(n)$ Zeit.

Funktionsrumpf 10. (F4)

```
SelectAgent_MinValue(Agentlist|MAG)
{
    Task := null;
    Agent := null
    MAG' := MAG;
    while(Task=null AND MAG<>∅)
        Task := Random(MAG');
        if (Task.Receiver = null)
            Task := null;
            MAG' := MAG' - Task;
    if (Task<>null)
        Agent = Task.Receiver
    return {Agent,Task,0}
}
```

F5 - Nutzenbestimmung eines Holons

Die Bewertung eines Holons wird anhand der beiden Funktionen 'Value($HypHolon|_{MAG}$)' und 'Value('$Actionlist|_{MAG}$)' durchgeführt. Die 'Value'-Funktion kann dabei zwei Typen von Eingabeparametern erhalten. Entweder erhält diese Funktion eine Holonenstruktur oder eine Actionlist, anhand der die entsprechende Holonenstruktur rekonstruiert werden kann, sodass das gleiche Bewertungsverfahren für beide Eingabetypen verwendet werden kann. Je nach Anwendungsszenario sind unterschiedliche Kostenfunktionen implementiert, die als Grundlage für diese Funktionen dienen. So wird für jede Aufgabe, unter Berücksichtigung der individuellen Eigenschaften des zugewiesenen Agenten, die Kosten bzw. der Nutzen ermittelt und über die Menge der Aufgaben der MAG aufsummiert. Die Wissensbasis des Holonenführers dient dabei als Basis für die Abschätzung der Agenteneigenschaften. Die Summe der Nutzenbewertungen aller Aufgaben einer MAG bildet so ein Maß für den Vergleich zweier Holonenstrukturen (Voraussetzung: alle Bewertungen beziehen sich auf die gleiche MAG). Eine Kostenfunktion dient der Berechnung der Kosten bzw. des Nutzens eines Agenten, wenn dieser eine Aufgabe zugewiesen bekommt. Zuvor ist es jedoch notwendig eine Normalisierung der bezüglich der Aufgabenstellung standardisierten und gewichteten Agentenvektoren durchzuführen, damit die Nutzen dieser Agenten bezüglich einer gegebenen Aufgabe vergleichbar sind. Die Kostenfunktion berechnet, basierend auf der Metrik, wie sie in Abschnitt 5.4.2 definiert ist, einen Wert, der mit den Nutzenwerten der anderen Agenten vergleichbar ist.

Funktionsrumpf 11. (F5)

```
Value(HypHolon|MAG)
{
    Value=0;
    foreach(Task in MAG)
        if(Task.Receiver <> null)
            Value := Value + Compute_Distance(Task, Task.Receiver);
    return Value;
}
```

Terminierung Die Funktion F5 terminiert, wenn die Funktion *Compute_ Distance(Task, Agent)* terminiert. Die Funktion Compute_Distance(Task, Agent) berechnet den Abstand zwischen dem idealen Agenten der übergebenen Aufgabe und dem angegebenen Agenten, basierend auf den Verfahren aus Abschnitt 5.4.2. Für eine endliche Anzahl von Attributen terminiert diese Funktion, da eine einfache Summation über den Attributwerten neben Grundrechenoperationen durchgeführt wird.

Laufzeit Für die Bewertung des Nutzens eines Agenten bezüglich des Holons wird eine Kostenfunktion eingesetzt, die eine gewichtetet Abstandsbestimmung durchführt, was in $O(l)$ pro Aufgabe der MAG durchgeführt werden kann. Für die Nutzenbestimmung aller Aufgaben der gegebenen MAG ergibt sich so eine Laufzeit für die Funktion F5 von $O(n \cdot l) = O(n)$. Dies führt zu einer maximalen Laufzeit für einen Simulationsdurchgang[7] von

$$c \cdot O(nm \cdot log(m)) + O(n) = O(nm \cdot log(m))$$

Insgesamt ergibt sich für die Initialisierungsphase eine Gesamtlaufzeit von:

$$O(nm) + O(nm \cdot log(m)) + O(n) = O(nm \cdot log(m))$$

F6 - Bilaterale Verhandlungen

'BilateralNegotiation($A, costs$)' beschreibt keine Funktion, sondern ein Protokoll das zwei Agenten ausführen, um eine Multiattributverhandlung durchzuführen. Die Schwierigkeit bei solchen Verhandlungen ist, dass über mehrere variable Parameter verhandelt werden muss. Jonker et al. beschreiben in [JT01] eine Möglichkeit der Realisierung einer Multiattribut-Verhandlung. Eine diesem Verfahren angelehnte Vorgehensweise ist folgende (siehe Abbildung 7.2):

1. Zwei Agenten A und B treten miteinander in Verhandlung. Zu Beginn legt jeder Agent seine individuelle untere Toleranzgrenze fest, bis zu welcher Abweichung vom eingegangenen und eigenen Gebot die Verhandlung erfolgreich beendet wird und die obere Toleranzgrenze, unterhalb dieser die Verhandlungen weitergeführt werden. Ein Überschreiten der oberen Toleranzgrenze führt zum erfolglosen Abbruch der Verhandlungen. Zusätzlich legt jeder Agent einen Zeitpunkt fest, wie lange die Verhandlungen geführt werden können. Wird dieser Zeitpunkt überschritten, bricht der entsprechende Agent die Verhandlungen erfolglos ab.

2. A ist Verhandlungsführer und eröffnet die Verhandlung mit einem Angebot bezüglich der zu verhandelnden Aufgabe. Der Vorschlag beschreibt die Ausprägungen der Attributwerte der Aufgabe.

3. Agent B berechnet ein eigenes Angebot und vergleicht dieses auf Ähnlichkeit (siehe Abschnitt 5.4.3) mit dem Angebot von A.

[7] c ist ein konstanter Wert für die Runden der Simulationsphase, da diese im Schema begrenzt sind.

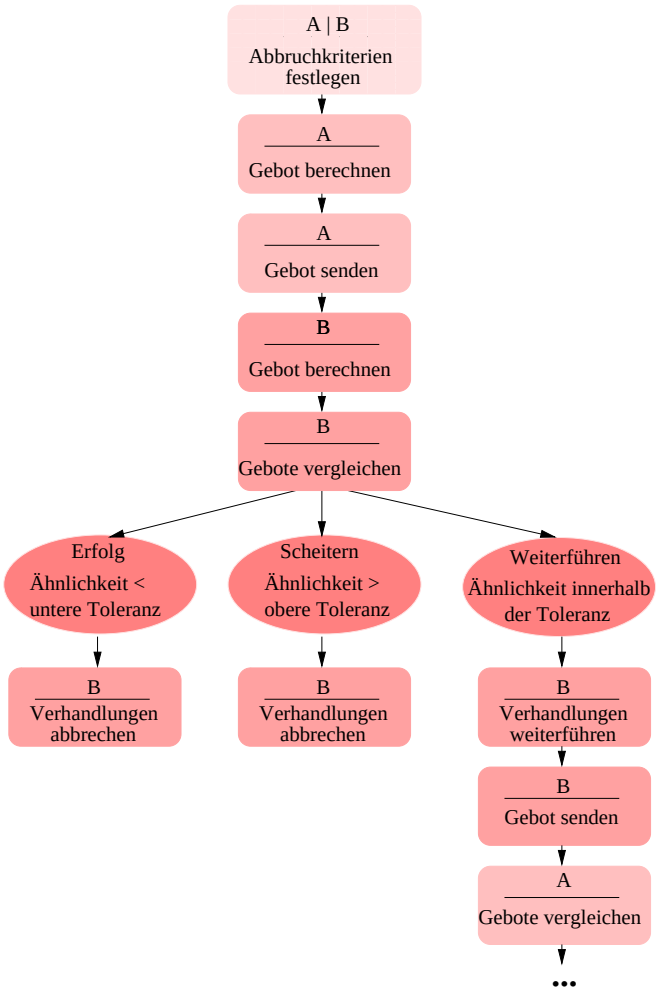

Abbildung 7.2: Bilaterales Verhandlungsprotokoll

4. Liegt der Ähnlichkeitsfaktor unterhalb der unteren Toleranzgrenze, so wird die Verhandlung erfolgreich beendet und das Angebot von A gilt als Vertrag.

5. Ist der Ähnlichkeitsfaktor größer als der Wert der oberen Verhandlungstoleranz, so scheitert die Verhandlung und wird abgebrochen.

6. Werden die Verhandlungen weitergeführt, berechnet der Agent ein neues Angebot, indem der Wert eines Attributs des eingegangenen Angebots durch den Wert seines Angebots ersetzt und diese Angebot an Agent A versendet wird.

7. A prüft in analoger Weise zu B die Ähnlichkeit seines Angebots zu dem von B und die Verhandlungen werden dann beendet oder durch ein erneutes Angebot weitergeführt.

Funktionsrumpf 12. (F6)

```
BilateralNegotiation(Agent, Task, Costs)
{
    Interval := Define_Continue_Criteria();
    Bid := null;
    Result := null;
    LoopCounter := 0;
    while(Result <> null AND LoopCounter<MaxNegotiationRounds)
        LoopCounter++;
        NewBid := Compute_Bid(Bid, Task);
        Bid := Send_Bid(NewBid, Agent);
        Wait(Bid);
        switch(Bid)
            case 'failed':
                Result := false;
            case 'accepted':
                Result := true;
            case 'negotiate':
                if (Compute_Distance(NewBid,Bid)> Interval.UpperBound)
                    % negotiation failed
                    Send_Bid('failed', Agent);
                    Result := false;
                if (Compute_Distance(NewBid,Bid)< Interval.LowerBound)
                    % negotiation successful
                    Send_Bid('accepted', Agent);
                    Result := true;
                if (Compute_Distance(NewBid,Bid)> Interval.LowerBound AND
                    Compute_Distance(NewBid,Bid)< Interval.UpperBound)
                    % negotiation goes on
                    Result := null;
    if (Result=null)
        Result = false;
    return Result;
}
```

Terminierung Aufgrund des Rundenbegrenzers 'MaxNegotiationRounds' und der Tatsache, dass die Funktion *Wait(Bid)* einen Timeout-Mechanismus enthält, kann der Ablauf nicht blockiert werden, und die Verhandlungen enden spätestens nach einer zuvor festgelegten Rundenanzahl. Die Funktion *Compute_Bid(Bid, Task)* erhält als Eingabe ein Gebot und die Aufgabe und berechnet ein neues Gebot, dessen Abstand kleiner ist, als der Abstand zwischen Eingabegebot und Aufgabe. Dazu werden alle bis auf einen Parameter des Eingabegebots für das neue Gebot übernommen und der freie Parameter dem entsprechen Parameterwert der Aufgabe angeglichen. Das Ergebnisangebot wird anschließend zurückgegeben und die Funktion wird beendet. Die Funktion *Compute_Distance(NewBid,Bid)* berechnet nach dem gleichen Verfahren, wie bei vorheriger Funktion, den Abstand zwischen zwei Vektoren. So terminiert auch diese Funktion und im Allgemeinen auch die Funktion F6.

Laufzeit Bei den Verhandlungen muss zwischen zwei Fällen unterschieden werden:

1. *Einfügen eines Agenten in die Holonenstruktur.* Die Laufzeit ist von dem zugrundeliegenden Verhandlungsprotokoll abhängig. In diesem Algorithmus werden bilaterale Verhandlungen eingesetzt, die durch einen konstanten Wert für die maximale Anzahl von Verhandlungsrunden begrenzt ist. Die beiden Funktionen *Compute_ Bid(Bid, Task)* und *Compute_ Distance(NewBid,Bid)* können in konstanter Zeit ausgeführt werden, da sie eine Aufsummierung über die Anzahl der Attribute durchführen. Beide Funktionen setzen eine Abstandsberechnung ein, die auf einer festen Rechenvorschrift basieren und in konstanter Zeit ausgeführt werden können. Die Anzahl der Attribute ist ebenfalls konstant, so dass eine while-Schleife in konstanter Zeit ausgeführt werden kann. Durch den konstanten Rundenbegrenzer kann die Verhandlung so in konstanter Zeit $O(1)$ ausgeführt werden.

2. *Entbinden eines Agenten von einer Aufgabe.* Je nachdem ob eine Bestätigung der Entbindung erforderlich ist, sind für diesen Vorgang ein bis zwei Kommunikationsschritte notwendig, so dass diese Aktion in konstanter Zeit $O(1)$ ausgeführt werden kann.

Insgesamt ergibt sich eine konstante Gesamtlaufzeit für die Verhandlungsphase im ungünstigsten Fall von $O(c \cdot c_{Verh}) = O(1)$.

F7 - Agentenbewertung

Neue Informationen über Agenten werden der Wissensbasis durch den Aufruf der Funktion 'Evaluate($Agentlist|_{MAG}$)' zugefügt. Der Ablauf dieser Funktion geschieht analog der 'Update_Agent_Information(MAG)' Funktion, jedoch mit umgekehrtem Informationsfluss. Die in der Verhandlungsphase gewonnenen Informationen über Agenten werden durch die Evaluationsfunktion dem WUA weitergegeben und so für die Bewertung der Agentengesellschaft nutzbar gemacht. Bei erfolgreich abgeschlossenen Verhandlungen werden die Attributwerte[8] der beteiligten Agenten in die Wissensbasis aufgenommen und entsprechend dem Verfahren der Funktion 'Update_Agent_Information(MAG)' bewertet. Die Werte von gescheiterten Verhandlungen werden nicht berücksichtigt, da sie keine abgesicherten Informationen enthalten. So werden die Fähigkeiten eines Agenten eher unterschätzt, jedoch niemals überschätzt.

Terminierung Der Kern dieser Funktion gleicht der Funktion F2 dieses Algorithmus und terminiert unter der Voraussetzung, dass die beiden Funktionen *Get_ Results_ Negotiation(MAG)* und *Send_ Agentlist_ to_ WUA(Agentlist)* terminieren. Die erste der beiden Funktionen erstellt für jede zugewiesene Aufgabe einen Agentenvektor, dessen Attributwerte gleich den Aufgabenwerten sind. Die Anzahl der Attribute und der Aufgaben einer MAG sind begrenzt, so dass die Funktion terminiert. Die zweite Funktion dient der Übertragung der aktualisierten Agentenliste an den WUA und terminiert ebenfalls. Somit terminiert die komplette Funktion F7.

[8] Das letzte Gebot des Agenten wird als Vorlage für die Attributwerte verwendet. Dieses Gebot spiegelt einen Ausschnitt der Attributwerte des Agenten wieder.

Funktionsrumpf 13. (F7)

```
Evaluate(Agentlist|MAG)
{
    NewAgentList := Get_Results_Negotiation(MAG);
    if (NewAgentList <> null)
        AgentEval := AgentEval + NewAgentList;
        if (AgentEval.Lenght > MaxLenght)
            Remove_Oldest_Entry(AgentEval);
        Agentlist := ComputeVariance(AgentEval, MAG);
    Send_Agentlist_to_WUA(Agentlist);
}
```

Laufzeit Für die Bewertungsfunktion F7 wird das gleiche Verfahren wie in Funktion F1 eingesetzt, wodurch die Laufzeiten dieser beiden Funktionen identisch sind mit $O(nm)$. Die zusätzliche Funktion *Get_ Results_ Negotiation(MAG)* hat mit einer Laufzeit von $O(n)$ keinen weiteren Einfluss auf die Gesamtlaufzeit von F7.

Gesamtlaufzeit

Die Gesamtlaufzeit beträgt:

$$O_{Initialisierung}(mn{\cdot}log(m)) = \left\{ \begin{array}{l} 1 \cdot \left(\begin{array}{l} O_{Update\_Agent\_Information()}(nm)+ \\ O_{Update\_MAG()}(n) \end{array} \right) \\ + \\ c \cdot \left(\begin{array}{l} O_{SelectAgent\_MaxValue()}(mn \cdot log(m))+ \\ O_{SelectAgent\_MinValue()}(n)+ \\ O_{Value()}(m) \end{array} \right) \end{array} \right.$$

$$O_{Verhandlung}(1) = O_{BilateralNegotiation()}(1)$$

$$O_{Evaluierung}(nm) = O_{Evaluate()}(nm)$$

$$\begin{aligned} &O_{Initialisierung}(mn \cdot log(m)) + O_{Verhandlung}(1) + O_{Evaluierung}(nm) = \\ &= O_{Gesamt}(nm \cdot (log(m) + 1)) = O_{Gesamt}(nm \cdot log(m)) \end{aligned}$$

Die Agenten werden in diesem Algorithmus zwar zufällig ausgewählt, jedoch anschließend auf ihre Tauglichkeit überprüft. Es wird hierbei solange ein Agent zufällig ausgewählt, bis er für die Aufgabe eingesetzt werden kann. So beschränkt sich die Trefferwahrscheinlichkeit, den besten Agenten für die Aufgabe zu finden, auf die Gruppe von Agenten, die relevant für eine Aufgabe sind (Agenten, die einer Dienstklasse zugewiesen sind, mit der die Aufgabe bearbeitet werden kann). Somit gilt:

$$p_2 = \frac{\#aufgabenrelevanteAgenten}{m}$$

7.3.3 Algorithmus 3 - wissensbasierter DHF-S

Der dritte Algorithmus, der auf dem DHF-S Schema basierend entwickelt wurde, ist eine Weiterentwicklung des zweiten Algorithmus und gleicht in der Realisierung der meisten Funktionen dem des zweiten Algorithmus. Unterschiede existieren in der Vorgehensweise zur Selektion von Agenten, die in das Holon aufgenommen, bzw. aus dem Holon entfernt werden sollen. Diese Auswahlfunktionen werden durch komplexere Funktionen ersetzt. Während bei den ersten beiden Algorithmen davon ausgegangen wird, dass die Kosten für die Ausführung der Verhandlungen geringer sind, als für die Pflege und Analyse der Agentenwissensbasis, wird im dritten Algorithmus davon ausgegangen, dass die Ersparnis durch Vermeidung von Verhandlungen größer ist, als die Kosten zum Aufbau einer Wissensbasis. Durch den Einsatz einer Wissensbasis wird es möglich eine genauere Vorhersage der Agentenattribute durchführen zu können, wodurch die Anzahl und somit die Kosten der Verhandlungen im Ganzen reduziert werden können.

Agentenselektion

Die Wahl der Agenten in der Simulationsphase wirkt sich auf die Qualität des gesamten Algorithmus aus. Enthalten die hypothetischen Holone häufig ungeeignete Agenten, so scheitern in der zweiten Phase öfters die Verhandlungen, was wiederum zu weiteren Verhandlungsrunden führt. Gelingt es hingegen Agenten in der Simulationsphase zu finden, die bestmöglich dem Aufgabenprofile entsprechen (im Idealfall stimmen die Anforderungen der Aufgabe genau mit den Attributwerten eines Agenten überein), so steigt die Wahrscheinlichkeit, Verhandlungen erfolgreich durchführen zu können. Ob jedoch mit einer Auswahl von Agenten erfolgreiche Verhandlungen durchgeführt werden können, hängt neben den 'Select'-Funktionen von der Qualität der eigenen Wissensbasis und von der aktuellen Situation der Agenten ab. Letzteres ist vom Holonenführer nur dann abschätzbar, falls die Agenten ihm ihre Pläne offen legen, was jedoch mit den in dieser Arbeit getroffenen Vorgaben an die Umwelt nicht möglich ist. Die Qualität der Wissensbasis steigt mit der Anzahl der aufgezeichneten Daten über die relevanten Agenten und mit der Anzahl von Agenten, die dem Holonenführer bekannt sind. Wesentlich für den Erfolg sind jedoch nachfolgende Select-Funktionen. Die Funktionen 'SelectAgent_MaxValue()' und 'SelectAgent_MinValue()' geben analog dem zweiten Algorithmus nach Ausführung ein 3-Tupel, bestehend aus einem selektierten Agenten, einer Aufgabe und die Kosten der Ausführung dieser Aufgabe durch den selektierten Agenten, als Ergebnis zurück.

Auswahl von Agenten mit maximalem Nutzen (F3) Die Funktion 'SelectAgent_MaxValue($Agentlist|_{MAG}$)' dient der Auswahl eines Agenten mit maximalem Nutzen für das Holon für eine noch nicht zugewiesene Aufgabe aus der Aufgabenmenge der MAG. Im Gegensatz zu den *Random* DHF-S-Ansätzen werden die Agenten bezüglich der Aufgaben analysiert und die Agenten ausgewählt, die den größten Nutzen für das Holon besitzen. Beim Aufruf dieser

Funktion werden zu allen Aufgaben der MAG (denen noch keine Agenten zugewiesen sind) der Nutzen bezüglich aller relevanter Agenten bestimmt. Die Nutzenberechnung basiert auf einem zuvor festgelegten Maß, mit dem die Agenten untereinander vergleichbar sind. So kann zu jedem Agenten bezüglich einer Aufgabe dessen Distanz zum *idealen* Agenten bestimmt werden (siehe Abschnitt 5.4.2). Der ideale Agent ist ein hypothetischer Agent, der in allen Parametern der Aufgabe bestmöglich entspricht. Für jede Aufgabe wird der Agent mit dem geringsten Abstand zum idealen Agenten bestimmt. Nachdem für alle offenen Aufgaben je ein Agent ermittelt wurde, wird abschließend das Aufgaben-Agenten-Paar ausgewählt, dessen Nutzen insgesamt am größten ist. So werden zuerst die Aufgaben mit Agenten belegt, deren Nutzen für das Holon am größten ist. Dadurch stehen Agenten unter Umständen für weitere Aufgabenzuweisungen nicht mehr zur Verfügung. Die Qualität der Lösung hängt also wesentlich von der Zuteilungsreihenfolge der Aufgaben ab. Durch die Randomisierung der Aktionen in der Simulationsphase, in der zufällig entschieden wird, ob Agenten in das Holon aufgenommen bzw. entfernt werden, entwickeln sich unterschiedliche Reihenfolgen der Aufgabenzuteilungen, so dass das Problem nicht auftritt.

Bemerkung. Mit diesem Ansatz wird nicht versucht eine global- (Holon-) optimale Lösung zu berechnen. Zum einen ist die Berechnungskomplexität der optimalen Lösung bei einer großen Agentengesellschaft wesentlich höher. Zum anderen ändert sich die Umwelt ständig, wodurch die Realisierung der hypothetischen Holone nicht immer vollständig durchführbar ist, so dass weitere Simulationsphasen initiiert werden müssen.

Funktionsrumpf 14. (F3)

```
SelectAgent_MaxValue(Agentlist|_MAG)
{
    BestAgent := null;
    BestDistance := infinity;
    BestTask := null;
    foreach(Task in MAG)
        if (Task.Receiver = null)
            IdealAgent := Build_Ideal_Agent(Task);
            foreach(Agent in Agentlist)
                Distance := Compute_Distance(Agent, IdealAgent);
                if (Distance < BestDistance)
                    BestDistance := Distance;
                    BestAgent := Agent;
                    BestTask := Task;
    return {BestAgent, BestTask, BestDistance}
}
```

Terminierung Die Funktion F3 terminiert, das sie aus zwei geschachtelten Schleifen und der Funktion *Compute_ Distance(Agent, IdealAgent)* besteht, von der bekannt ist, dass sie terminiert. Die Funktion *Build_ Ideal_ Agent(Task)* erstellt einen Agentenvektor, der genau die Attributwerte der Aufgabe enthält und dessen Abstand zum Aufgabevektor daher gleich Null ist. Diese Funktion terminiert ebenfalls, da hierzu nur die Attributwerte kopiert werden und somit auch die Funktion F3.

Laufzeit In der Funktion F3 werden zu jeder unbesetzten Aufgabe der MAG alle Agenten auf ihren Nutzen hin untersucht. Dazu wird der Abstand zwischen Agenten und idealen Agenten bestimmt, was in $O(1)$ Schritten möglich ist. Die Bestimmung des Aufgaben-Agenten Tupels (Ergebnis der Funktion) benötigt so $O(nm)$ Zeit.

Auswahl von Agenten mit minimalem Nutzen (F4) Die Funktion 'SelectAgent_MinValue($Agentlist|_{MAG}$)' bildet analog der Funktion 'SelectAgent_MaxValue($Agentlist|_{MAG}$)' *ideale* Agenten zu jeder Aufgabe, die bereits einem Agenten zugewiesen wurde. Anschließend werden die Distanzen zwischen den idealen Agenten und den zugewiesen Agenten berechnet und das Aufgaben-Agenten-Kosten Tupel ermittelt, das die größte Distanz zwischen dem idealen Agenten und dem zugewiesenen Agenten besitzt. Abschließend liefert diese Funktion das Aufgaben-Agenten-Kosten Tupel als Ergebnis zurück.

Funktionsrumpf 15. (F4)

```
SelectAgent_MinValue(Agentlist|MAG)
{
    WorstAgent = null;
    WorstDistance = 0;
    WorstTask := null;
    foreach(Task in MAG)
        if (Task.Receiver <> null)
            IdealAgent := Build_Ideal_Agent(Task);
            Distance := Compute_Distance(Agent, IdealAgent);
            if (Distance > WorstDistance)
                WorstDistance := Distance;
                WorstAgent := Agent;
                WorstTask := Task;
    return {WorstAgent, WorstTask, WorstDistance}
}
```

Laufzeit Die Funktion F4 analysiert jede besetzte Aufgabe der MAG nach dem Agenten, mit dem geringsten Nutzen für das Holon. Dazu wird ebenfalls der Abstand zwischen idealem Agenten und dem zugewiesenen Agenten ermittelt, was in konstanter Zeit möglich ist. Insgesamt benötigt diese Funktion $O(n)$ Zeit. Der restliche Ablauf der Initialisierungsphase gleicht dem des vorherigen Algorithmus. Die Gesamtlaufzeit für die Initialisierungsphase beträgt somit.

$$O(nm) + O(nm) + O(n) + O(n) = O(nm + n) = O(nm)$$

Phase 2 und 3 sind unverändert gegenüber dem zweiten Algorithmus, weshalb die Laufzeit direkt übernommen werden kann.

Gesamtlaufzeit

In diesem Algorithmus werden die gleichen Verfahren zur Realisierung der Funktionen F1 und F2 verwendet. So ergibt sich ein Zeitbedarf der *Preperations*-Phase von $O(nm) + O(n) = O(nm)$. Die Gesamtlaufzeit beträgt für einen

Durchlauf des dritten Algorithmus:

$$O_{Initialisierung}(mn) = \begin{cases} 1 \cdot \left(\begin{array}{l} O_{Update\_Agent\_Information()}(nm)+ \\ O_{Update\_MAG()}(n) \end{array} \right) \\ + \\ c \cdot \left(\begin{array}{l} O_{SelectAgent\_MaxValue()}(mn)+ \\ O_{SelectAgent\_MinValue()}(n)+ \\ O_{Value()}(n)) \end{array} \right) \end{cases}$$

$$O_{Verhandlung}(1) = O_{BilateralNegotiation()}(1)$$

$$O_{Evaluierung}(nm) = O_{Evaluate()}(nm)$$

$$O_{Initialisierung}(nm) + O_{Verhandlung}(1) + O_{Evaluierung}(nm) = O_{Gesamt}(nm)$$

Da bei diesem Verfahren jeweils der beste Agent zu einer Aufgabe ermittelt wird, ist hier eine Trefferwahrscheinlichkeit von $p_2 = 1$ gegeben.

7.3.4 Algorithmus 4 - SVM DHF-S

Für die Auswahl von relevanten Agenten in der Simulationsphase des vierten Algorithmus werden unterstützend *Support Vector Maschines* (SVM) eingesetzt. Der vierte Algorithmus erweitert den dritten Algorithmus. Ein SVM-Mechanismus wird eingesetzt, um die Auswahl möglicher Agenten einzuschränken, die in den Select-Funktionen untersucht werden. Die Erweiterung wirkt sich neben den Select-Funktionen auch auf die Evaluierungsfunktionen aus, die im Folgenden beschrieben werden. Mit Hilfe der SVM werden die Agenten nicht nur anhand ihrer Dienste klassifiziert, sondern werden noch spezieller in unterschiedliche Aufgabentypen je Dienstklasse aufgeteilt. Wenn keine Informationen über einen Agenten bezüglich seines Aufgabentyps existieren, wird dieser Agent zunächst der Dienstklasse im Allgemeinen zugewiesen.

Durch Erlernen der Fähigkeiten kann ein Agent schließlich speziellen Aufgabentypen zugewiesen werden. Durch diese Klassifizierung müssen zur Auswahl eines Agenten nicht mehr alle Agenten einer Dienstklasse untersucht werden, sondern nur die des entsprechenden Aufgabentyps und die allgemeinen Agenten des Dienstes. Durch das Erlernen der Agenteneigenschaften reduziert sich so die Menge der allgemeinen Agenten, bis schließlich alle Agenten Aufgabentypen zugewiesen sind. Mit dem Eingang von neuen Aufgaben erlernt der Agent die Abgrenzung zwischen den Aufgabentypen. Somit können die Agenten besser in aufgabentypische Gruppen einteilen werden (siehe Abbildung 7.3). Zu Beginn gehören alle Agenten der allgemeinen Dienstklasse an, sofern keine gesicherten Informationen über die Fähigkeiten der Agenten existieren. Anhand der Verhandlungen kann die Richtigkeit der Angaben zu den speziellen Dienstklassen der Agenten geprüft werden. Scheitert die Verhandlung aufgrund der Unfähigkeit, eine Aufgabe zu bearbeiten, wird ein Agent von der speziellen Dienstklasse

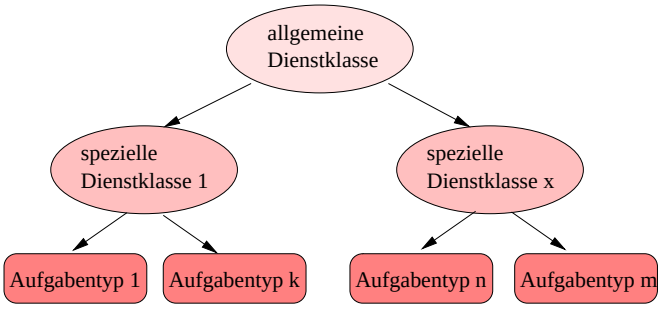

Abbildung 7.3: Klassifizierung der Agenten anhand von Aufgabentypen

in die allgemeine Dienstklasse verschoben. Konnte hingegen die spezielle Dienstklasse durch die Verhandlung bestätigt werden, ist es möglich den Agenten mit Hilfe einer SVM näher zu spezifizieren und einem Aufgabentyp zuzuweisen. Die Aufgaben bilden dabei die Lernmenge (klassifiziert durch den Typus), auf der die Agenten getestet und klassifiziert werden.

F1 - Aktualisierung der Agenteninformationen

Die Funktion 'Update_Agent_Information(MAG)' aktualisiert die Informationen über die relevanten Agenten bezüglich einer gegebenen Mehrschrittaufgabe (MAG). Die Vorgehensweise zum Erlernen der Agenteninformationen geschieht in analoger Weise nach der Funktion des dritten Algorithmus.

Nachdem die neuen Informationen durch die stochastische Bewertungsfunktion in die Wissensbasis aufgenommen wurden, müssen anschließend die Klassifizierungen der Agenten angepasst werden.

Funktionsrumpf 16. (F1)

```
Update_Agent_Information(MAG)
{
    NewAgentList := Get_Agent_Info_From_WUA(MAG);
    if (NewAgentList <> null)
        AgentEval := AgentEval + NewAgentList;
        if (AgentEval.Lenght > MaxLenght)
            Remove_Oldest_Entry(AgentEval);
        Agentlist := ComputeVariance(AgentEval, MAG);
        foreach (Agent in Agentlist)
            foreach ([Tasktype#Hyperplane] in Tasktypes)
                if (Item_In_Class(Agent, Hyperplane))
                    Agentlist := Agentlist - Agent;
                    Agentlist.Tasktype := Agentlist.Tasktype + Agent;
}
```

Terminierung Die Funktion gleicht der Funktion F1 aus vorherigen Algorithmus und terminiert, falls die zusätzlich Klassifikation der Agenten terminiert.

Die Klassifikation durch die Funktion *Item_ In_ Class(Agent, Hyperplane)* terminiert, da $f(x) = sgn(\sum_{m=1}^{M} \alpha_m \cdot d_m \cdot K^{\phi}(x_m, x) + b)$ berechnet werden muss, was in endlicher Zeit lösbar ist.

Laufzeit Die Funktion F1 entspricht denen der vorherigen Algorithmen, jedoch mit dem Zusatz der Klassifikation. Der erste Teil dieser Funktion kann bekannterweise in $O(nm)$ gelöst werden. Die Klassifikation der Agenten benötigt zusätzlich $O(nm)$ Zeit, um für jeden Agenten zu prüfen, welche Aufgabentypen er bearbeiten kann. Insgesamt wird für F1 also $O(nm)$ Zeit benötigt.

Agentenselektion

Die Funktionen 'SelectAgent_MaxValue()' und 'SelectAgent_MinValue()' geben nach Ausführung ein 3-Tupel, bestehend aus einem selektierten Agenten, einer Aufgabe und die Kosten der Ausführung dieser Aufgabe durch den selektierten Agenten als Ergebnis zurück.

Auswahl von Agenten mit maximalem Nutzen (F3) Die Funktion 'SelectAgent_MaxValue($Agentlist|_{MAG}$)' dient der Auswahl eines Agenten mit maximalem Nutzen für das Holon für eine noch nicht zugewiesene Aufgabe aus der Aufgabenmenge der MAG. Bevor Agenten für die Bearbeitung einer Aufgabe analysiert werden, wird die Agentenklasse bestimmt, die für die Bearbeitung dieses Aufgabentyps geeignet ist. Anschließend werden die relevanten Agenten in analoger Weise dem dritten DHF-S Verfahren auf ihre Tauglichkeit untersucht.

Terminierung Terminiert die Zusammenstellung relevanter Agenten, dann terminiert die komplette Funktion F3, da diese eine Erweiterung der Funktion F3 des dritten Algorithmus ist. Dass die Funktion *Item_ In_ Class(Task, Hyperplane)* terminiert, wurde zuvor in Funktion F1 gezeigt. Somit terminiert F3.

Funktionsrumpf 17. (F3)

```
SelectAgent_MaxValue(Agentlist|_MAG)
{
    BestAgent := null;
    BestDistance := infinity;
    BestTask := null;
    RelevantAgents := null;
    foreach(Task in MAG)
        if (Task.Receiver = null)
            IdealAgent := Build_Ideal_Agent(Task);
            foreach([Tasktype#Hyperplane] in Tasktypes)
                if (Item_In_Class(Task, Hyperplane))
                    RelevantAgents := RelevantAgents + Agentlist.Tasktype
            foreach(Agent in RelevantAgents)
                Distance := Compute_Distance(Agent, IdealAgent);
                if (Distance < BestDistance)
                    BestDistance := Distance;
                    BestAgent := Agent;
                    BestTask := Task;
    return {BestAgent, BestTask, BestDistance}
}
```

Laufzeit Funktion F3 wird in analoger Weise des dritten Algorithmus ausgeführt, jedoch mit dem Zusatz, dass die Menge relevanter Agenten durch den Einsatz der SVM reduziert werden kann. Die Berechnung, welche Agenten für die Bearbeitung einer Aufgabe in Frage kommen, kann pro Aufgabentyp in konstanter Zeit durchgeführt werden. Die Bestimmung des Aufgaben-Agenten Paars (Ergebnis der Funktion) benötigt so ebenfalls $O(nm)$ Zeit.

Auswahl von Agenten mit minimalem Nutzen (F4) Die Funktion 'SelectAgent_MinValue()' bildet analog der Funktion 'SelectAgent_MaxValue()' *ideale* Agenten zu jeder Aufgabe, die bereits einem Agenten zugewiesen wurde. Anschließend werden die Distanzen zwischen den idealen Agenten und den zugewiesen Agenten berechnet und das neue Aufgaben-Agenten-Kosten-Tupel ermittelt, das die größte Distanz zwischen dem idealen Agenten und dem zugewiesenen Agenten besitzt. Abschließend liefert diese Funktion das Aufgaben-Agenten-Kosten-Tupel als Ergebnis zurück. Für die Realisierung diese Funktion kann die Funktion F4 des dritten Algorithmus ohne Modifikation direkt übernommen werden, die terminiert und $O(n)$ Zeit benötigt.

F7 - Agentenbewertung

Neue Informationen über Agenten und Aufgaben werden der Wissensbasis durch den Aufruf der Funktion 'Evaluate($Agentlist|_{MAG}$)' zugefügt. Der Ablauf dieser Funktion geschieht analog des vorherigen Algorithmus, jedoch mit dem Zusatz des Erlernens der Aufgabenklassifikation. Es wird überprüft, ob neue Aufgaben in der MAG eingefügt wurden, bzw. Aufgaben von ihren Parametern modifiziert wurden. Diese Aufgaben werden in die Trainingsmenge entsprechend ihres Aufgabentyps aufgenommen. Anschließend werden die Hyperebenen neu berechnet. Da jedoch die Berechnung der Hyperebenen unter Umständen viel Zeit beansprucht, wird nicht bei jedem Funktionsaufruf eine Neuberechnung durchgeführt, sondern nur nach hinreichend vielen Modifikationen der Wissensbasis. Beispielsweise könnte eine Regel zur Aktualisierung der Hyperebenen lauten, erst zu aktualisieren, wenn mindestens 10% neue Daten in die Wissensbasis aufgenommen wurden.

Terminierung Der erste Teil dieser Funktion gleicht der Funktion F7 des vorherigen Algorithmus und terminiert daher. Der zweite Teil zur Aktualisierung der Hyperebenen terminiert, da zum einen die Liste der Aufgaben pro MAG endlich ist, wie auch die Trainingsliste, die alle Aufgabentypen enthält. Zur Ausführung der Funktion *Compute_Hyperplanes()* muss das quadratische Optimierungsproblem gelöst werden, wozu hinreichende Literatur und frei verfügbarer Programmcode zur Verfügung stehen, die nachweisbar terminieren. Somit terminiert die Funktion F7.

Laufzeit Da für die Bewertungsfunktion F7 das gleiche Verfahren wie Funktion F1 eingesetzt wird, sind auch die Laufzeiten dieser beiden Funktionen identisch mit $O(nm)$ für den ersten Teil dieser Funktion. Der Zweite Teil zur Klassifikation benötigt $O(n)$ Zeit für die Aktualisierung der Trainingsliste. Die Funkti-

on zur Neuberechnung der Hyperebenen mit Hilfe von SVM-Verfahren[9] kann in $O(m^3)$ berechnet werden [CL02a], [Bos02]. So liegt der Zeitbedarf dieser Funktion F7 bei $O(nm+n+m^3) = O(nm+m^3)$, falls die Hyperebenen neu berechnet werden müssen, sonst bei $O(nm)$.

Funktionsrumpf 18. (F7)

```
Evaluate(Agentlist|_MAG)
{
    NewAgentList := Get_Results_Negotiation(MAG);
    if (NewAgentList <> null)
        AgentEval := AgentEval + NewAgentList;
        if (AgentEval.Lenght > MaxLenght)
            Remove_Oldest_Entry(AgentEval);
        Agentlist := ComputeVariance(AgentEval, MAG);
    Send_Agentlist_to_WUA(Agentlist);
    i := 0;
    foreach(Task in MAG)
        if(Task_Changed(Task)
            Add_Task_Trainingslist(Task);
            i++;
    if ( i / Trainingslist.Lenght > 0.1)
        Tasktypes := Compute_Hyperplanes();
}
```

Gesamtlaufzeit

Die Gesamtlaufzeit steigt scheinbar gegenüber dem dritten Algorithmus. Dies ist jedoch nur dann der Fall, wenn die Hyperebenen neu berechnet werden müssen. Sonst reduziert der Einsatz einer SVM die Laufzeit der Simulationsphase, was jedoch in der O-Notation nicht ersichtlich wird (siehe Auswertung der Testläufe 8. Die Gesamtlaufzeit beträgt:

$$O_{Initialisierung}(mn) = \begin{cases} 1 \cdot \left(\begin{array}{l} O_{Update\_Agent\_Information()}(nm)+ \\ O_{Update\_MAG()}(n) \end{array} \right) \\ + \\ c \cdot \left(\begin{array}{l} O_{SelectAgent\_MaxValue()}(mn)+ \\ O_{SelectAgent\_MinValue()}(n)+ \\ O_{Value()}(n)) \end{array} \right) \end{cases}$$

$$O_{Verhandlung}(1) = O_{BilateralNegotiation()}(1)$$

$$O_{Evaluierung}(nm + m^3) = O_{Evaluate()}(nm) \; bzw. \; O_{Evaluate()}(nm + m^3)$$

$$O_{Initialisierung}(nm) + O_{Verhandlung}(1) + O_{Evaluierung}(nm) = O_{Gesamt}(nm)$$

bzw. im Fall, dass die Hyperebenen neu berechnet werden müssen, ergibt sich ein zusätzlicher Berechnungsaufwand in der Evaluierungsfunktion:

$$O_{Initialisierung}(nm)+O_{Verhandlung}(1)+O_{Evaluierung}(nm+m^3) = O_{Gesamt}(mn+m^3)$$

Da auch hier jeweils der beste Agent zu einer Aufgabe ermittelt wird, ist hier eine Trefferwahrscheinlichkeit von $p_2 = 1$ gegeben.

[9] Genauere Angaben siehe [SBS99].

7.4 Vergleichende Zusammenfassung

7.4.1 Vergleich der DHF-S Algorithmen

In Kapitel 7.3 habe ich vier Algorithmen beschrieben, die der DHF-S Klassen angehören. Die Ansätze implementieren unterschiedliche Methoden für den Aufbau von Wissen, Reasoning über diesem Wissen und der Ausführung von bilateralen Verhandlungen. Der erste Algorithmus verzichtet komplett auf die Pflege und Analyse einer Wissensbasis und setzt für die Realisierung von Verhandlungen einen 'One-Shot'-Auktionsmechanismus ein. Diese Vereinfachung hat einerseits den Vorteil, dass die Ausführung der Funktionen sehr effizient ist. Andererseits zeigt sich in der Gesamtheit des Algorithmus, dass aufgrund des fehlenden Wissens wesentlich mehr Simulationsrunden und damit auch Verhandlungen ausgeführt werden müssen, bis alle Aufgaben der MAG einem Agenten zugewiesen sind. Im ersten Algorithmus konnte so die Laufzeit für einen Durchgang reduziert werden, jedoch auf Kosten der Kommunikationslast, die abhängig von der Mächtigkeit der Agentengesellschaft ist und bei großen Gesellschaften höher gegenüber den anderen Algorithmen liegt. Diese Eigenschaft drückt sich in dem Faktor p_2 der Laufzeitanalyse aus. p_2 steht hierbei im umgekehrten Verhältnis zur Anzahl der erwartenden Durchläufe und ist somit auch ein Maß für die Kommunikationslast. Tabelle 7.2 zeigt den direkten Vergleich aller vier Algorithmen. Fragen bezüglich der Stabilität und der Effizienz beim Lernen der Algorithmen können erst anhand von praktischen Testläufen, wie sie im folgenden Kapitel durchgeführt werden, beantwortet werden. Dabei ist unter anderem von Interesse, welche Art von Stabilität die Algorithmen in dynamischen Umgebungen erreichen können.

	Random$^-$ DHF-S	**Random$^+$ DHF-S**	**wissensbasierter** DHF-S	**SVM DHF-S**
Wissensbasis	Nein	Ja	Ja	Ja
Select-Funktionen	Random	Random & Agentenanalyse	Aufgaben- & Agentenanalyse	Aufgabenanalyse & SVM
Laufzeit Phase 1	$O(n)$	$O(mn \cdot log(m))$	$O(nm)$	$O(nm)$
Laufzeit Phase 2	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Laufzeit Phase 3	$O(1)$	$O(nm)$	$O(nm)$	$O(nm + m^3)$/ $O(mn)$
Gesamtlaufzeit	$O(n)$	$O(mn \cdot log(m))$	$O(nm)$	$O(nm + m^3)$/ $O(nm)$
Trefferwahrscheinlichkeit	$p_2 = \frac{1}{m}$	$p_2 = \frac{\#relev.Agenten}{m}$	$p_2 = 1$	$p_2 = 1$
Verhandlungsart	1:1 Contract Net	Multiattribut	Multiattribut	Multiattribut

Tabelle 7.2: Vergleich der DHF-S Algorithmen

7.4.2 Vergleich mit dem Verfahren von Soh und TsaTsoulis

Soh & TsaTsoulis beschreiben in [ST02] ein Verfahren zur Bildung von Koalitionen, das vom Ablauf dem hier beschriebenen Schema ähnlich ist. Dieses Verfahren basiert auf einem spieltheoretischen Ansatz zur Nutzenbestimmung, entgegen dem hier vorgestellten DHF-S. Der Ansatz von Soh & TsaTsoulis teilt den Koalitionsformierungsprozess in drei Phasen auf, die sich wiederholen, bis allen Aufgaben ein Agent zugewiesen ist:

1. Der Initiator startet den Koalitionsformierungsprozess zuerst durch Auswahl von Agenten aus seiner *Nachbarschaft*, die für die Bestimmung einer Initiallösung anhand ihrer Fähigkeiten geeignet sind. Die Nachbarschaft eines Agenten besteht dabei aus allen Agenten, die dem Initiator bekannt sind. Die relevanten Agenten werden dann entsprechend ihrem Nutzen für die Koalition in einer geordneten Liste einsortiert.

2. Im zweiten Schritt startet der Initiator die Verhandlungen mit den am besten bewerteten Kandidaten, um sie für die Koalition zu gewinnen. Diese Verhandlungen finden parallel statt, was dazu führen kann, dass unter Umständen nicht alle ausgewählten Agenten der Koalition beitreten und somit das Koalitionsziel nicht erreicht werden kann.

3. Daher werden in einem dritten Schritt nach Beendigung aller Verhandlungen die Agenten über die Annahme der erfolgreich verhandelten Koalitionsbeitritte informiert, falls *alle* Verhandlungen erfolgreich abgeschlossen sind. Wird mindestens eine Koalitionsverhandlung nicht erfolgreich abgeschlossen, werden allen Agenten über das Scheitern der Verhandlungen informiert. Erst nach erfolgreicher Bestätigung nehmen die Agenten die Aufgaben an, planen sie in ihren lokalen Plan ein und sind fortan daran gebunden.

Um die Verhandlungsergebnisse zu verbessern, schlagen die Autoren vor, zusätzlich Lernmethoden in der ersten Phase des Verfahrens einzusetzen, um so bessere Bewertungen bezüglich der Agenten zu erhalten. Die Wahl der potentiellen Koalitionspartner ist bei diesem Verfahren entscheidend für die Qualität der gebildeten Koalitionen und besonders für die Suchdauer, bis eine Koalitionsstruktur gefunden wird, bei der alle Partner den Bedingungen des Initiators zustimmen und der Koalition beitreten. Nachfolgende Tabellen 7.3 und 7.4 zeigen einen Vergleich des vorgestellten Verfahrens von Soh & TsaTsoulis mit dem DHF-S Schema. Ein Vergleich ist möglich, da die Ansätze beider Verfahren, des DHF-S Schemas und der Vorgehensweise von Soh & TsaTsoulis, ähnliche Voraussetzungen an eine dynamische Umgebung stellen. Das Verfahren von Soh & TsaTsoulis ist hierbei von besonderem Interesse, da beide Verfahren rundenbasiert virtuelle Gruppenstrukturen aufbauen, um sie anschließend in einer Verhandlungsphase zu manifestieren.

	DHF-S	**Soh & TsaTsoulis**
Beziehungsmodell	Ja	Ja
Vertrauensmodell	Ja	Nein
Lernen	Ja	Ja
Können Agenten aus einer Koalition ohne technische Probleme ausscheiden?	Ja, durch die Agenten selbst, bzw. vom Holonenführer entlassen.	Nein, wenn eine Koalition gebildet ist, findet keine weiteren Anpassungen mehr statt, außer es treten unbeeinflussbare Störungen auf.
Bestrafung für einen Vertragsbruch?	Ja	Nein
Optimalität	Abhängig von der Anzahl der Agenten, die dem Holonenführer bekannt sind. Je größer die Anzahl, desto besser die Lösungen. Um die Bekanntmachung der Agenten untereinander zu verbessern wird u.a. der World-Utility-Agent eingesetzt.	Abhängig von der Anzahl der Agenten, die dem Koalitionsführer bekannt sind. Jedoch sind die Agenten bei der Suche nach neuen Agenten auf sich allein gestellt. Nur wenn zwei Koalitionen sich vereinen, kann sich das Agentenverzeichnis des Koalitionsführers erweitern. Informationen bzgl. neuer Agenten verbreiten sich nur sehr langsam. Zudem stellt sich die Frage, wie ein neuer Agent die Adressen von anderen Agenten erfährt?
Verhandlungsmethoden	Multiattribute negotiation	Argumentative negotiation
Verhandlungsreihenfolge	Sequentiell	Parallel
Kann die nachträgliche Bestätigung der zuvor verhandelten Verträge vermieden werden?	Ja. Verhandlungen enden direkt mit bindenden Verträgen. So wird vermieden, dass Bestätigungen von Koalitionsteilnehmern abgelehnt werden, obwohl die Verhandlung positiv verlaufen ist.	Nein. Erfolgreich geführte Verhandlungen müssen am Ende der Verhandlungsphase bestätigt werden, da ein Agent eine Bestätigung im Nachhinein ablehnen kann, obwohl die Verhandlungen positiv verlaufen sind. Da es keine Bestrafung gibt, besteht hierbei ein großes Sicherheitsrisiko für Betrug.
Bereitet der Algorithmus Reservekandidaten vor, für den Fall, dass eine Verhandlung scheitert?	Nein	Ja, da Verhandlungen mit allen möglichen Partnern durchgeführt werden und der Agent mit dem besten Verhandlungsergebnis den Zuschlag erhält, existiert somit ein *Reservekandidat*. Der Algorithmus entspricht dem Contract-Net Protokoll mit der Einschränkung, dass die Agentengesellschaft benevolent sein muss. Das Protokoll ist nicht vor Missbrauch durch nicht benevolente Agenten geschützt.

Tabelle 7.3: Vergleich DHF-S mit dem Algorithmus von Soh&TsaTsoulis-Teil I

	DHF-S	Soh & TsaTsoulis
Kommunikations-aufwand	Je besser der Holonenführer die Attribute der anderen Agenten erlernt hat, desto weniger Verhandlungen müssen tatsächlich durchgeführt werden. Das Kommunikationsaufkommen ist direkt abhängig vom Wissen des Agenten.	Kostenintensiv, da die Verhandlungen parallel ausgeführt werden und scheitern können, was zur Folge hat, dass bereits erfolgreich verhandelte Aufgaben wieder aufgelöst werde müssen.
Sind Verträge bindend?	Ja, bei Vertragsbruch erfolgt Bestrafung, mit Geld und Vertrauen.	Nicht notwendigerweise, da es keine direkte Bestrafung gibt.
Globaler Informationsträger	Ja, World-Utility-Agent	Nein
Optimierung existierender Holone zur Ausführungszeit?	Ja, notwendig, da der Holonenführer zu keinem Zeitpunkt vollständiges Wissen über die Agenten der Gesellschaft erlangen kann.	Nein, die Koalitionsstruktur ist fixiert. Nachteile sind: a) lösbare Probleme können u.U. nicht gelöst werden b) es können suboptimale Lösung entstehen.
Wie kann eine Koalition zerbrechen?	Durch interne und externe Effekte, d.h. von den Agenten gewollt oder durch äußere Einflüsse bewirkt, z.B. technische Probleme.	Nur durch äußere Einflüsse, da das Verhalten aller Agenten als benevolent angenommen wird.

Tabelle 7.4: Vergleich DHF-S mit dem Algorithmus von Soh&TsaTsoulis-Teil II

7.5 Zusammenfassung

Nachdem ich die Grundlagen von Verhandlungen zur Teambildung beschrieben habe, habe ich ein Schema zur dynamischen Holonenformierung entwickelt. Die Kernidee des Schemas ist, dass jeder Holonenführer neue Holonen mit potentiellen Partnern erst simuliert und anschließend (falls eine ´bessere´ Holonenstruktur gefunden wurde) verhandelt. Dabei bestehen Holone aus hierarchisch strukturierten Agentenverbänden, die aufgabenorientiert miteinander in geeigneten Unterverbänden kooperieren. Die Simulation derart bestimmter hypothetischer Holonenstrukturen erfolgt anhand der erlernten (unsicheren) Wissensbasis des Holonenführers eines jeden Holons. Bevor ein solch hypothetisches Holon durch Verhandlungen realisiert wird, wird zwischen dem Risiko des Fehlschlags bei der Umsetzung der simulierten Holonenstruktur und dem Nutzen für das zu modifizierende Holon abgewogen. Scheitert eine Verhandlung, treten Störungen auf, oder wurde keine Verbesserung durch die Simulation gefunden, werden solange neue Simulationen und Verhandlungen ausgeführt, bis eine Verbesserung hinsichtlich der Kosten-/Nutzenfunktion erreicht wird. Im Anschluss an die Verhandlungen werden die Ergebnisse evaluiert und das erworbene Wissen der Agenten über ihre jeweilige Umwelt entsprechend aktualisiert.

Das DHF-S Schema wurde in vier verschiedenen DHF-S Algorithmen realisiert, die sich im Wesentlichen in der Art und Weise unterscheiden, wie individuelles Wissen über die Verhandlung von den Agenten erworben und effektiv eingesetzt wird. Beispielsweise setzt der Algorithmus SVM-DHF-S ein um eine Support-Vector-Machine (SVM) erweitertes Reinforcement-basiertes Lernverfahren ein, um die für die jeweiligen Aufgaben geeignetsten Agenten gezielt statt nur zufällig nach 'trial-and-error' auszuwählen. Das Schema beschreibt eine

Klasse von Algorithmen, die einem gemeinsamen Ablauf folgen, der durch das Schema vorgegeben ist. Bei der Realisierung dieser vier Algorithmen habe ich die Verfahren aus den vorherigen Kapiteln Kapitel 4 bis Kapitel 6 eingesetzt, die die Grundlagen für die konkrete Instanziierungen des Schemas zu Algorithmen bilden. Im anschließenden Kapitel 8 werden die Algorithmen auch in Testläufen auf ihre Praxistauglichkeit untersucht.

Kapitel 8

Anwendung auf Szenarien und Evaluation

In den bisherigen Kapiteln habe ich die formale Basis für die Entwicklung von Algorithmen zur Bildung von Holonen in dynamischen, nicht-benevolenten Umgebungen geschaffen. Auf Grundlage des DHF-S Schemas wurden vier Algorithmen entwickelt, die in diesem Kapitel evaluiert werden[1]. Allgemeine Aussagen über Stabilität und Lernfähigkeit der Holone sind aufgrund der Eigenschaften dynamischer Umgebungen, sowie der Existenz von vagen Informationen über die Umwelt, nicht direkt möglich. Das Verhalten dieser Algorithmen muss daher durch geeignete Simulationen analysiert werden. Die Fragestellungen, die hier untersucht werden, sind: Welche Auswirkungen haben die Umstrukturierungsmöglichkeiten der Algorithmen auf die Stabilität der Holone, und führen bereits einzelne Störungen zu einer totalen Destabilisierung der Holonen. Bei den wissensbasierten DHF-S Algorithmen und dem SVM DHF-S Algorithmus ist zudem die Lernfähigkeit von besonderem Interesse. Die Anzahl der Verhandlungen ist sehr stark abhängig von der Qualität der Wissensbasis des Verhandlungsführers. Mit Hilfe der Simulationen wird unter anderem eine Abschätzung ermittelt, wie viele Evaluationsphasen des Algorithmus durchlaufen werden müssen, bis die Trefferwahrscheinlichkeit der Simulationsphase einen vorgegebenen Wert erreicht. Im folgenden Abschnitt 8.1 wird die Simulationsumgebung beschrieben. In Abschnitt 8.2 werden die Testszenarien und die Eigenschaften der Algorithmen angegeben, die getestet werden sollen. Anschließend wird in Abschnitt 8.3 die Testläufe ausgewertet und analysiert.

8.1 Testumgebung

Um die Simulationen der Algorithmen aus Abschnitt 7.3 durchführen zu können, muss die Testumgebung klar definiert sein. Das Testszenario wird so gewählt, dass zum einen die Ergebnisse der Algorithmen miteinander vergleichbar sind und zum anderen die dynamischen Effekte auf die Holonenbildung begrenzt

[1] Die Implementierung dieser Algorithmen habe ich in der Programmiersprache C# vorgenommen. Als Basis diente das Agricola-System, das auf den Grundlagen aus Kapitel 4 und Kapitel 5 aufgebaut wurde.

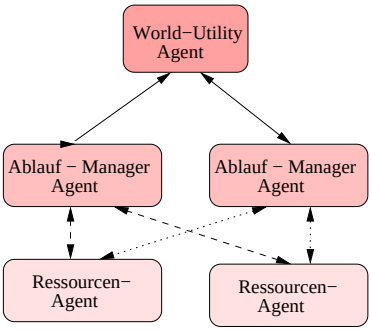

Abbildung 8.1: Relationen zwischen den Agententypen der Testumgebung

werden, so dass gezielte Aussagen über die unterschiedlichen Ansätze möglich sind. Die Durchführung der Testreihen geschieht iterativ. Dabei wird die Anzahl der MAGs, der Aufgaben und der Agenten in den aufeinanderfolgenden Testläufen kontinuierlich gesteigert. Gemessen werden die Gesamtkosten[2] der Holone zu definierten Zeitpunkten und die Anzahl von Verhandlungsversuchen der zweiten Phase des Schemas. Die Anzahl der Verhandlungsrunden gibt ein von der Hardware unabhängiges Maß für die Bearbeitungszeit wieder. Ein direkter Vergleich der vier Algorithmen kann nur unter identischen Randbedingungen vollzogen werden. Alle Algorithmen werden daher auf die gleichen Aufgaben-/MAG-Sätze hin getestet und den gleichen Störungen ausgesetzt. Für die Aufgaben, wie auch für die Agenten, sind die Ausprägungen der Attribute nicht auf einzelne Werte beschränkt, sondern beziehen sich jeweils auf ein Intervall. Diese Intervalle müssen sich nicht notwendigerweise vollständig überdeckend, so dass ein Agent Aufgaben mit unterschiedlichen Ausprägungen annehmen kann. Agenten sind einer Aufgabe nicht ad hoc zuweisbar, so dass das Erlernen der Agenteneigenschaften notwendig ist. Die Initialisierung der Agentengesellschaft für eine Testreihen wird unabhängig der Testszenarien immer nach folgendem Schema durchgeführt:

- Die Agentengesellschaft, bestehend aus einem World-Utility-Agent, mehreren Ablauf-Manager-Agenten (Holon) und mehreren Ressourcen-Agenten, wird initiiert.
- Jeder Ressourcen-Agent meldet sich bei allen Ablauf-Manager-Agenten an und ebenso jeder Ablauf-Manager-Agent beim World-Utility-Agent (siehe Abbildung 8.1).
- Zu Beginn besitzt kein Agent Wissen über die Fähigkeiten der anderen. Die Wissensbasis der Agenten wird somit erst durch Ausführung der Verhandlungen mit der Zeit gefüllt und verbessert.
- Jeder Ablauf-Manager-Agent erhält zu Beginn X Mehrschrittaufgaben, mit jeweils Y Aufgaben.

[2] Die Gesamtkosten ist eine Aufsummierung der Bearbeitungskosten aller Aufgaben.

- Die Ablauf-Manager-Agenten führen ihre Aktionen parallel aus. Dabei arbeiten sie ihre Mehrschrittaufgaben sequentiell ab. Während der Bildung von Holonen werden Störungen vom WUA erzeugt und an die Ablauf-Manager weitergegeben. Nachdem sich die Agentengesellschaft stabilisiert hat (falls eine stabile Lösung existiert), werden die Testergebnisse gesammelt und an den WUA weitergegeben (siehe Abbildung 8.2).

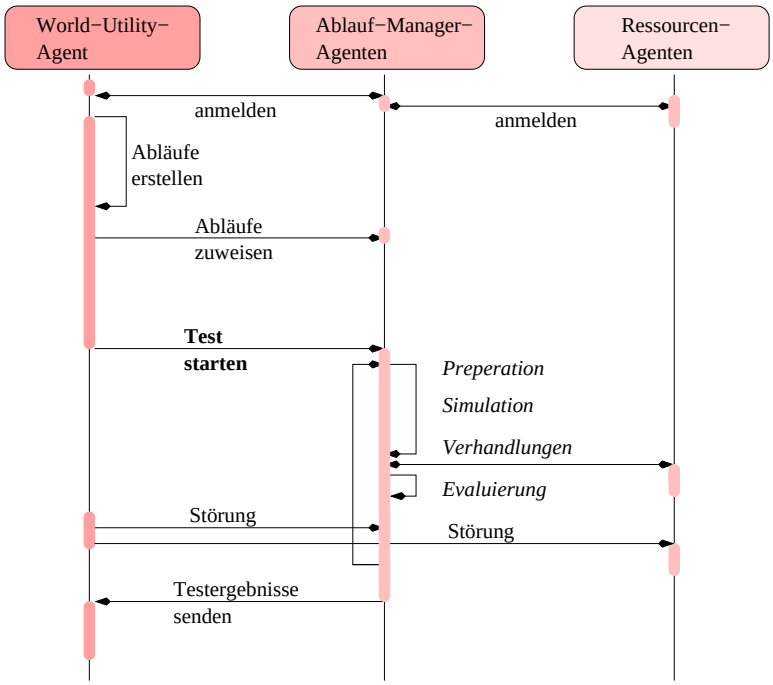

Abbildung 8.2: Allgemeiner Ablauf der Testszenarien

Bei der Durchführung der Testszenarien werden die Algorithmen mit zwei unterschiedlichen Wahrscheinlichkeitskonfigurationen für die *add*, *remove* und *noop* Operationen der Simulationsphase getestet. Zum einen mit einer Verteilung von 35% : 35% : 30% und zum anderen mit 55% : 15% : 30%. Die Testläufe werden solange durchgeführt, bis allen Aufgaben des Testszenarios Agenten zugewiesen sind und sich das System stabilisiert hat. Finden die Algorithmen nach einer begrenzten Rundenanzahl keine vollständige Zuweisung von Agenten auf Aufgaben und finden weitere Rekonfigurationen statt, wird der Algorithmus als instabil abgebrochen.

8.2 Testszenarien

8.2.1 Testszenario 1

Die vier in Kapitel 7.3 vorgestellten Algorithmen werden hinsichtlich des Nutzens der Pflege und Analyse der Wissensbasis gegenüber der zufälligen Auswahl von Agenten untersucht. Gemessen wird der Nutzen der wissensbasierten und

der SVM DHF-S Algorithmen gegenüber den Random Algorithmen anhand der Differenz zwischen der Anzahl durchgeführter Verhandlungen und der Anzahl von Aufgaben und Agenten der Testszenarien.

Testziel: *Qualität* der Select-Funktion

- Fragestellung: Welche Qualität haben die Algorithmen hinsichtlich der Gesamtkosten und der Anzahl von Verhandlungsrunden?
- Messkriterien
  - Kosten der Gesamtlösung bis eine asymptotische Stabilität erreicht wird
  - Anzahl Verhandlungen
- Testläufe über
  - Anzahl der Agenten
  - Anzahl der Aufgaben pro Holon
- eingesetzte Algorithmen: 1 - 4

Die Dynamik der Umgebung wird während eines Testdurchlaufs unterdrückt. So können Aussagen bezüglich der Qualität der Selektionsfunktion durch Angabe der Trefferquote geeigneter Agenten bezüglich der Mächtigkeit der Agentengesellschaft angegeben werden.

8.2.2 Testszenario 2

Im zweiten Testszenario soll die Stabilität der Algorithmen bei einfachen Störungen getestet werden. Dazu werden analog dem ersten Testszenario Aufgaben eingeplant und abgewartet bis die Holone sich stabilisiert haben. Anschließend wird eine einfache Störung erzeugt. Die Art der Störung wird zufällig initiiert und besteht aus dem Entfernen/Hinzufügen eines Agenten oder der Ausplanung einer Menge von Aufgaben, die anschließend wieder eingeplant werden müssen. Wird ein Agent entfernt, müssen alle ihm zugewiesenen Aufgaben erneut verplant werden. Die Störung tritt nur einmal auf, wodurch die Umplanungen in den Holonenstrukturen angestoßen werden. Gemessen wird, wie viele Verhandlungen zusätzlich notwendig sind, bis die Holone sich wieder stabilisieren haben.

Testziel: *Stabilität* bei einfacher Störung

- Fragestellung: Welche Art von Stabilität konnte unter welchen Bedingungen erreicht werden (asymptotisch, begrenzt oder keine Stabilität)?
- Messkriterien
  - Kosten der Gesamtlösung bis Lösung stabil ist
  - Anzahl Verhandlungen
- Testläufe über
  - Anzahl der Agenten
  - Anzahl der Aufgaben pro Holon
- eingesetzte Algorithmen: 1 - 4

8.2.3 Testszenario 3

In diesem Testszenario wird geprüft, wie sich die Algorithmen bei regelmäßigen Störungen verhalten. Das System wird von Beginn an gestört, d.h. es wird nicht gewartet bis sich eine stabile Lösung gebildet hat. Direkt nach der Aufgabenverteilung vom WUA an die Ablauf-Manager werden zehn Störungen in regelmäßigen Intervallen initiiert. Analog dem zweiten Testszenario besteht eine Störung entweder aus dem Entfernen oder Hinzufügen eines Agenten in die Gesellschaft, oder dem Ausplanen einer Menge von Aufgaben, die anschließend wieder eingeplant werden.

Testziel: *Stabilität* bei regelmäßiger Störung

- Fragestellung: Kann eine begrenzte Stabilität erreicht werden? Ab welchem Störungsgrad eskalieren die Verhandlungen?
- Messkriterien
  - Anzahl der offenen Aufgaben nach einer Störung
- Testläufe über
  - Anzahl der Agenten
  - Anzahl der Aufgaben pro Holon
- eingesetzte Algorithmen: 1 - 4

8.3 Resultat und Interpretation

Im Folgenden werden die Ergebnisse der empirischen Evaluation der zuvor vorgestellten DHF-S Algorithmen anhand der Agricola-Anwendungsdomäne vorgestellt. Der Fokus liegt in der Demonstration der Effizienz (hinsichtlich der Anzahl von Verhandlungen) und des Mehrwerts durch Pflege und Analyse einer Wissensbasis gegenüber randomisierten Verfahren, sowie einer Stabilitätsanalyse.

Bei der Bewertung der der Testergebnisse müssen die Kosten und die Anzahl der Verhandlungen in Relation zu einander betrachtet werden, da mit jeder Umstrukturierung zwei Ziele verfolgt werden: Minimierung der Gesamtkosten und vollständige Zuweisung aller Aufgaben auf die Agenten.

Bei den Testläufen wird zwischen zwei Gruppen unterschieden. Die Aufteilung der Testläufe dient dabei der Untersuchung, welche Auswirkungen die Gewichtung auf die zufällige Wahl des Aktionsoperators in der Simulationsphase hat. Es werden die Testszenarien sowohl mit einem Verhältnis 35% : 35% : 30% zwischen dem Operator 'Agent hinzufügen' (add), 'Agent von einer Aufgabe entbinden' (remove) und 'keine Aktion ausführen' (noop) durchgeführt, wie auch mit einem Verhältnis von 55% : 15% : 30% pro Algorithmus.

8.3.1 Ergebnisse aus Testszenario 1

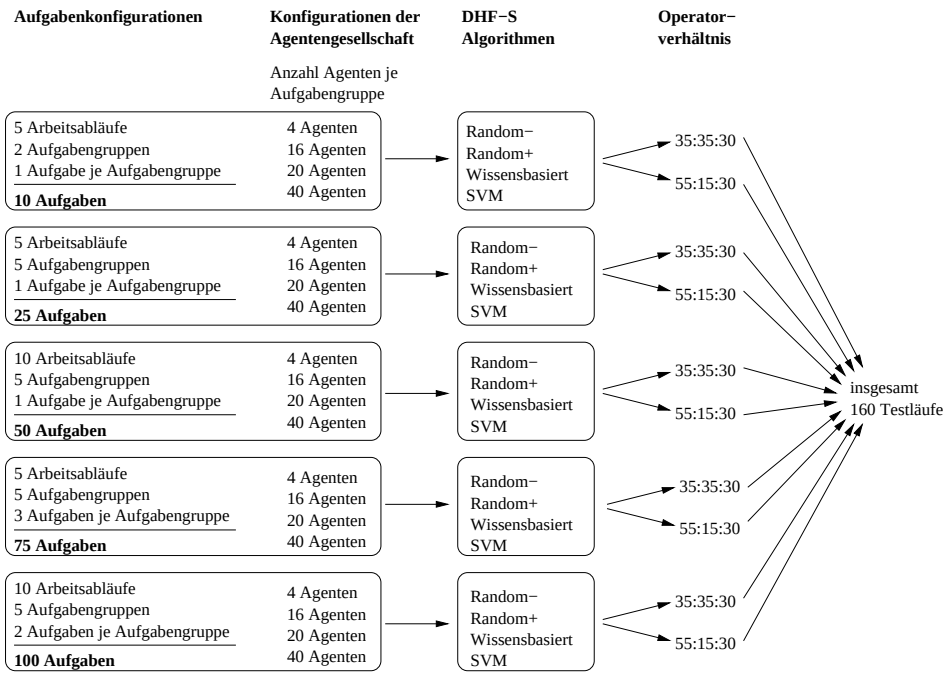

Abbildung 8.3: Konfigurationen der Testläufe

Die Algorithmen im ersten Testszenario wurden insgesamt 160 Testläufen unterzogen. Jeder der fünf Aufgabenkonfigurationen wurde auf die vier Agentengesellschaftskonfigurationen angewandt. Dies wurde jeweils für jeden der vier Algorithmen mit den Operatorverhältnissen 55:15:30 und 35:35:30 vollzogen (siehe Abbildung 8.3). Insgesamt ergibt sich für die vier Algorithmen $5 * 4 * 4 * 2 = 160$ Testläufe. Ein Testlauf ist charakterisiert durch die Anzahl von Agenten je Aufgabengruppe, der Anzahl von Arbeitsabläufen, der Anzahl von Aufgabengruppen je Arbeitsablauf und der Anzahl von Aufgaben je Arbeitsgruppe. Die Struktur eines Arbeitsablaufs wird zufällig erstellt und setzt sich aus mehreren Aufgabengruppen zusammen, die wiederum genau die vorgegebene Anzahl von Aufgaben je Testlauf enthält. Abbildung 8.4 zeigt einen Ablauf mit fünf Aufgabengruppen zu je sechs Aufgaben.

Fazit Testszenario 1 Abbildung 8.6 und Abbildung 8.7 zeigen die Ergebnisse der Testläufe des ersten Testszenarios. Auf der X-Achse ist die Anzahl der fünf Aufgabengruppen angetragen (10, 25, 50, 75 und 100 Aufgaben). Jeder Balken einer Aufgabengruppe steht für einen Algorithmus, beginnend mit dem Random$^-$, dem Random$^+$, dem wissensbasierten DHF-S und dem SVM DFH-S Algorithmus. In den linken Diagrammen steht die Höhe der Balken für die Anzahl der Verhandlungen eines Testlaufs und in den rechten Diagrammen für die Kosten aller Aufgaben nach Beendigung eines Testlaufs. Bei sehr kleinen Agenten- und Auftragsmengen unterscheiden sich die Ergebnisse hinsichtlich

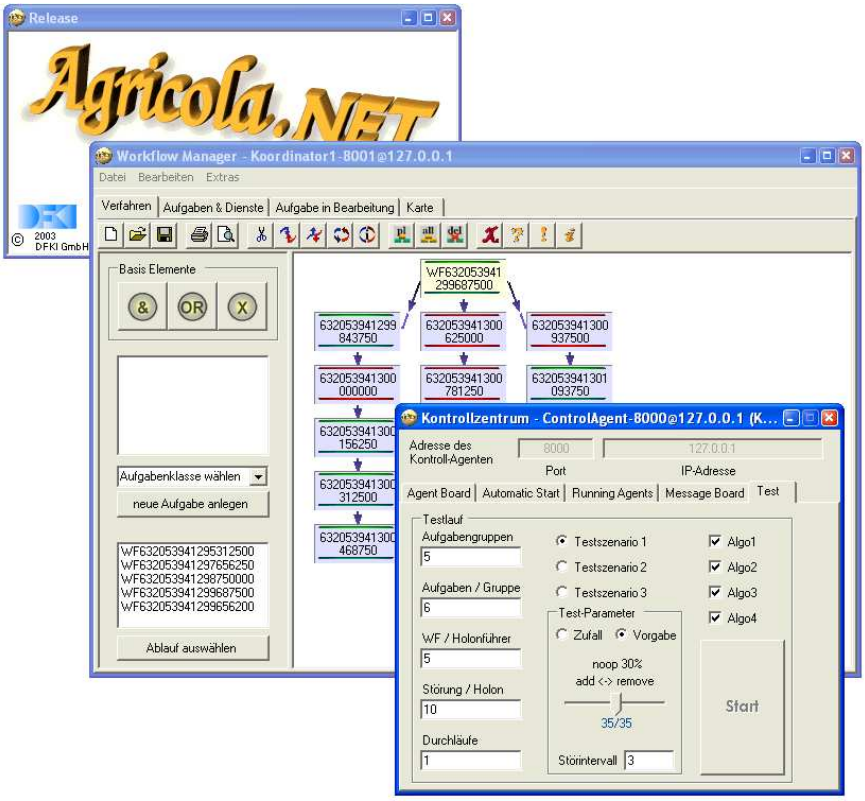

Abbildung 8.4: Konfiguration eines Testlaufs

der Anzahl von Verhandlungen und Kosten kaum von den anderen.

- Random$^-$ DHF-S: Die Kosten und damit die Nutzenoptimierung des ersten Algorithmus eskalieren mit der Vergrößerung der Agenten- oder Aufgabenmenge und übersteigen weit die Werte der anderen drei Algorithmen. Bedingt durch die randomisierte Suche, ohne Zugriff auf Wissen über die anderen Agenten, gelingt es nur in einigen Fällen (bei sehr kleinen Aufgabenmengen - hier 10 Aufgaben), annehmbare Ergebnisse zu erzielen. Die Kosten[3] übersteigen in der Regel weit die Drei-Millionengrenze und liegen sogar im Milliardenbereich[4].

- Random$^+$ DHF-S: Dieser Algorithmus wählt die Agenten zwar zufällig aus, prüft jedoch vor der Aufnahme in ein hypothetisches Holon, ob der Agent die Grundfertigkeit zur Erfüllen der jeweiligen Aufgabe besitzt. So

[3] Zur Berechnung der Kosten verwenden die Agenten den euklidischen Abstand von den Positionen in ihren Plänen hin zum Zielort der Aufgabenstellung. Diese Ergebnisse werden abschließend mit den 'Kosten pro Stunde'-Werten der jeweiligen Agenten multipliziert.

[4] Die Diagramme enden bereits bei maximalen Kosten von 3.000.000 Einheiten. Jeder Balken der diese Grenze erreicht, überschreitet diese bei weitem.

	Random$^-$	Random$^+$	wissensbasiert	SVM
35:35:30	6,86	3,28	2,55	2,48
55:15:30	3,76	2,19	1,60	1,56

Tabelle 8.1: Durchschnittliche Anzahl bilateraler Verhandlungen pro Aufgabe

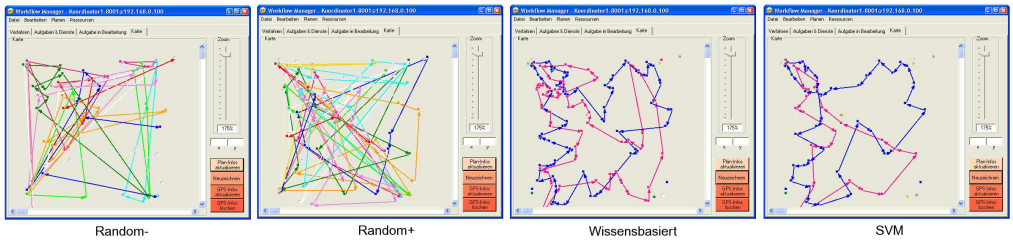

Abbildung 8.5: Graphische Darstellung der Agentenpläne

werden irrelevante Agenten bereits ausgefiltert, bevor mit den Verhandlungen begonnen wird. Wie geeignet die einzelnen Agenten hinsichtlich der jeweiligen Aufgabe sind, wird hierbei nicht berücksichtigt. Diese zwei Eigenschaften schlagen sich in den Testläufen nieder. Die Anzahl der Verhandlungen entspricht im Mittel der des dritten und vierten Algorithmus, jedoch mit größeren Schwankungen. Durch Fehlen der Nutzenbewertung steigen jedoch die Gesamtkosten der Holone für die Bearbeitung aller Aufgaben.

- Wissensbasierter DHF-S und SVM DHF-S: Der Vergleich zwischen dem dritten und vierten Algorithmus zeigt, dass aufgrund der Analyse während der Agentenauswahl (Select-Funktionen) beide Algorithmen Holonenkonfigurationen finden, die minimale Kosten aufweisen. Die Algorithmen finden die optimale Lösung hinsichtlich der Kosten und damit des Nutzens. Bei den Verhandlungen treten Unterschiede zugunsten des SVM-basierten Algorithmus auf. Tabelle 8.1 zeigt wie viele bilaterale Verhandlungen pro Aufgabe durchschnittlich geführt werden müssen, bis ein Agent gefunden wurde, der die Aufgabe ausführt.

Die graphische Darstellung der Agentenpläne (siehe Abbildung 8.5) weist deutliche Unterschiede zwischen den vier Algorithmen auf und bestätigt den zuvor aufgezeigten Trend. Die randomisierten Verfahren führen aufgrund der größeren Anzahl unterschiedlicher Agenten, die zur Aufgabenbearbeitung eingesetzt werden, zu wenig optimierten Plänen. Hingegen resultiert der Einsatz des wissensbasierten DHF-S bzw. des SVM DHF-S Algorithmus zu geordneten 'Touren' der Agenten. Der Vergleich der letzten beiden Algorithmen zeigt, dass durch Einsatz einer SVM die Routen weiter verbessert werden.

8.3.2 Ergebnisse aus Testszenario 2

Das zweite Testszenario kann in direktem Anschluss an das erste ausgeführt werden, da hierbei zunächst eine stabile Gesellschaft aufgebaut werden muss, welche das erste Testszenario erzeugt. Anschließend wird eine einfache Störung initiiert und die Auswirkungen gemessen, so dass auch hier insgesamt 160 Testläufe mit den gleichen Konfigurationen wie in Testszenario 1 (siehe Abbildung 8.3) ausgeführt werden. Nachdem die Holone einen stabilen Zustand erreicht haben, wird jeder Holonenführer angewiesen, eine zufällig ausgewählte Aufgabe auszuplanen. Entsprechend dem DHF-S Schemas werden Ersatzagenten gesucht und unter Umständen sogar Umstrukturierungen zur Nutzenmaximierung vorgenommen. Die Anzahl von Verhandlungen, die nach einer Störung ausgeführt werden, ist ein Indikator für die Stabilität der Algorithmen. Da jedoch nicht nur Ersatz für den simulierten Ausfall gesucht wird, sondern gleichzeitig auch eine Steigerung des Nutzens der Holonenstruktur gesucht wird, muss auch der Gesamtnutzen der Holone berücksichtigt werden. Um den Optimierungseffekt in Bezug auf die Verhandlungskosten zu analysieren, wird sowohl die Anzahl von Verhandlungen gespeichert, wie auch der Nutzen der Holone jeweils vor und nach Auftreten der Störung, um die Kostenentwicklung zu dokumentieren.

Fazit Testszenario 2 Das Auftreten einer einfachen Störung auf eine stabilen Holonengesellschaft (siehe Kapitel 6.3) hat schwerwiegende Auswirkung auf die Ergebnisse des ersten Algorithmus. Mit wachsender Agentenanzahl steigt das Risiko, dass eine Störung zu einer Destabilisierung der Holone führt. Die Kosten schwanken stark und steigen häufig um das 140-fache gegenüber der stabilen Lösung. Dabei zeigt sich, dass die Kostenschwankung größer ist, wenn die Anzahl der Agenten kleiner ist. Der erste Algorithmus destabilisiert gelegentlich. Diese Destabilisierung tritt jedoch vermehrt bei einem Operatorverhältnis von 35:35:30 im Gegensatz zu einem Operatorverhältnis von 55:15:30. Der Random$^+$ DHF-S Algorithmus destabilisiert dagegen nur in einzelnen Fällen, für die Kosten kann eine untere und obere Schranke [-15..10] angegeben werden. Holone unter Einsatz des dieses Algorithmus bilden SIBO stabile Holone. Der wissensbasierte und SVM DHF-S Algorithmus zeigen ein ähnliches Verhalten. Beide sind hinsichtlich einer Störung asymptotisch stabil. Sie finden nach einer Störung wieder eine alternative gute Ausgangslösung. Bei genauerer Betrachtung zeigen sich zwei Unterschiede:

- Der Nutzen einer SVM zeigt sich verstärkt bei einem Operatorverhältnis von 35:35:30
- Erst bei größeren Aufgabenmengen ist die Anzahl der Verhandlungen niedriger gegenüber dem wissensbasierten DHF-S Verfahren.

Ein gleichbleibender niedriger Quotient für die durchschnittliche Anzahl von Verhandlungen pro Aufgabe und Agent ist ein weiterer Indikator für eine asymptotische Stabilität. In Bezug auf dieses Bewertungskriterium zeigt der wissensbasierte und der SVM DHF-S Algorithmus deutlich besser Performanz gegenüber den ersten beiden Algorithmen. Auch hier erzielt der SVM DHF-S Algorithmus die besten Ergebnisse (siehe Abbildung 8.8).

8.3.3 Ergebnisse aus Testszenario 3

Die Testkonfiguration des dritten Testszenarios ist analog dem ersten und zweiten Testszenarios aufgebaut (siehe Tabelle 8.3). Für jede Testbedingung wird ein Durchlauf ausgeführt, so dass auch hier insgesamt 160 Testläufe ausgeführt wurden. Im Gegensatz zu den ersten beiden Testszenarien werden hier bereits Störungen erzeugt, bevor sich eine stabile Situation gebildet hat und weitere Störungen in Folge initiiert.

Fazit Testszenario 3 Von Bedeutung in diesem Testszenario ist die Anzahl noch offener Aufgaben nach Auftreten einer Störung (siehe Abbildung 8.9 und Abbildung 8.10). Auf der X-Achse sind die Störungen aufgetragen, die kontinuierlich auf das System einwirken. Jede Stelle der X-Achse steht für eine Störung. Zu jeder Testkonfiguration wurden fünf Testläufe mit jedem der vier Algorithmen durchgeführt, beginnend mit 10 Aufgaben, 25 Aufgaben, 50 Aufgaben, 75 Aufgaben und 100 Aufgaben. Gemessen wurde die Anzahl noch offener Aufgaben nach einer Störung, die keinem Agenten zugewiesen waren. Zu Beginn eines Testlaufs ist die Anzahl der offenen Aufgaben stets sehr hoch, da bereits erste Störungen auftreten, bevor eine stabile Lösung gefunden wurde. Ein Indikator für die Effizienz eines Algorithmus ist hierbei, wie schnell die Kurven sich der X-Achse nähern, also alle Aufgaben Agenten zuweist. Die Testergebnisse zeigen, dass trotz ständiger Störungen die Algorithmen die Anzahl offener Aufgaben kontinuierlich abbauen können. Ist erst einmal eine stabile Situation gefunden, verursachen weitere Störungen nur noch eine beschränkte Anzahl von Neuverhandlungen, mit Ausnahme des Random$^-$ DHF-S Algorithmus, der gelegentlich destabilisiert. Dieses Verhalten ändert sich jedoch mit einer stärkeren Gewichtung auf den *add*-Operator der Simulationsphase. Durch eine Verschiebung der Auswahlwahrscheinlichkeit zugunsten des *add*-Operators, verhält sich sogar der Random$^-$ DHF-S Algorithmus asymptotisch stabil. Die anderen drei Algorithmen hingegen verhalten sich generell asymptotisch stabil. Bei einem Operatorverhältnis von 55:15:30 bilden sich beim Einsatz des SVM DHF-S Algorithmus im günstigsten Fall ca. 1,5mal schneller stabile Holone als bei einem Verhältnis von 35:35:30. Dies korreliert mit dem Verhältnis zwischen den Verhandlungsrunden zum Aufbau einer stabilen Lösung ohne Störung (siehe Kapitel 8.3.1).

8.3.4 Fazit der Testläufe

Bei einem Operatorverhältnis von 55:15:30 liefert der SVM DHF-S Algorithmus die besten Ergebnisse. Als effizientester Algorithmus erreicht er eine Trefferquote von 1,56, d.h. der Holonenführer muss im Mittel 1,56 Verhandlungen führen, um einen Agenten zu finden, der die Aufgabe optimal bearbeiten kann[5]. Im Gegensatz dazu müsste man beim Einsatz des Contract-Net-Protokolls alle Agenten anschreiben[6], um den besten zu finden, weshalb das SVM basierte Verfahren

[5] Dieses Ergebnis ist abhängig von der Wissensbasis des Holonenführers.
[6] Unabhängig der Qualität der Wissensbasis.

sich besonders bei sehr großen Agentengesellschaften eignet. Die Kosteneffizienz des wissensbasierten DHF-S und des SVM DHF-S Algorithmus zeigt sich mit wachsender Agentenzahl. Bereits ab 16 Agenten sind die zu erwartenden Kosten um ein drittel kleiner als die gegenüber dem Random$^+$ DHF-S Algorithmus. Die Kosten des wissensbasierten DHF-S und des SVM DHF-S Algorithmus sind wie erwartet identisch, da die Erweiterung des wissensbasierten DHF-S Algorithmus durch eine SVM lediglich die Anzahl zu untersuchender Agenten einschränkt. Bei korrekter Klassifizierung wird der beste Agent gegebenenfalls mit einem geringeren Aufwand gefunden. Hingegen führt fehlerhaftes Wissen zu einer falschen Klassifizierung der Agenten, so dass unter Umständen der beste Agent nicht gefunden werden kann, wodurch die beiden Algorithmen unterschiedliche Kosten aufweisen müssten. Da jedoch die Kosten identisch sind, folgt, dass die Klassifikation durch die SVM korrekt ist. Wird eine Wissensbasis eingesetzt kann durch den Einsatz eines DHF-S Algorithmus **SIBO-** und **BIBO-Stabilität** garantiert werden. In der Regel verhält sich der wissensbasierte DHF-S und SVM DHF-S Algorithmus sogar asymptotisch stabil und finden dabei nach einer Störung wieder eine optimale Lösung hinsichtlich einer gegebenen Kosten-/Nutzenfunktion.

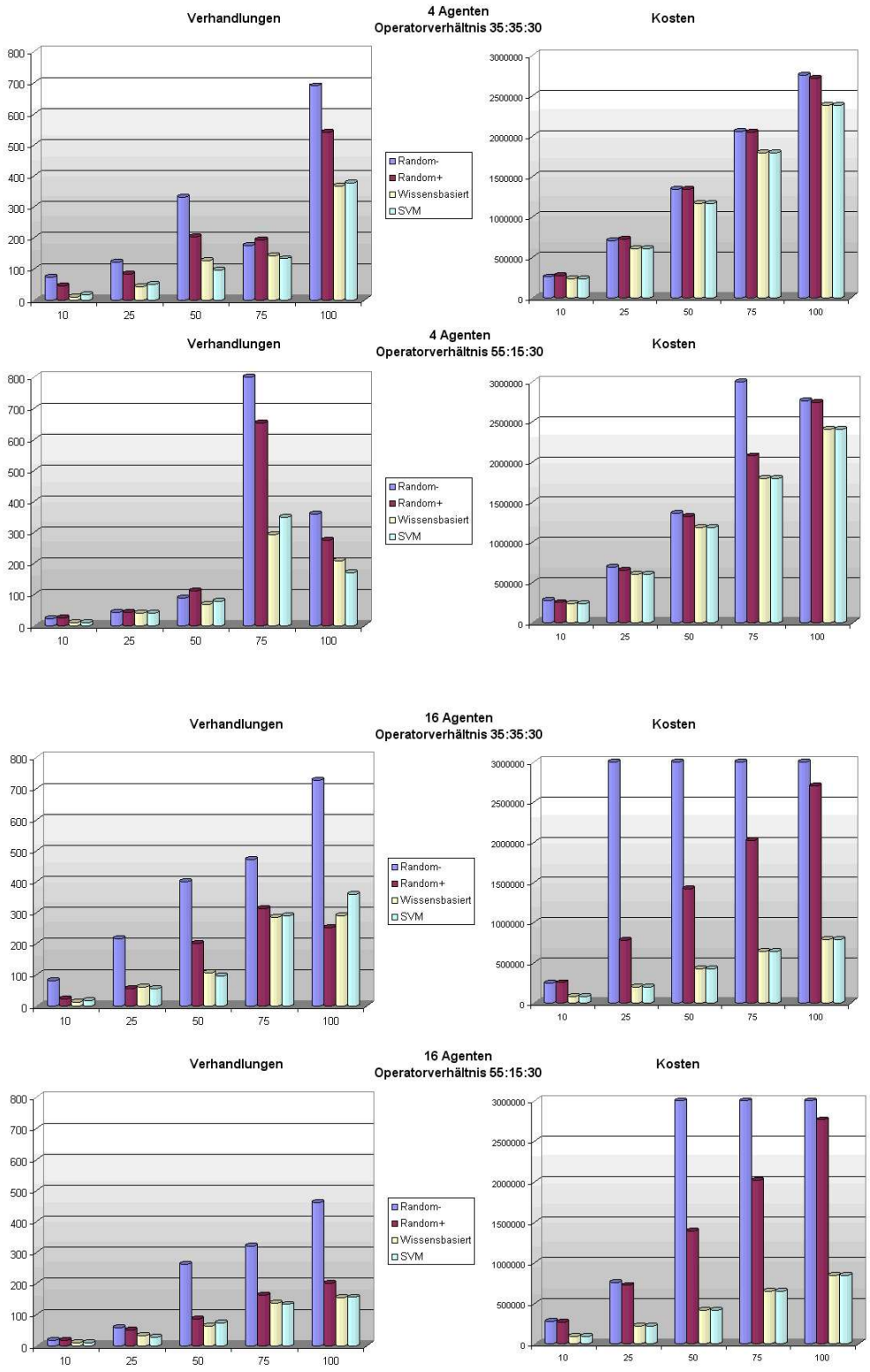

Abbildung 8.6: Testszenario 1 Testergebnisse - Teil I

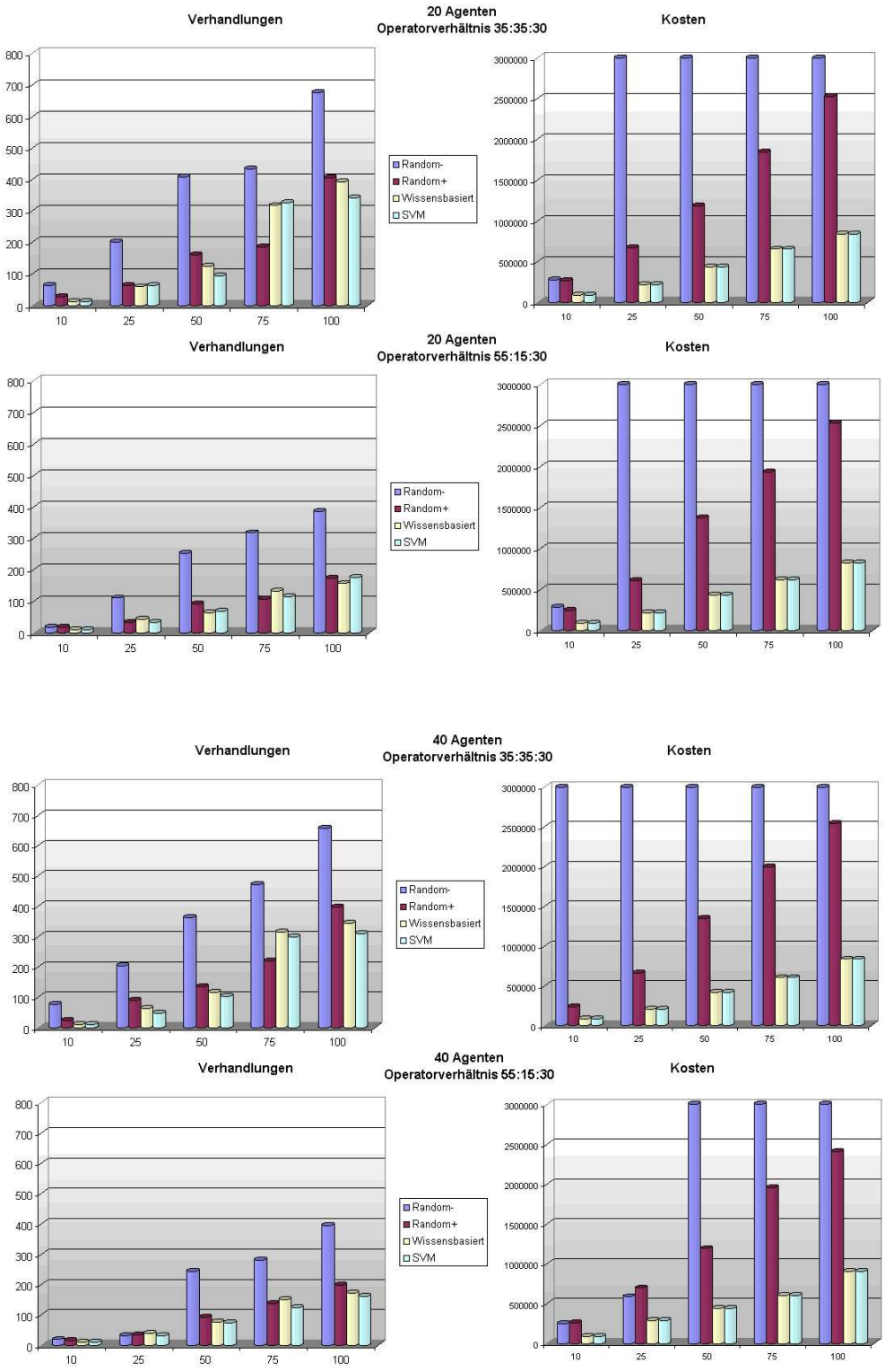

Abbildung 8.7: Testszenario 1 Testergebnisse - Teil II

4 Agenten - Operatorverhältnis 35:35:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	112	-7698,07	26	2,13	10	0,00	10	0,00
25	38	1,68	46	3,25	18	0,00	22	0,00
50	94	-0,44	136	0,88	22	0,00	36	0,00
75	34	-0,79	62	0,65	28	0,00	32	0,00
100	257	0,00	185	-0,71	39	0,00	47	0,00

4 Agenten - Operatorverhältnis 55:15:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	8	1,61	12	-2,94	10	0,00	10	0,00
25	46	0,86	24	-2,94	10	0,00	10	0,00
50	46	0,00	48	-1,56	18	0,00	16	0,00
75	297	98,91	36	-0,78	78	0,00	87	84,75
100	125	0,10	193	-0,59	35	0,00	25	0,00

16 Agenten - Operatorverhältnis 35:35:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	71	-8079,97	15	7,51	15	0,00	9	0,00
25	194	-24,86	74	9,87	16	0,00	16	0,00
50	417	90,33	75	3,47	23	0,00	29	0,00
75	357	10,17	71	1,89	69	0,00	59	0,00
100	626	24,89	134	-1,09	60	0,00	36	0,00

16 Agenten - Operatorverhältnis 55:15:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	11	1,04	7	-8,86	5	0,00	5	0,00
25	65	12,20	23	1,42	19	0,00	19	0,00
50	48	95,82	48	-1,42	12	0,00	14	0,00
75	271	80,66	46	-2,78	38	0,00	38	0,00
100	467	75,84	58	0,91	30	0,00	24	0,00

20 Agenten - Operatorverhältnis 35:35:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	95	-14320,20	17	-4,42	7	0,00	7	0,00
25	196	29,79	61	-5,13	27	0,00	29	0,00
50	384	70,78	53	-5,53	19	0,00	27	0,00
75	501	27,96	50	2,31	90	0,00	88	0,00
100	774	37,07	338	0,60	54	0,00	50	0,00

20 Agenten - Operatorverhältnis 55:15:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	41	24,99	15	-14,15	9	0,00	7	0,00
25	43	98,01	22	5,00	10	0,00	10	0,00
50	154	97,39	21	0,86	9	0,00	13	0,00
75	252	-91,82	28	-0,46	18	0,00	20	0,00
100	378	32,55	54	1,51	24	0,00	22	0,00

40 Agenten - Operatorverhältnis 35:35:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	81	99,39	14	-10,40	12	0,00	14	0,00
25	203	14,17	20	-3,35	22	0,00	32	0,00
50	396	14,12	50	5,29	24	0,00	26	0,00
75	477	46,88	70	0,37	72	0,00	56	0,00
100	648	48,66	180	-5,04	46	0,00	46	0,00

40 Agenten - Operatorverhältnis 55:15:30

Vergleich nach einer Störung

Anzahl der Aufgaben	Random- DHF-S		Random+ DHF-S		Wissensbasiert DHF-S		SVM DHF-S	
	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis	Verhandlungen	Kostenersparnis
10	17	-8,86	15	9,31	7	0,00	5	0,00
25	22	5,50	16	-0,42	14	0,00	20	0,00
50	209	0,28	48	-3,02	18	0,00	22	0,00
75	27	94,19	32	2,70	38	0,00	34	0,00
100	404	31,84	22	-2,85	28	0,00	26	0,00

Abbildung 8.8: Ergebnisse nach einem Störimpuls

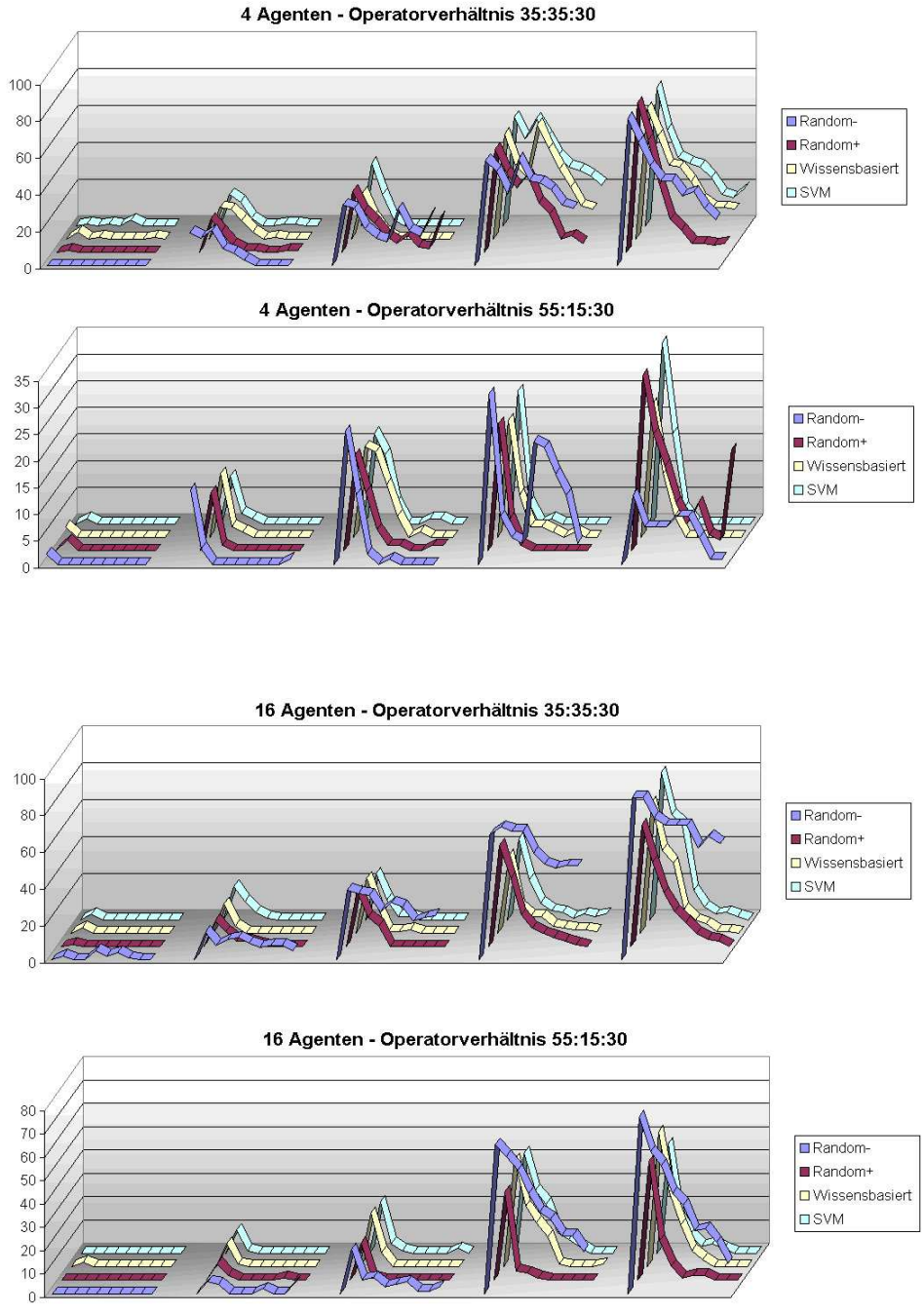

Abbildung 8.9: Testszenario 3 Testergebnisse - Teil I

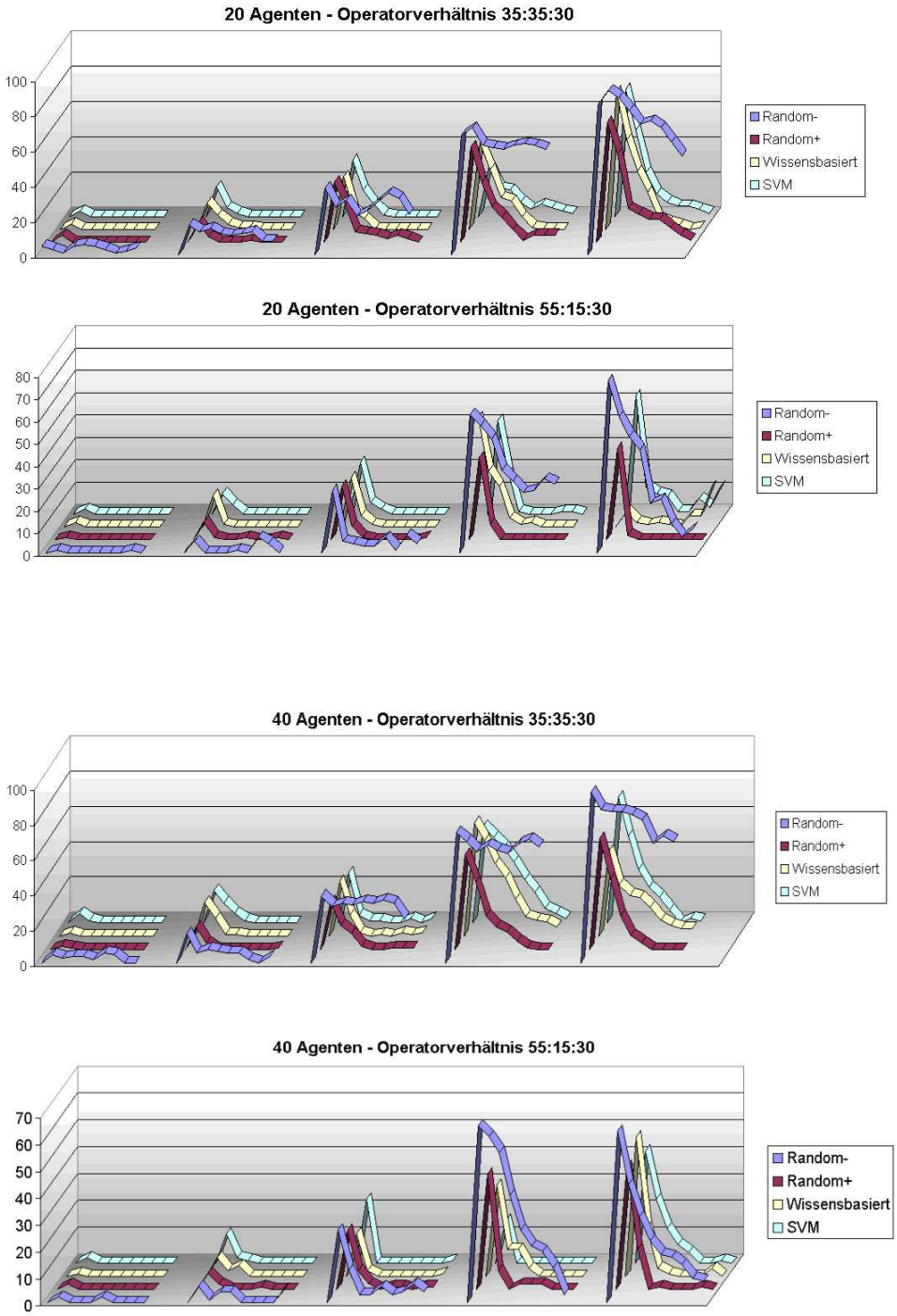

Abbildung 8.10: Testszenario 3 Testergebnisse - Teil II

Kapitel 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Das Thema dieser Dissertation ist die Entwicklung eines Verfahrens zur Bildung von Holonen in einer nicht benevolenten, dynamischen Umgebung. Um ein Verfahren zur Lösung dieser Fragestellung angeben zu können, habe ich eine Vektoralgebra zur Beschreibung von Arbeitsabläufen und den Entitäten (Agenten, Holone, Dienste und Aufgaben) eines Multiagenten-Systems entwickelt und genutzt. Darauf aufbauend habe ich ein Schema entwickelt, das es den Agenten ermöglicht, sich flexibel und eigenständig den Veränderungen ihrer Umgebung anzupassen. Die Notwendigkeit von Systemen, die sich dynamischen Veränderungen der Umgebung anpassen können, habe ich in Kapitel 3 anhand von drei Systemen gezeigt. Dabei wurde die zunehmende Wichtigkeit flexibler Systeme deutlich, die mit vagen und unvollständigen Wissen umgehen müssen. In dem Projekt Agricola habe ich die in dieser Arbeit entwickelte Formalismen und Verfahren erfolgreich praktisch einsetzen können und damit die abschließenden Testläufe aus Kapitel 8 gebildet. Zur Verwaltung von Arbeitsabläufen habe ich im Projekt Agricola ein Agentenwerkzeug entwickelt, mit dem Arbeitsabläufe zusammengestellt, die Planung und Ausführung einzelner Aufgaben überwacht und die Ressourcenverteilung dynamisch koordiniert werden kann.

Damit eine automatisierte Planung realisiert werden kann, musste ich zunächst einen geeigneten Formalismus zur Beschreibung von Aufgaben, Diensten, Agenten und Holonen entwickeln. Die Beschreibungen dieser Elemente eines Multiagenten-Systems müssen dabei der Beschreibung der Domäne folgen, die durch eine Menge von charakterisierenden Attributen definiert wird. Nur wenn auch die Aufgaben und Dienste der Agenten die gleiche Beschreibung wie die Agenten selbst besitzen, ist es möglich Zustände zu beschreiben, die Agenten erfüllen müssen, um Dienste anbieten und Aufgaben basierend auf ihren Diensten bearbeiten zu können. In Kapitel 5 konnte ich zeigen, dass durch eine geeignete Definition der Agenten und der Definition von Verknüpfungen der Agenten untereinander eine vektorbasierte Algebra zur Modellierung genommen werden kann. Somit steht die gesamte Theorie der Vektorrechnung zur weiteren Bearbeitung zur Verfügung. Beispielsweise kann anhand einer Abstandberechnung zwischen gegebenen Agentenvektoren und einem Aufgabenvektor der nächstlie-

gende (geeignetste / ähnlichste) Agent ermittelt werden, der die vorgegebene Aufgabe am besten lösen kann. Durch die Abbildung der Agenten auf Vektoren ist es unter anderem möglich, Agenten nicht nur anhand eines Attributes, sondern anhand einer Menge von Attributen miteinander zu vergleichen. Zudem kann durch eine geeignete Verknüpfungsvorschrift zweier Agentenvektoren eine Gruppe von Agenten auf einen Vektor abgebildet werden. Darauf aufbauend habe ich gezeigt, wie ein Holon durch eine Matrix, bestehend aus den Agentenvektoren seiner Subagenten und den Matrizen seiner Subholone, auf eine Holonenmatrix abgebildet werden kann. Die Abbildung eines Multiagenten-Systems auf einen Vektorraum ermöglicht so die formale Beschreibung der Gruppen- und Holonenbildung. Damit können Relationen zwischen den Agenten basierend auf einer Metrik für Matchmakingverfahren beschrieben werden, die Bestimmung alternativer Agenten zur Aufgabenbearbeitung über das Ähnlichkeitskriterium vollzogen werden, oder bei der Ermittlung des Ressourcenbedarfs komplexer Aufgaben mit Hilfe einer Linearkombination aus unterschiedlichen Agententypen eine Lösung gefunden werden. Zudem bietet der mathematische Formalismus Definitionen algebraischer Strukturen, wie z.B. Gruppen, die direkt auf Agenten übertragen werden können. Diese Definitionen geben Richtlinien an die Verfahren zur Holonenbildung vor, womit klare Aussagen bezüglich des Verhaltens von Agenten gegeben werden können. Beispielsweise spielt die Reihenfolge, in der Agenten in ein Holon aufgenommen werden, keine Rolle, wegen der Assoziativität einer Gruppe.

Im Kapitel 6 habe ich unterschiedliche Formen der Kooperation untersucht und basierend, auf dem im vorherigen Kapitel definierten Agentenvektorraum, ein Lernverfahren zur Verbesserung des unvollständigen und vagen Wissens der Agenten angegeben. Zusammen bilden die Abbildung eines Multiagenten-Systems auf einen Vektorraum und der Einsatz des entworfenen Lernverfahrens die Grundlage für die Entwicklung eines Schemas (Kapitel 7) zur Bildung von Holonen in dynamischen und offenen Umgebungen. Nach einer Diskussion des DHF-S Schemas habe ich vier Algorithmen der DHF-S Klasse entwickelt, die von komplett randomisierten Ansätzen bis hin zu Algorithmen reichen, die eine detaillierte Analyse und Pflege der Wissensbasis (mit den in den vorherigen Kapiteln beschriebenen Methoden) durchführen. Abschließend habe ich in Kapitel 8 Testreihen durchgeführt, um die vier Algorithmen *Random*$^-$, *Random*$^+$, *wissensbasierter DHF-S* und *SVM DHF-S* miteinander zu vergleichen. Die Analyse der Testergebnisse zeigt, welcher Stabilitätsgrad durch die unterschiedlichen Algorithmen erreicht werden kann und in welchem Verhältnis der Aufwand der Pflege einer Wissensbasis (zur Reduktion der ausgeführten Verhandlungen in Bezug auf die Größe der Agentengesellschaft) und Aufgabenvielfalt steht. Diese Arbeit baut also einen Formalismus auf, unter dessen Verwendung es möglich ist, Methoden formal zu beschreiben, die als Grundlage für die Entwicklung von Algorithmen dienen und das Problem der Holonenformierung in dynamischen und offenen Umgebungen lösen.

9.2 Ausblick

In dieser Arbeit wurde ein Formalismus zur Abbildung eines Multiagenten-Systems auf einen Vektorraum vorgestellt. Aufbauend auf diesem Formalismus ist es nun möglich, Eigenschaften von Algorithmen und Protokollen formal zu zeigen. Beispielsweise könnte die Konvergenz der hier vorgestellten Algorithmen, wie durch empirische Versuche gezeigt wurde, formal mit Hilfe des Banachsche Fixpunktsatzes [Heu84] gezeigt werden. Die Abbildung von Multiagenten-Systemen auf einen Vektorraum eröffnet die Möglichkeit, Sätze und Verfahren der klassischen Mathematik auf Multiagenten-Systemen zu übertragen. Somit steht die gesamte Theorie der Vektorrechnung zur Bearbeitung zur Verfügung, die im Einzelnen jedoch auf ihre Sinnhaftigkeit untersucht werden müsste. Zum Beispiel bietet eine solche Abbildung die Grundlage für den Einsatz von *Support Vector Machines* (SVM) [CV95a] im Bereich der Multiagenten-Systeme. SVMs können in diesem Kontext zur Selektion geeigneter Agenten bezüglich eines Bedarfs eingesetzt werden. Bei diesem Ansatz müssen jedoch die Entitäten der Umgebung durch Vektoren beschrieben sein, so dass es erst durch den Nachweis eines Multiagenten-System-Vektorraums es möglich wird, solche SVM-Verfahren einzusetzen. Der Aufbau einer solchen Abbildung wurde motiviert durch die enorme Leistungssteigerung im Bereich der Computergrafik, speziell bei dem Einsatz hoch performanter Verfahren zur Realisierung von 3D-Grafiken in Echtzeit, wie zum Beispiel in Computerspielen. Hierbei könnten Algorithmen zur Kollisionsberechung eines grafischen Objekts mit der Topologie der Bodenkarte zum Einsatz kommen, um eine effiziente Auswahl von Agenten mit bestimmten Eigenschaften durchzuführen. Dabei entsprächen die Agenten einer Agentengesellschaft den Polygonen der Bodenkarte einer 3D-Welt, die durch Vektoren beschrieben ist und der gesuchte Agent entspräche dem grafischen Objekt. Analog der Kollisionsberechnung, bei der mehrere Millionen Polygone der Umwelt mit einigen Hundert bis Tausend Polygonen pro grafischen Objekt gegeneinander getestet werden, müssen die Agentenvektoren gegenüber dem Zielvektor des gesuchten Agenten getestet werden. Verfahren zur Einschränkung der Komplexität, wie zum Beispiel der Einsatz von sogenannten Boundingboxen in der Computergrafik, können in Multiagenten-Systemen möglicherweise ebenso sinnvoll verwendet werden. Zudem finden Bestrebungen statt, Vektoroperationen auf die GPU auszulagern, die für solche Berechnung hoch optimiert ist. Für den Einsatz solcher Techniken in Multiagenten-Systemen bietet die vorgestellte Agentenvektorraumtheorie die Grundlage.

Der Einsatz von Multiagententechnologie im Bereich der Logistik zur Koordination von komplexen Abläufen und Verfahren (wie im Projekt Agricola) ist weiterhin eine essentielle Fragestellung, die zunehmend Wichtigkeit in realen Anwendungen gewinnt. So existieren bereits Bestrebungen, aufbauend auf dem Agricola-System, welches erste Lösungsansätze für die dynamische und flexible Planung und Überwachung von Arbeitsabläufen anbietet, ein Agentendienstnetzwerk aufzubauen, um nicht nur logistische Fragestellungen effizient mit dezentralen Ansätzen lösen zu können, sondern eine Dienstesuite anzubieten, die die logistischen Abläufe in die betriebswirtschaftlichen Abläufe existierender Systeme integriert. Dazu finden zur Zeit Verhandlungen zwischen

dem DFKI und dem Umweltministeriums des Saarlandes statt, das bestehende System weiterzuentwickeln und im Gegensatz zum Agricola-System nicht nur landesweit einzusetzen, sondern mit Unterstützung des Bundesmaschinenrings deutschlandweit ein solches Dienstnetzwerk aufzubauen. Von Interesse ist auch der Einsatz einer modifizierten Variante des Agricola-Systems im Bereich der Flugplatzlogistik, am Beispiel der FraPort AG des Frankfurter Flughafens. Wichtig hierbei ist eine dynamische und fehlertolerante Planung und Koordination der Bodeneinsatzfahrzeuge zur Versorgung der Flugzeuge. Gerade in diesem Anwendungsszenario finden häufig Umplanungen aufgrund verspäteter Flugzeuge statt, die mit aktuell im Einsatz befindlichen Systemen nicht mehr geplant bzw. optimiert werden können. Daher wird die Einsatzplanung weiterhin von Hand durchgeführt. Die Ansätze, wie sie im TeleTruck-System und in dem darauf aufbauenden Projekten CASA und Agricola entwickelt wurden, könnten in dieser Anwendungsdomäne ebenfalls nutzbringend eingesetzt werden.

Anhang A

Relevante Aspekte der Linearen Geometrie

A.1 Relationen

Definition A.1. Seien M und M' zwei Mengen. Unter einer Relation R zwischen M und M' versteht man eine *Teilmenge* R des kartesischen Produkts $M \times M' := \{(x;x')|\ x \in M \wedge x' \in M'\}$, also $R \subseteq M \times M'$. Gilt (x ; x') $\in R$, wobei x $\in$ M und x' $\in$ M' ist, dann stehen x und x' in der Relation R. Schreibweise: x R x'

Sonderfall:. Ist $M = M'$, so heißt R eine *Relation auf M.*

Beispiel A.1. Es sei $M := \{2,3,4,5\}$ und $M' := \{6,7,8,9\}$.
Die Menge $R := \{(2;6), (2;8), (3,6), (3;9), (4;8)\}$ ist eine Relation zwischen M und M'.
Man kann dafür unter Verwendung der Aussageform „x ist Teiler von x' " auch schreiben: $R := \{(x;x')\}|\ x \in M \wedge x' \in M' \wedge$ x ist Teiler von x' }.

Beispiel A.2. Es sei $M := \mathbb{Z}$ und $M' := \mathbb{N}_0 = \mathbb{N} \cup \{0\}$.
Die Menge $R := \{(x;x')|x \in \mathbb{Z} \wedge x' \in \mathbb{N}_0 \wedge x' = x^2\}$ ist eine Relation zwischen $\mathbb{Z}$ und $\mathbb{N}_0$.

Beispiel A.3. Sei M:= $\{1,2,3\}$.
Die Menge $R_1 := \{(1;1), (2;2), (3;3)\}$ und $R_2 := \{(1;1), (2;1), (3;1), (2;2), (3;2), (3;3)\}$ sind Relationen auf M. R_1 ist die Gleichheitsrelation auf M wegen $x = x'$, R_2 die größer-oder-gleich Relation auf M wegen $x \geq x'$.

A.1.1 Veranschaulichung von Relationen durch Pfeildiagramme

In Abbildung A.1, A.2 und A.3 sind die Relationen der vorherigen Beispiele durch sogenannte Pfeildiagramme veranschaulicht: M und M' werden durch Punktmengen dargestellt. Ist $(x;x') \in R$ mit $x \in M$ und $x' \in M'$, so zeichnet man einen vom Element x zum Element x' gerichteten Pfeil.

A.1.2 Relationen mit besonderen Eigenschaften

Relationen mit besonderen Eigenschaften haben einen eigenen Namen. Von Bedeutung sind insbesondere die *Äquivalenzrelationen, Ordnungsrelationen* und *Abbildungen.* Die Relationen von Beispiel A.2 ist eine Abbildung, die Relation R_1 aus Beispiel A.3 eine Äquivalenzrelation, die Relation R_2 von Beispiel A.3 eine Ordnungsrelation.

Äquivalenzrelation auf einer Menge M nennt man jede *reflexive, symmetrische und transitive Relation R auf M.* Dabei heißt eine Relation R auf eine Menge M

- *reflexiv*, wenn $\forall x \in M$ gilt:$(x;x) \in R \Rightarrow xRx$

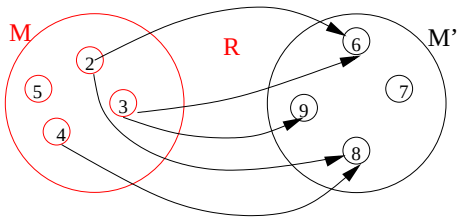

Abbildung A.1: Pfeildiagramm der Relation von Beispiel A.1

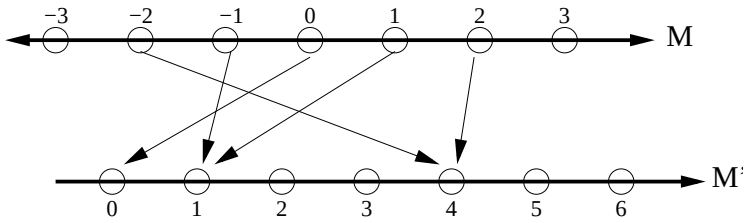

Abbildung A.2: Pfeildiagramm eines Teils der Relation von Beispiel A.2

- *symmetrisch*, wenn $\forall x, x' \in M$ gilt: $(x; x') \in R \Rightarrow (x'; x) \in R$ bzw. $xRx' \Rightarrow x'Rx$
- *transitiv*, wenn $\forall x, x', x'' \in M$ gilt: $(x; x') \in R \wedge (x'; x'') \in R \Rightarrow (x; x'') \in R$ bzw. $xRx' \wedge x'Rx'' \Rightarrow xRx''$.

Jede Äquivalenzrelation auf M bewirkt (induziert) eine Klasseneinteilung (*Partition*) von M, wobei man unter einer Klasseneinteilung von M eine Zerlegung von M in nichtleere Teilmengen (Klassen) versteht, bei der der Durchschnitt von je zwei Klassen leer und die Vereinigung aller Klassen von M gleich der Menge M ist. Umgekehrt induziert jede Klasseneinteilung von M eine Äquivalenzrelation auf M.

Beispiel A.4. In der Elementargeometrie wird der Begriff des Vektors durch eine Äquivalenzrelation erklärt: M sei die Menge aller Pfeile eines n-dimensionalen Raums, d.h. die Menge aller geordneten Punktepaare ($(X_1; X_2)$ Anfangs- / $(Y_1; Y_2)$ Endpunkt) des Raums. Die Erfüllbarkeitsmenge der Aussageform „Pfeil $(X_1; X_2)$ ist parallelgleich zu Pfeil $(Y_1; Y_2)$“ ist dann die Äquivalenzrelation auf M. Die Klasse aller zu einem Pfeil parallelgleicher Pfeile heißt *Vektor*. Jeder Pfeil aus der Klasse nennt man einen *Repräsentanten des Vektors*.

Beispiel A.5. Eine Klasseneinteilung auf der Menge $\mathbb{Z}$ ist gegeben durch:

$$A_0 = \{x \in \mathbb{Z} \mid x \; mod \; 3 = 0\},$$
$$A_1 = \{x \in \mathbb{Z} \mid x \; mod \; 3 = 1\},$$
$$A_2 = \{x \in \mathbb{Z} \mid x \; mod \; 3 = 2\},$$

da $A_0 \cap A_1 = A_0 \cap A_2 = A_1 \cap A_2 = \emptyset$ und $A_0 \cup A_1 \cup A_2 = \mathbb{Z}$. Die durch diese Klasseneinteilung induzierte Äquivalenzrelation ist gegeben durch „x und y haben bei der Division durch 3 denselben Rest“. Als Repräsentanten der drei Klassen A_0, A_1, A_2 können 0,1,2 gewählt werden.

Ordnungsrelation auf einer Menge M nennt man jede *reflexive, antisymmetrische (identitive) und transitive Relation R auf M*. Dabei heißt Relation R auf einer Menge M *antisymmetrisch*, wenn $\forall x, x' \in M$ gilt:

$$(x; x') \in R \wedge (x'; x) \in R \Rightarrow x = x'$$

bzw.

$$xRx' \wedge x'Rx \Rightarrow x = x'.$$

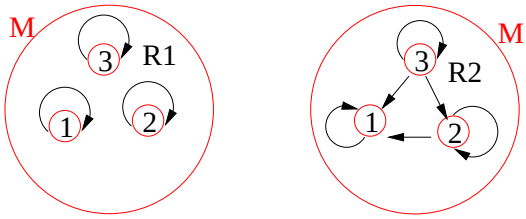

Abbildung A.3: Pfeildiagramme der Realtionen von Beispiel A.3

Ist eine Ordnungsrelation R zudem noch linear, d.h. gilt $\forall x, x' \in M$:

$$(x; x') \in R \vee (x'; x) \in R$$

so nennt man R eine *lineare Ordnungsrelation.*

Beispiel A.6. Auf der Menge $\mathbb{R}$ ist die durch „kleiner-oder-gleich“ definierte Relation eine lineare Ordnungsrelation, z.B. für $x = 1$ und $x' = 2$ gilt entweder $(1; 2) \in R$ $(1 \leq 2)$ bzw. $(2; 1) \in R$ $(2 \leq 1)$.
Auf der Potenzmenge *P(M)* einer Menge M, d.h. auf der Menge aller Teilmengen einer Menge M ist die durch „Teilmenge von“ zu charakterisierte Relation eine (nicht lineare) Ordnungsrelation, z.B. für $x = \{1, 3, 5\}$ und $x' = \{2, 4, 6\}$ gilt weder $x \subseteq x'$, noch $x' \subseteq x$.

A.2 Abbildungen

A.2.1 Definition einer Abbildung

Bei der im Beispiel A.2 angegebene Relation $R : x \to x^2$ steht jedes Element der Menge $M = \mathbb{Z}$ mit genau einem Element der Menge $M' = \mathbb{N}_0$ in Relation. Man sagt dann auch: Jedem Element von $\mathbb{Z}$ wird genau ein Element von $\mathbb{N}_0$ zugeordnet. Relationen dieser speziellen Art heißen Abbildung.

Definition A.2. Eine Relation R zwischen zwei Mengen M und M' heißt eine *Abbildung von M in M'*, wenn es zu jedem $x \in M$ genau ein $x' \in M'$ mit $(x; x') \in R$ gibt, also $(x; x') \in R \wedge (x; x'') \in R \Rightarrow x' = x''$ für alle $x \in M$ gilt.
Die Menge M heißt *Urbildmenge*, *Originalmenge* bzw. *Definitionsmenge* von R, die Menge M' *Zielmenge* von R, die Menge aller $x' \in M'$, für die es ein $x \in M$ mit $(x; x') \in R$ gibt, *Bildmenge* bzw. *Wertemenge* von M bei einer Abbildung R.

Bemerkung. Die Bildmenge ist also entweder eine echte Teilmenge oder gleich der Zielmenge M'.

Ist α eine Abbildung der Menge M in die Menge M' mit $\alpha : M \to M'$. Gilt $(x; x') \in \alpha$ für $x \in M$ und $x' \in M'$, dann wird x auf x' abgebildet: $x \mapsto x'$.

x heißt *Urbild* bzw. *Original* von x', x' das *Bild* von x bei der Abbildung α. Für x' schreibt man auch $\alpha(x)$, also $x' = \alpha(x)$. Für die Bild- bzw. Wertemenge von $\mathcal{A}$ bei der Abbildung α schreibt man $\alpha(M) := \{\alpha(x) \mid x \in M\}$. Ist $N \subseteq M$, so heißt $\alpha(N) := \{\alpha(x) \mid x \in N\}$ das Bild von N bei der Abbildung α. Für $\alpha(N)$ gilt: $\alpha(N) \subseteq \alpha(M) \subseteq M'$.

Sonderfall:. Ist $M = M'$, so heißt α eine Abbildung von M in sich. Ein Element x, für das $\alpha(x) = x$ gilt, heißt *Fixelement* von α. Gilt $\alpha(x) = x$ für alle $x \in M$, so heißt die Abbildung α die *Indentität* bzw. *identische Abbildung von M.*

A.2.2 Gleichheit von Abbildungen

Zwei Mengen A und B heißen gleich, wenn jedes Element von A auch Element von B ist und umgekehrt. Da Abbildungen Relationen, also Teilmengen von kartesischen Produkten sind, gilt:

Satz A.1. Zwei Abbildungen $\alpha : M \to M'$ und $\beta : M \to M'$ sind genau dann gleich, wenn $\alpha(x) = \beta(x)\ \forall x \in M$ gilt.

A.2.3 Festlegung einer Abbildung

Die Festlegung einer Abbildung $\alpha : M \to M'$ kann auf zwei Arten erfolgen:

1. durch Angabe (Aufzählen) aller Paare $(x; x') \in \alpha$
2. durch Angabe einer Abbildungsvorschrift (Zuordnungsvorschrift), d.h. einer Vorschrift, die für alle $x \in M$ eindeutig ein $x' \in M'$ festlegt.

Das erste Verfahren kann in Umgebungen eingesetzt werden, in der M eine endliche Menge ist. Ist M eine unendliche Menge bzw. verändert sich die Zusammensetzung der Menge über die Zeit (Elemente werden hinzugefügt bzw. aus der Menge entfernt), so läßt sich eine Abbildung nur auf die zweite Art (allgemeiner) festlegen.

A.2.4 Injektive, surjektive und bijektive Abbildungen

Abbildungen einer Menge M in eine Menge M', bei denen jedes Element von M' Bild von höchstens einem Element von M oder Bild von mindestens einem Element von M ist, haben spezielle Namen:

Definition A.3. Ist bei einer Abbildung $\alpha : M \to M'$ jedes Elements von M' Bild *höchstens* eines Elements von M, gilt also: $\alpha(x_1) = \alpha(x_2) \Rightarrow x_1 = x_2$, so heißt α eine injektive Abbildung von M nach M', bzw. *Injektion*. Zu beachten gilt, dass nicht notwendigerweise jedes Element aus M' eine Zuweisung besitzt ($\alpha(M) \subseteq M'$).

Definition A.4. Ist bei einer Abbildung $\alpha : M \to M'$ jedes Element von M' Bild von *mindestens* einem Element von M, gilt also: $\alpha(M) = M'$, so heißt α eine surjektive Abbildung von M auf M' bzw. *Surjektion*.

Definition A.5. Ist bei einer Abbildung $\alpha : M \to M'$ jedes Element von M' Bild von *genau* einem Element von M, ist also α sowohl surjektiv als auch injektiv, so heißt α eine eineindeutige (bijektive) Abbildung von M auf M' bzw. *Bijektion*.

Sonderfall:. Ist $M = M'$, so heißt α eine eineindeutige Abbildung von M auf sich, bzw. eine Bijektion auf M.

A.2.5 Umkehrabbildung

Bei einer bijektiven Abbildung $\alpha : M \to M'$ gibt es zu jedem $x' \in M'$ genau ein $x \in M$ mit $\alpha(x) = x'$. In diesem Fall ist durch die Abbildungsvorschrift $x \mapsto x'$ eine eineindeutige Abbildung von M' auf M mit $x' = \alpha(x)$ festgelegt.

Definition A.6. Es sei $\alpha : M \to M'$ eine bijektive Abbildung mit $\alpha(x) = x'$. Die durch $\alpha^{-1}(x') = x :\Leftrightarrow x' = \alpha(x)$ festgelegte Abbildung $\alpha^{-1} : M' \to M$ heißt dann die Umkehrabbildung von α bzw. die zu α *inverse Abbildung*.

Da α^{-1} eine bijektive Abbildung ist, gibt es auch zu α^{-1} eine inverse Abbildung. Für diese Abbildung $(\alpha^{-1})^{-1}$ gilt: $(\alpha^{-1})^{-1} = \alpha$, wegen $(\alpha^{-1})^{-1}(x) = \alpha(x)\ \forall x \in M$.

A.3 Algebraische Strukturen

A.3.1 Verknüpfungen

Abbildungen sind spezielle Relationen. Spezielle Abbildungen, die in der Mathematik häufig vorkommen, sind die sogenannten Verknüpfungen.

Definition A.7. Jede Abbildung $\alpha : M \times M \to M$ heißt eine *innere* bzw. *algebraische Verknüpfung auf* M. Gilt bei einer solchen Verknüpfung für $(a, b) \in M \times M$ mit $\alpha(a, b) = c \in M$, so schreibt man dafür a und b mit einem Verknüpfungszeichen, z.B mit $\square : c := a \square b$.

Bemerkung. Bei einer inneren Verknüpfung ist es hierbei sinnvoll zu fragen, ob die Verknüpfung kommutativ und assoziativ ist, d.h. ob $\forall a, b \in M$ gilt:
$a \square b = b \square a$ bzw. $\forall a, b, c \in M$ gilt $(a \square b) \square c = a \square (b \square c)$.

Beispiel A.7. Addition und Multiplikation sind innere Verknüpfungen auf den Mengen $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$. Die Subtraktion ist eine innere Verknüpfung auf den Mengen $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, dagegen nicht auf der Menge $\mathbb{N}$. Die Division ist eine innere Verknüpfung auf den Mengen $\mathbb{Q}\backslash\{0\}, \mathbb{R}\backslash\{0\}$.

Definition A.8. Jede Abbildung $\alpha : N \times M \to M$ mit $N \neq M$ heißt *äußere Verknüpfung* von M mit N. Gilt bei einer solchen Verknüpfung $(n; a) \mapsto b$ für $n \in N$ und $a, b \in M$, so schreibt man dafür a und b mit einem Verknüpfungszeichen, z.B. $\cdot : b := n \cdot a$.

Beispiel A.8. Es sei V die Menge aller Vektoren einer Ebene. Die Multiplikation eines Vektors mit einer reellen Zahl oder S-Multiplikation ist eine äußere Verknüpfung $\mathbb{R} \times V \to V$. (siehe Abbildung A.4)

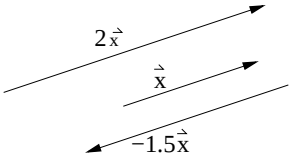

Abbildung A.4: Multiplikation eines Vektors

Definition A.9. Mengen zusammen mit Verknüpfungen heißen *algebraische Strukturen* oder *Verknüpfungsgebilde.*

A.3.2 Verknüpfungstreue Abbildungen

Unter den Abbildungen, bei denen auf der Urbildmenge M und der Zielmenge M' innere Verknüpfungen definiert sind, sind die „verknüpfungstreuen", bzw. „strukturerhaltenden" Abbildungen (die sogenannten Homomorphismen) von besonderer Bedeutung.

Definition A.10. M und M' seien Mengen mit je einer inneren Verknüpfung $\square$ bzw. $\square$'. Eine Abbildung $\alpha : M \to M'$ heißt ein *Homomorphismus* von M in M', wenn für alle $x_1, x_2 \in M$ gilt:

(H) $\alpha(x_1 \square x_2) = \alpha(x_1) \square' \alpha(x_2)$

Eine bijektive Abbildung $\alpha : M \to M'$ mit der Eigenschaft (H) heißt ein *Isomorphismus* zwischen M und M'.

Bei einem Homomorphismus $\alpha : M \to M'$ gilt für alle $x_1, x_2 \in M$, es ist gleichgültig, ob zuerst die Verknüpfung $\square$ auf M und dann die Abbildung α oder zuerst die Abbildung α und dann die Verknüpfung $\square$' auf M' ausgeführt wird. Mit anderen Worten: Das Bild der $\square$-Verknüpfung von x_1 und x_2 ist gleich der $\square$'-Verknüpfung der Bilder von x_1 und x_2.

Sonderfall:. Ist $M = M'$, so heißt α ein *Endomorphismus* von M; ist zudem α noch bijektiv, so heißt α ein *Automorphismus* von M.

Gibt es für zwei algebraische Strukturen einen Homomorphismus oder einen Isomorphismus, durch den die eine Struktur auf die andere abgebildet werden kann, so heißen die beiden algebraischen Strukturen *homomorph* bzw. *isomorph.*
Sind $(M, \square)$ und $(M', \square')$ isomorph, so schreibt man: $(M, \square) \cong (M', \square')$.

Beispiel A.9. Durch die Logarithmusfunktion $f : \mathbb{R}^+ \to \mathbb{R}$ mit $x \mapsto log(x)$ ist eine bijektive Abbildung von $\mathbb{R}^2$ auf $\mathbb{R}$ gegeben. Da $log(x_1 \cdot x_2) = log(x_1) + log(x_2)$ $\forall x_1, x_2 \in \mathbb{R}^+$ gilt, sind $(\mathbb{R}^+, \cdot)$ und $(\mathbb{R}, +)$ isomorph, also $(\mathbb{R}^+, \cdot) \cong (\mathbb{R}, +)$.

Bemerkung. Da für Verknüpfungen in homomorphen algebraischen Strukturen dieselben Gesetze gelten, ist es möglich, verschiedene mathematische Gegenstände einheitlich zu untersuchen. Es ergibt sich daraus eine außerordentlich große Straffung in der Behandlung spezieller Fragestellungen. Isomorphe Strukturen können sogar, wenn man von der speziellen Natur der Gegenstände, d.h. von der Deutung der Elemente und der Verknüpfung absieht, im abstrakten Sinne als gleich betrachtet werden. Z.B. im Fall der komplexen Zahlen mit der imaginären Einheit i sind $(\{D_0, D_1, D_2, D_3\}, \circ)$ und $(\{1, i, -1, -i\}, \cdot)$ verschiedene Deutungen (Modelle) für ein und derselben algebraische Struktur, mit $D_0 \mapsto 1, D_1 \mapsto i, D_2 \mapsto -1, D_3 \mapsto -i$ und der bijektiven Abbildung $\alpha : M \to M'$.

A.3.3 Halbgruppen

Sei $(M, \square)$ eine mathematische Struktur mit einer Menge M und einer auf ihr definierten Verknüpfung $\square$. Eine Untersuchung solcher Mengen ergibt, dass vier charakterisierende Eigenschaften existieren, anhand derer eine Einteilung möglich ist. Je nach dem, welche der vier Eigenschaften erfüllt ist, bilden diese Mengen unterschiedliche mathematische Strukturen. Die einfachste dieser Strukturen ist die Halbgruppe. Bei einer Halbgruppe wird lediglich verlangt, dass die Verknüpfung assoziativ ist.

Definition A.11. Eine Menge mit einer assoziativen Verknüpfung ist eine Halbgruppe.

(HG) *Assoziativgesetz* $\forall a, b, c \in M$ gilt: $(a \square b) \square c = a \square (b \square c)$

Beispiel A.10. Die Menge der natürlichen Zahlen mit der Addition $(\mathbb{N}, +)$ ist eine Halbgruppe. Ebenso ist $(\mathbb{N}, \cdot)$ eine Halbgruppe.

A.3.4 Monoide

Definition A.12. Eine Halbgruppe $(M, \square)$ ist ein Monoid, wenn sie ein neutrales Element enthält, d.h. folgendes Axiom erfüllt:

(M) *Existenz eines neutralen Elements.* Es gibt ein Element $e \in M$, so dass $\forall a \in M$ gilt: $a \square e = e \square a = a$

Beispiel A.11. Die Menge der natürlichen Zahlen mit der Multiplikation $(\mathbb{N}, \cdot)$ ist ein Monoid mit der 1 als neutralem Element. Die Menge der natürlichen Zahlen einschließlich der Null mit der Addition $(\mathbb{N}_0, +)$ ist ein Monoid. Die Null ist das neutrale Element bezüglich der Addition.

Satz A.2. Sei $(M, \square)$ ein Monoid. Dann ist das neutrale Element e eindeutig bestimmt, d.h. es gibt nicht mehrere neutrale Elemente in $(M, \square)$.

A.3.5 Gruppen

Definition A.13. Eine algebraische Struktur $(G, \square)$, die aus einer nichtleeren Menge G zusammen mit einer inneren Verknüpfung $\square$ besteht, heißt eine *Gruppe*, wenn folgende Axiome erfüllt sind:

(G_1) *Assoziativgesetz* $\forall a, b, c \in G$ gilt: $(a \square b) \square c = a \square (b \square c)$

(G_2) *Existenz eines neutralen Elements.* Es gibt ein Element $e \in G$, so dass $\forall a \in G$ gilt: $a \square e = e \square a = a$

(G_3) *Existenz von inversen Elementen* Zu jedem $a \in G$ gibt es ein Element $a^{-1} \in G$, so dass gilt: $a \square a^{-1} = a^{-1} \square a = e$

(G_4) *Abgeschlossenheit*: zwei Elementen $a, b \in M$ wird eindeutig ein Element $c = a \square b$ zugeordnet, das ebenfalls in M enthalten ist.

Gilt zudem noch

(G_5) *Kommutativgesetz* $a \square b = b \square a \; \forall a, b \in G$, so heißt eine solche Gruppe eine *kommutative* bzw. *Abelsche* Gruppe.

Bemerkung. Wegen der Assoziativität können Klammern ganz weggelassen werden, also $(a \square b) \square c = a \square (b \square c) = a \square b \square c$

Satz A.3. In jeder Gruppe gibt es genau ein neutrales Element.

Satz A.4. In jeder Gruppe gibt es zu jedem Element a genau ein inverses Element a^{-1}.

Satz A.5. In jeder Gruppe sind die Gleichungen $a \square x = b$ und $y \square a = b$ eindeutig lösbar.

A.3.6 Abelsche Gruppen

Definition A.14. Eine Gruppe $(G, \square)$ heißt *abelsche Gruppe*, wenn die Verknüpfung $\square$ kommutativ ist, d.h. wenn für alle Elemente $a, b \in G$ gilt:

$$a \square b = b \square a$$

Beispiel A.12. $(\mathbb{Z}, +)$ ist abelsche Gruppe.

Die Menge aller Permutationen von n Elementen mit der Komposition (Hintereinanderausführung von Abbildungen) als Verknüpfung ist eine Gruppe, jedoch keine abelsche Gruppe.

A.3.7 Untergruppen

Definition A.15. Sei $(G, \square)$ eine Gruppe und U eine Teilmenge von G. Ist durch die innere Verknüpfung $\square$ auf G eine innere Verknüpfung auf U (eine Einschränkung der Verknüpfung $\square$ auf U) gegeben, d.h. gilt $a \square b \in U \ \forall a, b \in U$ und bildet U zusammen mit dieser Verknüpfung $\triangleright$ (geschrieben: $(U, \square)\triangleright$) eine Gruppe, so heißt diese Gruppe eine *Untergruppe von* $(G, \square)$. $\square$ ist eine allgemeinere Gruppenvorschrift und $\triangleright$ eine speziellere Vorschrift, die eine Gruppe in Untergruppen zerteilen kann.

Für den Nachweis der Untergruppeneigenschaft kann folgendes Kriterium verwendet werden:

Satz A.6. Es sei $(G, \square)$ eine Gruppe und $U \subseteq G$. $(U, \square)$ ist genau dann eine Untergruppen von $(G, \square)$, wenn gilt:

(a) U ist nicht leer

(b) $a \square b \in U \ \forall a, b \in U$

(c) Gruppenaxiom (G_3) ist in $(U, \square)$ erfüllt

Beweis. *Diese Bedingungen sind notwendig, denn ist $(U, \square)$ eine Untergruppe, so gelten a), b) und c).*
Sie sind aber auch hinreichend: Nach a) ist U eine nichtleere Menge; nach b) ist $\square$ eine innere Verknüpfung auf U. Da $\square$ auf G assoziativ ist, gilt dies auch auf U. Es ist also Gruppenaxiom (G_1) in $(U, \square)$ erfüllt. Wegen c) gibt es zu jedem $a \in u$ ein Element $a^{-1} \in U$ mit $a \square a^{-1} = a^{-1} \square a = e$. Da $a \square b \in U \ \forall a, b \in U$, also auch $\forall a, a^{-1} \in U$ gilt, ist $e \in U$. Gruppenaxiom (G_2) ist daher in $(U, \square)$ ebenfalls erfüllt.
$\square$

A.3.8 Körper

Addition und Multiplikation sind innere Verknüpfungen. Diese beiden Verknüpfungen sind aber nicht voneinander unabhängig, sondern durch das Distributivgesetz miteinander verbunden. Eine Verallgemeinerung dieser Beziehung führt zum Begriffs des Körpers:

Definition A.16. Eine algebraische Struktur $(K, +, \cdot)$, die aus einer Menge K mit mindestens zwei Elementen zusammen mit zwei innere Verknüpfungen $+$ und $\cdot$ besteht, heißt ein *Körper*, wenn folgende Axiome erfüllt sind:

(K_1) $(K, +)$ ist eine kommutative Gruppe mit neutralem Element 0 (Nullelement)

(K_2) $(K \backslash \{0\}, \cdot)$ ist eine kommutative Gruppe mit neutralem Element 1 (Einselement)

(K_3) Für alle $a \in K$ gilt: $0 \cdot a = 0$

(K_4) Für beliebige $a, b, c \in K$ gilt das Distributivgesetz: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

Definition A.17. Sei dazu $(K^n, +, \cdot)$ ein Körper und $n \in \mathbb{N}$ mit $K^n = \{x = (x_1, ..., x_n) \mid x_i \in K,\ i = 1, ..., n\}$ die Gesamtheit der geordneten n-tupel von Elementen aus K^n. Für $x = (x_1, ..., x_n)$ und $y = (y_1, ...y_n)$ aus K^n erklärt man die *Addition* durch:

$$\begin{aligned} x + y &= (x_1, ..., x_n) + (y_1, ..., y_n) \\ &= (x_1 + y_1, x_2 + y_2, ..., x_n + y_n) \end{aligned}$$

Dann ist $(K^n, +, \cdot)$ eine *additive Abelsche Gruppe* mit dem Nullelement

$$0 = (0, ..., 0)$$

und den jeweiligen Inversen bzgl. ihres Original:

$$\begin{aligned} -x &= (-x_1, ..., -x_n) \in K^n \\ x &= (x_1, ..., x_n) \in K^n \end{aligned}$$

Definition A.18. Für jedes $\alpha \in K$ und jedes $x = (x_1, ..., x_n) \in K^n$ sei

$$\begin{aligned} K \times K^n &\to K^n \\ K \times K^n \ni (\alpha, x) &\mapsto \alpha \cdot x = (\alpha x_1, \alpha x_2, ..., \alpha x_n) \in K^n \end{aligned}$$

wobei $\alpha \cdot x_i$ die übliche Multiplikation in K bedeutet. Diese Rechenvorschrift wird als „Multiplikation von X mit dem Skalar $\alpha \in K$“ bezeichnet.

Bemerkung. Da $\alpha \in K$ und $x \in K^n$ aus verschiedenen Mengen sind, ist dies keine algebraisch Verknüpfung gemäß Definition A.7. Solche Verknüpfungen nennt man *äußere algebraischen Verknüpfungen* zwischen K und K^n.

A.4 Vektorräume

Die Theorie der Vektorräume ermöglicht einen axiomatischen Aufbau der Geometrie, wodurch formale und algebraische Aussagen über die jeweiligen geometrischen Lagebeziehungen zwischen Objekten ermöglicht werden.

A.4.1 Vektorräume und affine Punkträume

Definition A.19. Die Menge K^n der n-Tupel über K mit der Addition (siehe Definition A.17) und der Multiplikation mit Skalaren $\alpha \in K$ gemäß Definition A.18 heißt der *arithmetische Vektorraum K^n über K*; die Elemente $x \in K^n$ nennt man auch *arithmetische Vektoren* aus K^n.

Die Struktur des K-Vektorraums tritt mit den verschiedenartigsten konkreten Bedeutungen in der Algebra, in der Analysis, in der Geometrie, in der Physik und anderen Anwendungsgebieten auf. So lassen sich auch Multiagenten-Systeme durch die Eigenschaften des K-Vektorraums beschreiben, wie im folgenden Kapitel 5 beschrieben wird.

Definition eines affinen Punktraums

In der Elementargeometrie sind Vektoren durch (geordnete) Punktepaare definiert (siehe Abbildung A.5): Zu jedem (geordneten) Punktepaar $(A; B)$ gibt es genau einen Pfeil mit A als *Anfangspunkt* und B als *Zielpunkt*. Da dieser Pfeil Repräsentant einer ganzen Klasse parallelgerichteter Pfeile ist, gibt es also zu $(A; B)$ genau einen Vektor. Man bezeichnet ihn mit $\vec{AB}$ und nennt ihn den *Verbindungsvektor* der Punkte A und B. Für alle Punkte und Vektoren des Anschauungsraums gilt:

(A_1) Zu jedem Anfangspunkt A und zu jedem Vektor $\vec{v}$ gibt es genau einen Punkt B mit $\vec{AB} = \vec{v}$.

(A_2) Für je drei Punkte A, B, C ist die Summe der Verbindungsvektoren $\vec{AB}$ und $\vec{BC}$ gleich dem Verbindungsvektor $\vec{AC}$, als $\vec{AB} + \vec{BC} = \vec{AC}$ (*Satz von Chasles*). Siehe Abbildung A.5.

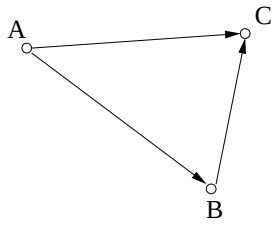

Abbildung A.5: Geometrische Interpretation des Satz von Chasles

Die Verallgemeinerung der Zusammenhänge für die Punkte und Vektoren dieses Anschauungsraums führt zu dem Begriff des affinen (verwandten) Punktraumes.

A.4.2 Ortsvektoren

Die Punkte eines Punktraumes P über einem Vektorraum V können eineindeutig den Vektoren von V zugeordnet werden, d.h. es gibt eine bijektive Abbildung von P auf V:

Wählt man nämlich einen Punkt $O \in P$, also einen gemeinsamen, festen Anfangspunkt aller Vektoren, so gibt es nach der Definition eines affinen Punktraumes zu jedem Punkt $X \in P$ genau einen Vektor $\vec{x} = \vec{OX} \in V$. Man sagt auch, der Punktraum P sei zentriert.

Umgekehrt gehört zum Punkt $O \in P$ und zu jedem Vektor $\vec{x} \in V$ nach Axiom (A_1) genau ein Punkt $X \in P$ mit $\vec{OX} = \vec{x}$. Der für diese Abbildung von P auf V fest gewählte Punkt $O \in P$ (ihm ist der Nullvektor $\vec{o} \in V$ zugeordnet) heißt *Ursprung* in P; der Vektor $\vec{x} := \vec{OX}$, also das Bild von X, heißt *Ortsvektor* von X bezüglich O.

Bemerkung. Geometrisch lässt sich die Abbildung von P auf V so interpretieren: Dem Punkt $X \in P$ wird diejenige Pfeilklasse zugeordnet, für die der Pfeil mit Anfangspunkt O und Zielpunkt X ein Repräsentant ist.

A.4.3 Vektoraddition

Auf der Menge V der Vektoren wird in der Elementargeometrie durch ein Konstruktionsverfahren eine innere Verknüpfung, die *Vektoraddition*, definiert. Zwei Vektoren $\vec{v}$ und $\vec{w}$ ordnet man dadurch einen Vektor $\vec{v} \oplus \vec{w}$ bzw. $\vec{v} + \vec{w}$ zu.

Abbildung A.6 zeigt die Konstruktion zweier Vektoren $\vec{v}$ und $\vec{w}$ und erläutert weiter, dass die Vektoraddition nicht von der speziellen Wahl der Vektoren abhängt.

Mit dem Nullvektor $\vec{0}$ als neutrales Element bildet die Menge V aller Vektoren zusammen mit der Vektoraddition, also die Struktur $(V, \oplus)$, eine *kommutative Gruppe*.

A.4.4 S-Multiplikation

Entsprechend Abbildung A.7 kann man jeder reellen Zahl $r \in \mathbb{R}$ und jedem Vektor $\vec{v} \in V$ einen Vektor $r \odot \vec{v}$ bzw. $r\vec{v}$ zuordnen. Man spricht von der *Multiplikation eines Vektors mit einer reellen Zahl* bzw. *S-Multiplikation*. Sie ist eine äußere Verknüpfung des Vektorraums der reellen Zahlen V mit der Menge der Reellenzahlen $\mathbb{R}$.

Geometrisch ergeben sich für die S-Multiplikation folgende Eigenschaften: $\forall r_1, r_2 \in \mathbb{R}$ und $\forall \vec{v}, \vec{w} \in V$ gilt:

(V_1) Assoziativgesetz: $r_1 \odot (r_2 \odot \vec{v}) = (r_1 \odot r_2) \odot \vec{v}$

(V_2) Distributivgesetz: $(r_1 + r_2) \odot \vec{v} = (r_1 \odot \vec{v}) \oplus (r_2 \odot \vec{v})$
$r_1 \odot (\vec{v} \oplus \vec{w}) = (r_1 \odot \vec{v}) \oplus (r_1 \odot \vec{w})$

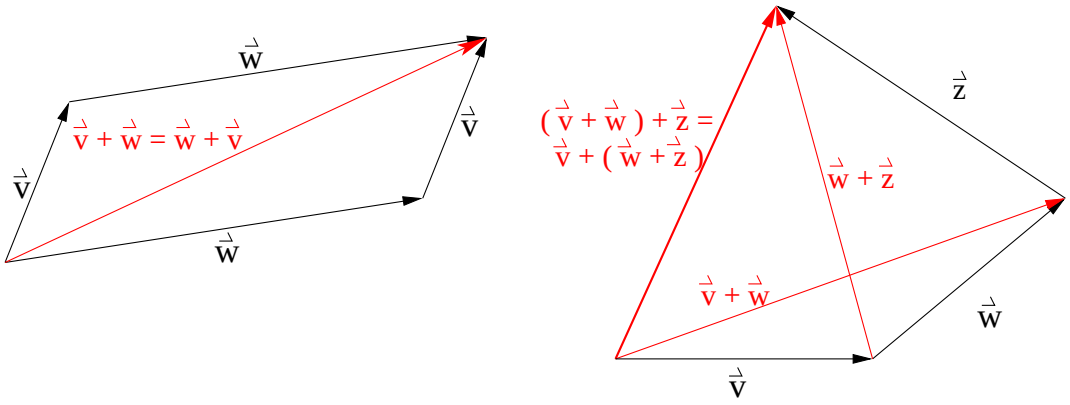

Abbildung A.6: Erläuterung des Kommutativ- und des Assoziativgesetzes

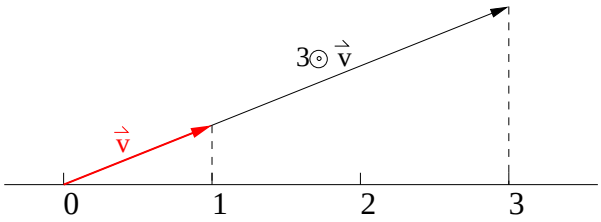

Abbildung A.7: Die S-Multiplikation

(V_3) Unitäres Gesetz: $1 \odot \vec{v} = \vec{v}$

Die Menge aller Pfeilklassen (Vektoren) zusammen mit der Vektoraddition und der S-Multiplikation nennt man *Vektorraum der Pfeilklassen* bzw. auch *Vektor-Kalkül*.

A.4.5 Definition eines Vektorraums

Definition A.20. Eine Menge V, die zusammen mit einer inneren Verknüpfung $\oplus$ eine kommutative Gruppe bildet, und ein Körper $(K, +, \cdot)$ zusammen mit einer äußeren Verknüpfung $\odot$ von V mit K heißt ein *Vektorraum* über K (geschrieben $(V, K, \oplus, \odot)$ oder einfach V), wenn für alle Elemente in K und V folgende Axiome erfüllt sind:

(V_1) Assoziativgesetz

(V_2) Distributivgesetz

(V_3) Unitäres Gesetz

Die innere Verknüpfung $\oplus$ (eine Abbildung $V \times V \to V$) heißt Vektoraddition, die äußere Verknüpfung $\odot$ (eine Abbildung $K \times V \to V$) heißt S-Multiplikation. Das neutrale Element bei der Verknüpfung $\oplus$ nennt man *Nullvektor* von V, geschrieben $\vec{0}$, das zu $\vec{v} \in V$ bzgl. $\oplus$ inverse Element *Gegenvektor* zu $\vec{v}$, geschrieben $-\vec{v}$.

Im allgemeinen verwendet man für die Verknüpfung $\oplus$ und $+$ dasselbe Zeichen, nämlich $+$; analog wird in der Schreibweise der Verknüpfungen $\odot$ und $\cdot$ kein Unterschied gemacht, indem man bei beiden das Verknüpfungszeichen weglässt.

Satz A.7. Für alle $k \in K$ und für alle $\vec{v} \in V$ gilt:

a) $k\vec{v} = \vec{o} \Leftrightarrow k = 0 \vee \vec{v} = \vec{o}$

b) $k(-\vec{v}) = (-k)\vec{v} = -(k\vec{v})$

A.4.6 Untervektorräume

Definition A.21. Sei V ein Vektorraum über einem Körper K und U eine Teilmenge von V. Ist U bezüglich der auf V definierten Vektoraddition und der S-Multiplikation selbst ein Vektorraum über K, so heißt U ein *Untervektorraum* bzw. Unterraum von V.

Ähnlich den Untergruppen (Satz A.6) erleichtert auch hier ein einfacheres Kriterium den Nachweis der Untervektorraumeigenschaft:

Satz A.8. Sei U eine Teilmenge eines Vektorraums V. U ist genau dann ein Untervektorraum von V, wenn gilt:

a) U ist nicht leer

b) $\vec{u_1} + \vec{u_2} \in U \ \forall \vec{u_1}, \vec{u_2} \in U$

c) $k\vec{u} \in U \ \forall k \in K, \forall \vec{u} \in U$

A.4.7 Linearkombination

Abbildung A.8 zeigt gemäß den Konstruktionsverfahren aus (A.4.3 und A.4.4) eine Linearkombination $\vec{z} = 2\vec{v_1} + \frac{3}{2}\vec{v_2}$. Fasst man beispielsweise $2\vec{v_1}$ und $\frac{2}{3}\vec{v_2}$ als physikalische Kräfte auf, so it $\vec{z}$ die „resultierende“ Kraft.

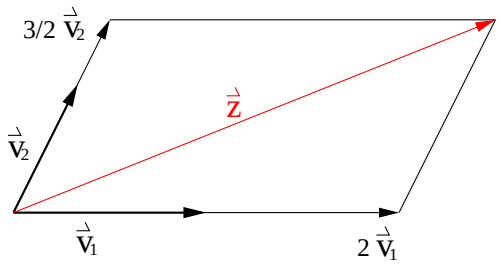

Abbildung A.8: Linearkombination zweier Vektoren $\vec{v_1}$ und $\vec{v_2}$

Definition A.22. Sei V ein Vektorraum über einem Körper K; $\vec{v_1}, \vec{v_2}, ..., \vec{v_n} \in V$, $k_1, k_2, ..., k_n \in K$. Der Vektor $k_1\vec{v_1} + k_2\vec{v_2} + ... + k_n\vec{v_n}$ heißt dann eine *Linearkombination* der Vektoren $\vec{v_1}, \vec{v_2}, ..., \vec{v_n}$.

Erzeugendensystem

Aus Abbildung A.8 erkennt man, dass jeder Vektor in der Ebene als Linearkombination von $\vec{v_1}$ und $\vec{v_2}$ dargestellt werden kann.

Definition A.23. Eine Menge $W := \{\vec{v_1}, ..., \vec{v_n}\} \subseteq V$ heißt ein Erzeugendensystem des Vektorraums V, wenn jeder Vektor aus V Linearkombination der Vektoren $\vec{v_1}, ..., \vec{v_n}$ ist.

Lineare Abhängigkeit und Unabhängigkeit

Definition A.24. Sei V ein Vektorraum über einem Körper K. Die Vektoren $\vec{v_1}, ..., \vec{v_n} \in V$ heißen *linear abhängig*, wenn es Skalare $k_1, ..., k_n \in K$ gibt, die nicht alle den Wert Null besitzen, so dass gilt: $k_1\vec{v_1} + ... + k_n\vec{v_n} = \vec{o}$.

Satz A.9. Die Vektoren $\vec{v_1}, ..., \vec{v_n} \in V (n \geq 2)$ sind genau dann linear abhängig, wenn sich (mindestens) einer von ihnen als Linearkombination der anderen darstellen lässt.

Definition A.25. Sei V ein Vektorraum über einem Körper K. Die Vektoren $\vec{v_1}, ..., \vec{v_n} \in V$ heißen *linear unabhängig*, wenn sie nicht linear abhängig sind.

Die lineare Unabhängigkeit der Vektoren $\vec{v_1}, ..., \vec{v_n}$ ist somit gleichbedeutend mit der Aussage: $k_1\vec{v_1} + ...k_n\vec{v_n} = \vec{o} \Rightarrow k_1 = k_2 = ... = k_n = 0$

Basis eines Vektorraums

Definition A.26. V sei ein Vektorraum über einem Körper K. Eine Teilmenge $B := \{\vec{b}_1, ..., \vec{b}_n\}$ von V heißt *Basis* des Vektorraums V, wenn die Vektoren $\vec{b}_1, ..., \vec{b}_n$ linear unabhängig sind und ein Erzeugendensystem bilden, d.h. eine Basis ist ein Erzeugendensystem mit *minimaler* Mächtigkeit.

Definition A.27. Die Mächtigkeit n der Basis eines Vektorraums V heißt die *Dimension* dieses Vektorraums. Schreibweise: $dim(V) = n$.

A.4.8 Vektorräume mit Norm

Die in der folgenden Definition gestellten Forderungen an eine Norm korrespondieren direkt mit der Vorstellung des „Längenbegriffs“ und erweitert den Vektorraum zum metrischen Vektorraum.

Definition A.28. Sei V ein reeller Vektorraum. Eine Abbildung $||\cdot|| : V \to \mathbb{R}, \vec{x} \mapsto ||\vec{x}||$ heißt *Norm* auf V, wenn $\forall \vec{x}, \vec{y} \in V$ und $s \in \mathbb{R}$ gilt:

(N_1) $||\vec{x}|| \geq 0$, wenn dabei $||\vec{x}|| = 0 \Rightarrow \vec{x} = \vec{0}$

(N_2) $||s \cdot \vec{x}|| = |s| \cdot ||\vec{x}||$

(N_3) $||\vec{x} + \vec{y}|| \leq ||\vec{x}|| + ||\vec{y}||$

Bei der Suche nach einer Norm ist es möglich, dass diese drei Forderungen $(N_1), (N_2), (N_3)$ einen Widerspruch in sich bergen. Deshalb muss stets die Widerspruchsfreiheit der drei Forderungen durch Angabe einer Abbildung von V in $\mathbb{R}$, die $(N_1), (N_2)$ und (N_3) erfüllt, gezeigt werden.

Beispiel A.13. Die durch $||\vec{x}||_\infty := max\{|x_1|, |x_2|, ..., |x_n|\}$ definierte Abbildung $||\cdot||_\infty : \mathbb{R}^n \to \mathbb{R}$ ist eine Norm. Denn dass $||\cdot||_\infty$ (N_1) und (N_2) erfüllt ist klar; (N_3) weist man mit Hilfe der Dreiecksungleichung für den Betrag reeller Zahlen nach:

$$\begin{aligned} ||\vec{x} + \vec{y}||_\infty &= max\{|x_1 + y_1|, |x_2 + y_2|, ..., |x_n + y_n|\} \leq \\ &\leq max\{|x_1| + |y_1|, |x_2| + |y_2|, ..., |x_n| + |y_n|\} \leq \\ &\leq max\{|x_1|, |x_2|, ..., |x_n|\} + max\{|y_1|, |y_2|, ..., |y_n|\} = \\ &= ||\vec{x}||_\infty + ||\vec{y}||_\infty \end{aligned}$$

Diese Norm heißt *Maximumnorm*.

Durch Definition A.28 wird auf $\mathbb{R}^n$ die Maximumnorm nicht als einzige Norm festgelegt. So gibt es weitere Normen, wie z.B. die *euklidische Norm* ($||\vec{x}||_2 := \sqrt{x_1^2 + x_2^2 + ...x_n^2}$), oder die Betragssummennorm ($||\vec{x}||_1 := |x_1| + |x_2| + ...|x_n|$).

Beispiel A.14. Länge von Transportwegen:
In einer Halle mit rechtwinkligem Grundriss, die nach den Himmelsrichtungen ausgerichtet ist, lagern Gegenstände, die zur Weiterverarbeitung an den Ausgang der Halle transportiert werden sollen. Jeder einzelne Platz in der Werkshalle wird durch einen Vektor $\vec{x} := (x_W; x_N) \in \mathbb{R}^2$ charakterisiert, wobei x_W und x_N die Vielfachen einer Längeneinheit angeben, um die ein dort lagernden Gegenstand in Richtung Westen und Norden transportiert werden muss, damit er zum Ausgang gelangt. Welche Länge hat der durch $(x_W; x_N) \in \mathbb{R}^2$ gegebene Weg, welche Norm im $\mathbb{R}^2$ beschreibt ihn sinnvoll?

Könnte man den Gegenstand schräg durch die Halle transportieren, wäre die *euklidische Norm* zu wählen. Andererseits ist (wegen den lagernden Gegenständen und Regalen) dieser Weg verstellt. Müssen daher vorgegebene rechtwinklige, parallel zu den Hallenmauern freigelassene Korridore für den Transport benutzt werden; dann beschreibt die *Betragssummennorm* die Länge der wirkliche zurückzulegende Wegstrecke.

Die Wahl einer Norm hängt von den Gegebenheiten und der speziellen Fragestellung eines Problems ab. So ist die Möglichkeit Transportwege in einem Lager mit Hilfe der *Betragssummennorm* berechnen zu können, die einzig sinnvolle.

Beispiel A.15. Im affinen Punktraum $P := \mathbb{R}^n$ über dem Vektorraum $\mathbb{R}^n$ mit der euklidischen Norm $||\cdot||_2$ wird der Abstand $d_2(X,Y)$ zweier Punkte $X := (x_1;...;x_n)$, $Y := (y_1;...;y_n) \in P$ durch $d_2(X,Y) = \sqrt{(x_1-y_1)^2+...+(x_n-y_n)^2}$ festgelegt. d_2 nennt man *euklidische Abstandsfunktion*.

Beispiel A.16. Im affinen Punktraum $P := \mathbb{R}^n$ über dem Vektorraum $\mathbb{R}^n$ mit Betragssummennorm $||\cdot||_1$ wird der Abstand $d_1(X,y)$ zweier Punkte $X := (x_1;...;x_n)$, $Y := (y_1;...;y_n) \in P$ durch $d_1(X,Y) = \sum_{i=1}^{n} |x_i - y_i|$ festgelegt. Häufig nennt man d_1 *Manhattandistanz* (siehe Abbildung A.9).

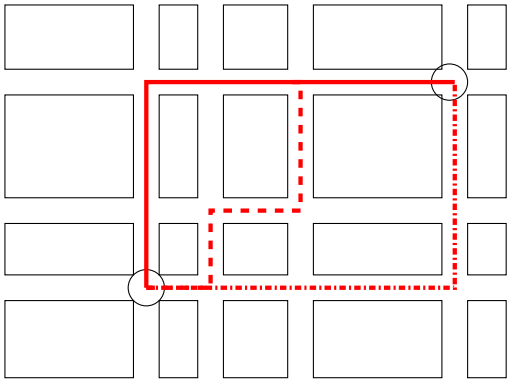

Abbildung A.9: Manhattandistanz

Anhang B

Stochastik - Einführung

Die im Folgenden beschriebenen Grundlagen sind eine Zusammenfassung der für diese Arbeit relevanten Definitionen, Sätze und Verfahren der Stochastik basierend auf den Arbeiten von [BH85], [AGP90] und [Sch85].

B.1 Grundlagen der Wahrscheinlichkeitsrechnung

Die Definition $P(A) = \frac{\text{Anzahl der für A günstige Ausgänge}}{\text{Anzahl aller möglichen Ausgänge}}$, die die Wahrscheinlichkeit $P(A)$ eines Ereignisraums A für ein Laplace-Experiment abgibt, hat bei der historischen Entwicklung der Wahrscheinlichkeitsrechnung eine wesentliche Rolle gespielt.

- *Die klassische Definition der Wahrscheinlichkeit*
  Die Anfänge der Wahrscheinlichkeitsrechnung reichen bis ins 17. Jahrhundert zurück. Die angestellten Überlegungen gingen davon aus, dass z.B. beim Werfen eines Würfels alle sechs Augenzahlen *gleich möglich* sind. Jakob Bernoulli (1654-1705) und P. S. de Laplace (1749-1827) verstanden unter der Wahrscheinlichkeit eines Ereignisses den genannten Quotienten $P(A)$.
- *Die statistische Definition der Wahrscheinlichkeit*
  Bei dem Versuch der Wahrscheinlichkeitsrechnung, Anwendungsgebiete zu erschließen, die über den Bereich der Glücksspiele hinausreichen, erwies sich die der klassischen Definition zugrundeliegende Annahme, dass alle möglichen Ausgänge *gleich wahrscheinlich* sind, als entscheidendes Hindernis. Diese Annahme verbietet es bereits, die Wahrscheinlichkeitsrechnung auf einen nichtidealen (verfälschten) Würfel anzuwenden. Um auch solche Fälle einzubeziehen, musste die Definition der Wahrscheinlichkeit anders gefasst werden. Man versuchte, von der Erfahrungstatsache auszugehen, dass die relativen Häufigkeiten $h(a)$ eines Ereignisses A bei fortgesetzter Wiederholung eines Zufallsexperiments einer bestimmten Zahl zuzustreben scheinen (vgl. B.3). R. v. Mises (1883-1953) fasste die Wahrscheinlichkeit von A als

  $$lim_{n\to\infty} h(A)$$

  also als Grenzwert der relativen Häufigkeiten bei unbegrenzter Wiederholung eines Zufallsexperiments, auf (*Statistische Definition der Wahrscheinlichkeit*). Bei dem Versuch, eine logische Präzisierung der Begriffsbildung aufzubauen, stieß man auf unüberwindbare Schwierigkeiten; es erwies sich die statistische Definition der Wahrscheinlichkeit als ungeeignet für den Aufbau einer Wahrscheinlichkeitstheorie.
- *Die axiomatische Definition der Wahrscheinlichkeit*
  Eine logisch befriedigende Fassung des Wahrscheinlichkeitsbegriffs wurde erst möglich durch den Verzicht auf eine direkte Definition. Statt einer solchen beschränkte man sich darauf, für Wahrscheinlichkeiten lediglich gewisse Eigenschaften zu fordern (*axiomatische Definition der Wahrscheinlichkeit*). A. Kolmogoroff (*1903) zeigte, dass sich bereits aus wenigen Axiomen (die im wesentlichen mit den Forderungen in B.4 übereinstimmen) eine Wahrscheinlichkeitstheorie entwickeln lässt.

B.2 Relative Häufigkeiten von Ereignissen

Häufigkeiten treten immer dann auf, wenn ein Zufallexperiment wiederholt durchgeführt wird, wie z.B. bei wiederholte Verhandlungen der Agenten. In den vorherigen Abschnitten wurde die absolute bzw. relative Häufigkeit von Ausgängen betrachtet. Den Ausgängen entsprechen im Ereignisraum die 1 elementigen Ereignisse (Elementarereignisse). Neben diesen tritt bei n-maliger Durchführung eines Zufallsexperiments aber auch jedes andere Ereignis mit einer bestimmten Häufigkeit auf. Ist der Ereignisraum sehr groß (z.B. die Menge der reellen Zahlen), so bietet es sich an, den Ereignisraum in Intervalle zu unterteilen, die für die jeweilige Anwendung von Bedeutung sind. So kann die Komplexität geeignet reduziert werden, jedoch unter Verringerung der Genauigkeit des Ereignisraums.

Definition B.1. Ist ein Ereignis A bei n Durchführungen eines Zufallsexperiments n_1-mal eingetreten, so nennt man n_1 die *absolute Häufigkeit* und $n_1 : n$ die *relative Häufigkeit* von A an dieser Serie. Es ist also:

$$h(A) = \frac{n_1}{n}$$

Die als relative Häufigkeiten auftretenden Zahlen haben besondere Eigenschaften, und führen zu folgenden Satz.

Satz B.1. Für die Elementarereignisse $E_1, ..., E_n$ eines Ereignisraumes gilt:

(R_1) $0 \leq h(E_i) \leq 1$ für $i = 1, ..., n$

(R_2) $h(E_1) + ... + h(E_n) = 1$

(R_3) $h(A) = h(E_1) + ... + h(E_k)$, wenn $A = E_1 \cup .. \cup E_k$ ist.

Relative Häufigkeiten haben außer den Eigenschaften aus Satz B.1, noch eine weitere wichtige Eigenschaft, die jedoch weniger leicht zugänglich ist:

B.3 Der empirische Befund

Beispiel B.1. Ein Würfel wurde 1000mal geworfen. Nach 50, 100, 200, 500, 1000 Würfen wurden jeweils die relativen Häufigkeiten der einzelnen Augenzahlen ermittelt. Es ergibt sich somit folgende Werte:

n	h(1)	h(2)	h(3)	h(4)	h(5)	h(6)
50	0.120	0.300	0.220	0.140	0.120	0.100
100	0.150	0.270	0.200	0.160	0.130	0.090
200	0.135	0.285	0.180	0.170	0.135	0.095
500	0.148	0.290	0.170	0.162	0.120	0.110
1000	0.141	0.293	0.173	0.159	0.126	0.108

Im dem angegebenen Beispiel scheint es offensichtlich, dass die Augenzahl 6 seltener auftritt als die Augenzahl 2. Eine solche Aussage ist sehr bedeutsam, denn bei dieser Betrachtungsweise wird die Serienlänge n vollständig außer acht gelassen. Eine solche Aussage beschreibt die Beschaffenheit des Würfels und damit, wie sich dieser Würfel in Zukunft verhalten wird. Wie entstehen solche Aussagen? Es ist zu beobachten, dass die Einträge der obigen Tabelle innerhalb einer *Spalte* nicht irgendwelche Zahlen zwischen 0 und 1 enthält, die nichts miteinander zu tun haben. Es fällt vielmehr auf, dass diese Zahlen eine deutlich wahrnehmbare gemeinsame Größenordnung besitzen. Sie ist umso besser wahrnehmbar, desto größer n ist. So liegt folgende Aussage nahe: Wenn der Würfel sehr oft geworfen wird, so sind die für $h(1), ..., h(6)$ sich ergebenden Zahlen von $0.14, 0.29, 0.17, 0.16, 0.13$ und 0.11 nicht allzu sehr verschieden.

Diese Tatsache - und allein sie - gibt uns die Möglichkeit einer Vorhersage; sie berechtigt die Behauptung, dass z.B. bei 10.000 Würfen die Augenzahl 2 rund $0.29 \cdot 10.000$mal, d.h. 2900mal, die Augenzahl 6 hingegen nur 1100mal auftreten wird.

B.4 Die mathematische Wahrscheinlichkeit

Der in Beispiel B.1 beobachtete Sachverhalt, wonach relative Häufigkeiten bei sehr vielen Durchführungen eines Zufallsexperiments einigermaßen stabil sind, trifft praktisch bei allen Zufallsexperimenten zu (sofern sich bei den Wiederholungen die ursprünglichen Gegebenheiten nicht ändern, also der Würfel nicht ausgetauscht oder beschädigt wird). Ziel ist es nun, den eben beschriebenen Sachverhalt durch eine mathematische Theorie oder auch *mathematisches Modell der Wirklichkeit* zu beschreiben.

Mathematisch bedeutsam in dem genannten Beispiel B.1 sind nicht die konkreten Ausprägungen der speziellen Häufigkeiten, sondern vielmehr dass es sie gibt. Als mathematisch bedeutsam ist:

Annahme es existiert ein Zufallsexperiment bzw. Ereignisraum mit den Elementarereignissen $E_1, ..., E_k$. Jedem Elementarereignis wird eine reelle Zahl zugeordnet. Es existiert jedoch doch keine verbindliche Vorschrift, welche Zahl dem Ereignis zugeordnet werden soll. Da die zugeordneten Zahlen insbesondere keine berechneten relativen Häufigkeiten sind, haben sie auch nicht automatisch deren Eigenschaften. Da sie jedoch stellvertretend für relative Häufigkeiten verwendet werden, wird man hinsichtlich Satz B.1 fordern, dass sie zwischen 0 und 1 liegen und die Summe 1 haben. Daraus ergibt sich folgende Definition:

Definition B.2. Ist jedem Elementarereignis E_i mit $1 \leq i \leq k$ eines Ereignisraums eine reelle Zahl $P(E_i)$ so zugeordnet, dass

(W_1) $0 \leq P(E_i)$ für $1 \leq i \leq k$

(W_2) $P(E_1) + ... + P(E_k) = 1$ gilt,

so heißen die Zahlen $P(Ei)$ *Wahrscheinlichkeiten*. Eine Abbildung P, die jedem Elementarereignis eines Ereignisraums eine Wahrscheinlichkeit zuordnet, heißt *Wahrscheinlichkeitverteilung* oder auch *Wahrscheinlichkeitsfunktion*.

Beispiel B.2. Aus Beispiel B.1 kann folgende Zuordnung getroffen werden:

E_i	E_1	E_2	E_3	E_4	E_5	E_6
$P(E_i)$	0.14	0.29	0.17	0.16	0.13	0.11

Die Forderungen (W_1) und (W_2) sind erfüllt; es handelt sich also um eine Wahrscheinlichkeitsfunktion, die z.B. E_6 die Wahrscheinlichkeit 0.11 zuordnet.

Die in Definition B.2 genannten Forderungen (W_1) und (W_2) entsprechen den in Satz B.1 geforderten Eigenschaften (R_1) und (R_2) relativer Häufigkeiten. Satz B.1 stellt noch eine weiter Forderung auf, welche die folgende Verallgemeinerung von Definition B.2 auf beliebige Ereignisse nahe legt:

Definition B.3. Sei A ein Ereignis eines Ereignisraums mit den Elementarereignissen $E_1, ..., E_k$ und einer Wahrscheinlichkeitsfunktion P. Dann setzen wir

(W_3) $P(A) = P(E_1) + ... + P(E_r)$, falls $A = E_1 \cup ... \cup E_r$,

$P(A) = 0$, falls A das unmögliche Ereignis $\emptyset$ ist,

$P(A) = 1$, falls A das sichere Ereignis S ist

und nennen $P(A)$ die *Wahrscheinlichkeit des Ereignisses A*.

Durch Definition B.3 wird, wenn die Wahrscheinlichkeiten der Elementarereignisse bekannt sind, jedem Ereignis eines Ereignisraums eine Wahrscheinlichkeit zugeordnet. Die zunächst nur auf den Elementarereignissen erklärte Wahrscheinlichkeitsfunktion wird damit auf den ganzen Ereignisraum fortgesetzt. Die fortgesetzte Funktion wird üblicherweise wiederum mit P bezeichnet. Der Ereignisraum zusammen mit P wird auch *Wahrscheinlichkeitsraum* genannt.

B.5 Folgerungen aus den Axiomen der Wahrscheinlichkeit

Wahrscheinlichkeiten haben außer den in den Definitionen B.2 und B.3 geforderten Eigenschaften noch weitere wichtige Eigenschaften, von denen die wichtigsten hier folgen. Dabei ist zu beachten: Wahrscheinlichkeiten sind keine relativen Häufigkeiten, sondern im anschaulichen Sinne „willkürlich“ festgelegte Zahlen, die den Bedingungen (W_1), (W_2) und (W_3) genügen. Die folgenden Eigenschaften treffen für Wahrscheinlichkeiten stets zu, da sie offensichtlich eine Folge von (W_1), (W_2) und (W_3) sind.

Satz B.2. Weitere Eigenschaften von Wahrscheinlichkeiten:

(W_4) $0 \leq P(A) \leq 1$ für alle Ereignisse A.

(W_5) $P(\hat{A}) = 1 - P(A)$ für alle Ereignisse A. $\hat{A}$ ist das Gegenereignis von A.

(W_6) Ist $A \subset B$, so ist $P(A) \leq P(B)$.

(W_7) Sind A und B Ereignisse mit $A \cap B = \emptyset$, so gilt: $P(A \cup B) = P(A) + P(B)$

Beweis (W_6)**.** *Für* $A = E_1 \cup ... \cup E_r$ *und* $B = E_1 \cup ... \cup E_r \cup E_{r+1} \cup ... \cup E_s$ *ist* $P(B) = P(E_1) + ... + P(E_r) + P(E_{r+1} \cup ... \cup E_s) = P(A) + P(E_{r+s} \cup ... \cup E_s)$. *Hierin ist nach* W_1 $0 \leq P(E_{r+s} \cup ... \cup E_s) \leq 1$ *und daher* $P(B) \geq P(A)$. □

Bemerkung. Die Eigenschaft (W_7) folgt unmittelbar aus Eigenschaft (W_6). Die Bedingung $A \cap B = \emptyset$ bedeutet, dass die Ereignisse A und B niemals gleichzeitig eintreten. Sie schließen sich somit gegenseitig aus, bzw. sind unvereinbar.

B.6 Streuungsmaße einer Häufigkeitsverteilung

Bei quantitativen Daten sind neben Lagemaßen (z.B. arithmetisches Mittel) noch sogenannte *Streuungsmaße* üblich, die eine Beschreibung geben, wie die Daten um ein Lagemaß gruppiert sind.

B.6.1 Variabilität von Stichprobenwerten

Beispiel B.3. Die beiden nachfolgenden Tabellen zeigen zwei verschiedene Häufigkeitsverteilungen, die offensichtlich deutliche Unterschiede aufweisen. Trotzdem haben beide Verteilungen dasselbe arithmetische Mittel von 10.

a_i	8	9	10	11
n_i	1	5	17	7

a_i	9.8	9.9	10	10.1	10.2
n_i	9	8	4	6	10

Das Beispiel B.3 zeigt, dass ein Lagemaß eine Häufigkeitsverteilung evtl. nur sehr grob beschreibt. Es erscheint daher wünschenswert, eine weitere statistische Maßzahl einzuführen, welche die Abweichungen der Stichprobenwerte vom Lagemaß, die sogenannte Variabilität der Stichprobenwerte, misst.

B.6.2 Varianz und Standardabweichung

Es liegt nahe, die Variabilität von Daten x_i durch die Summe ihrer Abweichungen $x_i - \hat{x}$ von arithmetischen Mittel messen zu wollen. Wie man zeigen kann, hat diese Summe hat jedoch stets den Wert 0 (d.h. positive und negative Abweichungen heben sich gegenseitig auf), sie eignet sich daher nicht als Streuungsmaß. Um ein echtes Streuungsmaß zu bekommen, müsste man die Absolutbeträge der Abweichungen in Betracht ziehen. Rechnerisch und aus theoretischen Erwägungen empfiehlt es sich die Eliminierung der Vorzeichen durch Quadrieren der Abweichungen zu vollziehen. Außerdem werden durch Quadrieren große Abweichungen stärker gewichtet als kleinere. Um schließlich zu verhindern, dass mit zunehmender Anzahl

der Daten zwangläufig auch der Wert des Streuungsmaßes größer wird, dividiert man die Summe der quadrierten Abweichungen durch die Anzahl der Daten. Man bestimmt also den Mittelwert der quadratischen Abweichung.

Definition B.4. Sind $x_1, ..., x_n$ Stichproben mit dem arithmetischen Mittel $\hat{x}$, so heißt die Zahl

$$\hat{s}^2 = \frac{1}{n}[(x_1 - \hat{x})^2 + ... + (x_n - \hat{x})^2] =$$

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x})^2$$

Varianz (oder *Streuung*) von $x_1, ..., x_n$. Die Quadratwurzel aus der Varianz wird mit $\hat{s}$ bezeichnet und *Standardabweichung* genannt. Kommen unter den Stichprobenwerte $x_1, ..., x_n$ die Merkmalsausprägungen $a_1, ..., a_k$ mit den absoluten Häufigkeiten $n_1, ..., n_k$ vor, so ist

$$\hat{s}^2 = \frac{1}{n}[(a_1 - \hat{x})^2 \cdot n_1 + ... + (a_k - \hat{x})^2 \cdot n_k] =$$

$$= \frac{1}{n}\sum_{i=1}^{k}(a_i - \hat{x})^2 \cdot n_i.$$

Sind $h_1, ..., h_k$ die relativen Häufigkeiten von $a_1, ..., a_k$, so ist

$$\hat{s}^2 = (a_1 - \hat{x})^2 \cdot h_1 + ... + (a_k - \hat{x})^2 \cdot h_k =$$

$$= \sum_{i=1}^{k}(a_i - \hat{x})^2 \cdot h_i.$$

B.7 Das Gesetz der großen Zahl

Satz B.3. *(Gesetz der großen Zahl)*
Die Wahrscheinlichkeit dafür, dass bei n unabhängigen Durchführungen eines Zufallsexperiments die relative Häufigkeit eines Ereignisses A von der Wahrscheinlichkeit $P(A)$ um weniger als eine beliebig vorgegebene Zahl $e > 0$ abweicht, strebt für $n \to \infty$ gegen 1.

Die Bedeutung dieses Satzes wird deutlich, wenn man die Aussage mit dem in B.3 Gesagten vergleicht. Wird ein Zufallsexperiment unter gleichbleibenden Bedingungen n-mal durchgeführt, so schwanken die relativen Häufigkeiten für ein Ereignis A mit zunehmenden Werten von n immer weniger. Es scheint als ob sich die relativen Häufigkeiten auf eine gewisse Zahl hin stabilisieren. Dieser Befund ist jedoch als empirisches Ergebnis mathematisch grundsätzlich nicht beweisbar.

Als mathematisches Analogon liefert uns die Theorie den obigen Satz B.3. Er besagt, dass ein Ereignis A, dessen Wahrscheinlichkeit $P(A)$ ist, bei n Durchführungen eines Zufallsexperiments mit einer relativen Häufigkeit auftreten muss, die mit zunehmenden n immer wahrscheinlicher in der Nähe einer festen Zahl liegen. Der Satz liefert damit die theoretische Begründung für den ursprünglich beobachteten empirischen Sachverhalt. Durch seine Aussage, dass es sich bei dieser festen Zahl um die Wahrscheinlichkeit $P(A)$ handelt, rechtfertigt er gleichzeitig theoretisch die näherungsweise Bestimmung von Wahrscheinlichkeiten mittels relativer Häufigkeiten.

B.8 Erwartungswert

Zur Beschreibung einer Häufigkeitsverteilung bei relativen Häufigkeiten wird unter anderem z.B. das arithmetische Mittel

$$\hat{x} = \sum_{i=1}^{k} a_i \cdot h(a_i)$$

verwendet. Betrachtet man anstelle der Merkmalsausprägungen a_i die diesen zugeordneten Werte x_i einer Zufallsvariablen X, so ist

$$\hat{x} = \sum_{i=1}^{n} x_i \cdot h(X = x_i)$$

der entsprechende Mittelwert. Es handelt sich um das arithmetische Mittel derjenigen Werte, die man erhält, wenn man das zugrundeliegende Zufallsexperiment m-mal durchführt und aus den dabei erhaltenen Werten von X das arithmetische Mittel bildet. Ist m groß, so ändern sich die relativen Häufigkeiten $h(X = x_i)$ mit zunehmendem m nur wenig. Dies legt nahe (vgl. B.3) und unter Verwendung der Aussage von Satz B.3,

$$\hat{x} = \sum_{i=1}^{n} x_i \cdot P(X = x_i)$$

als Lagemaß für die Wahrscheinlichkeitsverteilung von X zu verwenden.

Definition B.5. Ist X eine Zufallsvariable, welche die Werte $x_1, ..., x_n$ annimmt, so heißt die reelle Zahl $E(X)$ mit

$$E(X) = \sum_{i=1}^{n} x_i \cdot P(X = x_i)$$

Erwartungswert der Zufallsvariablen X.

B.9 Bernoulli-Kette

Die einfachsten Zufallsexperimente sind solche mit genau 2 Ergebnissen, wie z.B. das einmalige Werfen einer Münze. s ist üblich, eines der Ergebnisse als *Treffer*, das andere als *Niete* zu bezeichnen und für den Treffer 1 und für die Niete 0 zu schreiben. Damit ist $\omega_0 := \{0; 1\}$ ein Ereignisraum für ein solches Zufallsexperiment.

Definition B.6. Ein Zufallsexperiment heißt *Bernoulli-Experiment* mir dem Parameter p, wenn für seinen Wahrscheinlichkeitsraum gilt:

1. Der Ereignisraum ist $\omega_0 = \{0; 1\}$.
2. $P(\{1\}) = p$, d.h. die Trefferwahrscheinlichkeit ist p.

In vielen Problemen treten Serien von Bernoulli-Experimenten auf, z.B.:

- der n-fache Münzwurf mit 'Alder' jeweils als Treffer
- der n-fache Würfelwurf mit 'Sechs' jeweils als Treffer

Das Kennzeichen bei solchen Serien ist, dass die Wahrscheinlichkeit für einen Treffer von Versuch zu Versuch gleich bleibt und dass sich die Versuche gegenseitig nicht beeinflussen.

Aus diesen Beispielen kann das stochastische Modell der *Bernoulli-Kette* abstrahiert werden, das eine Serie von Bernoulli-Experimenten beschreibt.

Definition B.7. Ein Zufallsexperiment heißt *Bernoulli-Kette* der Länge n mit dem Parameter p, wenn für seinen Wahrscheinlichkeitsraum (ω, P) gilt:

1. $\omega = \{0; 1\}^n$ = Menge aller n-Tupel auf {0;1}.
2. Its ω ein n-Tupel mit genau k Einsen, so ist $P(\{\omega\}) := p^k(1-p)^{n-k}$, d.h. die Wahrscheinlichkeit für eine bestimmte Serie mit genau k Treffern ist $p^k(1-p)^{n-k}$, $k \in \{0, 1, ..., n\}$.

Bemerkung. Für die Anwendung von Bernoulli-Ketten ist zu beachten:

1. Für $p = 0$ und $k = 0$ bzw. $p = 1$ und $k = n$ versagt die Formel für $P(\{\omega\})$, da sich der unbestimmte Faktor 0^0 ergibt. Man überlegt sich leicht, dass es sinnvoll ist, in beiden Fällen $P(\{\omega\}) = 1$ zu setzen.
2. Üblicherweise setzt man $q := 1 - p$, so dass $P(\{\omega\}) = p^k q^{n-k}$ gilt.

3. Für $n = 1$ ist die Bernoulli-Kette natürlich ein Bernoulli-Experiment.

Die in Definition B.7 festgelegte Bernoulli-Kette hat genau die Eigenschaften, die von einer Serie von Bernoulli-Experimenten erwartet wird. Es gilt:

Satz B.4. Bei einem Zufallsexperiment mit dem Ereignisraum $\omega = \{0; 1\}^n$ und einer Wahrscheinlichkeitsverteilung P bedeute *Treffer an der i-ten Stelle* das Ereignis $A_i :=$ Menge aller n-Tupel mit 1 an der i-ten Stelle. Ein solches Zufallsexperiment ist eine Bernoulli-Kette der Länge n mit dem Parameter p genau dann, wenn gilt:

1. $P(A_i) = p$ für alle i.
2. Die A_i sind stochastisch unabhängig.

Literaturverzeichnis

[AGP90] H. Athen, H. Griesel, and H. Postel. *Mathematik heute - Leistungskurs Stochastik.* Number ISBN 3-507-83093-0. Schroedel - Schöningh, 1990.

[All] *URL: http://space.tin.it/computer/csadun/software/allen_alg/#2.*

[All83] J. F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM 26*, 1983.

[Art] *URL: http://www.honikel.de/gg8_11.htm/.*

[Axe84] R. Axelrod. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[BC98] P. Bocheck and S.-F. Chang. Content-based Dynamic Resource Allocation for VBR Video in Bandwidth Limited Networks. *Proceedings of the Sixth IEEE/IFIP International Workshop on Quality of Service (IWQoS'98), Napa, California*, 1998.

[BD03] C. H. Brooks and E. H. Durfee. Congregation Formation in Multiagent Systems. *Proceedings of the Conference of Autonomous Agents and Multiagent Systems (AAMAS'03)*, pages 145–170, 2003.

[BFV98a] H.-J. Bürckert, K. Fischer, and G. Vierke. Transportation Scheduling with Holonic MAS - The TeleTruck Approach. *Proceedings of the Third International Conference on Practical Applications of Intelligent Agents and Multiagents (PAAM´98)*, 1998.

[BFV98b] H.-J. Bürckert, K. Fischer, and G. Vierke. Transportation Scheduling with Holonic MAS - The TeleTruck Approach. In *PAAM*, 1998.

[BG98] A. H. Bond and L. Gasser. Readings in Distributed artifical Intelligence. *Morgan Kaufmann Publishers Inc.*, 1998.

[BGV92] B. E. Boser, L. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, D. Haussler, Ed.*, pages 144–152, 1992.

[BH85] F. Barth and R. Haller. *Stochastik.* Number ISBN 3-431-02511-0. Ehrenwirth Verlag, 1985.

[BHM92] A. Bachem, W. Hochstättler, and M. Malich. Simulated Trading: A New Approach For Solving Vehicle Routing Problems. Technical report, Mathematisches Institut der Universität zu Köln, Dezember 1992.

[Bic99] M. Bichler. Decision Analysis - A Crtitical Enabler for Multi-Attribute Auctions. *Global Networked Organisations, Proceedings of the 12th International Bled Electronic Commerce Conference*, 1999.

[BJT00] F. M. T. Brazier, C. M. Jonker, and J. Treur. Compositional Design and Reuse of a Generic Agent Model. *Applied Artificial Intelligence Journal*, (Vol 14):491–538, 2000.

[BKS03] B. Blankenburg, M. Klusch, and O. Shehory. Kernel-Stable Coalitions Between Rational Agents. *Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS)*, 2003.

[BN95] H.-J. Bürckert B. Nebel, and. Reasoning About Temporal Relations: A Maximal Tractable Subclass of Allen's Interval Algebra. *Journal of the ACM, Volume 42, Issue 1*, 1995.

[Bos02] D. Boswell. Introduction to Support Vector Machines. 2002.

[BS98] C. Beam and A. Segev. Auctions on the Internet: a Field Study. *University of Califoria, Berkeley, The Fisher Center for Management and Information Technology (CITM), Publication 98-WP-1032*, 1998.

[BSS96] C. Beam, A. Segev, and J. G. Shantikumar. Eletronic Negotiation through Internet-based Auctions. *CITM working Paper 96-WP-1019*, 1996.

[Car98] E. Caruana. *Der Krankendienst, 71 Jahrgang Fachlexikon d. soz. Arbeit (1993) Hrsg.: Deutscher Verein für öffentliche und private Fürsorge.* Eigenverlag, 1998.

[CG97] H. Corsten and R. Gössinger. Entwurf eines konzeptionellen Rahmens für ein Multiagentensystem zur integrativen Unterstützung der Produktionsplanung und -steuerung. Technical Report Nr. 13, Universität Kaiserslautern, 1997.

[CH98] A. K. Caglayan and C. G. Harrison. *Intelligente Software- Agenten: Grundlagen, Technik und praktische Anwendung im Unternehmen.* Carl Hanser Verlag München Wien, 1998.

[Che99] F. C. Cheong. *Internet Agents: Spiders, Wanderers, Brokers, and Bots.* New Riders Publishing, Indianapolis, 1999.

[CL91] P. R. Cohen and H. J. Levesque. Teamwork. *Nous, Vol. 25*, pages 487–512, 1991.

[CL02a] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for Support Vector Machines. *http://www.csie.ntu.edu.tw/ cjlin/papers/libsvm.ps.gz*, 2002.

[CL02b] Chih-Chung Chang and Chih-Jen Lin. Libsvm: Introduction and Benchmarks. *http://www.csie.ntu.edu.tw/ cjlin/papers/q2.ps.gz*, 2002.

[CMM97] A. Chavez, A. Moukas, and P. Maes. Challenger: A Multi-agent System for Distributed Resource Allocation. *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, 1997.

[CS95] R. Conte and J. S. Sichman. DEPNET: How to benefit from social dependence. *Journal of Mathematical Sociology, 20(2-3)*, pages 161–177, 1995.

[CV95a] C. Cortes and V. Vapnik. *Support vector networks. Machine Learning, 20.* 1995.

[CV95b] C. Cortes and V. N. Vapnik. Support Vector Networks. *Machine Learning, Vol.20*, pages 273–297, 1995.

[Dom] *URL: http://domino.mpi-sb.mpg.de/internet/news.nsf/Spotlight/20031031.*

[Dom93] I. Domowitz. Automating the Continous Double Auction in Practice: Automated Trade Execution Systems in Financial Markets. *Friedman, D. und J. Rust, The Double Auction Market - Institutions, Theories, and Evidence, Proceedings of the Workshop on Double Auction Markets, Addison-Wesley, Massachusetts*, 1993.

[EBT01] V. Ermolayev, S. Borue, and V. Tolok. Co-Operation Tasks Execution by the Coalitions of Rational Software Agents. *Fifth International Workshop CIA-2001 on cooperative information agents*, 2001.

[ESR+00] N. Elouardi, J. Steuer, M. Radimirsch, H. Fette, and K. Jobmann. Proactive Dynamic Resource Distribution Based on Borrowing and Control Engineering Methods for GSM-like Systems. *PIMRC 2000 -The 11th IEEE International Symposium on Personal, Indoor and Mobile Radio Communication-18-21 September 2000, London UK*, 2000.

[Fer01] J. Ferber. *Multiagentensysteme: Eine Einführung in die Verteilte Künstliche Intelligenz.* Addison-Wesley Verlag, 2001.

[FK93] K. Fischer and N. Kuhn. A dai approach to modeling the transportation domain. Technical report, RR-93-25, Deutsches Forschungszentrum für Künstliche Intelligenz, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, 1993.

[FN71] R. E. Fikes and N. Nilsson. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, (5(2)):189–208, 1971.

[Fri93] D. Friedman. The Double Auction Market Institution: A Survey. *Proceedings of the Workshop on Double Auction Markets, Addison-Wesley, Massachusetts*, 1993.

[Gab97] Gabler. *Gabler Wirtschaftlexikon, 14. Auflage.* Gabler, Wiesbaden, 1997.

[Ger99a] A. Gerber. Erweiterung des R/3-Warenwirtschaftssystems durch Integration des TeleTruck-Transportplanungssystems. Diplomarbeit, Universität des Saarlandes, 1999.

[Ger99b] C. Gerber. *Self-Adaptation and Scalability in Multi-Agent Societies.* PhD thesis, Universität des Saarlandes, 1999.

[GJ98] M. A. Gibney and N. R. Jennings. Dynamic Resource Allocation by Market-Based Routing in Telecommunications Networks. *Intelligent Agents for Telecommunication Applications, Proceedings of the Second International Workshop on Intelligent Agents for Telecommunication (IATA'98)*, 1998.

[GK01] A. Gerber and M. Klusch. CASA: Agents for Mobile Integrated Commerce in Forestry and Agriculture. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01), August 4-10, Saettle, Washington, USA, AAAI Press / Morgan Kaufmann Publisher*, 2001.

[GK02] A. Gerber and M. Klusch. *CASA: Agent-Based Integrated Services Network for Timber Production and Sales. Journal of Intelligent Systems (Vol.1) - IEEE.* 2002.

[GK03] A. Gerber and M. Klusch. Forming Dynamic Coalitions of Rational Agents by use of the DCF-S Scheme. *Proceedings of the The second International Joint Conference on AUTONOMOUS AGENTS and MULTI-AGENT SYSTEMS (AAMAS03)*, 2003.

[GK04] A. Gerber and M. Klusch. AGRICOLA - Agenten fuer mobile Planungsdienste in der Landwirtschaft. *Journal of Kuenstliche Intelligenz - in Press. Vol 1/04 (February) arendtap Desktop Publishing - Agentur, Verlags- und Vertriebs GmbH*, 2004.

[GKK02] A. Gerber, N. Kammenhuber, and M. Klusch. CASA: A Distributed Holonic Multiagent Architecture for Timber Production. *Proceedings of the The first International Joint Conference on AUTONOMOUS AGENTS and MULTI-AGENT SYSTEMS (AAMAS02)*, 2002.

[GKRZ01] A. Gerber, M. Klusch, C. Ruß, and I. Zinnikus. Holonic Agents for the Coordination of Supply Webs. *Proceedings of the 5th International Conference on Autonomous Agents*, (ACM ISBN 1-58113-326-X):73–75, 2001.

[GL03] N. Griffiths and M. Luck. Coalition formation through motivation and trust. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia*, pages 17–24, 2003.

[GM98] R. Gutman and P. Maes. Cooperative vs. Competitive Multi-Agent Negotiation in Retail Electronic Commerce. *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98), Paris*, 1998.

[GR00] A. Gerber and C. Ruß. A Holonic Coordination Infrastructure for Agent-Based Supply Webs. *Proceedings of the Eight Annual Workshop on Information Technologies and Systems WITS'2000*, pages 7–12, 2000.

[GR01a] A. Gerber and C. Ruß. A Holonic Multi-Agent Co-Ordination Server. *Proceedings of the 14th International FLAIRS Conference*, (ISBN 0-1-57735-133-9):200–204, 2001.

[GR01b] A. Gerber and C. Ruß. A Holonic Multi-Agent Infrastructure for Electronic Procurement. *Proceeding of the 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (AI 2001), Advances in Artificial Intelligence - Springer*, (ISBN 3-540-42144-0):16–25, 2001.

[GR01c] A. Gerber and C. Ruß. Holonic Agents for the Simulation of Supply Webs. *Proceedings of the second Workshop on Agent Based Simulation II*, (ISBN 1-56555-215-6):30–35, 2001.

[GRK02] A. Gerber, C. Ruß, and M. Klusch. An Agent-Based Approach for Integrating Logistics Services into a B2B Marketplace. *Proceedings of the International Conference of Internet Computing (IC'02) (Volume III)*, (ISBN 1-892512-37-8):705–712, 2002.

[GRK03] A. Gerber, C. Ruß, and M. Klusch. Supply Web Co-ordination by an Agent-based Trading Network with Integrated Logistics Services. *International Journal of Electronic Commerce Research and Applications. Vol 2, Issue 2 - Elsevier*, (ISSN 1567-4223):133–145, 2003.

[GS01] J. A. Giampapa and K. Sycara. Conversational Case-Based Planning for Agent Team Coordination. *Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR 2001, Springer-Verlag, Berlin Heidelberg, Vol. 2080*, pages 189–203, 2001.

[GSV99a] C. Gerber, J. Siekmann, and G. Vierke. Flexible Autonomy in Holonic Agent Systems. *Proceedings of the 1999 AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.

[GSV99b] C. Gerber, J. Siekmann, and G. Vierke. Holonic Multi-Agent Systems. Technical Report Research Report RR-99-03, DFKI, 1999.

[GSW98] Gomber, Schmidt, and Weinhardt. Efficiency incentives and computational tractability in the coordination of multiagent systems. *Proceedings of the Workshop Kooperationsnetze und Elektronische Koordination*, 1998.

[GVZ00] A. Gerber, G. Vierke, and I. Zinnikus. Generic Modelling of Multi-Agent Solutions for Transportation Domains. *Proceedings of the Workshop 'Agent Technologies and Their Application Scenarios in Logistics' on the 14th European Conferene on Artificial Intelligence (ECAI 2000 - Berlin)*, pages 29–31, 2000.

[Hau98] C. Haug. *Erfolgreich im Team.* Deutscher Taschenbuch, 1998.

[Heu84] H. Heuser. *Lehrbuch der Analysis, Teil 1.* Teubner, Stuttgart, 1984.

[HI01] M. J. Holler and G. Illing. *Einführung in die Spieltheorie.* Springer Heidelberg, 4th edition, 2001.

[His] URL: http://library.thinkquest.org/2705/history.html.

[HW98] J. Hu and M. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 242–250, 1998.

[Jen95] N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence, Vol. 75*, pages 195–240, 1995.

[JT01] C. M. Jonker and J. Treur. An Agent Architecture for Multi-Attribute Negotiation. *Proceedings of the senteenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, (ISBN 1-55860-777-3, Volume 2):1195–2001, 2001.

[Kat93] J. R. Katzenbach. *Der Schlüssel zur Hochleistungsorganisation.* 1993.

[KBF+02] M. Klusch, H.-J. Bürckert, P. Funk, A. Gerber, and C. Ruß. *Applications of Information Agents Systems. Journal of L.C. Jain (Ed.), Intelligent Agents and Their Applications.* Number ISBN 3-7908-1469-5. Physica Verlag Heidelberg, 2002.

[Ket94] S. P. Ketchpel. Forming coalitions in the face of uncertain rewards. *Proceedings of AAAI94*, pages 414–419, 1994.

[KG01] M. Klusch and A. Gerber. An Agent-Based Mobile e-Commerce Service Platform for Forestry and Agriculture. *Proceedings of the second Asia-Pacific Conference on Intelligent Agent Technology (IAT 2001), Intelligent Agent Technology - World Scientific Publishing Co. Pte. Ltd, Singapore 912805*, (ISBN 981-02-4706-0):119–123, 2001.

[KG02a] M. Klusch and A. Gerber. DCF-S: A Dynamic Coalition Formation Scheme for Rational Agents. *Proceedings of the Workshop of the first International Joint Conference on AUTONOMOUS AGENTS and MULTI-AGENT SYSTEMS (AAMAS02)*, 2002.

[KG02b] M. Klusch and A. Gerber. *Dynamic Coalition Formation among Rational Agents. Journal of Intelligent Systems (Vol.3) - IEEE.* 2002.

[KG02c] M. Klusch and A. Gerber. Issues of Dynamic Coalition Formation Among Rational Agents. *Tate. A. (ed.) Proceedings of the Second International Conference on Knowledge Systems for Coalition Operations (KSCO-2002), Toulouse, France, 23- 24 April 2002.*, pages 91–102, 2002.

[Klu96] M. Klusch. *Rational kooperative Erkennung von Interdatenbankabhängigkeiten.* PhD thesis, Chrsitian-Albrechts-Universität zu Kiel, 1996.

[Klu01] M. Klusch. Information Agent Technology for the Internet: A Survey. *Journal on Data and Knowledge Engineering, Special Issue on Intelligent Information Integration, D. Fensel (Ed.), Vol. 36(3)*, 2001.

[KP93] K. Kirchner and G. Pink. *36 Spielregeln für mehr Erfolg im Team. In: Bäck,H.: Schriftenreihe Winning-Teams.* Köln, 1993.

[KR84] J.P. Kahan and A. Rapoport. *Theories of coalition formation.* Lawrence Erlbaum Associates, New Jersey, 1984.

[Kra97] S. Kraus. Negotiation and cooperation in multi-agent environments. *Journal of Artificial Intelligence, Vol. 94(1-2)*, pages 79–98, 1997.

[Kra01] S. Kraus. *Strategic negotiation in multiagent environment.* Number ISBN 0-26211264-7. A Bradford book, 2001.

[KS96] M. Klusch and O. Shehory. A polynomial kernel-oriented coalition formation algorithm for rational information agents. *Proceedings of ICMAS-96*, pages 157–164, 1996.

[LP01] A. Latteier and M. Pelletier. *The Zope Book.* Number ISBN: 0735711372. Macmillan Computer Pub, 2001.

[LR57] R. D. Luce and H. Raiffa. *Games and Decisions.* John Wiley & Sons, 1957.

[LSM97] R. Lewicki, D. Saunders, and J. Monton. Essentials of Negotiation. *Irwin*, 1997.

[Lux95] A. Lux. *Kooperative Mensch-Maschine Arbeit - Ein Modellierungsansatz und dessen Umsetzung im Rahmen des Systems MEKKA*. PhD thesis, Universität des Saarlandes, 1995.

[MB83] J. Malik and T.O. Binford. Reasoning in Time and Space. *Proceedings of the 8th IJCAI*, pages 343–345, 1983.

[MC93] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. Technical Report Technical Report No.157, Cambridge, MA: MIT Centre for Coordination Science, 1993.

[Mey90] *Meyers Konversations-Lexikon, vierte Auflage, Band IV*. Verlag des Bibliographischen Instituts, Leipzig und Wien, 1890.

[MH69] J. McCarthy and P. J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[Mit97] T. Mitchell. *Machine Learning*. Number ISBN 0070428077. McGraw Hill, 1997.

[Mül94] J. P. Müller. A Conceptual Model of Agent Interaction. *Deen, S. M., editor, Draft proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS-94)*, (DAKE Centre, University of Keele, UK.Müller and Pischel):389–404, 1994.

[Mül96] J. P. Müller. *An Architecture for Dynamically Interacting Agents*. PhD thesis, Universität des Saarlandes, 1996.

[MM87] R. P. McAfee and J. McMillan. Auctions and Biding. *Journal of Economic Literature*, pages 699–738, 1987.

[MMR+01] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Transactions on Neural Networks, Vol 12, No 2*, 2001.

[MP94] J. P. Müller and M. Pischel. Modelling Interacting Agents in Dynamic Environments. *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)*, (Amsterdam):709–713, 1994.

[MPI] *URL: http://www.mpi-sb.mpg.de/units/ag4/*.

[MPT94] J. P. Müller, M. Pischel, and M. Thiel. A Pragmatic Approach to Modelling Autonomous Interacting Systems. *Wooldridge, M. and Jennings, N. R., editors, Pre-proceedings of the 1994 Workshop on*

Agent Theories, Architectures, and Languages, (Amsterdam):226–240, 1994.

[MSS02] L. Mönch, M. Stehli, and R. Schulz. An Agent-Based Architecture for Solving Dynamic Resource Allocation Problems in Manufactoring. *14TH EUROPEAN SIMULATION SYMPOSIUM AND EXHIBITION - Simulation in Industry Modeling, Simulation and Optimization*, (ISBN: 3-936150-22-2), 2002.

[Mül96] J. P. Müller. *The Design of Intelligent Agents - a Layered Approach.* Number Volume 1177. Lectures notes in artificial intelligence, Springer, 1996.

[Nah97] S. Nahmias. *Production and operations analysis, 3. Auflage.* Homewood, Ill.: Irwin, 1997.

[Nas50] J. F. Nash. Equilibrium Points in N-Person games. *Proceedings of the National Academy of Sciences of the Untied States of America 36*, pages 48–49, 1950.

[NAS00] H. S. Nwana, N. Azarmio, and R. Smith. The Rise of Machine Intelligence. 2000.

[NLJ96] H. S. Nwana, L. Lee, and N. Jennings. Coordination in Software Agent Systems. *BT Technology Journal, 14(4)*, pages 79–88, 1996.

[OFG96] E. Osuna, R. Freund, and F. Girosi. Support vector machines: Training and applications. *A.I. Memo AIM-1602, MIT A.I. Lab*, 1996.

[OR99] M. J. Osborne and A. Rubinstein. *A Course in Game Theory.* The MIT Press, ISBN 0-262-65040-1, 1999.

[OS01] C. Ollmert and H. Schinzer. Multiagentensysteme - Aufbau und Anwendungsbereiche in der Wirtschaftsinformatik. *HMD Praxis der Wirtschaftsinformatik, Heft 220*, 2001.

[Par98] V. Parunak. Characterizing Multi-Agent Negotiation. *Proceedings of the International Workshop on Multiagent-Systems, Negotiation Sience and Technology: Opportunities and pitfalls*, 1998.

[Pla99] J. Platt. Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods - Support Vector Learning. B. Schölkop, C.J.C. Burges and A.J. Smola, Eds, Cambridge,MA*, (185-208), 1999.

[R39] *R3-Online Dokumentation, Version 4.5A, Oktober 1998.*

[Rap70] A. Rapoport. *N-Person Game Theory.* University of Michigan. Ann Arbor, Michigan, 1970.

[RN95] S. J. Russel and P. Norvig. *Artificial Intelligence - A Modern Approach.* Number ISBN 0-13-360124-2 (number-feld). Prentice-Hall International, Inc., Englewood Cliffs, New Jersey 07632, 1995.

[RS81] J. G. Riley and W. F. Samuelson. Optimal auctions. *American Economic Review, 71/3*, 1981.

[San93] T.W. Sandholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, pages 295–308, May 1993.

[SB98] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press, Cambridge, 1998.

[SBS99] B. Schölkopf, C. J. C. Burges, and A. J. Smola. Advances in kernel methods - support vector learning. *MIT Press, Cambridge, MA*, 1999.

[Sch70] A. Schick. *in Louis C. Gawthrop, page 32.* 1970.

[Sch85] A. Schmid. *Wahrscheinlichkeitsrechnung und Statistik.* Number ISBN 3-12-739390-3. Ernst Klett Verlag, Stuttgart, 1985.

[Sch95] D. Schier. Ein Multiagentenansatz zum Lösen von Fleet-Scheduling-Problemen. Master's thesis, Universität des Saarlandes, 1995.

[SF96] J. H. Siekmann and K. Fischer. DFKI Script zu Multiagentensysteme. *URL: http://www.dfki.uni-sb.de/ kuf/mas.html*, 1996.

[SGB+02] J. A. K. Suykens, T. V. Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle. *Least Squares Support Vector Machines.* Number ISBN 981-238-151-1. World Scientific, 2002.

[SGR98] E. Stickel, H.-D. Groffmann, and K.-H. Rau. *Gabler Wirtschaftsinformatik Lexikon.* Betriebswirtschaftlicher Verlag Dr. Th. Gabler GmbH, 1998.

[She01] O. Shehory. Optimality and Risk in Purchase at Multiple Auctions. *Proceedings of 5th International Workshop on Cooperative Information Agents (CIA 2001), M Klusch, F Zambonelli (eds.), Lecture Notes in Artificial Intelligence, Vol. 2182, Springer*, 2001.

[Shi95] K. I. Shimomura. The bargaining set and coalition formation. Technical report, Brown University, Department of Economics, November 1995.

[SK96] O. Shehory and S. Kraus. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. *In Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96) / MIT Press*, pages pages 330–337, 1996.

[SK98] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Computational Intelligence, 15(3)*, pages 218–251, 1998.

[SK99] O. Shehory and S. Kraus. Feasible Formation of Stable Coalitions among Autonomous Agents in Non-super-additive Environments. *Computational Intelligence, 15(3)*, pages 218–251, 1999.

[SL95] T. Sandholm and V. R. Leser. Coalition formation among bounded rational agents. *Proceedings of IJCAI-95, Morgan Kaufman, San Francisco, California*, pages 662–669, 1995.

[SL97] T. Sandholm and V. R. Lesser. Coalition formation among bounded rational agents. *Artificial Intelligence 94(1-2), Special issue on Principles of Multi-Agent Systems*, pages 99–137, 1997.

[SLA+99] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. TohmAE. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence*, 1999.

[SMB+99] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Müller, C. Rätsch, and A. J. Smola. Input space vs. Feature space in kernal-based methods. *IEEE Transactions on Neural Networks, Vol. 10, No. 5*, pages 1000–1017, 1999.

[SPR+03] P. Scerri, D. Pynadath, P. Rosenbloom, M.Si, N. Schurr, and M. Tambe. A Prototype Infrastructure for Distributed Robot-Agent-Person Teams. *Proceedings of the second International Joint conference on agents and multiagent systems (AAMAS)*, 2003.

[SPS+01] B. Schölkopf, J. C. Platt, A. J. Smola, J. Shawe-Taylor, and R.C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation 13 (7)*, pages 1443–1471, 2001.

[SRG99] M. P. Singh, A. S. Rao, and M. P. Georgeff. Formal methods in dai: Logic-based representing and reasoning. *in G. Weiß, (ed), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, MIT Press: Cambridge, MA*, pages 331–376, 1999.

[SS01] A. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 2001.

[SSJ98] O. Shehory, K. Sycara, and S. Jha. Multi-agent Coordination through Coalition Formation. *Agent Theories, Architectures, and Languages*, 1998.

[SSM99] B. Schölkopf, A. J. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation, Vol. 10*, pages 1299–1319, 1999.

[ST02] L. K. Soh and C. TsaTsoulis. Real-Time Saticficing Multiagent Coalition Formation. *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002)*, 2002.

[SV98] T. Sandholm and N. Vulkan. Bargaining with Deadlines. *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on on Innovative Applications of Artificial Intelligence*, (ISBN 0-262-51106-1):44–51, 1998.

[SVMa] *URL: http://www.kernel-machines.org.*

[SVMb] *URL: http://www.support-vector.net/tutorial.html.*

[SYIV02] O. Shehory, J. Yen, T. Ioerger, and J. Vassileva. Workshop on Teamwork and Coalition Formation. *AAAMAS, Bologna, Italy*, (http://julita.usask.ca/coatea/index-1.htm), 2002.

[Tem99] H. Tempelmeier. *Material-Logistik. Modelle und Algorithmen f¸r die Produktionsplanung und -steuerung und das Supply Chain Management.* Berlin: Springer, 1999.

[TF99] T. Tesch and P. Frankhauser. Arbitration and Matchmaking for Agents with Conflicting Interests. *Cooperative Information Agents III, Third International Workshop, CIA' 99, Uppsala, Sweden, July 31 - August 2, 1999, Proceedings. Lecture Notes in Computer Science, Vol. 1652, Springer, 1999, ISBN 3-540-55325-8*, 1999.

[Tic03] C. Tichy. Intelligente, agenten-basierte Ressourcenverwaltung innerhalb der CASA-Dienste-Suite. Master's thesis, Universität des Saarlandes, 2003.

[TS00a] K. Sycara T. Shintani, T. Ito. Multiple Negotiations among Agents for a Distributed Meeting Schedulers. *Arbitration and Matchmaking for Agents with Conflicting Interest*, pages 435–436, 2000.

[TS00b] M. Tsvetovat and K. Sycara. Customer coalitions in the electronic marketplace. *Proceedings of 4th International Conference on Autonomous Agents, Barcelona, June, ACM Press*, 2000.

[Var99] H. R. Varian. *Grundzüge der Mikroökonomik.* 4. Auflage. Oldenbourg, 1999.

[VFS01] G. Vauvert and A. El Fallah-Seghrouchni. Coalition formation among strong autonomous and weak rational agents. *Proceedings of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW), Annecy, France, May*, 2001.

[Vic61] W. Vickrey. Counterspeculation, Auctions, and Competitive Sealed Tenders. *Journal of Finance 16*, pages 8–37, 1961.

[Vie00] G. Vierke. *TeleTruck - A Holonic Multi-Agent System for Telematics.* PhD thesis, Universität des Saarlandes, 2000.

[Voh95] R. Vohra. Coalitional non-cooperative approaches to cooperation. Technical report, Brown University, Department of Economics, June 1995.

[Wei99] G. Weiss. *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[Wie93] G. Wiendieck. *Einführung in die Arbeits- und Organisationspsychologie*. Fernuniversität Hagen TÜV Rheinland GmbH, 1993.

[WST01] D. H. Wolpert, J. Sill, and K. Tumer. Reinforcement Learning in Distributed Domains: Beyond Team Games. *Proceedings of the senteenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, (ISBN 1-55860-777-3, Volume 2):819–830, 2001.

[YkS01] J. Yamamoto and k. Sycara. A stable and efficient buyer coalition formation scheme for e-marketplaces. *Proceedings 5th International Conference on Autonomous Agents, Montreal, Canada, ACM Press, May/June*, 2001.

[ZR94] G. Zlotkin and J. S. Rosenschein. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domai. *Proceedings of AAAI94*, pages 432–437, 1994.