SimMarket:
Simulation des Abverkaufsverhaltens von Artikeln des Einzelhandels mit probabilistischen Agenten

Dissertation

zur Erlangung des Grades des Doktors der Ingenieurwissenschaften (Dr.-Ing.) der Naturwissenschaftlich-Technischen Fakultäten der Universität des Saarlandes

von

Björn Patrick Stahmer

Saarbrücken im Jahr 2006

Tag des Promotionskolloquiums:

08.09.2006

Dekan der Fakultät 6 – Naturwissenschaftlich-Technische Fakultät I:

Prof. Dr. Thorsten Herfet

Prüfungsvorsitzender:

Prof. Dr. Reinhard Wilhelm

Berichterstatter:

Prof. Dr. Jörg Siekmann

Prof. Dr. Joachim Hertel

Prof. Dr. Anthony Jameson

Akademischer Beisitzer:

Dr. Christoph G. Jung

Diese Arbeit ist meinen Eltern gewidmet.

Danksagung

Die vorliegende Arbeit entstand im Projekt *SimMarket.XT*, das vom *Bundesministerium für Bildung und Forschung* (*BMBF*) am *Deutschen Forschungszentrum für Künstliche Intelligenz* (*DFKI*) in Saarbrücken gefördert worden ist.

Ein besonderer Dank gilt Prof. Dr. Jörg Siekmann, der für die Leitung des faszinierenden Umfeldes, in dem die vorliegende Arbeit entstand, verantwortlich ist. Ich danke ihm für die Betreuung meiner Arbeit, für zahlreiche Anregungen und kreative Vorschläge.

Ich danke Prof. Dr. Anthony Jameson für die Bereitschaft die vorliegende Dissertation als Zweitgutachter zu beurteilen.

Allen Mitarbeitern und Studenten des DFKI gilt mein Dank für die angenehme und produktive Arbeitsatmosphäre. Insbesondere danke ich diesbezüglich meinem Gruppenleiter Klaus Fischer, der mir immer so gut es ging den Rücken frei gehalten hat.

Ein ganz besonderer Dank geht an meine Kollegen und Freunde Sven Jacobi und Arndt Schwaiger (die Kommissare!), die während der vergangenen drei Jahre das Büro mit mir geteilt haben. Meinem engen Freund und Projektkollegen Arndt Schwaiger bin ich zu besonderem Dank verpflichtet.

Des Weiteren möchte ich mich bei dem gesamten *Dacos*-Team bedanken, das mir bei der Umsetzung des SimMarket-Systems tatkräftig zur Seite stand. Besonderen Dank gilt Wolfgang Barth, der während des gesamten Projektes mein Mentor in praxisnahen und domänenspezifischen Fragestellungen war und Prof. Joachim Hertel, der mir die Gelegenheit gegeben hat, mich an dem Neuaufbau der Firma Dacos sogar in Form eines Gesellschafters zu beteiligen. Auch bei allen anderen Gesellschaftern der Dacos möchte ich mich für die Aufnahme bedanken.

Schließlich möchte ich mich von ganzem Herzen bei meiner Familie: meinen Eltern, meiner Schwester und meiner Frau Galina für die moralische und seelische Unterstützung in den letzten Jahren bedanken. Auch bei meinen Freunden, allen TKDlern und insbesondere bei meinem langjährigen Freund Oliver Janz, möchte ich mich für den Rückhalt bedanken, ohne sie wäre die Fertigstellung dieser Arbeit nicht möglich gewesen.

Björn Stahmer, April 2006

Kurzzusammenfassung

Diese Arbeit ist in den Bereichen Business Intelligence, multiagentenbasierte Simulation und probabilistische Netzwerke angesiedelt. Das Ziel der Arbeit ist die Entwicklung eines Entscheidungsunterstützungssystems für das Sortimentsmanagement im Einzelhandel, um die Optimierung von Preisen, Promotionen und der Sortimentszusammensetzung zu ermöglichen. Es wird gezeigt, wie man mit probabilistischen Agenten das Abverkaufsverhalten von Artikeln im Einzelhandel ex ante simulieren kann. Dazu wurde ein probabilistisches holonisches Multiagentensystem (PH-MAS) entwickelt, bei dem die Wissensbasen der Agenten mittels Data-Mining-Verfahren aus den Realdaten der Händler extrahiert werden. Die gewonnenen Abverkaufsmuster werden in so genannten Verhaltensnetzen kodiert, die für eine Simulation der repräsentierten Artikel verwendet werden. Es wird gezeigt, wie der Kern der Verhaltensnetze durch erweiterte Bayes'sche Netze realisiert werden kann. So werden die neuen Evidenzarten Soft- und Extrapolationsevidenz eingeführt und in das Simulationssystem integriert. Für die Modellierung und Simulation von globalen Abhängigkeiten zwischen Artikelagenten wird ein Verschmelzungsalgorithmus vorgestellt, der die probabilistischen Verhaltensnetze der Agenten in holonische Metanetze fusioniert. Des Weiteren wird eine mehrdimensionale Simulationssprache (MSL) für beliebige Verhaltensnetze und andere mehrdimensionale Wissensrepräsentationsformen vorgestellt. Schließlich wird eine selbst optimierende Simulationsroutine präsentiert, die beliebige zu simulierende Szenarien in Abfolgen von Netzkonfigurationen konvertiert und damit effiziente Simulationen auf der Basis von Verhaltensnetzen ermöglicht. Die entwickelten Technologien wurden vollständig in Agenten des PH-MAS integriert und mit Hilfe des neu entwickelten verteilten Agentenframeworks MASg auf der Basis der .NET-Technologie realisiert. Es wird beschrieben, wie dieses generische Multiagentensystem sukzessiv zu einem umfangreichen Simulationssystem für die Prognose von Artikelabverkaufsverhalten ausgebaut wurde.

Abstract

This work is part of the areas Business Intelligence, multiagent-based simulation and probabilistic networks. The goal of this work is the development of a decision support system for category management in the retail domain for optimizing pricing, promotions and sales mix. I will show how to simulate ex ante the sales behaviour of products with probabilistic agents. The basis of the system is a new developed probabilistic holonic multi-agent system (PH-MAS), where the knowledge bases of the agents are extracted by data mining retailers' real data. The patterns of sale will be encoded into so-called behaviour networks, which will be used for simulating the represented items. We will see how the core of the behaviour networks is realised with extended Bayesian networks. New kinds of evidences – soft and extrapolation evidences – are introduced and concretised. For modelling and simulating global dependencies between item agents a merging algorithm is presented for fusing the probabilistic behaviour networks of the agents into holonic meta networks. Additionally, I will present a concept of a multi-dimensional simulation language (MSL) for arbitrary behaviour networks and other multi-dimensional knowledge representation formalisms. Finally, a self-optimising simulation routine is presented, which converts arbitrary simulation scenarios into a sequence of network configurations for efficient simulation based on behaviour networks. All developed technologies of this work are integrated into the agents of the PH-MAS, which is realised by using the new developed distributed agent framework MASg and the .NET technology. I will show how this generic multi-agent system is successively expanded to a massive simulation system to forecast the sales behaviour of products.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Einzelhandel erlebt zurzeit eine Phase verschärfter Konkurrenz: Durch die zunehmende Globalisierung drängen immer mehr internationale Anbieter in bisher von lokalen Handelsunternehmen beherrschte Märkte vor. Gleichzeitig steigen der Margen- und damit der Kostendruck durch die Preisgestaltung von Discountern. Supermarkt- und Drogeriemarktketten sind daher gezwungen, neue Strategien zu entwickeln, um ihre Kunden ohne weitere Margeneinbußen zu binden und dadurch konkurrenz- und wettbewerbsfähig zu bleiben. Bei der Entwicklung solcher Strategien haben Handelsunternehmen heute mehr denn je die Kunden mit ihren Konsumbedürfnissen, ihrer Erwartungshaltung und den daraus resultierenden *Abverkaufsmustern* im Blick. Gelingt es einem Handelsunternehmen, das Abverkaufsmuster der Artikel zu erkennen und darauf aufbauend eine Promotions-, Preis- und Sortimentsstrategie zu entwickeln, die den Kundenerwartungen und den Unternehmenszielen entspricht, so ist dieses auch bei einem schwierigen Marktumfeld in der Lage, seine Wettbewerbsposition zu festigen oder gar zu stärken.

Ein wesentlicher Schlüssel zu verstärkter Wettbewerbsfähigkeit liegt daher für ein Handelsunternehmen im detaillierten Wissen über die internen und externen Geschäftsprozesse (engl. *Business Intelligence/Business Insight*) (Kapitel 2). Insbesondere das Wissen über das Abverkaufsverhalten der Artikel und eine darauf aufbauende Implementierung eines Systems für Sortimentsmanagement, welches optimal auf die Bedürfnisse, Erwartungen und das regional variierende Kaufverhalten seiner Kunden ausgerichtet wurde, ist für ein Einzelhandelsunternehmen wichtig, da hierdurch sowohl margenstabilisierende oder gar margenerhöhende Kaufanreize als auch eine festere Kundenbindung erreicht werden können. Dafür müssen die Marketing- und Sortimentsverantwortlichen (*Category Manager*) eines Handelsunternehmens aus der immensen Menge an gelisteten und einlistbaren Artikeln eine geeignete Teilmenge auswählen und wiederum für jeden einzelnen der ausgewählten Artikel entscheiden, ob für ihn eine Einlistung, Auslistung, Preisänderung, Platzierungsänderung oder eine Promotionsaktion durchgeführt werden soll. Je nach Handelsunternehmen kann das Sortiment aus bis zu 150.000 verschiedenen Artikeln bestehen, die es gilt aus Millionen verfügbarer Artikel auszuwählen. Aus dieser kombinatorisch explodierenden Menge an möglichen Handlungsalternativen diejenigen auszuwählen, die im Zusammenspiel mittel- und langfristig Neukunden anziehen und Stammkunden binden, ist ein überaus komplexes Entscheidungsproblem. Darüber hinaus gilt es, eine Vielzahl an sich gegenseitig bedingenden Einflussfaktoren zu berücksichtigen und auf der Basis von Erfahrungswerten diejenigen Sortimentsentscheidungen zu treffen, die sowohl warengruppenintern als auch warengruppenübergreifend das gewünschte Kundenkaufverhalten induzieren.

Die Problematik dabei besteht in erster Linie in der riesigen Anzahl der kombinatorisch möglichen Sortimentsänderungen sowie in der enormen Komplexität der externen Einflüsse und internen Abhängigkeiten. So haben alle Änderungen bezüglich eines einzelnen Artikels, z. B. eine Preis- oder Platzierungsänderung, Auswirkungen auf die Abverkäufe vieler anderer Artikel (Kannibalisierungs-, Push- oder Pulleffekte). Eine zeitnahe Rückbetrachtung der Effekte umgesetzter Veränderungen in Sortiment- und Preisgestaltung im Markt gehört heute zum Tagesgeschäft eines Sortimentsmanagers. Aus vorhandenen Erfahrungen jedoch auf die Zukunft zu schließen, unterliegt bei der Vielzahl an Einflussfaktoren (engl. *Demand Influence Factors*) ohne den Einsatz entsprechender Werkzeuge einem hohen Wagnis. Denn hierbei gilt es über die eigeninduzierten Einflüsse auf das Artikelabverkaufsverhalten hinaus, externe Einflüsse wie die Sortimentsgestaltung und zugehörige Maßnahmen der Konkurrenz, psychologische Gesichtspunkte sowie generelle Umwelteinflüsse zu beachten, wie z. B. Markttrends, saisonale Aktionen wie SSV und WSV, Sonderverkäufe zu Weihnachten, Ostern, Valentinstag, allgemeine Wirtschaftslage, Jahreszeiten, Wetterlage, demografische Daten, usw. So ist es verständlich, dass die Marketing- und Sortimentsverantwortlichen auf Grund der enormen Flut an entscheidungsrelevanten Informationen häufig nicht in der Lage sind, optimale Entscheidungen zu treffen. Zum einen, weil die Komplexität des Entscheidungsproblems die menschliche Fähigkeit der Informationsverarbeitung übersteigt. Und zum anderen, weil es Einzelhändler an Systemen fehlt, die die Auswirkungen der geplanten Entscheidungen des Sortimentsmanagements im Vorfeld ihrer Umsetzung quantitativ simulieren und prognostizieren können. Eine solche Software muss interdisziplinäre Erkenntnisse aus dem Marketing, die praktischen Erfahrungen von Sortimentsverantwortlichen mit den vielfältigen Einflussfaktoren und deren Interdependenzen und das Wissen über neueste Technologien des Data Mining, der Wissensrepräsentation, der Computersimulation und der Künstlichen Intelligenz miteinander verbinden, um das Abverkaufsverhalten der Artikel bei Maßnahmen des Händlers mit Hilfe einer Simulationssoftware (Kapitel 3.1.3) mit hoher Güte abschätzen zu können.

Da eine Software, die dies im vollen Umfang leisten kann, Handelsunternehmen gegenwärtig nicht zur Verfügung steht, werden die oben beschriebenen Entscheidungen bzw. Maßnahmen des Sortimentsmanagements von den Verantwortlichen heute in vielen Fällen inkrementell und „aus dem Bauch heraus“ getroffen und (bestenfalls) erst im Nachhinein (ex-post) evaluiert. Seit Längerem werden diese Entscheidungsprozesse der Sortimentsverantwortlichen zwar rechnerunterstützt durchgeführt, aber die Computernutzung beschränkt sich dabei in den meisten Fällen auf das Einsehen von internen Marktinformationen und das Erstellen von einfachen Reports, die keine komplexen Zusammenhänge analysieren können oder das „Durchspielen“ beliebiger „Was-wäre-wenn-Szenarien“ erlauben. Ein Beispiel für ein solches Szenario wäre: die Änderung der Preise, der Bewerbung und der Platzierung mehrerer Artikel unter der Annahme, dass neue Produkte ins Sortiment aufgenommen werden, die Mitbewerber ebenfalls den Preis einiger ähnlicher Artikel ändern und entsprechende Werbemaßnahmen durchführen, und das Weihnachtsgeschäft vor der Tür steht.

Gesucht wird also ein System, das alle wichtigen Entitäten der Einzelhandelswirtschaft, unter Berücksichtigung zahlreicher externer Einflussfaktoren und Zusammenhänge, möglichst exakt modellieren und realistisch simulieren kann. Der Schlüssel für ein solches Verfahren ist die Modellierung der Abverkaufsmuster aller Artikel, so dass diese Modelle nicht nur für einfache Prognosen, sondern für beliebige Simulationen von Was-wäre-wenn-Szenarien genutzt werden können. Besäße man das perfekte Wissen über das Abverkaufsverhalten aller Artikel und ein Verfahren dieses in ein geeignetes Modell zu überführen, so könnte man zukünftige Entwicklungen unter Berücksichtigung aller Faktoren simulieren und damit das gesamte Kundenverhalten eines Unternehmens vorhersagen.

Natürlich besitzt man kein *vollständiges* Wissen über sämtliche Einflussfaktoren, die auf das Abverkaufsverhalten aller Artikel wirken, aber nachdem sich im Einzelhandel in den letzten Jahren der Einsatz von Scannerkassen etabliert hat und sogar vereinzelt bereits Data Warehouses (Kapitel 2.2) im Einsatz sind, liegt eine Datengrundlage vor, aus der man wesentliche Erkenntnisse über das Abverkaufsverhalten gewinnen kann.

Doch wie kann man die enorme Masse der vorhandenen Daten für die Modellierung, die Analyse und die Simulation von Artikelabverkaufsverhalten nutzen? Selbst bei den innovationsfreudigsten Unternehmen, die schon seit über 10 Jahren Data Warehouses im Einsatz haben und Vorreiter bei der Implementierung von Business Intelligence (Kapitel 2) sind, ist in den letzten Jahren eine gewisse Ernüchterung eingetreten. Man hat einsehen müssen, dass die bisherige Anwendung von Business Intelligenz mit den Schwerpunkten Data Warehousing und Data Mining in vielen Fällen nicht den gewünschten Erfolg gebracht haben. Das liegt vor allem daran, dass sich die meisten Analyseverfahren nicht für Prognose und Optimierung und umgekehrt sich Prognoseverfahren nicht für die Analyse und Optimierung eignen. Klassische Optimierungsverfahren sind hoch komplex in der Modellierung und Ausführung und eignen sich meist nicht für die Analyse und Prognose. Deshalb hat der Analyst Lou Agosta in [Agosta04] die provokante These propagiert:

„Data Mining is dead – long live Predictive Analytics" [Agosta04]

Unter *Predictive Analytics* versteht er Verfahren, die Hypothesen nicht nur validieren, sondern auch neue Hypothesen aufstellen und die Zukunft unter Berücksichtigung von unstetigen Veränderungen vorhersagen können. Tabelle 1 zeigt die Unterschiede zwischen den Business-Intelligence-Paradigmen.

Die Definition des relativ neuen Begriffs *Predictive Analytics* hat sehr viele Ähnlichkeiten mit einer anderen Forschungsrichtung, die es schon seit vielen Jahren gibt – *Simulation*. Deshalb war es sinnvoll, zusätzlich zu den klassischen Bereichen Statistik und Data Mining auch in anderen Bereichen nach potenziellen Lösungen für die Analyse, Prognose und Optimierung von Artikelabverkäufen zu suchen, um eine geeignete Kombination von Verfahren aus den relevanten Forschungsbereichen – Simulation, Business Intelligence, Data Mining, Künstliche Intelligenz insbesondere Multiagentensysteme und Statistik – für die Realisierung der Ziele zu finden.

Data Warehousing	Klassisches Data Mining	Predictive Analytics
Query- und Reporting-Funktionen	Statistische Analyse	Prediktive Algorithmen
Statische Sichtweise	Kontinuierliche Änderungen	Auch unstetige Änderungen
Beschreibt Gegenwart und Vergangenheit	Prognostiziert die Vergangenheit	Sagt die Zukunft voraus
Setzt Hypothesen voraus	Validiert Hypothesen	Entwickelt und validiert Hypothesen

Tabelle 1: Predictive Analytics

Die Frage zu Beginn dieser Arbeit lautete deshalb: Welche vorhandenen Techniken sind für die Umsetzung eines solchen komplexen Simulationssystems geeignet, wie müssen diese Technologien zu einem Gesamtkonzept vereint werden und an welchen Stellen müssen neue Ideen entwickelt werden, um zu einem in der Praxis anwendbaren System zu kommen?

Eine Antwort auf diese Fragen zu finden ist auch ein Ziel des Projektes *SimMarket*[1], das unter Kooperation

- des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI),
- der Dacos Software GmbH und
- des Instituts für Handel und Internationales Marketing (H.I.MA.)

durchgeführt wurde. Dazu wurden zunächst die bereits genannten Technologien aus den Bereichen Business Intelligence (Kapitel 2) insbesondere Data Mining, klassische Prognoseverfahren, Künstliche Intelligenz und speziell Multiagententechnologie recherchiert und evaluiert. Anschließend wurde die beste Kombination entsprechender Konzepte und Verfahren in ein Multiagentensystem integriert (Kapitel 3) und dieses für die Implementierung eines simulationsbasierten Prototypen (Kapitel 4) für das Sortimentsmanagement verwendet. Kapitel 6 beschreibt schließlich die behandelten Anwendungsfälle und deren Evaluation unter Verwendung von Realdaten.

Um dem Projekt die nötige Praxisnähe durch Supermarktdaten zu verschaffen wurde zu Begin des Projektes eine Partnerschaft mit den drei Handelsunternehmen

- *tegut…*,
- *dm-drogerie markt* und

[1] SimMarket ist der Name eines vom BMBF (Bundesministerium für Bildung und Forschung) geförderten Projektes am DFKI (Deutsches Forschungszentrum für Künstliche Intelligenz), das insgesamt mit einem Projektvolumen von 3,6 Millionen Euro gefördert worden ist.

- *Globus (St. Wendel)*

geschlossen. Dabei stellen diese Unternehmen die zur Modellierung benötigten Daten zur Verfügung, wie beispielsweise Artikelabverkaufsdaten, Informationen über Werbemaßnahmen, Platzierungsinformationen, bewegliche und statische Artikelstammdaten und Informationen über die Aktivitäten der Konkurrenz. Durch Standort-, Sortiments- und Datenstrukturunterschiede differenzieren sich die Datengrundlagen der o. g. Handelsunternehmen (u. a. demografisch), so dass durch die heterogene Datengrundlage eine möglichst standort-, sortiments- und unternehmensunabhängige Funktionsweise des Prototyps gewährleistet wird.

Darüber hinaus fließen die von den Sortimentsverantwortlichen der Handelsunternehmen gesammelten Erfahrungen, die in gemeinschaftlichen Workshops ermittelt wurden, als Domänenwissen in das SimMarket-System ein.

Die Dacos Software GmbH, die als Neugründung aus der bereits 1977 erstmalig gegründeten Dacos GmbH hervorgegangen ist und schon damals Software für den Handel entwickelt hat, stand dem SimMarket-Projekt von Anfang an beratend zur Seite und hat das DFKI-Team bei der Entwicklung der Benutzeroberflächen und dem Software Engineering unterstützt. Da das Potenzial des Prototyps sehr schnell erkennbar war, begann die Firma Dacos recht bald mit der parallelen Umsetzung eines Produktes, das die Funktionalität des Forschungsprototyps in einer vermarktbaren Software realisiert. Das Institut für Handel und Internationales Marketing (H.I.MA.) von Prof. Zentes hat als Partner ebenfalls Erfahrungen im Handel und Marketing in das SimMarket-Projekt eingebracht.

Die besonderen Eigenschaften der zu modellierenden Domäne erforderten eine neuartige Herangehensweise, da bisherige Verfahren mindestens eine der folgenden Einschränkungen haben:

1. Es können nur lineare Modelle erzeugt werden.
2. Modelle müssen manuell erstellt und/oder konfiguriert werden.
3. Es können zu wenig Einflussfaktoren modellieren werden
4. Das Verfahren ist zu speziell auf eine Problemstellung angepasst.
5. Die Ergebnisse sind nicht interpretierbar und somit für Anwender nicht nachvollziehbar.
6. Es kann kein Domänenwissen und
7. kein unsicheres Wissen verarbeiten werden.
8. Die Beantwortung von Was-wäre-wenn-Fragen ist nicht möglich.

Deshalb werden in meiner Arbeit die Artikelmodelle automatisch mittels *Data-Mining-Verfahren bzw. maschinellem Lernverfahren* aus den vorhandenen Rohdaten der Händler extrahiert, um

1. die manuelle Erstellung und Parametrisierung zu vermeiden,
2. hinreichend exakt quantifiziert und
3. komplex genug zu sein, um *Simulationen* zu ermöglichen.

Die speziellen Eigenschaften der Domäne, die Datengrundlage der Händler und die Absicht vorhandenes Expertenwissen in die Modelle zu integrieren machte die Verwendung *probabilistischer Modelle* notwendig. Nach einer Analyse verschiedener infrage kommender Modelle entscheiden wir uns für die Theorie der *Bayes'schen Netze*, die in dieser Arbeit sukzessiv durch Anpassung an die Domäne zu *Verhaltensnetzen* ausgebaut werden (Kapitel 3.4.3, 4.5, 4.6). Nach der Evaluation existierender Bayes'scher Netztools (Anhang A), von denen keines alle benötigten Funktionalitäten realisierte, entschieden wir uns eine eigene Bibliothek zu implementieren.

Die nächste Problematik bestand darin, die große Komplexität des Modells eines vollständigen Supermarkts mit allen Artikeln und deren Interdependenzen in den Griff zu bekommen. Gelöst wurde es durch die Entwicklung eines verteilten holonischen Multiagentensystems. Der prinzipielle Ansatz von SimMarket um die Komplexität zu verringern besteht darin, die Artikel und Warengruppen eines Marktes durch *Agenten* (siehe Kapitel 3.1.1) zu modellieren, d. h., jeder einzelne Artikel eines Marktes wird innerhalb des SimMarket-Simulationssystems durch ein eigenes Softwareobjekt repräsentiert, das spezifische Designkriterien erfüllt. Das Abverkaufsverhalten wird dabei aus realen Daten mit Hilfe spezieller Lernverfahren extrahiert (Kapitel 3.4.4) und in der Wissensbasis des (Artikel-)Agenten (Kapitel 4.3) in geeigneten Repräsentationsformaten (Kapitel 3.4.3) gespeichert. Um vorhandene Rechnerressourcen auszunutzen, können sich die mobilen Agenten auf verschiedene Computer in einem Netzwerk verteilen (*Load Balancing*).

Zusätzlich zu Artikeln und Warengruppen können weitere wichtige Entitäten des Marktes und externen Einflussfaktoren wie die Wettbewerber, das Wetter, Konsumgüterhersteller etc. als einzelne Agenten, die zusammen ein verteiltes *Multiagentensystem (MAS)* bilden, modelliert und mitsimuliert werden. Mit der Modellierung einer weiteren sehr wichtigen Entität, nämlich den Kunden eines Supermarktes, hat sich Arndt Schwaiger in seiner Arbeit [Schwaiger06] beschäftigt. Die Kunden eines Marktes werden ebenfalls als Agenten modelliert und anschließend virtuell einkaufen geschickt. Das Multiagentensystem besteht bei SimMarket – grob skizziert – aus einem Modell eines realen Supermarktes und dessen Umfeldes, für das man beliebige Szenarien definieren kann, die den Agenten während einer Simulation (Kapitel 4.6) präsentiert werden und auf die diese mit ihren gelernten Verhaltensmuster entsprechend reagieren.

Da das SimMarket-System in erster Line als Entscheidungs- und Arbeitsunterstützung gedacht ist, das dem Anwender dabei hilft, aus einer großen Anzahl von Handlungsaltnativen die Geeignetste auszuwählen, handelt es sich um kein vollständig selbst optimierendes System, dass ohne Zutun eines Anwenders das Sortiment optimiert. Die wäre auch auf Grund der Komplexität der Domäne nicht möglich. Dennoch ist es das erklärte Ziel von SimMarket bestimmte Fragestellungen mit optimierenden Funktionen anzugehen. Dies ist allerdings nicht Thema dieser Arbeit. Vielmehr ist der aktuelle Entscheidungs- und Arbeitsprozess eines Sortimentsverantwortlichen, wie er z. B. bei den Partnern *Globus*, *tegut…* und *dm-drogerie markt* im Einkauf, im Marketing und im Warengruppenmanagement tagtäglich abläuft, die Ausgangsbasis dieser Arbeit. Der Sortimentsverantwortliche steht vor der Aufgabe ein

bestimmtes Teilsortiment unter sehr engen Bedingungen zu optimieren, indem er Artikel einlistet, auslistet, bewirbt, die Platzierung oder den Preis ändert. Solche Bedingung sind z. B. Verträge, die mit den Herstellern der Artikel geschlossen worden sind und eine bestimmte Mindestlaufzeit und einen fixen Preis festlegen. Andere Artikel sind so genannte „Muss-Artikel“, die vielleicht keinen Gewinn bringen, aber imagefördernd sind oder für die Kunden essenziell sind. Für andere Artikel bekommt der Händler Werbekostenzuschüsse (WKZ), die auch mit bestimmten Bedingungen z. B. der Platzierung im Regal oder im Markt verbunden sind usw. Auf Grund dieser zahlreichen Constraints ist eine automatisch optimierende Software zum jetzigen Zeitpunkt nicht möglich, da die meisten der benötigten Informationen bisher in keinem System erfasst sind. Zudem wäre die Akzeptanz unter den potenziellen Anwendern und das Vertrauen in eine solche Software nicht gegeben. Deshalb definiert der Sortimentsverantwortliche selbst die infrage kommenden zukünftigen Szenarien mit allen Randbedingungen und konfiguriert welche Parameter in welchem Umfang variabel sind. Daraus ergibt sich eine Menge alternativer Szenarien, die nun vom System unter Berücksichtigung der internen Abhängigkeiten und externen Einflüsse simuliert und nach definierten Kriterien bewertet werden können. Das Ergebnis ist ein nach den Unternehmenszielen sortiertes Ranking der Szenarien.

Vor der Simulation wird zunächst auf der Basis der definierten Szenarien ein geeignetes Multiagentensystem mit den benötigten Agenten initialisiert und Szenarien an diese übergeben. Die eigentliche Simulation und Prognose (Kapitel 4.6) besteht darin, dass man den Artikel- und Warengruppenagenten die aktuelle Szenariovariation präsentiert und sie nach ihren individuellen Verhaltensmustern reagieren lässt, die in den Verhaltensnetzen kodiert sind. Die Bewertung seitens der Artikelagenten besteht darin, dass alle direkt oder indirekt von den Maßnahmen betroffenen Gruppenagenten und deren Artikelagenten ebenfalls die zu erwartenden Verkaufszahlen für den definierten Simulationszeitraum berechnen. Während der Simulationsphase findet zudem ein Zusammenspiel zwischen Artikel- und Warengruppenagenten statt. Beispielsweise interagieren die einzelnen Artikelagenten einer Warengruppe miteinander, um die Interdependenzen zwischen Artikeln zu simulieren. Das SimMarket-System ist dadurch in der Lage, für beliebige Einzelprodukte und Mengen von Produkten die Effekte von Preisänderungen und verschiedensten Promotionsaktionen auf den Abverkauf in einem Markt in Abhängigkeit von Wochentagen, Jahreszeiten, saisonalen Einflüssen, etc. zu simulieren. Nachdem alle Agenten auf das Szenario reagiert und die Artikelagenten entsprechende Abverkäufe prognostiziert haben, werden die Ergebnisse summiert und zur Berechnung relevanter Kennzahlen herangezogen. Dabei stellt die Komplexität der zu Grunde liegenden Domäne hohe Anforderungen an die Architektur des Systems, das in der Lage sein muss, sehr viele verschiedene Einheiten und deren Relationen detailliert zu modellieren und anschließend realistisch zu simulieren. Aus diesem Grunde kommt im Rahmen des SimMarket-Projektes das neu entwickelte Multiagenten-Framework MASg (Kapitel 3.3) zum Einsatz.

Eine weitere Eigenschaft der Artikelagenten, die bisher kein anderes mir bekanntes Prognosesystem realisiert, ist die quantitative Korrelations- und Substitutionsanalyse, die es erlaubt quantitative Abhängigkeiten zwischen den Artikeln bei einer Simulation zu berücksichtigen. Dies geschieht durch die so genannte Holonisierung (Kapitel 3.4.7), bei der Agenten bei Bedarf (temporär) zu Holonen [SchSta05b] – eine spezielle Form von Agentengruppen – verschmelzen. Bei diesem Vorgang werden die probabilistischen Wissensbasen der Agenten zu Metanetzen [StaSch04] fusioniert infolgedessen neue Relationen zwischen Einfluss- und Effektknoten emergieren können. Dieses Verfahren wurde zusammen mit dem SimMarket-Prototypen ([SchSta03], [SchSta04], [SchSta05]), der im Entwicklungsstand August 2004 bereits über 450.000 Zeilen Code verfügt, erfolgreich auf der IEEE-Konferenz IAT 2004 (Intelligence Agent Technologie) präsentiert [StaSch04], wo das System mit dem gemeinsamen Preis „*Best Demo Award*“ der Konferenzen IAT und WI (Web Intelligence) ausgezeichnet wurde. Auf Grund der wirtschaftlichen Relevanz für ein mögliches Produkt wurde das Verfahren außerdem in Zusammenarbeit mit der Dacos Software GmbH zum Patent [SchStaRus05] angemeldet.

Ein weiterer Kern des Lösungsansatzes, der in dieser Arbeit präsentiert wird, besteht darin, aus den historischen Daten, die zuvor für alle Händler in ein geeignetes Datenmodell überführt werden mussten (Anhang D), mit Hilfe maschineller Lernverfahren nicht lineare Abhängigkeiten zwischen Einflüssen und Effekten automatisch zu erkennen und diese als probabilistische Netze, den Verhaltensnetzen (Kapitel 3.4.3), zu speichern und sie als Wissensbasen in die Agenten zu integrieren. Die Verhaltensnetzen wurde in dieser Arbeit durch die Verwendung Bayes’scher Netze realisiert. Prinzipiell könnten diese aber auch durch andere Technologien mit ähnlichen Eigenschaften umgesetzt werden. Ein wesentlicher Vorteil der Verhaltensnetze (z. B. im Gegensatz zu Neuronalen Netzen (Kapitel 3.5)) besteht jedoch darin, dass es sich nicht um Blackbox-Modelle handelt, deren Funktionsweisen schon während der Lernphase nicht mehr nachvollziehbar sind, sondern im Gegenteil können diese Lernverfahren vorgegebenes Domänenwissen vollständig berücksichtigen. Zudem ist die grafische Repräsentation der Netze für den potenziellen Anwender interpretier- und manipulierbar. Mit Hilfe dieser Netze können die Auswirkungen verschiedener Zustandskombinationen von Einflüssen simuliert werden, indem neues Wissen über Einflussvariablen in das Netz eingegeben wird und anschließend durch Anwendung von Inferenz-Methoden (Kapitel 3.4.5) die Werte der Effektknoten berechnet werden. Dabei können Knoten je nach Fragestellung sowohl als Einfluss- als auch als Effektvariablen angesehen werden, so dass man durch ein einziges Verfahren sowohl Diagnosen, Analysen als auch Prognosen durchführen kann. Beispielsweise lässt sich einerseits prognostizieren, welche Auswirkungen Preise und Promotionen auf den Absatz eines Artikels haben, und andererseits kann der Verkaufspreis sowie die nötige Bewerbungsmaßnahme ermittelt werden, die zum Erreichen eines konkreten Absatzziels führen.

In der Theorie der Bayes’schen Netze wird neues Wissen, das in die Netze integriert werden soll, als Evidenz bezeichnet. Die Standardliteratur zum Thema Bayes’sche Netze

unterscheidet zwischen harten, negativen und virtuellen Evidenzen. Vomlel, Kim und Valtora führen in ihrem Papier [ValKimVom01] eine neue Evidenzart, so genannte Softevidenzen ein, die es ermöglichen das Wissen über eine konkrete Wahrscheinlichkeitsverteilung in ein Netz zu integrieren. Dies ermöglicht die Interpolation beliebiger Werte, die so nicht in der Trainingsmenge vorgekommen sind. In einem Schwerpunkt dieser Arbeit habe ich mich mit der Konkretisierung des Konzepts *Softevidenzen* und deren Implementierung in den SimMarket-Prototypen beschäftigt. Des Weiteren führe ich eine neue Evidenzart, so genannte Extrapolationsevidenzen ein. Diese ermöglichen nicht nur die Interpolation von beliebigen Werten, sondern auch eine Extrapolation. Soft- und Extrapolationsevidenzen sind für das Setzen von beliebigen Preisen in das Simulationsmodell, die in der Vergangenheit nicht vorgekommen sind, essenziell.

Des Weiteren habe ich mich damit beschäftigt, wie für bestimmte Teilaspekte die Verhaltensnetze mit klassischen statistischen Methoden, beispielsweise für die *Trendanalyse* und *–korrektur* (siehe Kapitel 4.6.4) kombiniert werden können.

Die Flexibilität der Verhaltensnetze erlaubt es, das SimMarket-System für die Lösung verschiedener Fragestellungen der Einzelhändler zu verwenden. In Anhang E sind alle Fragestellungen aufgelistet, mit denen sich die Partner bei ihrer täglichen Arbeit konfrontiert sehen und die mit SimMarket behandelt wurden bzw. durch zukünftige Erweiterungen behandelt werden sollen. Dieser Katalog wurde in zahlreichen Interviews und Workshops mit verschiedenen Gruppen von potenziellen Anwendern, vom Regional- und Filialleiter über Marketingverantwortliche bis zum Einkäufer und Warengruppenmanager, mit den Partnern erarbeitet.

Die Einbettung der Verhaltensnetze in die Agenten führte schließlich zu einer neuen Agentenarchitektur, den so genannten PH-Agenten (probabilistische holonische Agenten) (Kapitel 3.4), die zusammen mit dem neuen Multiagenten-Framework MASg (Kapitel 3.3) das probabilistische holonische Multiagentensystem PH-MAS (Kapitel 3) bilden. PH-MAS ist rekursiv in dem Sinn, dass sich Artikel- und Warengruppenagenten zu neuen Warengruppen oder Holonen zusammentun können. Dadurch ist das Simulationssystem beliebig skalierbar: von einem einzelnen Artikel, über Warengruppen bis hin zur Filiale oder sogar dem gesamten Unternehmen. So wird beispielsweise die Warengruppen-Hierarchie eines Händlers auf die Agentenhierarchie abgebildet.

2 Business Intelligence: Informationsgewinnung und Prognose in betriebswirtschaftlichen Anwendungen

In diesem Kapitel sollen die relevanten Forschungsgebiete identifiziert und das SimMarket-System in das aktuelle betriebswirtschaftliche Anwendungsfeld eingeordnet werden. Wie wir sehen werden, fällt diese Arbeit in den Forschungsbereich *Business Intelligence (BI)*, der im Folgenden ausführlich vorgestellt wird. Dazu schauen wir uns relevante Modelle und Methoden aus den Bereichen Data Warehousing, Data Mining und Prognose an. Der Leser, der mit diesen Themen vertraut ist, kann dieses Kapitel überspringen.

Ziel: Ein System das einem Sortimentsverantwortlichen erlaubt, beliebige zukünftige „Was-wäre-wenn-Szenarien“ – sprich Kombinationen von z.B. Preisänderungen und Promotionsvorhaben – zu definieren und deren Auswirkungen (auf der Basis von gespeicherten Realdaten) auf das gesamte Sortiment, eine Warengruppe oder auf einzelne Artikel zu prognostizieren.

Anhand dieser Zieldefinition kann man die relevanten Forschungsgebiete identifizieren, die einen Beitrag zu Lösung des Problems bieten können. Da alle Aussagen des Systems auf „harten Fakten“, d.h. den realen Abverkaufsdaten der Händler, basieren sollen, ist es nötig umfangreiche Datensammlungen mit möglichst vollständigen Vergangenheitsdaten anzulegen, und den effizienten Abruf sicherzustellen. Aus diesem Grund werden wir uns als erstes dem Themenkomplex *Data Warehousing* zuwenden. Aus der gespeicherten Datenmenge müssen Modelle extrahiert werden, die bei gegebenen Szenarien hinreichend genaue Prognosen liefern können. Zu diesem Zweck werden wir uns der Themenkomplexe *Data Mining* und *Multiagentensysteme* widmen. Da abzusehen ist, dass in der Supermarktdomäne nicht mit einfachen binären Entscheidungen gearbeitet werden kann, wird unser Hauptaugenmerk auf Modellen liegen, die die Repräsentation von Unsicherheit berücksichtigen. Da ein wesentlicher Kern des Systems die Prognosefähigkeit ist, werden wir uns auch dem Themenkomplex *klassische Prognoseverfahren* zuwenden. Es ist allerdings anzunehmen, dass mit klassischen Prognoseverfahren nicht die Auswirkungen von Änderungen an einem oder mehreren Artikeln auf das gesamte oder Teile des Sortimentes prognostiziert werden können. Dazu bedarf es einer komplexen Simulation des gesamten Unternehmens. Deshalb wird sich diese Arbeit auch eingehend mit dem Themenkomplex Simulation beschäftigen. Der letzte noch zu erwähnende Themenbereich betrifft die

Visualisierung der Ergebnisse des Systems. Die geeignete und ergonomische Präsentation in einem entscheidungsunterstützenden System ist mindestens genauso wichtig wie die Genauigkeit der Ergebnisse. Auf Grund des Umfanges der anderen Themen liegt der Fokus dieser Arbeit jedoch nicht auf diesem Bereich. Sehr interessante Ansätze für selbst erklärende, benutzeradaptive und multimodale Benutzerschnittstellen wurden beispielsweise unter der Leitung von Wolfgang Wahlster im Projekt *SmartKom* (siehe [Wahlster03]) im Forschungsbereich *Intelligente Benutzerschnittstellen* am DFKI entwickelt.

2.1 Business Intelligence

Handelsunternehmen mangelt es selten an Daten. Auf alle nur denkbaren Fragen im Unternehmensalltag liegen die Antworten meistens irgendwo in Datenbanken, Verzeichnisstrukturen oder in alten Mainframe-Servern versteckt. An diesem Punkt setzen Business-Intelligence[2]-Anwendungen an. BI-Anwendungen sammeln Informationen aus einer Vielzahl von Datenquellen, bereiten diese auf und präsentieren sie Entscheidungsträgern über ein benutzerfreundliches Interface. BI umfasst also das Sammeln, das Speichern, das Analysieren, das Vorhersagen und das Präsentieren von sehr großen Datenmengen bzw. Informationen. Dabei kommen Techniken aus den Bereichen statistischer Analyse, Data Mining, Prognose, Data Warehousing, ***O**n-**L**ine **A**nalytical **P**rocessing (OLAP)*, Anfrage- und Reportgenerierung und Visualisierung zur Anwendung. Ein weiterer Themenbereich ist der Bereich der Simulation. Die Techniken dieses Gebietes werden in den nächsten Jahren nach Meinung vieler Experten die wichtigsten Neuerungen bringen und den Bereich BI um eine neue Dimension bereichern.

2.1.1 Anwendungsgebiete von Business Intelligence

BI-Anwendungen kommen heute in allen Unternehmensbereichen zum Einsatz. Der Schwerpunkt liegt dabei im dispositiven Bereich, also im mittleren bis hohen Management.

---

[2] *Intelligence* ist in diesem Fall nicht mit Intelligenz, sondern mit *Information* zu übersetzen. *Business-Intelligence-System* bedeutet so viel wie Unternehmensinformationssystem. *BI* wird allerdings auch als Synonym neuere Technologien genutzt, die bei der Entwicklung von BI-Systemen zum Einsatz kommen und dient damit auch der Abgrenzung herkömmlicher Ansätze (siehe [Kemper03]).

In der Vergangenheit führte man Unternehmen auf der Basis nur weniger Kennzahlen, die zudem noch aufwändig in Handarbeit aus operativen Systemen gewonnen werden mussten. Operative Systeme sind transaktionsorientierte Administrations-, Dispositions- und Abrechnungssysteme, die man auch unter dem Begriff *ERP – Enterprise Resource Planning* zusammenfasst. Das Aufkommen von ERP ist gleichzusetzen mit der Erfolgsgeschichte von *SAP (Systemanalyse und Programmentwicklung)*. SAP besitzt mit seinen Systemen (z.B. dem Warenwirtschaftssystem R/3) seit den 90er-Jahren eine unangefochtene Vormachtstellung im ERP-Markt. Auf die operativen Systeme setzen die dispositiven Systeme *Management Support Systems (MSS)* auf, die Informationen aus verschiedenen operativen Systemen sammeln, verdichten und dem Management visuell aufbereitet präsentieren. Dispositive Systeme segmentiert man nach Einsatzgebiet in die drei Ebenen Lower-, Middle- und Topmanagement. Traditionell kommen für die verschiedenen Ebenen unterschiedliche Arten von Systemen zum Einsatz. Im unteren Management werden berichtsorientierte *Management Information Systems (MIS)* mit schwach verdichteten internen Daten verwendet (z.B. Crystal Reports). Im mittleren Management werden algorithmisch ausgerichtete modellgestützte Analysen benötigt, die mit Hilfe von *Decision Support Systems (DSS)*[3] erzeugt werden (z.B. auf Excel basierend). Im hohen Management sind einfache berichtsorientierte Systeme gefordert, die hoch verdichtete interne und externe Informationen bieten. Diese Systeme werden als *Executive/Enterprise Information Systems (EIS)* bezeichnet. Unter EIS fallen auch Systeme für das *Corporate Performance Management (CPM)*[4] bei denen es um die Erfolgsmessung von Unternehmen geht.

Solche Informationssysteme sind heutzutage in allen großen Unternehmen im Einsatz und finden im zunehmenden Maße Einzug in mittelständige Unternehmen, die im größer werdenden Konkurrenzdruck Management-Konzepte der großen Unternehmen adaptieren. Der Erfolg von modernen Managementkonzepten wie Balanced Scorecards ([Hindle01]), *Six Sigma*[5] und *Activity Based Management (ABM)*[6] hängt ganz wesentlich von der Messbarkeit der Unternehmensleistung ab. Dies ist nur mit modernen Informationssystemen möglich. Die

---

[3] Im Deutschen: *Entscheidungsunterstützungssysteme (EUS)*

[4] Ähnliche Konzepte sind unter den Begriffen *Business Performance Management (BPM)*, *Enterprise Performance Management (EPM)* und *Strategic Performance Management (SPM)* bekannt.

[5] http://www.sixsigma.de/,
http://www.triz-online.de/innovation/six_sigma/six_sigma.htm

[6] http://www.sas.com/offices/europe/germany/solutions/ps_fi_3.html

Realität in vielen Unternehmen sieht leider so aus, dass es eine Vielfalt von Systemen mit gewachsener dispositver und operativer Datenhaltung gibt, die nur durch eine vertikale Integration eines BI-Systems zu einer einheitlichen Datenbasis vereint werden können.

Den meisten Unternehmen wird die Notwendigkeit von BI-Systemen klar, wenn sie mit der Umsetzung von modernen Prozessoptimierungsverfahren beginnen. Fast alle Verfahren setzen auf den extensiven Einsatz von BI-Systemen.

Im Handelssegment werden diese Optimierungskonzepte von der Initiative *Efficient Consumer Response (ECR)*[7] vorangetrieben. Das Ziel der Initiative ist die „unternehmensübergreifende Prozessoptimierung vom Kunden des Kunden bis zum Lieferanten des Lieferanten". Im Mittelpunkt aller Aktivitäten von ECR steht der Konsument. Dabei werden sowohl für die versorgungsorientierte Seite als auch für die angebotsorientierte Seite Prozessempfehlungen angeboten. Grundlage aller ECR-Prozesse sind einheitliche Identifikations- und Kommunikationsstandards, die auch *Enabling Technologies* genannt werden. Ein Beispiel für einen etablierten Identifikationsstandard für Produkte ist die *EAN*-Nummer[8] (Europäische Artikelnummer), die über einen Barcode ausgelesen werden kann und dann weltweit verwendet wird. Aktuell werden Standards für die berührungslose Identifikation von Artikeln mittels Radio-Tags[9] (**RFID** - **R**adio **F**requency **ID**entification) ausgearbeitet. Im Gegensatz zu früheren Initiativen wird bei ECR der Fokus nicht auf das Unternehmen gelegt, sondern immer erst auf den Kunden. Dadurch ergeben sich in vielen Fällen neue Sicht- und Herangehensweisen. Beim *Category Management (CM)* [ZenJanMor99] beispielsweise optimieren Händler und Konsumgüterhersteller gemeinsam die Zusammensetzung von Warengruppen. Die Segmentierung der Produkte in Warengruppen erfolgt nicht aus Sicht des Händlers, sondern nach den Bedürfnissen der Kunden. Beim CM werden in einem 8-stufigen Prozess Warengruppenzusammensetzungen, Platzierungen, Promotionen und Preise optimiert. CM umfasst die folgenden acht Stufen:

1. Kategorie-Definition: Es werden Antworten auf die folgenden Fragen gesucht. Wie segmentiert der Kunde das Sortiment des Händlers aus seiner Sicht in Warengruppen? Welche Unterkategorien gibt es und wie kann man den Erfolg einer Kategorie messen?

[7] http://www.ecr.de

[8] http://www.ean.de

[9] http://www.ean.de/ean/Inhalt/e4/e64

2. Kategorie-Rolle: Im zweiten Schritt werden die Rollen von Warengruppen ermittelt bzw. festgelegt. In der Regel unterscheidet man die Rollen Profilierungs-, Pflicht-, Ergänzungs-, Saison- und Impulsrolle. Mit dieser Einteilung differenziert sich ein Unternehmen von seinen Mitbewerbern. Die Rolle bestimmt außerdem die Priorität und somit auch die Ressourcen-Zuordnung einer Warengruppe.
3. Kategorie-Bewertung: In diesem Schritt wird die Leistung einer Warengruppe ermittelt. Daten werden erhoben, aufbereitet und analysiert. Es werden sowohl Marktanteile, Käuferreichweiten und Bedarfsdeckungsraten als auch Umsatz, Gewinn und Rendite(-potenzial) ermittelt.
4. Kategorie-Zieldefinition. In diesem Schritt werden gemeinsame Ziele von Hersteller und Händler für eine Warengruppe definiert, die den Category Managern als Benchmark dienen. Die Ziele sind meistens quantitativer Natur und müssen natürlich mit den Kategorie-Rollen übereinstimmen.
5. Kategorie-Strategie: Die Strategie legt fest wie die Kategorie-Rollen und Leistungsziele erreicht werden. Beispiele sind die Erhöhung der Frequenz, des Gewinns, des Cashflows, des Marktanteiles, des Images oder der Konsumentenakzeptanz.
6. Kategorie-Taktiken: Erst in diesem Schritt werden konkrete Maßnahmen zur Umsetzung der Kategorie-Strategie und somit der Rollen und Ziele bestimmt. Zuerst sammeln Händler und Hersteller ihre Optionen und machen eine Erfolgsabschätzung. Dann wird entschieden welche Maßnahmen umgesetzt werden. Konkrete Maßnahmen sind beispielsweise Sortimentsänderungen, Promotionen, veränderte Platzierungen und Preisänderungen.
7. Kategorie-Implementierungsplan: Nun werden Termine und Verantwortlichkeiten für alle Beteiligten festgelegt, um die taktischen Maßnahem durchzuführen.
8. Kategorie-Überprüfung: Im letzten Schritt werden kontinuierlich die definierten Ziele auf Erfolg überprüft. Bei Bedarf werden Anpassungen bei einzelnen CM-Schritten gemacht.

Insbesondere die Schritte 3, 6 und 8 erfordern den Einsatz von BI-Anwendungen und stimmen mit den Zielen von SimMarket überein, einem Category Manager bei der Analyse, bei der Prognose und bei der (kontinuierlichen) Erfolgsmessung zu unterstützen.

Ein weiterer großer Bereich auf der *ECR-Demand-Side* (angebotsorientierten Seite) ist das *Customer Relationship Management (CRM)* [Schneider04]. Dabei geht es u. a. darum, mit *Customer Behaviour Modelling (CBM)* Methoden individuelle Kunden im System zu repräsentieren, um beispielsweise One-to-One-Marketing [Zentes00] umzusetzen.

Auf der *ECR-Supply-Side* (versorgungsorientierten Seite) sind ebenfalls zahlreiche Prozessstandards erarbeitet worden, die von vielen Unternehmen erfolgreich umgesetzt

worden sind. Dazu gehören Standards im *e-Procurement* (elektronische Beschaffung) und im *Supply Chain Management (SCM)* wie z.B. das *Continuous Replenishment Program (CRP)* [10] und das *Collaborative Planning Forecasting and Replenishment (CPFR)*[11]. Bei beiden Modellen geht es um die Optimierung des Bestands- und Bestellmanagements. Das Ziel der Modelle ist es eine kontinuierliche und effiziente Warenversorgung entlang der gesamten logistischen Kette zu erreichen. Durch unternehmensübergreifende Betrachtung und Steuerung der Lieferkette sind erhebliche Effizienz- und Einsparungspotenziale möglich, die bei der Betrachtung von Einzelunternehmen nicht realisierbar sind. Der Schlüssel zum Erfolg der Prozesse ist dabei der transparente Informationsaustausch zwischen den Beteiligten der Supply Chain (Vorlieferanten, Hersteller und Handelsunternehmen). Es werden gemeinsame Geschäftspläne erstellt, Event- und Promotionsplanungen abgeglichen, Daten ausgetauscht und gemeinsame Prognosen durchgeführt. Die erhöhte Datendichte führt außerdem zu einer deutlich verbesserten Prognosegenauigkeit. Bei diesen Modellen benötigt man infolgedessen unternehmensübergreifende Business-Intelligence-Systeme. Auch in diesem Bereich können die in dieser Arbeit vorgestellten Konzepte die Performanz der Systeme verbessern. Die Simulation von Artikeln und anderen Entitäten kann insbesondere dabei helfen die Prognosekomponenten der Dispositionssysteme zu verbessern.

Ein weiteres sehr verbreitetes Modell zur Modellierung einer Supply Chain ist das ***S**upply **C**hain **O**peration **R**eference Model* (**SCOR**) [Bolstorff03]. Es wurde Anfang 1996 von der *Supply Chain Council (SCC)* entworfen und wird seitdem kontinuierlich weiterentwickelt. Inzwischen liegt SCOR bereits in Version 6.0 vor und wird von über 1000 Mitgliedern des Konzils weltweit verwendet. Auch in diesem Modell wird eine BI-Lösung für die reale Implementierung in einem Unternehmen benötigt (Modellebene 4, siehe auch [12]).

2.1.2 Architektur eines BI-Systems

Die Integration eines BI-Systems in die EDV-Landschaft eines Unternehmens ist ein vertikaler Prozess, der am unteren Ende auf die vorhandenen operativen Systeme aufsetzt und sich je nach Komplexität des BI-Systems bis auf das hohe Management ausdehnt. Die Grundlage für alle Informationssysteme sind die bereits existierenden operativen Systeme für

---

[10] http://www.ecr.de/e21/e24/e251

[11] http://www.cpfr.org

[12] http://www.net-lexikon.de/SCOR-Modell.html

das *Supply Chain Management*, für das *Enterprise Ressource Planning*, das *Customer Relationship Management* und das *e-Procurement*.

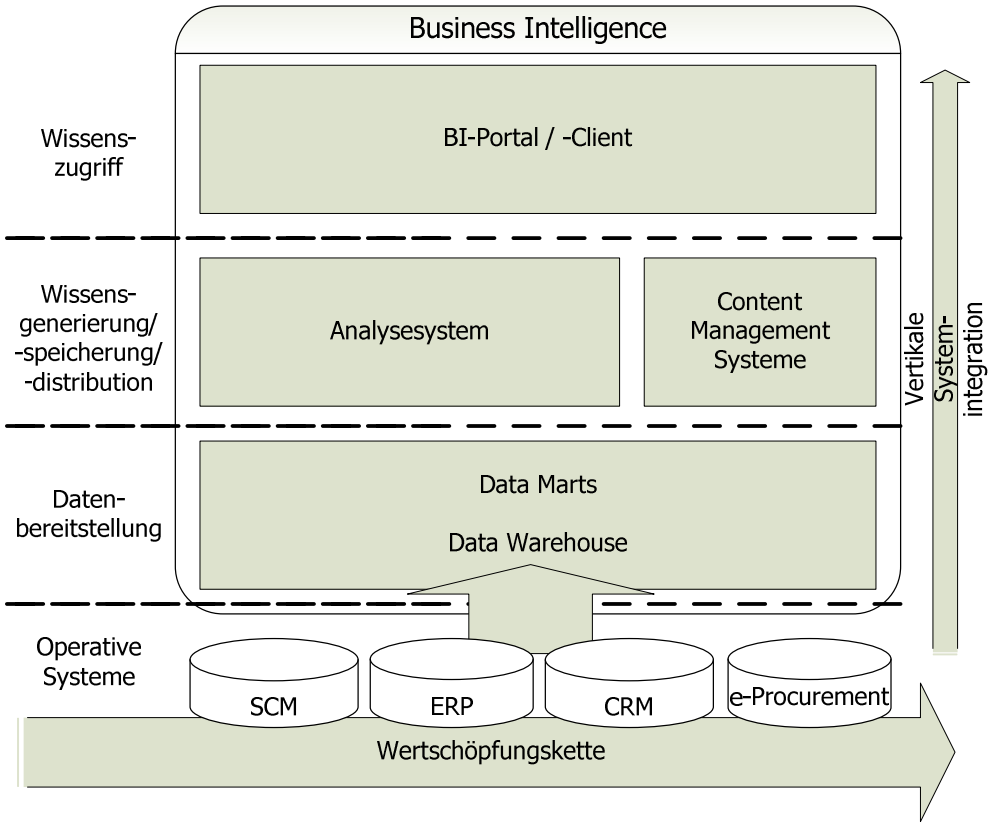


Abbildung 1: Architektur eines Business-Intelligence-Systems

In jüngster Zeit wurde begonnen eine automatische Rückkopplung zwischen DWH und operativen Systemen zu integrieren, um die Informationen nicht nur den Analytikern im dispositiven Bereich zur Verfügung zu stellen, sondern diese wieder zurück in operative Systeme fließen zu lassen. Diese Prozesse werden bei SAP „Close-the-Loop"-Szenarien genannt.

Die unterste Ebene eines BI-Systems ist die Datenbereitstellungsebene. ETL-Prozesse (Extraktions-, Transformations- und Ladeprozesse) sammeln und transformieren Daten aus den operativen Systemen in ein Data Warehouse (siehe Kapitel 2.2) bzw. in Data Marts. Data Marts sind kleinere Data Warehouses, die nur Daten einer Filiale oder sogar nur einer Fachabteilung beinhalten. In übergeordneten DWH werden Data Marts zusammengefasst und ermöglichen Unternehmensweite Analysen für das obere Management. Data Marts haben prinzipiell die gleiche Funktionalität wie DWH sind aber auf ihren Einsatz als Wissensbasen für das niedrige bis mittlere Management oder sogar als Informationslieferanten für operative

Systeme konzipiert. Die Kapselung von Unternehmensdaten in Data Marts hat außerdem die Vorteile einer erhöhten Datensicherheit, einer verbesserten Ausfallsicherheit und eine deutlich verbesserte Performance. Es wird sichergestellt, dass Fachabteilungen nur auf für sie relevante Daten Zugriff haben. Sollte ein Data Mart ausfallen können andere Fachabteilen weiter arbeiten. Der deutlich geringere Datenbestand von Data Marts erhöht außerdem die Performanz des Systems. In DWH bzw. Data Marts werden Daten strukturiert in mehrdimensionalen Cubes (Würfeln) gespeichert und für häufige Anfragen vorab verdichtet (OLAP – Kapitel 2.2). Des Weiteren werden Informationen über diese Daten in Form von Meta-Daten in DWH abgelegt.

In der übergeordneten Ebene eines BI-System befinden sich Analysesysteme zum Extrahieren und Generieren von neuem Wissen. Hier werden konzeptorientierte Analysen durchgeführt, um beispielsweise Kennzahlen für eine *Balanced Scorecard* zu ermitteln oder Datenmuster mittels Data-Mining-Verfahren (siehe Kapitel 2.3) zu extrahiert. Hier werden ebenfalls Vorschläge für modellorientierte Decision Support Systeme (z.B. Excel) und berichtsorientierte Management Informationssysteme (z.B. Crystal Reports) generiert. Bei modernen Systemen können diese Informationen außerdem wieder zurück in operative Systeme fließen.

Auf der Basis der so gewonnen Informationen werden bei Bedarf Modelle für alle in der Domäne vorhandenen Entitäten erstellen, so dass diese für Simulationen verwendet werden können. Durch Simulationen ist man in der Lage die Auswirkungen von beliebigen Was-wäre-wenn-Szenarien zu prognostizieren, so dass man durch die Kombination eines BI-Systems mit einem (Mikro-)Simulationssystem (siehe Kapitel 4) deutlich komplexere Fragestellungen behandeln kann als mit klassischem Data Mining oder Prognoseverfahren (siehe Kapitel 2.4).

In der höchsten Ebene eines BI-Systems werden schließlich die Verteilung und der Zugriff auf das gewonnene Wissen realisiert. Das Wissen kann beispielsweise in ein Content Management System (CMS) wie z.B. ZOPE integriert werden, dass den Anwendern den Zugriff über ein BI-Portal gestattet. Wie solch ein BI-Portal gestaltet wird hängt von der Art des Anwenders (operativ vs. dispositiv) und des bereits verwendeten Informationssystems ab. Prinzipiell gibt es die Möglichkeit die neu gewonnenen Informationen in vorhandene Systeme zu integrieren oder über eigenständige Anwendungen (Rich- oder Thin-Clients) anzubieten. Auf Grund der Komplexität des Systems haben wir in SimMarket von Anfang an das Prinzip eines Rich-Clients favorisiert.

Beispielsweise konnte Amazon.com durch die Implementierung von Business-Intelligence-Techniken die Betrugsrate halbieren[13]. Einer der Vorreiter bei der Nutzung von BI im Einzelhandel ist die Firma *dm-drogerie markt*, die bereits seit 1995 erfolgreich BI operativ einsetzt[14]. Auch *tegut…* setzt BI für die Analyse von Kundengruppen und deren Beziehungen untereinander und deren Kaufgewohnheiten ein, um das Sortiment so gut wie möglich auf die Bedürfnisse der Kunden anzupassen[15].

2.2 Data-Warehouse-Systeme

2.2.1 Einleitung

Data-Warehouse-Systeme sind Datenbank-Systeme, die um Technologien und Konzepte erweitert wurden, um mit ihnen große Datenmengen aus einer Vielzahl von heterogenen Systemen periodisch extrahieren, transformieren und konsistent speichern zu können. Der Begriff *Data Warehousing* bezeichnet dabei nicht nur die Verwendung von Data-Warehouse-Technologien, sondern auch einen betriebswirtschaftlichen Prozess, der den Umgang eines Unternehmens mit seinen Daten beschreibt. Data-Warehouse-Systeme bilden den Kern von Business-Intelligence-Anwendungen. Data Warehousing ist ein Prozess, der die technische Heterogenität einer bestehenden Systemlandschaft von operativen Systemen überbrückt indem Daten gereinigt, strukturiert und konsistent an einem zentralen Ort mit beschreibenden Informationen abgelegt werden. Daten können beliebig aufbereitet und aggregiert werden, wobei Meta-Informationen beschreiben wo die Daten liegen, was der Ursprung der Daten ist und wie die Daten zu interpretieren sind. Daten werden periodisch in ein Data Warehouse (DWH) überführt ohne alte Daten zu überschreiben. Dadurch werden historische Betrachtungen wie Trendanalysen und Erfolgsanalysen möglich. Die grundlegende Idee eines DWH ist die multidimensionale Speicherung und Analyse (OLAP – On-Line Analytical Processing) von Daten. Daten werden nicht als „flache“ Tabellen, sondern als mehrdimensionale Würfel (engl. Cubes) gespeichert. Die Kanten des Datenwürfels stellen den Analysekontext und die Zellen die Auswertungsgegenstände dar. Abbildung 2 zeigt das Beispiel eines mehrdimensionalen Würfels, welcher die Anzahl von Artikelabverkäufen nach dem Hersteller, der Warengruppe und des Artikeltyps ausweist. Durch Einschränken einer

---

[13] http://www.sas.com/success/amazon.html

[14] http://www.microstrategy.at/files/Kundenbericht%20dm%20drogerie%20markt.pdf

[15] http://www-306.ibm.com/software/de/db2/bi.html

Kante auf einen bestimmten Wert (z.B. Ketchup von Kraft), kann man durch einen Würfel navigieren und Analysen durchführen. Durch Hinzunahme einer Dimension z.B. Verkaufspreis lässt sich die Darstellung verfeinern.

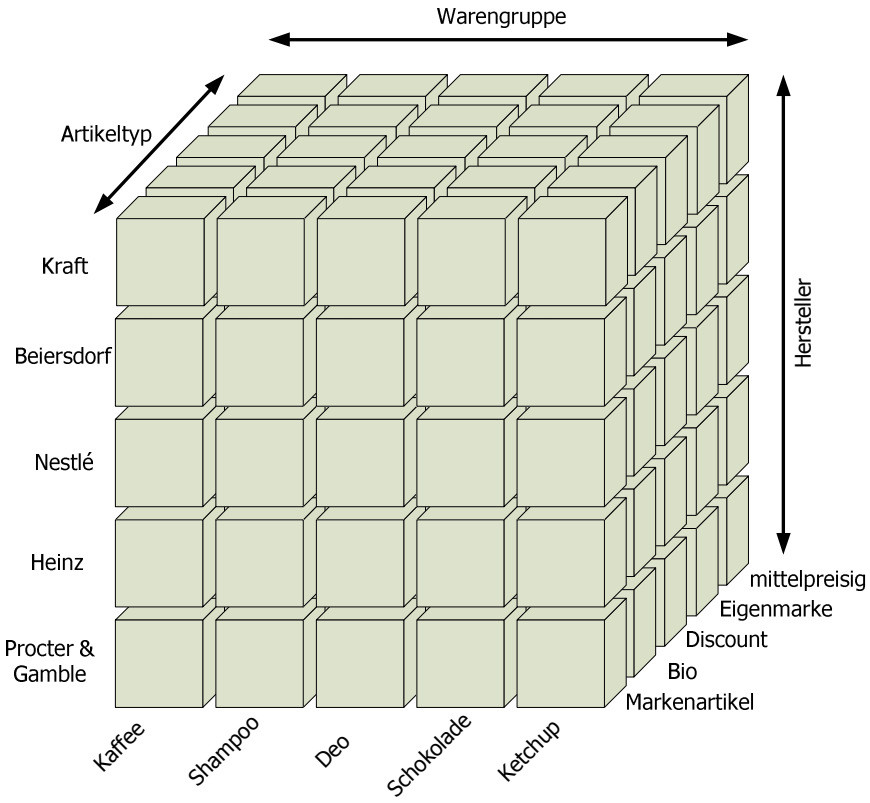


Abbildung 2: Ein mehrdimensionaler Datenwürfel (Cube)

Data Warehousing ist ein Konzept, dass aus verschieden Disziplinen und somit Sichtweisen zusammengewachsen ist. Betriebswirte waren seit jeher bemüht Prozesse innerhalb eines Unternehmens software-technisch zu modellieren und zu optimieren. Insbesondere das Berichtswesen wurde durch immer komplexere Software optimiert. Die Entwickler von Datenbank-Systemen waren hingegen bemüht die immer umfangreicher werdenden Datenmengen in den Griff zugekommen und die heterogene historisch gewachsene Systemlandschaft mittels eines einheitlichen Datenmodells in ein System zu integrieren. Weiterhin waren Statistiker bemüht ihre erfassten Werte zu speichern, auszuwerten und Modelle für Vorhersagen zu generieren (z.B. für komplexe Hochrechnungen). Diese verschiedenen Systeme sind mittlerweile in vielen Unternehmen zu einem DWH zusammengewachsen und die gewonnenen Erkenntnisse und Methoden unter dem Begriff

Data Warehousing zusammengefasst. Eine „offizielle“ Definition für *Data Warehouse* oder *Data Warehousing* gibt es nicht. Die am häufigsten zitierte Definition ist von *Bill Inmon*:

> *„A data warehouse is a subject-oriented, integrated, time-varying, non-volatile collection of data in support of the management's decision making process.” [Inmon96], [Inmon99]*

Klassische DWH-Systeme werden durch diese Systeme sehr gut beschrieben. Für moderne Systeme ist allerdings eine erweiterte Fassung sinnvoll. Wolfgang Lehner [Lehner03] erweitert die Definition wie folgt:

> *„…ein Data-Warehouse-System ist eine Sammlung von Systemkomponenten und einzelnen Datenbanken, die in einem mehrstufigen Prozess basierend auf einer Vielzahl von Quellsystemen abgeleitet sind und folgenden Eigenschaften zu genügen haben:*
>
> - *Auswertungsorientierte Organisation der Daten*
> - *Integration von Daten aus unterschiedlichen Quellen*
> - *Keine Aktualisierung durch den Benutzer*
> - *(Optionale) Historisierung mit expliziter temporaler Modellunterstützung“*

Der Begriff Data Warehouse beschreibt somit ein System von einzelnen Komponenten und Strukturen, wogegen der Begriff Data Warehousing den Prozess ein DWH zu planen, aufzubauen und insbesondere zu betreiben bezeichnet.

Die Definitionen von Inmon und Lehner beschreiben ein DWH sehr umfassend schränken allerdings die Anwendung auf die Unterstützung von Managemententscheidungen ein. Das für moderne Systeme nicht mehr korrekt. Wie bereits erwähnt, arbeiten viele Projekte daran Rückkanäle von DWH zu operativen Systemen zu implementieren. Viele Analyseergebnisse wie z.B. von Kundenklassifizierungen sind auch im operativen Bereich von Nutzen. So gibt es CRM-Systeme in Call-Centern, die während eines Verkaufsgespräches am Telefon anhand des Gesprächsverlaufs den Kunden klassifizieren und noch während des Gespräches neue Produktvorschläge generieren (Cross-Selling). Dazu greifen diese Systeme auf Analyseergebnisse in DWH zurück oder führen sogar online Analysen durch. Diese Systeme sind somit keine Entscheidungsunterstützungssysteme für das Management.

2.2.2 Aufbau eines Data Warehouses

Ein Data Warehouse besteht aus einer Vielzahl von Systemen, die untereinander gekoppelt sind. Die Abbildung 3 aus [Lehner03] zeigt ein vollständiges DWH-System mit den einzelnen Bereichen und den zugehörigen Datenbasen. In der Realität müssen nicht immer alle Komponenten eines DWH existieren.

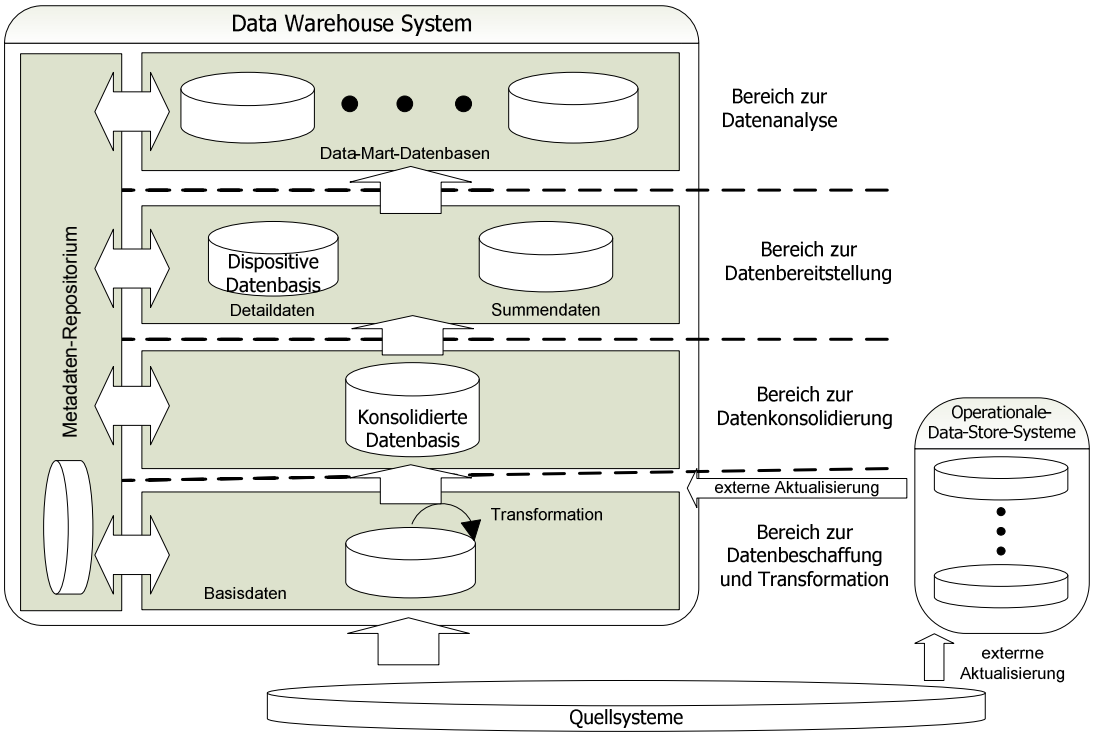


Abbildung 3: Data-Warehouse-Architektur

Die folgenden Begriffe werden in Abbildung 3 veranschaulicht:

- *Quellsysteme*
  Die Quellsysteme umfassen alle direkten oder indirekten Datenlieferanten, die vom DWH angesprochen werden. Aus diesen werden Daten extrahiert und in das DWH eingefügt.
- *Datenbeschaffung und Transformation*
  Dieser Bereich ist für das Ansprechen der Quellsysteme und das Extrahieren der Daten zuständig. Unter Umständen werden dabei lokale Quellsysteme zur Aktualisierung über externe Datenquellen angestoßen. In diesem Bereich werden

außerdem extrahierte Daten bereinigt, transformiert, nach fest definierten Qualitätskriterien geprüft und anschließend zusammengefügt.

- *Datenkonsolidierung*
  Bei der Datenkonsolidierung werden neue Daten nach festen Regeln dem DWH-Datenbestand hinzugefügt. Bei der resultierenden Datenbasis handelt es sich um eine organisationsweite und anwendungsunabhängige Speicherung.
- *Datenbereitstellung*
  In diesem Bereich wird aus der anwendungsunabhängigen Datenrepräsentation eine an die jeweilige Anwendung angepasste und optimierte Speicherung. Daten werden verdichtet, um spätere Analysevorgänge zu beschleunigen.
- *Datenanalyse*
  Der Bereich Datenanalyse beinhaltet alle Systeme, die aus der vorhandenen Datenbasis neue Informationen extrahieren und dem DWH hinzufügen. Data-Mining-Verfahren (siehe Kapitel 2.3) können aus Performanzgründen direkt in ein DWH implementiert werden und bei einer Aktualisierung des Datenbestandes automatisch ausgelöst werden.
- *Metadaten-Repositorium*
  Im Metadaten-Repositorium werden alle Datenquellen mit Extraktions- und Transformationsvorschriften und Zieldatenbanken gespeichert. Alle Vorgänge, Strukturen, Ausführungspläne und Datenquellen werden hier abgelegt und dokumentiert.

Schritte der Datenintegration

Nachdem Daten aus den Quellsystemen vom DWH abgerufen worden sind müssen diese zunächst in drei Schritten aufbereitet werden bevor sie ins DWH integriert werden können:

- *Datenbereinigung (engl. data cleansing / data scrubbing)*
  In diesem ersten Schritt werden die Daten mit Techniken aus der Mustererkennung, der künstlichen Intelligenz oder auch manuell von Fehlern bereinigt. Außerdem werden Zeichensätze in das Zielformat konvertiert. Offensichtlich falsche Datensätze müssen korrigiert oder herausgefiltert werden bevor sie ins DWH übernommen werden können.
- *Schematransformation (engl. schema transformation)*
  Bei der Schematransformation werden instanzneutrale Transformationsvorschriften für die Konvertierung der Quelltabellen in das Format der Zieltabellen definiert. Dabei kann es zu einfachen Formatkonvertierungen oder auch zur Zerlegung und Komposition von (Teil-)Tabellen bzw. Attributen kommen. Ein klassisches Problem

ist die Konvertierung von Datumsangaben. Das Format (z.B. englische oder deutsche Schreibweise) muss erkannt und in das Zielformat gewandelt werden.

- *Datentransformation (engl. data transformation)*
  Bei der Datentransformation werden schließlich die Schematransformationen auf konkrete Instanzen angewendet. Dabei unterscheidet man zwei Transformationsarten:
  - *Datensatztransformation*
    Hierbei werden Datensätze geändert. Zu Beginn werden Filterregeln auf die Datensätze angewandt, da viele Daten aus operativen Systemen im DWH nicht benötigt werden und gänzlich weggelassen werden können. Als Nächstes werden die Daten wie in den Schematransformationen definiert umgewandelt. Dabei werden häufig mehrere Datensätze auch aus unterschiedlichen Quellen miteinander verknüpft. Alle Datensätze müssen dafür zuvor bereinigt worden sein.
  - *Attributtransformation*
    Transformationen können auch auf der reinen Attributebene stattfinden. In diesem Schritt werden beispielsweise einfache neue Attribute aus zuvor zusammengeführten Daten berechnet. Wenn z.B. im vorigen Schritt aus zwei verschiedenen Quellen Verkaufszahlen und Preise zu einem Datensatz zusammengelegt worden sind kann nun das im Zielschema neu definierte Attribut ‚Umsatz' errechnet werden.

Nun können die Daten im Rahmen der Datenkonsolidierung nach festgelegten Regeln ihren Zielsystemen bzw. Zieltabellen hinzugefügt werden.

Für eine detaillierte Einführung in die Technologie von DWH siehe [Lehner03].

2.3 Data Mining

2.3.1 Was ist Data Mining?

Data Mining (DM) ist eine sehr junge interdisziplinäre Forschungsrichtung, die aus verschiedenen Fachrichtungen gewachsen ist und als Erstes – mittlerweile aber nicht mehr ausschließlich – im betriebswirtschaftlichen Umfeld Anwendung fand.

Definition Data Mining:

Data Mining ist ein Prozess bei dem automatisiert neues bzw. verstecktes Wissen aus (großen) Datenmengen extrahiert wird.

Andere Definitionen beschreiben das gewonnene Wissen näher als Korrelationen, Regeln und Muster, das für die Entscheidungsfindung im betriebswirtschaftlichen Bereich Verwendung findet. Der Begriff *Data Mining* sollte aber nicht durch die Art des gewonnenen Wissens eingeschränkt werden, da alle denkbaren Repräsentationen von Wissen das Ergebnis eines Data-Mining-Verfahrens seien können. Von einfachen Kennzahlen, über Wenn-dann-Regeln bis hin zu komplexen Modellen, die sowohl Korrelationen, Regeln mit Unsicherheit, als auch Kausalität beinhalten ist vieles möglich. Auch die Beschränkung von Data Mining auf ein bestimmtes Anwendungsfeld wie betriebswirtschaftliche Fragestellungen ist nicht korrekt. Ein großes Anwendungsgebiet von Data Mining ist beispielsweise die Bioinformatik, bei der es u. a. darum geht, Gene in unbekannten DNA-Sequenzen mittels DM-Verfahren zu finden bzw. vorherzusagen (Genvorhersage, engl. *gene finding* oder *gene prediction*).

Die Entwicklung erster Data-Mining-Verfahren begann Ende der 80er-Jahre, damals noch unter der Bezeichnung *„Knowledge Discovery in Databases (KDD)"*. Data Mining und KDD werden heutzutage als Synonyme bebraucht, wobei der Begriff Data Mining mittlerweile geläufiger ist. Je nach Fachrichtung gibt es noch weitere (nicht vollständig) gleich bedeutende Begriffe. Hier ein Überblick über die gängigsten Bezeichnungen:

- **Lernen:** Lernen ist ein Begriff aus der Biologie und Psychologie und bezeichnet den Prozess des Aneignens von Hypothesen über Gesetzmäßigkeiten der Welt anhand von Erfahrungen.
- **Maschinelles Lernen (ML):** Maschinelles Lernen kommt aus dem Gebiet der Künstlichen Intelligenz (KI) und orientiert sich an Konzepten aus der Biologie und Psychologie.
- **Funktionsapproximation:** Dieser Begriff stammt aus der Mathematik.
- **Adaption:** Die Ingenieurwissenschaften sprechen nicht von lernenden, sondern von adaptiven Systemen.
- **Parameterschätzung:** In der klassischen Statistik wird ein Modell durch Parameterschätzung an Erfahrungswerte angepasst.
- **Knowledge Discovery in Databases (KDD):** Ebenfalls eine weitere etwas ältere Bezeichnung für Data Mining, die aus dem Bereich Datenbanksysteme stammt.
- **Induktive (Logic-)Programmierung (ILP):** Dieser Begriff stammt aus dem Bereich der automatischen Induktion von Programmen.

2.3.1.1 Anwendungsfelder von Data Mining

Data Mining kann prinzipiell in allen Bereichen, in denen große Datenmengen vorliegen, angewendet werden, um aus diesen neue Informationen zu extrahieren. Besonders häufig wird DM im betriebswirtschaftlichen Umfeld (insbesondere Handel, Automobilbranche, Versicherungen und Banken) eingesetzt. Im Marketing werden DM-Methoden beispielsweise

genutzt um *Absatzprognosen*, *Kundensegmentierungen* (*Analytisches-CRM*), *Cross Selling* (*Warenkorbanalyse*) und *Missbrauchserkennungen* (engl. *fault detection*) durchzuführen und Werbebanner im WWW auf Kundengruppen zu optimieren (Web-Mining). Im Finanzbereich [HanKam01] werden DM-Verfahren für die Prognose von Aktienkursen oder für die Bonitätsprüfung von Kunden genutzt. Im Personalwesen können Personalauswahl und Mitarbeiterfehlleistungserkennung durch DM unterstützt werden, im Strategischen Management werden DM-Verfahren für die Bestimmung optimaler Standorte angewendet. In der Fertigung werden durch DM die Fehlererkennung in der Qualitätssicherung verbessert und optimale Produktsreihenfolgen festgelegt. Im Bereich des Controllings dient DM der Implementierung von Früherkennungssystemen von Unternehmenskrisen. In der Bioinformatik bzw. in der Pharmaindustrie werden mit Hilfe von Data Mining Analysen von Genexpressionsdaten [FriLinNac00] durchgeführt, Proteinstrukturen vorhergesagt [AbeMam97] und *Protein-Motifs* [CooHolSu01] entdeckt.

Im Gesundheitswesen wird DM von Krankenkassen, Pharmazieherstellern und in der medizinischen Versorgung eingesetzt [Flügel03]. Die Universität von San Francisco nutzt DM beispielsweise, um Korrelationen in ihren umfangreichen klinischen und genetischen Datenbeständen zu finden, um neue Erkenntnisse über Alzheimer oder die Entstehung anderer neurologischer Erkrankungen zu gewinnen[16].

Frank Köster nutzt DM-Methoden, um Wissensbasen von Agenten zu füllen, die die Basis von Assistenzsystemen für e-Learning-Anwendungen bilden [Köster02]. Die Agenten extrahieren mit DM-Methoden Leistungsprofile von Lernenden aus umfangreichen Sammlungen zeitlich strukturierter Interaktionsdaten, die der Lernende zuvor produziert hat. Die Leistungsprofile dienen den Agenten im Assistenzsystem zur Anpassung des tutoriellen Systems an den Benutzer. Weitere benutzeradaptive Systeme wurden von Jameson [Jameson03], Bohnenberger [Bohn04] und Wittig [Wittig03] entwickelt.

Energieversorger nutzen die Techniken des Data Mining, um beispielsweise Störungen im Gasnetz nach geografischer Lage und Wichtigkeit zu clustern (Gruppenbildung), um damit die Planung von Baumaßnahmen effizienter zu gestalten. Außerdem könnten die Ursachen von Störungen in Form von Regeln ermittelt werden. Die verwendeten Verfahren ermitteln wie stark sich die Faktoren Alter der Leitungen, Materialart, Lage (z. B. unter einer stark befahrenen Straße) und die Nähe zu einem Gewässer auf die Lebensdauer einer Leitung

[16] http://www.bio-itworld.com/news/111803_report3792.html

auswirken [Kirchner02]. Umweltschutzbehörden nutzten in ähnlicher Art und Weise Verfahren des Data Mining.

Mittlerweile wird Data Mining sogar zur Verbrechensbekämpfung eingesetzt. Das *Westmidlands Police Department* in Großbritannien nutzt beispielsweise die DM-Software *Clementine* von *SPSS*, um Muster und Trends bei ungelösten Straftaten aufzudecken, sowie neue Verhaltens- und Tatmuster frühzeitig zu erkennen[17]. Auch Organisationen wie FBI[18] (Federal Bureau of Investigation) und CIA[19] (Central Intelligence Agency) nutzen DM zur Verbrechens- und Terrorismusbekämpfung.

Die Telekommunikationsbranche wiederum nutzt Data Mining im Rahmen des *Churn Prevention*. Dabei geht es darum eine Beschreibung eines typischen Kündigers wie z.B. „unter 30 Jahre alt, männlich, Umsatz über 70,- EUR“ zu bekommen. Diese Beschreibung kann dann auf die Kundendatenbank angewendet werden, um wahrscheinliche zukünftige Kündiger im Voraus zu erkennen. Diese können dann mit zielgruppenorientiertem Marketing oder sogar individuell angesprochen werden. Auch bei der Bonitätsprüfung kommen Informationen, die mittels Data Mining ermittelt worden sind, zum Einsatz.

Data Mining findet auch im Bereich Knowledge Management Anwendung. Text-Mining-Methoden sammeln und extrahieren das Wissen eines Unternehmens aus allen im Firmennetzwerk vorhandenen Dokumenten und speichern dieses in kompakter Form in einem *Knowledge Warehouse*. Web-Mining-Methoden können in ähnlicher Weise Informationen aus großen Mengen von Webseiten gewinnen.

2.3.2 Der Data-Mining-Prozess

Der Begriff Data Mining umfasst nicht nur entsprechende Algorithmen, sondern auch den Prozess der vor und nach deren Anwendung durchzuführen ist. Bei jedem Data-Mining-Projekt steht vor der Verwendung der Modelle ein mehrstufiger Extraktion- und Modellierungsprozess. Da dieser Prozess im Wesentlichen domänen- und modellunabhängig ist und sich bei allen Data-Mining-Projekten wiederholt, wurde dieser Prozess von DaimlerChrysler, SPSS und NCR im Jahr 1999 als ein Standardprozessmodell formuliert. Der *CRoss-Industry Standard Process for Data Mining (CRISP-DM)* [CrispDM00] versteht

[17] http://www.spss.ch/_pdf/WestmidlandsPoliceData.pdf

[18] http://www.fcw.com/fcw/articles/2002/0603/news-fbi-06-03-02.asp

[19] http://dc.internet.com/news/article.php/1576771

sich als industrie-, tool- und applikationsunabhängiges Modell, das nicht proprietär und frei verfügbar für alle Anbieter und Anwender von Data Mining ist.

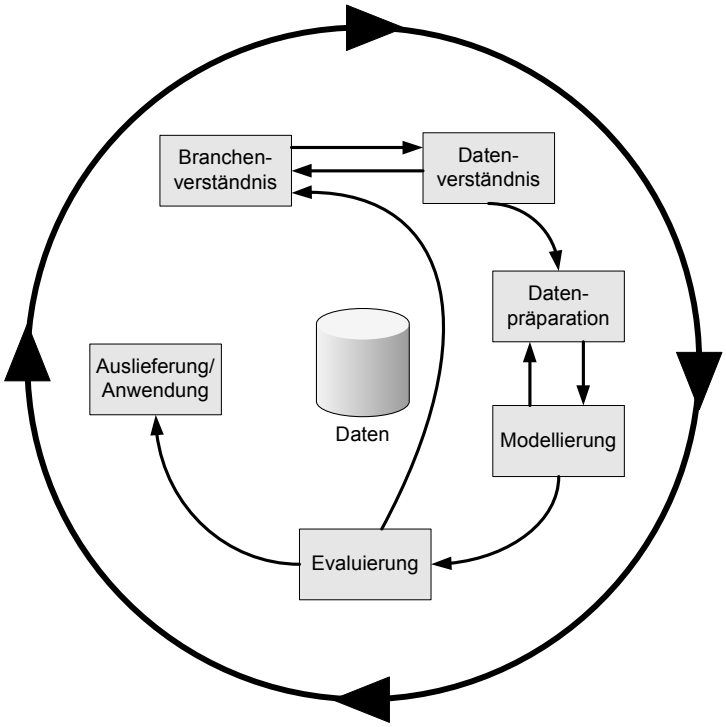


Abbildung 4: Das CRISP-DM-Prozessmodell

CRISP-DM beschreibt einen sechsstufigen Prozess:

1. **Branchenverständnis**
   Diese erste Phase zielt darauf ab, die Projektziele und Anforderungen aus einer Business-Perspektive zu verstehen, um dann dieses Wissen in eine zielgerichtete DM-Problemdefinition und eine Projektspezifikation zu überführen.

2. **Datenverständnis**
   Die Phase des Datenverstehens beginnt mit einer initialen Datensammlung, um mit den Daten vertraut zu werden, erste Einblicke zu bekommen, Qualitätsprobleme und interessante Untermengen in den Daten zu erkennen und, um erste Hypothesen bzgl. versteckter Informationen zu formulieren.

3. Datenpräparation

In der Phase der Datenpräparation wird aus dem Datenrohmaterial die finale Datenmenge, die in das Data-Mining-Tool einfließen soll, konstruiert. Datenpräparationen werden meistens mehrfach und in einer nicht festgelegten Form durchgeführt. Die Aufgaben dabei umfassen die Tabellen-, Rekord- und Attributselektion, ebenso wie das Transformieren und das Bereinigen der Daten.

4. Modellierung

In dieser Phase werden die Modellierungstechniken ausgewählt und auf die Daten angewandt. Dabei müssen meisten Parameter festgelegt und optimiert werden. Normalerweise existieren für ein und dasselbe Data-Mining-Problem mehrere Modellierungstechniken, die allerdings unterschiedliche Anforderungen an die Datenbasis haben können. Diese erfordert es häufig einen Schritt zurück in die Datenpräparationsphase zu gehen.

5. Evaluierung

In der Evaluierungsphase muss überprüft werden, ob einerseits fehlerfrei implementiert worden ist und anderseits, ob es korrekte Vorhersagen im Sinne des angedachten Modells liefert. Des Weiteren muss überprüft werden, ob das ausgewählte Modell in der konkreten Anwendung das Richtige gewesen ist und die gewünschten Informationen liefert. Bevor man das Ergebnis als endgültig betrachten kann, muss evaluiert werden, ob die in Phase 1 definierten Projektziele erreicht worden sind und die Ergebnisse den erforderlichen Nutzen haben.

6. Auslieferung/Anwendung

Die Erzeugung und Evaluierung eines adäquaten Modells bildet normalerweise noch nicht den Abschluss eines Projektes. Jetzt geht es darum, das Modell im großen Stil anzuwenden und tatsächlich neues Wissen aus den Rohdaten zu extrahieren. Dieses Wissen muss anschließend organisiert und dem Kunden verständlich präsentiert werden. Der Nutzen der Informationen sollte dem Kunden sofort ersichtlich sein. Häufig erwartet dieser bereits Handlungsoptionen, die aus den Daten abgeleitet worden sind bzw. durch diese plausible erscheinen, um ein vorgegebenes Ziel zu erreichen. Dies ist besonders dann notwendig, wenn der Kunde noch keine eingehenden Erfahrungen auf dem Gebiet des Data Mining gemacht hat. Die Auslieferungsphase kann von einer einfachen Übergabe eines Reports bis hin zur Implementierung des Data-Mining-Modells in die vorhandene IT-Umgebung des Kunden reichen. Die letzte Möglichkeit erfordert natürlich einen deutlich höheren Aufwand, da der gesamte DM-Prozess beim Kunden und vor allem vom Kunden durchgeführt werden muss, was in

Umstellungen der vorhandenen IT resultieren kann und auf alle Fälle Schulungen des Personals erfordert.

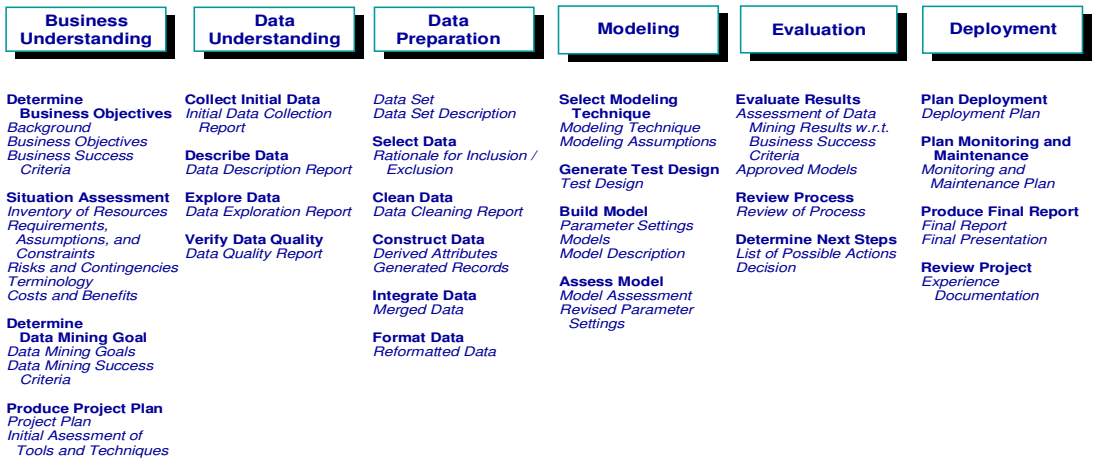


Abbildung 5: Die CRISP-DM-Phasen im Detail

2.3.3 Data-Mining-Methoden

2.3.3.1 Kategorisierung von Data-Mining-Methoden

Die Kategorisierung von Data-Mining-Methoden ist ein mehrdimensionales Problem, dass nicht in allen Fällen eindeutig lösbar ist.

Möglichkeiten der Kategorisierung von DM-Methoden sind:

- Kategorisierung nach Quelldateneigenschaften
- Kategorisierung nach Art des Ergebnisses
- Kategorisierung nach Art der eigentlichen DM-Methode
- Kategorisierung nach Aufgabenstellung

Bei der Kategorisierung nach Quelldateneigenschaften teilt man die Methoden nach der Fähigkeit mit unterschiedlichen Klassen von Quelldaten umzugehen. So gibt es spezielle Methoden für die Erkennung und Prognose von Trends oder für die Prognose von deterministischen „Ausreißern“ (engl. Outlier Analysis). Diese Methoden können somit nur auf bestimmte Arten von Daten angewendet werden. Zudem können einige Verfahren nur mit kategorialen (diskreten) andere auch mit kontinuierlichen (Eingabe-)Variablen umgehen.

Bei der Kategorisierung nach Art des Ergebnisses unterscheidet man Methoden mit deskriptiven und prediktiven Ergebnis bzw. Modellen. Ein Ergebnis oder ein resultierendes Modell kann der reinen Informationsgewinnung dienen, eine untersuchte Entität genauer

beschreiben oder klassifizieren oder sogar für eine Prognose verwendet werden. Außerdem unterscheidet man Modelle anhand des Typs der Ausgabevariablen. Wie bei den Eingabevariablen gibt es Modelle mit kategorialen und kontinuierlichen Ausgabevariablen.

Meistens trennt man die DM-Methoden allerdings nach der verwendeten Technologieklasse also nach der Art der eigentlichen Methode wie z.B. Neuronale Netze, Bayes'sche Netze, klassische Regressionsanalyse etc. Häufig beinhaltet diese Kategorisierung eine indirekte Trennung nach Fachrichtung auf Grund der historischen Entwicklung der Technologien.

Weiterhin unterscheidet man zwischen Verfahren für die Dependenz- und die Interdependenzanalyse. Bei der Interdependenzanalyse wird die Ähnlichkeit eines Objektes bezüglich weiterer Objekte oder die Abhängigkeit und Wirkung auf andere Objekte ermittelt. Bei der Dependenzanalyse wird versucht die Abhängigkeiten von Eigenschaften eines Objektes zu anderen Eigenschaften des Objektes herauszufinden.

Eine weitere wichtige Modelleigenschaft ist die Fähigkeit mit Unsicherheit oder Unschärfe umzugehen. Modelle mit Unsicherheit liefern nicht ein Ergebnis, sondern eine Menge möglicher Ergebnisse, die mit einer Wahrscheinlichkeit verknüpft sind. Modelle mit Unschärfe berücksichtigen zusätzlich, dass Ereignisse selbst häufig nur unscharf definiert werden können. Bsp.: „Kunde X findet Artikel Y attraktiv und kauft ihn deshalb". Modelle mit Unsicherheit würden diese Aussage mit einer Wahrscheinlichkeit verknüpfen: „Kunde X findet Artikel Y attraktiv und kauft ihn deshalb mit Wahrscheinlichkeit w (mit Wahrscheinlichkeit 1-w kauft er ihn nicht)". Unsichere Informationen werden durch *Unscharfe Maße* ausgedrückt. Im Gegensatz dazu modellieren Modelle mit Unschärfe unpräzise Informationen wie „attraktiv" durch *Unscharfe Mengen*. Unscharfe Mengen werden durch eine Funktion f: M $\rightarrow$ [0,1] beschrieben, die den Grad der Zugehörigkeit f zu einer *scharfen* Menge M ausdrückt. Ein Beispiel für Modelle mit Unsicherheit sind Bayes'sche Netze (siehe Kapitel 3.4.3.1). Unschärfe wird häufig mit Fuzzy-Logik modelliert.

Die letzte Möglichkeit zur Kategorisierung, die hier genannt werden soll, ist die Einteilung der Methoden nach ihrer Eignung für unterschiedliche Aufgabenstellungen. Eine mögliche Einteilung ist beispielsweise die Folgende:

- Datenanalyse,
- Prognose,
- Simulation und
- Optimierung.

Datenanalyse kann man wiederum weiter unterteilen in

- Abhängigkeitsanalyse,
- Klassifikation und
- Clusteranalyse.

2.3.3.2 (Bivariate) Abhängigkeitsanalysen

Bivariate Abhängigkeitsanalysen umfassen Abhängigkeitstests wie z.B. die Korrelationsanalyse nach Pearson und Abhängigkeitsentdeckungsverfahren wie die Assoziationsanalyse. Mit Hilfe der bivariaten Abhängigkeitsanalysen können bestehende Zusammenhänge zwischen zwei Variablen überprüft (Abhängigkeitstest) bzw. entdeckt (Abhängigkeitsentdeckungsverfahren) werden. Abhängigkeitsanalysen mit mehr als zwei Variablen werden mit dem Attribut ‚multivariat' gekennzeichnet.

Korrelationsanalyse

Bei der Korrelationsanalyse, die u. a. bei der Holonisierung (Kapitel 4.5.3) Anwendung findet, geht es um die Analyse von Abhängigkeiten zwischen zwei Zufallsvariablen X und Y. Dazu existiert eine Vielzahl an unterschiedlichen Korrelationskoeffizienten, die für unterschiedliche Aufgabenstellungen und Skalenniveaus geeignet sind. Möchte man beispielsweise Aussagen treffen, wie ‚je größer Variable X ist, desto größer ist auch Variable Y' oder ‚je größer Variable X, desto kleiner ist Y', dann verwendet man einen *Rangkoeffizienten* (z. B. nach *Spearman*), der den Grad der Monotonie zwischen den Variablen misst. Diese Verfahren können nur bei mindestens ordinalskalierten oder dichotomen[20] Variablen angewendet werden. Nominal skalierte Variablen mit mehr als zwei Kategorien können damit nicht behandelt werden.

Wenn allerdings nicht nur der proportionale Zusammenhang zwischen den Variablen von Interesse ist, sondern der lineare Zusammenhang, dann verwendet man den Pearsonschen Korrelationskoeffizienten. Der Pearsonsche Korrelationskoeffizient misst den Grad der Linearität zweier Variablen, die mindestens (intervall-)proportionalitätsskaliert sind.

Es existieren auch multivariate Korrelationskoeffizienten, die den Zusammenhang zwischen mehr als zwei Variablen untersuchen können.

Pearsonscher Korrelationskoeffizient

Um den Pearsonschen Korrelationskoeffizienten auf zwei Variablen X und Y anwenden zu können müssen die folgenden Modellvoraussetzungen erfüllt sein:

1. X und Y müssen mindestens intervallskaliert sein, um Mittelwerte und Streuungen berechnen zu können.

[20] dichotom = binär

2. (X, Y) besitzen eine zweidimensionale Normalverteilung. Aus dieser Bedingung folgt, dass u. a. Tests auf stochastische Unabhängigkeit und die Berechnung von Konfidenzintervallen möglich sind. Ob eine Normalverteilung vorliegt kann man mit dem Kolmogorow-Test oder dem χ^2-Anpassungstest prüfen.

Sind diese Voraussetzungen erfüllt, lässt sich der Grad der linearen Abhängigkeit von X und Y durch den folgenden Korrelationskoeffizienten berechnen:

$$\sigma(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}}$$

Dabei ist Cov(X, Y) die Kovarianz zwischen X und Y und Var(X) bzw. Var(Y) die Varianzen.

Es gilt.

1. $-1 \leq \sigma(X, Y) \leq 1$
2. Ist $\sigma(X, Y) = 0$, dann sind X und Y stochastisch unabhängig und bezeichnet diese als unkorreliert.
3. Ist $\sigma(X, Y) = 1$, dann hängen X und Y mit positivem Anstieg zusammen.
4. Ist $\sigma(X, Y) = -1$, dann hängen X und Y mit negativem Anstieg zusammen.

Ist $\sigma(X, Y) \neq 0$ und $|\sigma(X, Y)| \leq 0{,}5$ so nennt man X und Y schwach korreliert, gilt $0.5 < |\sigma(X, Y)| \leq 0{,}8$, dann spricht man davon, dass X und Y korrelieren und gilt $0.8 < |\sigma(X, Y)| \leq 1$ so heißen X und Y hoch korreliert. Im Fall $\sigma(X, Y) > 0$ korrelieren die Variablen positiv, bei $\sigma(X, Y) < 0$ negativ.

Auf der Basis dieses Modells lassen sich nun verschiedene Arten von Test durchführen. So lässt sich mit dem Korrelationskoeffizienten die Hypothese der stochastischen Unabhängigkeit prüfen oder falls keine stochastische Unabhängigkeit vorliegt der Grad des linearen Zusammenhangs ermitteln.

Der Pearsonsche Korrelationskoeffizient identifiziert allerdings nicht jede funktionale Abhängigkeit. Dies ist nur mit einer Regressionsanalyse möglich. So kann $\sigma(X, Y) \approx 0$ sein und trotzdem ein funktionaler Zusammenhang zwischen X und Y bestehen, z.B. bei $Y=X^2$ ist $\sigma(X, Y) = -0{,}076$.

2.3.3.3 Klassifikation

Die Klassifikation ordnet ein Objekt anhand einer Menge von Merkmalen einer von mehreren Klassen zu, um Gruppenunterschiede zu erklären. Der Unterschied zur Clusteranalyse besteht darin, dass bei der Klassifikation Gruppen oder Prototypen von Gruppen vorgegeben werden

und neue Objekte durch geschickten Vergleich der Merkmale einer der vorgegebenen Gruppen zugeordnet werden. Dagegen wird bei der Clusteranalyse eine Menge von Objekten ohne Trainingsmenge in unterschiedliche Gruppen unterteilt. Bei der Klassifikation spricht man deshalb von strukturprüfenden ‚Supervised'-Lernverfahren. Die Clusteranalyse gehört zur Klasse der strukturlernenden ‚Unsupervised'-Lernverfahren.

Anders ausgerückt ist das Ziel der Klassifikation das Approximieren einer Funktion $f(X1_1,..,X1_n) \rightarrow [K_i]$, aus einer gegebenen Trainingsmenge D, die ein Tupel $(X1_1,..,X1_n)$ von Objektmerkmalen des Objektes X1 auf eine vorgegebene Klasse K_i von i = 1…k Klassen abbildet. Dabei unterscheidet man zwei Arten von Klassifikationsverfahren, Verfahren, die nominal skalierte Merkmale voraussetzen und Verfahren die mit metrischen Skalenniveaus umgehen können. In beiden Klassen gibt es Verfahren deren Zielvariablen rein nominal skalierte Werte sind. Andere Verfahren dagegen können den „Abstand" zwischen zwei Objekten bzw. zwischen Objekt und Gruppe als metrischen Wert angeben. Die Gruppenzugehörigkeit kann dadurch bei diesen Verfahren durch Schwellenwerte definiert werden.

Ein typisches Beispiel für die Anwendung von Klassifikationsverfahren ist die Kategorisierung von Artikeln anhand ihrer Abverkaufscharakteristiken zur Auswahl geeigneter Prognoseverfahren. Beispielsweise werden für häufig verkaufte Artikel meistens andere Verfahren verwendet als für selten verkaufte Artikel.

Einen Überblick über die wichtigsten Klassifikationsverfahren ist in Anhang B zu finden.

2.3.3.4 Clusteranalyse

Im Gegensatz zur Klassifikation sind bei der Clusteranalyse (oder Clustering) die Klassen der zur Verfügung stehenden Objekte nicht bekannt. Clustering ist der Prozess des Gruppierens von Daten bzw. Objekten in Klassen (oder Cluster), so dass Objekte, die in einer Klasse sind eine höhere Ähnlichkeiten zueinander haben, als zu Objekten anderer Klassen. Jedes Objekt wird in der Regel durch einen oder mehrere n-dimensionale Merkmalsvektoren beschrieben, die beschreibende (aggregierte) Attribute oder auch Zeitreihen von Attributen enthalten. Mit der Clusteranalyse lassen sich beispielsweise Gruppen von ähnlichen Artikeln – z.B. Artikel mit ähnlichem Abverkaufsverhalten – eines Supermarktes herausfinden, für die dann unterschiedliche Prognose- oder weitere Analyseverfahren zur Anwendung kommen. Da für die Clusteranalyse keine Trainingsmenge mit bekannter Klassenzuordnung existiert, gehört die Clusteranalyse zur Klasse der ‚unsupervised'-Lernverfahren und somit zur Klasse der strukturentdeckenden Verfahren.

In der Literatur existiert eine große Anzahl unterschiedlicher Clusteralgorithmen, die man grob in fünf Kategorien einteilen kann:

Partitionierende Methoden

Bei gegebener Datenbasis von n Objekten unterteilen die partitionierenden Methoden diese in $k \leq n$ Klassen. Dabei muss jede Klasse mindestens ein Objekt haben und kein Objekt darf Mitglied mehrerer Klassen sein. Die Partitionierung wird durch einen iterativen Algorithmus erreicht, der die Objekte solange zwischen Klassen hin- und herschiebt bis ein zuvor definiertes Qualitätskriterium erfüllt wird. Ein generelles Kriterium lautet, dass Objekte innerhalb einer Klasse größere Ähnlichkeit besitzen als Objekte aus verschiedenen Klassen. Um ein optimales Clusterergebnis zu bekommen müsste man theoretisch alle möglichen Partitionen durchtesten. Da das bei größerem n nicht praktikabel ist verwenden die meisten Verfahren eine der zwei folgenden Heuristiken:

1. Den k-means-Algorithmus, der jedes Cluster durch den Mittelwert (Zentroiden) aller Objekte in der Klasse repräsentiert, oder
2. den k-medoid-Algorithmus, der jedes Cluster durch ein Objekt, das dem Zentrum des Cluster am nächsten liegt, repräsentiert.

Diese Heuristiken funktionieren sehr gut bei sphärisch geformten Cluster und kleinen bis mittelgroßen Datenbasen. Für komplexer geformte Cluster und sehr große Datenmenge sind die partitionierenden Methoden meist nicht gut geeignet. Beispiele für partitionierende Methoden sind Clusteralgorithmen auf der Basis von Assoziationsregeln wie z.B. ARCS (Association Rule Clustering System).

Hierarchische Methoden

Hierarchische Methoden unterteilen eine Datenbasis nicht einfach in k Klassen, sondern in eine Hierarchie von Klassen. Dadurch erhält man zusätzliche Aussagen über die Nähe von Klassen untereinander und kann Klassen von Klassen bilden. Es gibt zwei grundsätzlich unterschiedliche Ansätze für die Bildung einer Hierarchie: *Aggregation* und *Diversifikation*. Aggregation ist ein *bottom-up*-Ansatz, der bei den einzelnen Objekten beginnt und ähnliche Objekte sukzessiv zu Gruppen zusammenfügt. In den nächsten Schritten werden dann die Gruppen schrittweise verschmolzen bis man auf der höchsten Hierarchieebene nur noch eine Gruppe hat. Die Diversifikation ist ein *top-down*-Ansatz und beginnt mit der Gesamtmenge der Objekte und teilt diese in mehreren Iterationsschritten in immer kleinere Cluster. Die Diversifikation terminiert, wenn jedes Objekt ein eigenes Cluster bildet oder, wenn eine zuvor definierte Abbruchbedingung erfüllt ist. Das Problem der hierarchischen Methoden ist, das ein einmal getaner Schritt, sei es eine Teilung oder eine Verschmelzung, nicht wieder

rückgängig gemacht werden kann. Nachoptimierende Prozesse sind dadurch nicht möglich. Der Vorteil der schnellen Verarbeitung dieser Algorithmen resultiert aus diesem Nachteil. Es gibt zahlreiche Methoden die versuchen den genannten Nachteil durch eine ausführliche Voranalyse vor jeder Partitionierung zu minimieren. Andere Verfahren iterieren über die Hierarchiebildung selbst, um die Struktur in jedem Schritt zu verbessern. Die genannten Geschwindigkeitsvorteile gegenüber anderen Verfahren gehen bei diesen Varianten natürlich verloren.

Dichtebasierte Methoden

Die meisten Clusteralgorithmen verwenden ein Distanzmaß, um die Zugehörigkeit eines Objektes zu einem Cluster zu beurteilen. Diese Verfahren haben den Nachteil, dass damit nur sphärisch geformte Cluster und keine beliebigen Formen gefunden werden können. Dichtebasierte Verfahren haben diesen Nachteil nicht. Die Idee der dichtebasierten Methoden ist, ein Cluster so lange zu vergrößern bis die Dichte (die Anzahl an Objekten) in der „Nachbarschaft“ einen bestimmten Wert überschreitet. Das bedeutet, dass für jedes Objekt im Cluster eine minimale Anzahl an (Nachbar-)Objekten innerhalb eines gegebenen Radius existieren muss. Mit dieser Methode sind beliebige Clusterstrukturen möglich, „Ausreißer“ (engl. *outliers*) werden automatisch herausgefiltert.

Gitterbasierte Methoden

Gitterbasierte Verfahren quantisieren den Objektraum in eine endliche Anzahl an Zellen, die eine Gitterstruktur (engl. *grid*) bilden und auf der alle Clusteroperationen durchgeführt werden. Gitterbasierte Verfahren haben in der Regel eine von der Anzahl der Objekte unabhängige Laufzeit, da diese nur von der Anzahl der Zellen in jeder Dimension des quantisierten Raumes abhängt.

Modellbasierte Methoden

Modellbasierte Verfahren generieren für jeden Cluster ein Modell und ordnen die Objekte dem Cluster zu, die am Besten auf das Modell passen. Ein modellbasierter Algorithmus könnte beispielsweise eine Dichtefunktion lernen, die die räumliche Verteilung der Objekte abbildet und dabei die Cluster lokalisiert. Diese Methoden kommen meistens aus der klassischen Statistik und ermitteln häufig automatisch die Anzahl der Cluster. Bei den meisten anderen Verfahren muss die Anzahl der Cluster manuell vorgegeben werden oder durch eine Voranalyse ermittelt werden, und dient den Algorithmen als Abbruchbedingung.

In der Praxis kombinieren viele Verfahren mehrere der oben genannten Methodenklassen, um die Nachteile einzelner Verfahren zu beseitigen. Deshalb ist es häufig schwierig ein Verfahren eindeutig einer der obigen Klassen zuzuordnen.

2.4 Prognose

Prognoseverfahren sind die mit Abstand wichtigsten Komponenten in einem modernen Business-Intelligence-System und werden in vielen Sektoren wie z.B. der Absatzwirtschaft, Lagerhaltung und Finanzierung verwendet.

Im Allgemeinen analysieren Prognoseverfahren eine Menge vergangener Ereignisse, um aus ihnen Gesetzmäßigkeiten abzuleiten, die das Schließen auf zukünftige Ereignisse ermöglichen.

Die Beobachtungen liegen immer in Form von Zeitreihen, also einer Menge von zeitlich geordneten Werten, vor. Dies unterscheidet Prognoseverfahren von Klassifikations- und anderen Data-Mining-Verfahren. Klassifikationsverfahren beispielsweise generieren ebenfalls eine Aussage über ein Objekt anhand eines gegebenen Vektors x, die für Schlussfolgerungen über die Zukunft des Objektes genutzt werden kann. Diese Aussage basiert wie die Prognose auf Beobachtungen (die Trainingsmenge) und einer Theorie (die Beschreibung der Klassen). Im Gegensatz zur Prognose müssen die Beobachtungen keine Zeitreihe darstellen, sondern können auch (aggregierte) unabhängige Werte sein.

Prinzipiell ist eine Prognose nur dann möglich, wenn die *Zeitstabilitätshypothese* erfüllt ist, die besagt, dass eine Prognose nur dann sinnvoll ist, wenn die bei der Analyse entdeckten Gesetzmäßigkeiten in der Zukunft weiter gültig sind. In der Praxis ist diese Hypothese leider nie ganz erfüllt. Dennoch ist die Hypothese in vielen Fällen *im Wesentlichen* erfüllt, weshalb Prognosen überhaupt erst möglich sind. Im nächsten Abschnitt werden die verschieden Arten von Prognosemodellen vorgestellt und somit eine Klassifizierung der sehr zahlreichen Prognosemodelle vorgenommen.

2.4.1 Kategorisierung von Prognosemodellen

Qualitative und quantitative Modelle: Qualitative Modelle prognostizieren im Gegensatz zu qualitativen Modellen nur tendenzielle Aussagen z.B. „der Trend geht zum Convenience-Produkt“ und werden häufig bei langfristigen Prognosen angewendet. Quantitative Methoden erheben den Anspruch einen exakten Wert für einen bestimmten Zeitpunkt in der Zukunft vorhersagen zu können.

Univariate und multivariate Modelle: Univariate Verfahren analysieren nur die Zeitreihe der zu prognostizierenden Variable ohne Berücksichtigung weiterer Einflussfaktoren. Der Faktor „Zeit“ ist der einzige erklärende Wert der Prognose. Glättungsverfahren und autoregressive Verfahren, die eine gegebene Kurve von Vergangenheitsdaten „weiterzeichnen“ sind Beispiele für univariate Modelle. Multivariate Verfahren wie z.B. die Regressionsanalyse verknüpfen n (kausale) Einflussfaktoren mit einer zu prognostizierenden Variablen.

Länge des Prognosezeitraumes: Eine weitere mögliche Kategorisierung von Prognoseverfahren kann anhand der Länge des Prognosezeitraumes vorgenommen werden. Im Allgemeinen unterteilt man die Verfahren in Modelle für kurz-, mittel- und langfristige Prognosen für die leider keine festen Definitionen in der Literatur existieren. Die (mögliche) Länge des Prognosezeitraumes eines Verfahrens hängt natürlich auch vom zeitlichen Abstand der zur Verfügung stehenden Beobachtungswerte (Tag, Woche, Monat, Quartal, Jahr etc.) ab.

Lineare und nicht lineare Modelle: Die meisten klassischen Prognoseverfahren arbeiten nur mit linearen Gleichungen um den Wert einer Variablen vorherzusagen, d.h., dass Ergebnis einer Prognose entsteht rein additiv ohne Berücksichtigung von Abhängigkeiten zwischen den erklärenden Variablen. Nicht lineare Modelle können prinzipiell jeden Zusammenhang in den Trainingsdaten modellieren. Neuronale Netze sind ein Beispiel für nicht lineare Verfahren. Häufig gibt es Ansätze, um lineare Verfahren auf den nicht linearen Fall zu erweitern. Dies geschieht in der Praxis meistens durch eine Approximation des nicht linearen Falles durch mehrere lineare Gleichungen.

Ein- und Mehr-Gleichungsmodelle: Des Weiteren unterscheidet man Modelle die nur eine Variable und Modelle die mehrere interdependente Variablen vorhersagen können. Für Ersteres benötigt man nur eine Gleichung es handelt sich somit um ein Ein-Gleichungsmodell, für Letzteres benötigt man mehrere Gleichungen und spricht in Folge dessen von Mehr-Gleichungssystemen.

Modelle für unterschiedliche Bedarfsschwankungen: In vielen Fällen drückt die zu prognostizierende Variable den (schwankenden) Bedarf an einem bestimmten Objekt zu einer bestimmten Zeit und für einen bestimmten Zeitraum aus. In der Literatur unterscheidet man insbesondere Modelle nach der Fähigkeit *saisonale* Bedarfsschwankungen, *sporadischen* Bedarf und Bedarf*trends* zu prognostizieren.

Um dem Ziel dieser Arbeit gerecht zu werden ist es notwendig ein selbst lernendes, adaptives, Simulations- und Prognosesystems zu entwickeln, das quantitative Prognosen auf der Basis eines nicht linearen, multivariaten Mehr-Gleichungsmodells erzeugt und für alle denkbaren Längen von Prognosezeiträumen und für unterschiedliche Bedarfsschwankungen hinreichend gute Prognosen liefert.

2.4.2 Prognoseverfahren

Im Folgenden sollen nun die wichtigsten klassischen Prognoseverfahren skizziert werden.

2.4.2.1 Gleitende Durchschnitte

Prognosen auf der Basis von gleitenden Durchschnitten sind sehr einfach. Es werden Werte einer bestimmten Anzahl von vergangenen Zeitabschnitten gemittelt und dieser Wert als Prognosewert genommen:

$\hat{x}_t = \frac{x_{t-1} + x_{t-2} + ... + x_{t-n}}{n}$, wobei $\hat{x}_t$ der Prognosewert für die aktuelle Planungsperiode und n die Anzahl der gemittelten Zeitabschnitte ist.

Diese Art von Prognose wird häufig durch eine Gewichtung der vergangenen Werte verfeinert. Einzelne Werte, die in der Prognose eine besondere Rolle spielen oder Werte der jüngeren Vergangenheit können stärker gewichtet werden.

2.4.2.2 Exponentielle Glättung

Die exponentielle Glättung ist ebenfalls ein Prognoseverfahren, das jüngere Erfahrungswerte stärker gewichtet als ältere. Der Begriff „Glättung“ ist dabei lediglich eine andere Bezeichnung für Durchschnittsbildung, wie anhand der Formel für die exponentielle Glättung 1. Ordnung zu erkennen ist:

$$\hat{x}_t = \hat{x}_{t-1} + \alpha \cdot (x_{t-1} - \hat{x}_{t-1})$$

α ist ein festzulegender Glättungsfaktor, der Werte zwischen 0 und 1 annehmen kann. Ein neuer Prognosewert $\hat{x}_t$ berechnet sich aus der vorhergehenden Prognose $\hat{x}_{t-1}$, die mit dem Produkt aus Glättungskoeffizienten α und Prognosefehler $e_{t-1} = x_{t-1} - \hat{x}_{t-1}$ korrigiert wurde. Wählt man für α den Wert 1, so wird die vorangegangene Prognose um den vollen Prognosefehler korrigiert. Bei niedrigem α wird der neue Prognosewert nur geringfügig geändert. Die oben stehende Formel lässt sich in die folgende Darstellung umformen:

$$\hat{x}_t = \alpha \cdot x_{t-1} + \hat{x}_{t-1} \cdot (1 - \alpha)$$

Ersetzt man nun den „alten“ Prognosewert $\hat{x}_{t-1}$ durch die zugehörige Formel, dann ergibt sich der folgende Ausdruck:

$$\hat{x}_t = \alpha \cdot x_{t-1} + \alpha \cdot (1-\alpha) \cdot x_{t-2} + \alpha \cdot (1-\alpha)^2 \cdot x_{t-3} + \alpha \cdot (1-\alpha)^3 \cdot x_{t-4} + \ldots$$

An dieser Formel erkennt man, dass die Prognose einen Mittelwert darstellt, der aus den gewichteten Vergangenheitswerten berechnet wird. Man sieht ebenfalls, dass weiter zurück liegende Werte auf Grund des größer werdenden Exponenten weniger stark berücksichtigt werden.

Darüber hinaus existieren zahlreiche Erweiterungen der exponentiellen Glättung, wie beispielsweise die exponentielle Glättung mit Trendkorrektur oder exponentielle Glättung höherer Ordnungen, um unterschiedliche Arten von Bedarfsschwankungen besser behandeln zu können.

2.4.2.3 Prognoseverfahren nach *Winters*

Die exponentielle Glättung ist in der Lage einfache Trends in Zeitreihen fortzuschreiben. Saisonale bzw. zyklische Schwankungen können mit ihr nicht modelliert werden. Das *Verfahren nach Winters* behebt dieses Problem indem es die Zeitreihe in drei Komponenten zerlegt. Dieses Verfahren soll hier stellvertretend für *komponentenbasierte Prognoseverfahren* vorgestellt werden. Die Komponenten sind im Einzelnen:

1. Eine Trendkomponente, die die langfristige Entwicklung wieder gibt,
2. eine Saisonkomponente, die regelmäßige Schwankungen in der Zeitreihe modelliert und
3. einen Grundwert, der alles enthält, was nicht auf Saison- oder Trendeinflüsse zurückzuführen ist.

Ein Prognosewert wird durch folgende Formel berechnet:

$$\hat{x}_{t+i} = \left(\hat{a}_t + i \cdot \hat{b}_t\right) \cdot \hat{s}_{l,j} \text{ , wobei}$$

$\hat{x}_{t+i}$ = Prognosewert, der zum Zeitpunkt t für i Zeiteinheiten im Voraus berechnet wird.

$\hat{a}_t$ = Grundwert zum Zeitpunkt t.

$\hat{b}_t$ = Trendfaktor zum Zeitpunkt t.

$\hat{s}_{l,j}$ = Saisonfaktor zum Zeitpunkt t und $l = t - L + (i \bmod L)$ und Ordnungsnummer $j = (t + i) \bmod L$, wobei L die Länge des Saisonzyklus ist.

Der Prognosewert setzt sich aus dem linearen Trendwert $\hat{a}_t + i \cdot \hat{b}_t$ und dem Saisonfaktor $\hat{s}_{l,j}$ zusammen, die miteinander multipliziert werden. Wird die Saisonkomponente nicht von der Trendkomponente beeinflusst können diese Werte auch addiert werden.

Der Grundwert $\hat{a}_t$ zum Zeitpunkt t entspricht dem Grundwert $\hat{a}_{t-1} + \hat{b}_{t-1}$, der zusätzlich um den gewichteten Prognosefehler korrigiert wurde. Die resultierende Formel hat somit sehr große Ähnlichkeit zur Grundformel des exponentiellen Glättens:

$$\hat{a}_t = \left(\hat{a}_{t-1} + \hat{b}_{t-1}\right) + \alpha \left\{ \frac{x_t}{\hat{s}_{t-L,\,t \bmod L}} - \left[\hat{a}_{t-1} + \hat{b}_{t-1}\right] \right\},$$

wobei x_t der aktuelle Beobachtungswert der Zeitreihe zum Zeitpunkt t ist und $0 < \alpha < 1$ der Glättungsfaktor ist. Der verwendete Saisonfaktor entspricht dem alten Wert, da der neue zum Zeitpunkt der Berechnung des Grundwertes der neue Wert systembedingt noch nicht vorliegt.

Für jede Zeitreihe liegt ein Vektor der Länge L von Saisonfaktoren vor, die auf ähnliche Weise durch exponentielles Glätten aus vorangegangenen Saisonfaktoren des Zeitpunktes t - L berechnet werden. In jedem Schritt wird nur der aktuell betroffene Saisonfaktor angepasst.

$$\hat{s}_{t,j} = \hat{s}_{t-L,j} + \beta \left\{ \frac{x_t}{\hat{a}_t} - \hat{s}_{t-L,j} \right\},$$

wobei $0 < \beta < 1$ der Glättungsfaktor der Saisonkomponenten ist.

Der Trendfaktor wird analog durch exponentielles Glätten weiterentwickelt:

$$\hat{b}_t = \hat{b}_{t-1} + \chi \left[(\hat{a}_t - \hat{a}_{t-1}) - \hat{b}_{t-1} \right],$$

wobei $0 < \chi < 1$ der Glättungsfaktor der Trendkomponente ist.

2.4.2.4 Multiple Regressionsanalyse

Alle bisher genannten Prognoseverfahren sind univariate Methoden, die einen Prognosewert nur anhand des Faktors ‚Zeit' aus vergangenen Werten einer Zeitreihe berechnen. Diese Verfahren zerlegen die Kurve einer Zeitreihe in mehrere lineare Komponenten wie Trends und Saisoneffekte und nutzen diese Komponenten, um die Kurve einer Zeitreihe weiter zu „zeichnen". Die multiple Regression ist dagegen ein Verfahren, das die Abhängigkeiten zwischen einer Zielvariablen und mehreren unabhängigen Variablen untersucht.

Beispielsweise lässt sich mit der multiplen Regression analysieren inwieweit das *Gehalt* eines leitenden Angestellten von der Anzahl seiner zu betreuenden *Mitarbeiter* und dem zu erwartenden *Gewinn* seines Projektes abhängt:

$$Gehalt = \alpha * Mitarbeiter + \beta * Gewinn$$

Um diese Frage beantworten zu können werden auf Grund aller vorliegenden Daten die Koeffizienten α und β berechnet. Das Ergebnis ist eine so genannte Regressionsgrade, die visuell betrachtet durch die Menge der Erfahrungswerte verläuft und bei der die Summe der quadratischen Abweichungen (zwischen Gerade und Erfahrungswert) minimal ist. Mit Hilfe dieser Gleichung lässt sich für jeden einzelnen Mitarbeiter prüfen, ob sein Gehalt unterhalb der Regressionsgerade liegt und er somit unterbezahlt ist oder überhalb liegt. Im zweidimensionalen Falls lässt sich diese Gerade grafisch sehr schön darstellen, im mehrdimensionalen Fall handelt es sich um eine (Hyper-)ebene, die grafisch meistens nicht mehr sinnvoll dargestellt werden kann. Im multiplen Fall hat die zu ermittelnde Regressionsgleichung die folgende Form:

$$y_t = a_0 + \sum_{i=1}^{n} a_i x_{i,t-k_i} + e_t \text{, mit:}$$

n	=	Anzahl der Indikatoren
e_t	=	Fehlervariable
y_t	=	Vergangenheitswert in Periode t
a_0	=	Regressionskonstante
a_i	=	Regressionskoeffizienten (i = 1,…,n)
$x_{i,t-k_i}$	=	Indikator i mit Vorlauf von k_i Perioden

Zur Ermittlung der Regressionsparameter wird die Methode der kleinsten Quadrate eingesetzt, die die Summe der quadratischen Fehlerabweichungen der Erfahrungswerte von der Regressionsgeraden minimiert. Je kleiner die Varianz der Fehlerabweichungen von der Regressionsgeraden ist, desto genauer ist die Prognose. Existiert beispielsweise zwischen X und Y keine Abhängigkeit, dann ist das Verhältnis der Fehlervarianz der Y Variabeln zu der Originalvarianz gleich 1. Sind X und Y dagegen identisch ist das Verhältnis der Varianzen gleich 0. In der Praxis liegt der Wert meistens irgendwo zwischen 0 und 1. Der Wert 1 minus diesem Verhältnis bezeichnet man als R^2 oder Determinationskoeffizient (siehe auch 0). Dieser Wert ist ein Indikator in wieweit das Model den Daten angepasst ist und lässt sich direkt interpretieren. So bedeutet ein Wert von 0,4, dass die Schwankungen der Y-Werte um

die Regressionsgerade 1,0 – 0,4 = 0,6-mal der Originalvarianz entsprechen oder anders ausgedrückt, 40% der Originalschwankungen erklärt werden und 60% nicht.

Um diese Art von Regressionsanalyse durchführen zu können müssen zwei wichtige Annahmen gemacht werden:

1. Die Abhängigkeiten zwischen den Variablen sind linear und
2. die Residuen, d.h. die Abweichungen von vorhergesagten und beobachteten Werten, sind normalverteilt.

Die erste Annahme ist in der Realität leider nie ganz erfüllt und muss vor der Anwendung der linearen Regression überprüft werden. Bei Bedarf können dann nicht lineare Regressionsverfahren oder Filter angewendet werden, die nicht lineare Komponenten identifizieren und entsprechend modellieren bzw. rausfiltern können. Die zweite Annahme ist ebenfalls nicht immer erfüllt und muss gegebenenfalls getestet werden.

In der Praxis ist die Linearität der Regression ein erheblicher Nachteil. Eine Abverkaufskurve eines Artikels beispielsweise lässt sich selten durch eine einzelne lineare Funktion exakt beschreiben. Um das auszugleichen müssen mittels aufwändiger Tests Strukturbrüche erkannt und die einzelnen Abschnitte durch verschiedene (lineare) Funktionen modelliert werden. Nicht lineare Ansätze der Regressionsanalyse sind sehr komplex und aufwändig in der Handhabung, so dass eine Automatisierung für eine große Anzahl an Artikeln unmöglich werden kann. Die Komplexität der resultierenden Gleichungssysteme erreicht schnell, besonders bei mehreren Einflussfaktoren, eine in der Praxis nicht berechenbare Größe.

2.4.3 Gütemaße für Prognosen

Das Ziel jeder Prognose ist die Ableitung von Entscheidungen (auf der Basis der prognostizierten Werte), die zu einem zuvor definierten Ziel führen. Die Qualität der Entscheidungen kann nur so gut sein wie die Qualität der Prognosen, deshalb ist die Prognosegüte das entscheidende (aber nicht alleinige) Kriterium für den Einsatz eines bestimmten Prognoseverfahrens. Die Prognosequalität eines einzelnen Wertes wird ex post statistisch anhand des Prognosefehlers gemessen. Der Prognosefehler e_t ist die Differenz von realem Wert x_t und prognostiziertem Wert $\hat{x}_t$.

- Prognosefehler: $e_t = x_t - \hat{x}_t$
- Prognosezeitraum: $t = T + 1,\ldots,T + k$

Aus dem Prognosefehler leitet man weitere Kennzahlen ab:

- Absoluter Prognosefehler: $AE_t = |e_t|$
- Prozentualer Prognosefehler: $PE_t = \frac{e_t}{x_t} \cdot 100$
- Absoluter prozentualer Prognosefehler: $APE_t = \left|\frac{e_t}{x_t}\right| \cdot 100$
- Relativer absoluter Prognosefehler: $RE_t = \frac{|e_t|}{x_t}$
- Quadratischer Prognosefehler: $SE_t = e_t^2$

Um die Prognosegüte eines Verfahrens zu messen lassen sich aus den oben stehenden Prognosefehlern zahlreiche Fehlermaße konstruieren. Die folgende Übersicht zeigt die wichtigsten quantitativen Gütemaße, die auch im SimMarket-System implementiert worden sind und für die Evaluation einer Simulation verwendet werden können.

- Mittlerer Prognosefehler (ME):

$$ME = \frac{1}{k} \sum_{t=T+1}^{T+k} e_t$$

- Mittlerer absoluter Prognosefehler (MAE):

$$MAE = \frac{1}{k} \sum_{t=T+1}^{T+k} |e_t|$$

- Mittlerer relativer Prognosefehler (MRE):

$$MRE = \frac{1}{k} \sum_{t=T+1}^{T+k} \frac{|e_t|}{x_t}$$

- Mittlerer absoluter prozentualer Prognosefehler (MAPE):

$$MAPE = \frac{1}{k} \sum_{t=T+1}^{T+k} \left|\frac{e_t}{x_t}\right| \cdot 100$$

- Summer der quadratischen Prognosefehler (SSE):

$$SSE = \sum_{t=T+1}^{T+k} e_t^2$$

Die Quadrierung des Prognosefehlers bewirkt, dass große Abweichungen stärker gewichtet werden und das Fehlermaß stärker ansteigt. Dies ist in vielen Fällen erwünscht, hat aber den Nachteil, dass einzelne sehr große „Ausreißer“ das Fehlermaß erheblich verschlechtern.

- Mittlerer quadratischer Prognosefehler (MSE):

$$MSE = \frac{1}{k} \sum_{t=T+1}^{T+k} e_t^2$$

- Wurzel des mittleren quadratischen Prognosefehlers (RMSE):

$$RMSE = \sqrt{MSE}$$

Die bisher genannten Fehlermaße ergeben alle den Wert *null*, wenn die Prognose mit der Realität exakt übereinstimmt. Es existiert jedoch kein kritischer Wert, der zwischen einer guten und einer schlechten Prognose unterscheidet. Der Theil’sche Ungleichheitskoeffizient U ist ein normiertes Maß, dass versucht diesen Nachteil aufzuheben. U nimmt ebenfalls den Wert *null* an, wenn die Prognose „perfekt“ ist und nimmt den Wert 1 an, wenn die Prognose einer „naiven“ Prognose entspricht. Bei der naiven Prognose dient der letzte beobachtete Wert als Prognosewert, d. h. $\hat{x}_t = x_{t-1}$, wobei t = T + 1,…,T + k. Jeder Prognosewert ergibt sich aus dem letzten bekannten Wert, der bei Zeitreihenprognosen dem Wert x_{t-1} entspricht. U nimmt also einen Wert kleiner 1 an, wenn das verwendete Prognoseverfahren besser als die naive Prognose ist und einem Wert größer 1, wenn es schlechtere Werte als die naive Prognose liefert. Da die Werte zischen 0 und 1 nicht linear abgebildet werden, sind auch Werte knapp unterhalb von 1 schon als recht gut anzusehen.

- Ungleichheitskoeffizient von Theil (U):

$$U = \sqrt{\frac{\sum_{t=T+1}^{T+k} (x_t - \hat{x}_t)^2}{\sum_{t=T+1}^{T+k} (x_t - x_{t-1})^2}}$$

Der nun folgende Determinationskoeffizient wurde vom allgemeinen Korrelationskoeffizienten nach Pearson (siehe 0) abgeleitet und nimmt bei perfekter Korrelation zwischen x und $\hat{x}$ – und damit bei einer perfekten Prognose – den Wert 1 an. Er wird in der Literatur häufig als Fehlermaß für Prognosen verwendet, obwohl dieses Maß nur den Grad an Korrelation angibt. Somit ist r^2 auch dann gleich 1, wenn der Prognosefehler erheblich ist, aber immer den gleichen Abstand in der gleichen Richtung zum realen Wert besitzt. Dies lässt r^2 nicht gerade als ideales Gütemaß für Prognosen erscheinen.

- Determinationskoeffizient (r^2):

$$r(x,\hat{x})^2 = \frac{\mathrm{Cov}(x,\hat{x})^2}{\mathrm{Var}(x)\mathrm{Var}(\hat{x})}$$

Die Distanzmaße nach *Clark* und *Canberra* sind vektorraumbasierte Maße, die den Abstand zweier Vektoren basierend auf dem Winkel und den Längen der Vektoren angeben. Bei der Verwendung dieser Maße für die Gütemessung von Zeitreihenprognosen, werden die prognostizierten Werte und die realen Werte als Vektoren aufgefasst und der Abstand zwischen den Vektoren als Fehlermaß interpretiert.

- Distanz nach *Clark*:

$$d(x,\hat{x}) = \sqrt{\sum_{t=T+1}^{T+k} \frac{\left|x_t - \hat{x}_t\right|^2}{\left|x_t + \hat{x}_t\right|^2}}$$

- Distanz nach *Canberra*:

$$d(x,\hat{x}) = \sum_{t=T+1}^{T+k} \frac{\left|x_t - \hat{x}_t\right|}{\left|x_t + \hat{x}_t\right|}$$

3 PHMAS – Ein probabilistisches holonisches Multiagentensystem

3.1 Einleitung

3.1.1 Multiagentensysteme

Komplexe Probleme der Informatik können oft solange in kleinere Teilprobleme zerlegt bis die einzelnen Teilprobleme lösbar sind (*divide-and-conquer*). Der Vorteil ergibt sind in der Regel durch die schnellere Ausführbarkeit der kleineren Teilprobleme und durch die geschickte Auswahl einer hinreichenden Untermenge aller Teilprobleme, die für die Lösung des Gesamtproblems nötig ist. Diese Vorgehensweise hat allerdings den Nachteil, dass die Menge aller Problemlöser bei sehr komplexen Systemen koordiniert werden muss. Die Forschungsrichtung der *Multiagentensysteme* (MAS) beschäftigt sich unter anderem mit diesem Problem der Koordination autonomer Problemlöser, so genannter *Agenten* und bietet dafür effiziente Mechanismen an. Eine allgemeingültige Definition für den Begriff *Agent* existiert nicht. Russell und Norvig [RusNor95] definieren einen Agenten folgendermaßen:

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

Die meisten Autoren allerdings definieren den Agentenbegriff mit Hilfe von hinreichenden Eigenschaften, die eine Berechnungseinheit mindestens haben muss. Die folgende Liste umfasst alle Attribute die sowohl für die Definition eines *Agenten* als auch für die Definition der davon abgeleiteten Agententypen verwendet werden:

- autonom
- intelligent
- reaktiv
- proaktiv
- deliberativ
- adaptiv
- rational
- sozial
- mobil

Auf der Basis dieser Attribute unterscheidet man die folgenden grundlegenden Agententypen:

- reaktive Agenten
- deliberative Agenten

- soziale Agenten
- mobile Agenten
- Informations-/Internetagenten: Meistens soziale und mobile Agenten, die je nach Anwendung auch personalisiert und deliberativ sind.
- hybride Agenten: Agenten, die mehrere Eigenschaften, meist autonome, reaktive, deliberative und soziale, besitzen.

Der Vorteil der Multiagentensysteme z. B. gegenüber dem verteilten Problemlösen liegt in der Dynamik der Systeme zur Laufzeit. Anzahl und Art der Agententypen werden nicht vor Systemstart von einer zentralen Instanz festgelegt, sondern werden dynamisch zur Laufzeit den aktuellen Bedingungen dezentral angepasst. Bond und Gasser [BonGas98] definieren ein Multiagentensystem folgendermaßen:

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

Die MAS-Forschung beschäftigt sich damit, wie Agenten ihr Wissen, ihre Ziele, ihre Fähigkeiten und ihre Pläne koordinieren, um erfolgreich in Kooperation untereinander Probleme zu lösen.

Eine wichtige Eigenschaft von Multiagentensystemen, die deren Entwicklung besonders motiviert hat, ist *Emergenz*: Ein System ist *emergent*, wenn es eine Eigenschaft besitzt, die nicht direkt durch die Eigenschaften einzelner Teile des Systems erklärbar ist, sondern erst durch das Zusammenwirken von mehreren (unterschiedlichen) Teilen des Systems entsteht.

Ein Beispiel: Ein einzelnes Molekül kann keine Temperatur besitzen, denn Temperatur ist nur messbar, wenn viele Moleküle vorhanden sind. Daher ist Temperatur eine emergente Eigenschaft.

Ein weiteres Beispiel aus der Biologie ist der Ameisenhaufen: Die kognitiven Fähigkeiten einer einzelnen Ameise sind beschränkt. Dennoch ist ein Ameisenvolk in der Lage ein höchst komplexes Gebilde – den Ameisenhaufen – zu bauen. Dies ist das Resultat des Zusammenwirkens einer großen Anzahl sehr unterschiedlicher Ameisentypen (Arbeiter, Königinnen, Larven etc.), deren Fähigkeiten und Bedürfnisse sich optimal ergänzen.

Die wachsende Komplexität heterogener MAS erhöht die Notwendigkeit Strukturen und Organisationsformen in diese Systeme zu integrieren (siehe auch [Schillo04], [PanJen01]). Über die normale Form der Kooperation hinaus, werden dynamisch zur Laufzeit Gruppen von Agenten und „Gruppen von Gruppen“ gebildet, die durch eine rekursive Definition der Agenten realisiert werden. Gruppen von Agenten erscheinen nach außen hin wiederum als Agenten (Gruppenagenten) mit gleicher Schnittstelle wie Einzelagenten. Durch

Gruppenbildung lassen sich Hierarchien von Agentengruppen bilden, die die natürliche Struktur einer Domäne widerspiegelt. In SimMarket werden beispielsweise Artikelagenten durch Warengruppenagenten zusammengefasst und dadurch die Warengruppenhierarchie eines Warenhauses nachgebildet. Bei der Suche nach geeigneten Kooperationspartnern in einem MAS können zunächst die Gruppenagenten angesprochen werden, so dass nicht alle Agenten einzeln angesprochen werden müssen. Die Theorie der *holonischen MAS* beschreibt die dynamische Gruppenbildung und soll im Folgenden vorgestellt werden.

Das Wort *Holon* [GerSieVie99] stammt von dem griechischen Wort *holos* (Ganzes) und dem Suffix *on* (Teil) ab. Ein Holon ist ein einzelner Agent, der in der Lage ist mit anderen Holonen zu einem (Super)-Holonen zu verschmelzen. Da ein (Super-)Holon wiederum mit anderen Agenten zu komplexeren Holonen verschmelzen kann, lässt sich eine Agentenpopulation rekursiv definieren. Auf Grund der rekursiven Definition bezeichnet man in einem holonischen MAS sowohl die einzelnen Agenten, als auch die Super-Holonen in der Regel einfach als *Holon*. Im Weiteren, falls nicht anderes vermerkt, ist mit *Holon* immer eine Gruppe von Agenten gemeint.

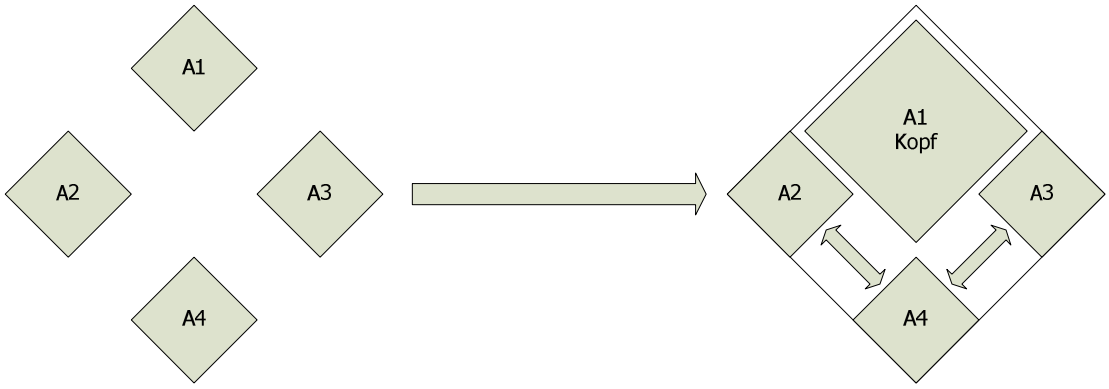


Abbildung 6: Vier Agenten bilden einen Holon mit Agent A1 als Kopf

Das besondere eines Holons ist, dass er nach außen wie ein einzelner Agent und nicht wie eine Gruppe von Agenten erscheint. Dies wird durch den *Kopf* (engl. *head*) des Holons realisiert, der den Holon nach außen repräsentiert und intern die Verwaltung der Subagenten übernimmt. Die Rolle des Kopfes kann entweder von einem einzelnen Agenten, der dem Holon beigetretenen ist oder durch einen bei der Holonenbildung neu erzeugten Agenten übernommen werden. Der Kopf kann entweder die Funktionen der Subagenten erben oder er delegiert Aufgaben bzw. Teilaufgaben an die entsprechenden Subagenten. Die einzelnen Agenten eines Holon verlieren allerdings nicht ihre Autonomie, sondern können den Holon

wieder verlassen [Schillo04]. Holonen können dynamisch zur Laufzeit gebildet und auch wieder aufgelöst werden.

In MAS gibt es viele Situationen in denen ein einzelner Agent der Hilfe anderer Agenten bedarf, um seine Aufgaben erfüllen zu können. Bei sehr vielen und örtlich getrennten Agenten (z. B. bei mobilen Agenten) kann der Koordinierungs- und damit der Kommunikationsaufwand erheblich werden. Besonders in Datenanalyse-Szenarien, in denen mittels eines mobilen Multiagentensystems verteiltes Data Mining durchgeführt wird, können schnell Gigabytes an Daten anfallen, die unter Umständen zwischen den Agenten ausgetauscht werden müssen. In solchen Fällen ist es sinnvoll die Agenten zuvor lokal zu Holonen zu verschmelzen, statt die Analyse mit aufwändigen Protokollen über das Netzwerk durchzuführen. Die Holonenbildung vermeidet zusätzlich Redundanz von Daten im Netzwerk, da nicht mehr alle Agenten das vorhandene Wissen benötigen, sondern nur noch der Kopf eines Holons Zugriff auf das gesamte Wissen hat.

Holonische Multiagentensysteme wurden bereits erfolgreich in mehreren Projekten am DFKI angewendet:

- TeleTruck: TeleTruck [BürFisVie98] ist ein Online-Dispositionssystem, mit dem Speditionen die Planung und Überwachung der Ausführung von Kundenaufträgen interaktiv durchführen können. Alle logischen und physikalischen Einheiten – wie z. B. die Spedition, die Fahrer, die Fahrzeuge und die Ware – werden im System als autonom planende und kooperierende Holonen modelliert. Mittels Kommunikation handeln alle potenziellen Auftragnehmer (Fahreragenten) einen optimalen Plan zur Erfüllung des Auftrages aus. Dazu verwendet TeleTruck für den Verhandlungsprozess eine erweitere Form des *Contract-Net-Protokolls* [Sandholm93] und den Optimierungsprozess *Simulated Trading* [BacHocMal93].
- Casa: Casa [Gerber04] ist ein weiteres Logistiksystem, das auf holonischen Agenten basiert. Casa ist eine Plattform für forstwirtschaftliche Logistik und Vertrieb, die alle Teilnehmer der Wertschöpfungskette wie Hersteller, Käufer, Zwischenhändler und Logistikfirmen als Agenten modelliert und in ihren Prozessen unterstützen soll. Die Hersteller werden bei der Planung und Überwachung des Produktionsablaufs und bei dem anschließenden Verkauf über eine Internetplattform unterstützt. Agenten übernehmen den Verkauf in Online-Auktionen, berechnen Einsatzpläne mit den resultierenden Kosten und bieten rollenspezifische Informationen u. a. über mobile Dienste wie WAP an.
- Agricola: Agricola [Gerber04] ist ein verteiltes Multiagentensystem, das eine Just-in-Time-Planung von landwirtschaftlichen Maschinenparks realisiert. Maschinenbesitzer können in einem Netzwerk von Landwirten ungenutzte Maschinen anbieten, die von anderen Landwirten angefordert werden können.

Lokale Maschineneinsatzplanungen von Landwirten werden mit Angeboten der Maschinenbesitzer dynamisch hinsichtlich Witterung, Ausfall von Maschinen, Standort, Zeitbedarf und Personalbedarf abgeglichen und mögliche Verteilungspläne erstellt. Alle Entitäten des Systems – wie Landwirte, Maschinenbesitzer und Koordinatoren – werden durch Agenten repräsentiert, die die Planung und Koordination im Netzwerk übernehmen.

3.1.2 Agentenplattformen

Die meisten Agentenprojekte verwenden objektorientierte Programmiersprachen wie C++, Java oder C# (.NET). Aktuelle Entwicklungsumgebungen realisieren bereits sehr viele Funktionen, die für Agentenplattformen (engl. *agent framework*) gebraucht werden. Dennoch gibt es viele agentenspezifische Funktionalitäten, die aktuelle Entwicklungsumgebungen (noch) nicht bieten und von Agentenplattformen realisiert werden müssen.

Abhängig von dem zu realisierenden Agententyp, unterscheidet man die folgenden Arten von Agentenplattformen:

- Deliberative Agentenplattformen
- Mobile Agentenplattformen
- Soziale Agentenplattformen
- Hybride Agentenplattformen

Eine Agentenplattform ist eine Art verbesserte *Middleware*[21], die auf vorhandene Entwicklungs- und Betriebssystemumgebungen aufsetzt, diese um agentenspezifische Komponenten und Funktionalitäten erweitert und die folgenden zwei Bereiche umfasst:

1. Die Agenteninfrastruktur und
2. die Agentenarchitektur.

---

[21] Entsprechend dazu sind Agentenplattformen oder -frameworks aus betriebswirtschaftlicher Sicht vor allem *EAI*-Systeme. *Enterprise Application Integration* umfasst die Planung, die Methoden und die Software, um heterogene, autonome Anwendungen - ggf. unter Einbeziehung von externen Systemen - prozessorientiert zu integrieren. Im Unterschied zur Middleware bietet EAI auch die Möglichkeit, Prozesslogik abzubilden.

Die Agenteninfrastruktur regelt die Verwaltung der Agenten und deren Kooperation bzw. Kommunikation. Sie umfasst einen Verzeichnisdienst der „*White-Pages*"- und optional auch „*Yellow-Pages*"-Funktionalitäten anbietet. Bei ersterem handelt es sich um ein Verzeichnis das die Adressen aller Agenten verwaltet, bei Letzterem werden zusätzlich die Funktionalitäten der Agenten mitverwaltet, wodurch eine Suche nach Agenten mit bestimmten Fähigkeiten möglich wird. Im Allgemeinen sollte eine Agentenplattform komponentenbasiert aufgebaut und dadurch flexibel konfigurierbar sein.

Die Anforderungen an eine Agenteninfrastruktur sind:

- Interoperabilität, zur Interaktion mit anderen Systemen und zur Bereitstellung von Informationen für die Agenten.
- Stabilität, Programmfehler in einzelnen Agenten sollten das System nicht nachhaltig beeinflussen oder gar abstürzen lassen (Fehlertoleranz).
- Skalierbarkeit, damit sich das System dynamisch zur Laufzeit an die gegebenen Bedingungen anpassen kann. Auch die Anpassung an neue Domänen sollte ohne großen Aufwand möglich sein.
- Transparenz und Einfachheit, um eine möglichst schnelle und saubere Entwicklung von neuen Systemen zu ermöglichen. Dazu gehören ebenfalls gute Wergzeuge, Dokumentation und Benutzeroberflächen.
- Konsistenz, z. B. bei mobilen Agenten, wo ein Agent nach einer Migration nur einmal existieren darf.
- Effizienz.

Zur Realisierung von verteilten und mobilen Agentensystemen kommen je nach Plattform unterschiedliche Technologien zum Einsatz. Bei Verwendung von *C++* wird häufig *Corba (Common Object Request Broker Architecture)* mit *IIOP (Internet Inter-Orb Protocol)* oder *DCOM (Distributed Component Object Model)* von *Microsoft* verwendet. Unter Java wird *RMI (Remote Method Invocation)* für RPC (Remote Procedure Calls) verwendet. *.NET* bietet mit *.NET Remoting* ebenfalls einen sehr fortgeschrittenen Mechanismus für verteilte Systeme an.

Eine Stufe über diesen low-level Technologien liegen *Webservices*, die mit Hilfe von Webservern Dienste im Netzwerk anbieten. Die Kommunikation mit Webservices basiert auf dem *XML*-basiertem (***EX****tensible* ***M****arkup* ***L****anguage*) *SOAP*-Protokoll (*Simple Object Access Protocol*) und geschieht in der Regel über das Internetprotokoll *http* oder *https*. Die Beschreibung, Integration und Suche von Webservices wird über das *UDDI*-Protokoll (*Universal Description, Discovery and Integration Protocol*) geregelt. UDDI-Server dienen als Verzeichnisdienst für Webservices und ermöglichen die gezielte Suche nach Webservices mit bestimmten Fähigkeiten und Eigenschaften.

Eine weitere sehr wichtige Anforderung an die Infrastruktur einer Agentenplattform ist Sicherheit. Eine Agentenplattform ohne komplexes Sicherheitsmodell ist nicht mehr denkbar und würde sich auf Grund von Akzeptanzproblemen seitens der Anwender nicht durchsetzen. Ein Sicherheitsmodell umfasst mindestens die folgenden vier Sicherheitskomponenten:

- Verschlüsselung: Alle Kommunikations- und Migrationskanäle müssen verschlüsselbar sein, um unauthorisiertes Mithören bzw. Abfangen von Agenten zu unterbinden. In Agentensystemen bieten sich vor allem Mischformen zwischen symmetrischen (z.B. *DES – Data Encryption Standard oder AES – Advanced Encryption Standard*) und asymmetrischen Verfahren (z.B. *RSA – Rivest, Shamir, Adleman*) wie z.B. *SSL (Secure Socket Layer)* über *https* an, um einerseits eine hohe Sicherheit und andererseits eine hohe Performanz zu bekommen.
- Authentisierung: Alle Agenten und Benutzer müssen sich eindeutig mittels Signaturen (z.B. *XML-Sig*) ausweisen können und einen eindeutigen Sicherheitskontext z.B. auf der Basis von Principals, Rollen und Credentials haben.
- Autorisierung: Nach erfolgreicher Authentisierung muss der Zugriff der Agenten bzw. Benutzer auf Daten oder Objekte auf der Basis eines Vertrauensmodells - wie z.B. Kerberos über ActiveDirectory und bei Webservices über WSS (Web Service Security) - geregelt werden. Die Steuerung kann entweder programmatisch durch speziell implementierten Code oder deklarativ mit Hilfe von annotiertem Code erfolgen. Wobei letztere Möglichkeit, wenn möglich, zu bevorzugen ist.
- Auditing: Alle Aktionen innerhalb des Agentenframeworks sollten vollständig und asynchron protokollierbar sein, um das Aufspüren von Sicherheitslücken und bösartigen Agenten oder Benutzer möglich zu machen. Deshalb sollten Aktionen von Benutzern und Agenten erst nach erfolgreicher Authentisierung erlaubt werden.

Die zweite Hauptkomponente einer Agentenplattform ist die Agentenarchitektur und deren funktionale Komponenten. Das *FIPA*-Konsortium bietet einen einheitlichen Standard für alle Komponenten eines Agenten, um die Interoperabilität zwischen verschiedenen Agentenplattformen zu erhöhen.

Die meisten Plattformen bieten einen Basisagenten mit einer Kontrollarchitektur, die einen Scheduler und eine Auswahl an Basiskomponenten bietet, die die Fähigkeiten der Agenten implementieren. Soziale Agenten haben eine komplexe Kommunikations- und Koordinationskomponente, deliberative Agenten eine Planungskomponente. Die Kommunikationskomponenten unterscheiden sich vor allem durch die umgesetzten Kommunikationsprotokolle (z. B. *Contract-Net*) und der verwendeten Agentensprache (z. B. *KQML* oder *FIPA-ACL – Agent Communication Language*). Des Weiteren existieren

spezielle Inhaltssprachen (z.B. *FIPA-SL* Semantic Language oder *FIPA-KIF* Knowledge Interchange Format), die den semantischen Inhalt eines sprachlichen Ausdrucks mit Hilfe von Ontologien (z.B. *DAMIL* oder *OIL*) besser beschreiben. Die Art der verwendeten Sprache hängt stark von der Art der Wissensbasis des Agenten ab. Ist die Wissensbasis in der Lage mit semantischen und probabilistischen (z.B. *Bayes'sche Netze*) Wissen umzugehen, muss auch die Agentensprache entsprechen konstituiert sein.

Eine Auflistung und Bewertung fast aller verfügbarer Agentenplattformen ist unter[22] zu finden.

3.1.3 Agentenbasierte Simulation

Eine Simulation ahmt das Verhalten eines realen komplexen Systems nach, wie die täglichen Operationen einer Bank, den Wert eines Aktienportfolios, den Ablauf eines Montagebandes in einer Fabrik oder das Kaufverhalten von Kunden eines Supermarktes. Eine Simulation im engeren Sinne ist das Ausführen eines Modells, das durch ein Computerprogramm repräsentiert wird und Informationen über das zu untersuchende System liefert. Die Modelle können aufwändig mit der Hand gestaltet, oder wie bei dem in dieser Arbeit präsentierten Ansatz automatisch aus einer vorliegenden Datenmenge extrahiert werden. Das Ziel der automatischen Modellgenerierung ist die Abstraktion von der reinen Datengrundlage auf ein allgemeingültigeres Modell, das einerseits das bereits beobachtete Verhalten eines Systems genau nachahmt, dass anderseits aber allgemein genug ist, um auch zukünftiges, noch nicht aufgetretenes Verhalten vorhersagen zu können. Idealerweise sollte es möglich sein den extrahierten Simulationsmodellen ein beliebiges Szenario zu präsentieren, auf das sie dann in der Simulation möglichst „realistisch" reagieren. Ein *Szenario* besteht aus einer Menge von manipulierbaren Einflussfaktoren und einer Menge von Zufallsvariablen, d.h. eine Menge von nicht beeinflussbaren Faktoren, deren Wahrscheinlichkeitsverteilungen bekannt sein können aber nicht müssen. Darüber hinaus können Zielvorgaben definiert werden, die während der Simulation durch Anpassung veränderlicher Variablen angenähert werden sollen.

[22] http://wwwmath.uni-muenster.de/u/lammers/EDU/AgentenBaukastenBewertungen/php/db-main.php?db_action=overview

Eine Simulation kann für die folgenden Aufgabenstellungen verwendet werden:

- *Prognose*, um das zukünftige Verhalten eines Systems bei gleich bleibendem Szenario vorherzusagen. WIE wird sich das System bei gleich bleibenden Bedingungen verhalten?
- *Diagnose*, um das vergangene Verhalten eines Systems zu analysieren. WARUM hat sich das System in der Vergangenheit so verhalten?
- *Was-wäre-wenn-Analyse*, um das zukünftige Verhalten eines Systems bei verändertem Szenario vorherzusagen. WAS wäre, wenn man die Konfiguration des Systems folgendermaßen verändert?
- *How-to-achieve-Analyse*, um zu ermitteln unter welchen Bedingungen sich ein System in einer bestimmten Weise verhält. WELCHE Bedingungen müssen eintreten, damit sich das System in gewünschter Weise verhält?
- *Optimierung*, um veränderbare Variablen des Systems nach bestimmten Richtlinien zu optimieren. WAS ist die optimale Konfiguration, um ein bestimmtes Ziel zu erreichen?

Abhängig von der zu simulierenden Domäne, der verwendeten Modelle und der Simulationsinfrastruktur spricht man von unterschiedlichen Typen von Simulationen [ZePrKi00]. *Qualitative* Simulationen geben nur Tendenzen als Ergebnisse aus, wohingegen *quantitative* Simulationen exakte numerische Werte liefern. Eine Simulation kann dabei auf *virtuellen* Werten oder auf *realen* Daten basieren. Ersteres wird dabei meistens im Zusammenhang mit qualitativen Simulationen verwendet. Modelle die auf realen Daten basieren werden entweder *manuell* oder mit *maschinellen Lernverfahren* erzeugt. Der Simulationsprozess kann entweder *sequenziell* oder *parallel* ablaufen. Bei der parallelen, nebenläufigen Simulation wird eine Ablaufsteuerung oder spezielle Protokolle benötigt, um die verschiedenen Simulationspipelines zu synchronisieren. Werden alle Entitäten einer Domäne bis auf die kleinste Ebene herunter als einzelne Objekte modelliert, dann spricht man von einer *Mikrosimulatio*n. Das Gegenteil von Mikrosimulation ist die *Makrosimulation* bei der die einzelnen Entitäten einer Domäne zu größeren Einheiten zusammengefasst werden. Bei sehr komplexen agentenbasierten Simulationen empfiehlt es sich, die Agenten auf verschiedenen Rechner, die über ein Netzwerk verbunden sind, zu verteilten und *verteilte* Simulationen durchzuführen.

Das Ziel des SimMarket-Projektes war es, eine Plattform für quantitative und verteilte agentenbasierte (Mikro-)Simulationen [Klügel01] zu schaffen, bei der die Wissensbasen der Agenten mit maschinellen Lernverfahren (Data Mining) aus realen Daten extrahiert werden können.

3.1.4 Verwandte Arbeiten

Bevor der Entschluss gefallen ist, eine eigene verteilte Multiagenten-Plattform für Simulationen zu entwickeln, wurden zahlreiche verwandten Arbeiten aus dem Bereich agentenbasierte Simulation auf die Anwendbarkeit in SimMarket analysiert. Die interessantesten dabei waren die Folgenden:

Das *Consumat*-Modell von W. Jager [Jager00] ist eine multiagentenbasierte Umsetzung eines psychologischen Metamodells, das auf verschiedenen zum Verständnis des Konsumentenverhaltens relevanten Theorien basiert. Es überträgt ein psychologisches Konsumentenmodell in ein berechenbares Multiagentenmodell. Die Motivationsfaktoren der Konsumenten beruhen auf den verfügbaren Konsummöglichkeiten, ihren Fähigkeiten, Bedürfnissen, ihrer Zufriedenheit sowie ihrer Unsicherheit. Diese Faktoren bestimmen welche kognitiven Prozesse im Konsumenten-Agenten ablaufen. Falls ein Konsument unzufrieden ist, wird er durchführbare Konsummöglichkeiten suchen, um eine zufrieden stellende Situation zu erreichen. Wenn er unzufrieden aber immer noch unsicher ist, wird er soziale Vergleiche anstellen und verschiedene Möglichkeiten zur Nachahmung anderer Konsumenten betrachten. Wenn der Konsument zufrieden und sicher ist, wird er einfach sein bisheriges Verhalten wiederholen. Häufige Wiederholungen bilden die kognitive Basis von gewohntem Verhalten. Letztlich wird jemand, der zufrieden aber unsicher ist, einfach das Verhalten anderer imitieren, die dieselben Möglichkeiten haben. Dieses konzeptuelle Modell wurde in einem Multiagentensystem implementiert, in welchem jeder Konsument durch einen *Consumat-Agenten* repräsentiert wird. Dieses Modell abstrahiert sehr stark von realen Kaufakten, es fehlt eine Abbildung von dem abstrakten Modell auf einen realen Markt. Der benötigte Mechanismus, der aus realen Kunden- und Artikeldaten automatisiert ein berechenbares Modell generiert, fehlt.

Bei dem *SIMSEG* Projekt der Wirtschaftsuniversität Wien [BucMaz01] handelt es sich um eine Simulationsumgebung für die Analyse von Marktsegmentierungen und Positionierungsstrategien. Im Gegensatz zum *Consumat*-Ansatz setzt *SIMSEG* auf ein wahrnehmungsbasiertes Kundenmodell. Jede Marke wird durch eine Wahrnehmungsklasse repräsentiert, die die Eigenschaften dieser Marke beschreibt. Beispielsweise unterscheidet man bei Bier drei Wahrnehmungsdimensionen mit jeweils vier Wahrnehmungsindikatoren:

- Intensität des Geschmacks (stark, schmackhaft, würzig, schwer),
- Leichtigkeit (niedriger Alkoholgehalt, wenig Kalorien, erfrischend, leicht),
- Lifestyle (cool, jung, ‚in', dynamisch).

Ein Kunde wird ebenfalls durch ein Wahrnehmungsprofil modelliert, das seine Konsumpräferenzen abbildet. Diese Profile ermöglichen die Segmentierung der Kunden in verschiedene Kundengruppen, die dann für eine gezielte Simulation genutzt werden können. Dazu wird für eine oder mehrere Marken eine Wahrnehmungsklasse angelegt bzw. für bereits vorhandene Marken geändert und anschließend den modellierten Kunden in einer Simulation präsentiert. Zuvor definierte Positionierungsstrategien können so auf Erfolg getestet werden. *SIMSEG* basiert ebenfalls auf einem sehr abstrakten Modell, bei dem es in erster Linie nicht darum geht einen realen Supermarkt zu simulieren, also z. B. genaue Abverkaufsprognosen zu erstellen, sondern vielmehr um die langfristigen Auswirkungen von Image- bzw. Wahrnehmungsänderungen auf das Kundenverhalten auf Markenebene.

Brannon et al. präsentieren in ihrem Papier [BrUlAnPr00] einem Top-Down-Ansatz um virtuelle Verbraucher in der Bekleidungsbranche agentenbasiert zu simulieren. Ihr Ansatz basiert ausschließlich auf manuell definierten Fakten, die aus der Literatur abgeleitet worden sind. Zunächst muss der Benutzer für alle virtuellen Kunden ein Verbraucherprofil und für alle Artikel ein Produktprofil anlegen. Das Verbraucherprofil besteht aus Attributen wie Einkommen, Lifestyle, Alter, Modevorlieben, Bekleidungsbedürfnisse etc. Das Produktprofil besteht aus Fakten wie Preis, Kaufhäufigkeit, Prestige, Style etc. Bevor eine Simulation gestartet wird, generiert ein Modul aus dem Verbraucherprofil ein Verbraucherbedürfnisprofil. Dieses Profil wird dann während der Simulation mit den Produktprofilen verglichen. Je mehr Übereinstimmungen gefunden werden, desto höher ist die Wahrscheinlichkeit, dass der Kunde dieses Produkt kauft.

Said, Bouron und Drogoul [SaiBouDro02] verwenden ebenfalls einen Top-Down-Ansatz, um agentenbasierte Interaktionsanalysen von Kundenverhalten zu erstellen. Das Ziel ihrer Arbeit ist es, eine virtuelle Kundenpopulation zu erzeugen, die für die Simulation der Auswirkungen von Marketingstrategien in einen Wettbewerbsumfeld verwendet werden kann. Das Kundenverhaltensmodell basiert auf psychologischen Faktoren wie Imitation, Konditionierung und Innovationsfreude, die in manuell erstellten Formeln das Kaufverhalten eines Kunden ausdrücken. Zusätzlich kann das globale Verhalten der Agentenpopulation mit Hilfe von genetischen Algorithmen an reales Marktverhalten angepasst werden. Auf Grund der einfachen psychologischen Attribute wurden keine exakten quantitativen Abverkaufsprognosen generiert, sondern qualitative Aussagen gewonnen.

3.2 Definition des PHMAS

Bei der Entwicklung des SimMarket-System wurde schnell klar, dass die Systemanforderungen weit über das hinausgehen, was derzeit verfügbare Agentenplattformen

bieten. Es wird ein System benötigt, um agentenbasierte Simulationen mit komplexen Multiagentensystemen mit tausenden mobilen Agenten durchzuführen. Die Agenten müssen die Fähigkeit besitzen, holonische Gruppen zu bilden und probabilistisches Wissen aus Terabytes an Daten zu extrahieren, zu modellieren und daraus mittels Inferenzen wiederum neues Wissen zu deduzieren: Eine solche Plattform stand seinerzeit nicht zur Verfügung.

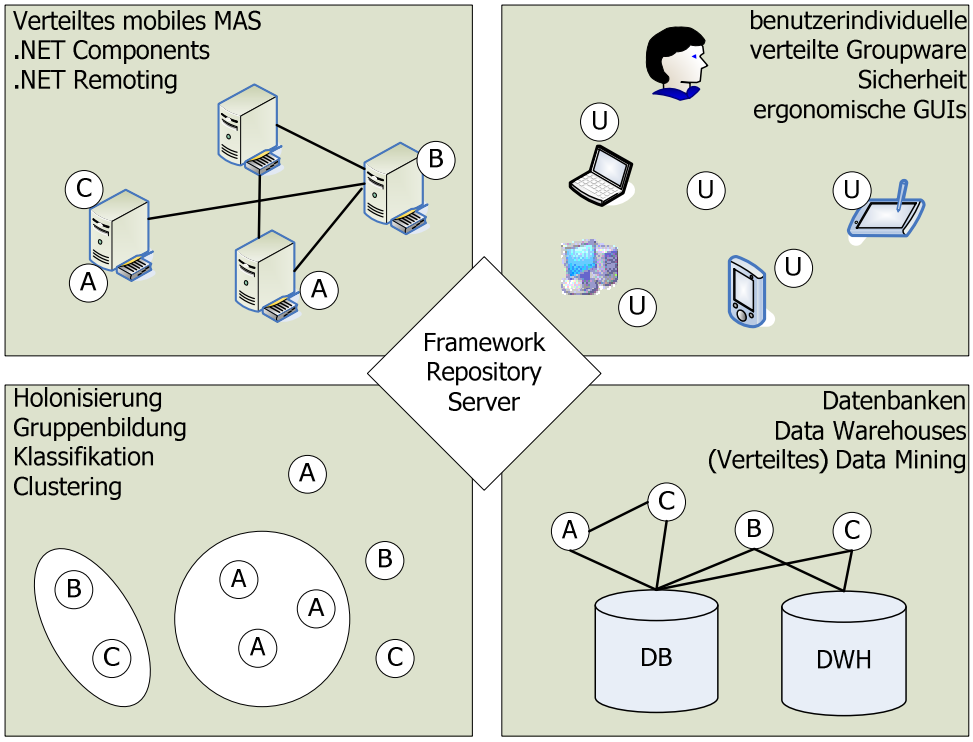


Abbildung 7: Das SimMarket-Agentenframework

Um alle Entitäten komplexer Domänen modellieren zu können, entwickelten wir das probabilistische holonische Multiagentensystem *PHMAS,* das aus einer Menge von probabilistischen holonischen Agenten (*PH-Agent*) $A_i = \{A_1^a,...,A_n^z\}$ mit unterschiedlichen Agententypen $AT = \{a,...,z\}$, dem MAS-*Grid* (*MASg*) und der *Simulationskomponente* besteht.

Das MASg besteht aus einem Repository-Framework *RF* und einer Menge von Agenten-Frameworks $AF = \{AF_1,...,A_m\}$. Die Agenten-Frameworks werden auf $\leq$ m Computern in einem Netzwerk distribuiert, sie verwalten Mengen $A_j \subseteq A_n$ von Agenten. Ein AF kann Agenten empfangen, neue Agenten initialisieren und Agenten aus dem Framework entfernen. Dieses ist notwendig, weil ein Agent normalerweise nicht im Stande ist, sich zu entfernen, während andere Agenten Referenzen auf diesen Agenten halten. Außerdem können Agenten im Normalfall, schon aus Sicherheitsgründen, nicht einfach von einem

Computer zum anderen migrieren. Eine Infrastruktur, die Agenten empfängt, ausführt und sich mit Sicherheitsfragen beschäftigt, ist erforderlich. Das RF stellt allen Agenten und allen Benutzern des Systems „White Page“ and „Yellow Page“ Funktionalitäten bereit. Das MAS unterstützt unterschiedliche Arten der Agentengruppierung und insbesondere die Methode der Holonisierung (Verschmelzung).

Jeder Agent kann ein Teil von einem oder mehreren Holonen H_i mit gleichen Agententypen $\{A_1^x,...,A_n^x\}$ oder unterschiedlichen Agententypen $\{A_1^a,...,A_n^z\}$ sein. Das RF verwaltet die Referenzen auf alle Teile des PHMAS $RF_{Ref} := \{Ref(A_i)\} \cup \{Ref(H_i)\} \cup \{Ref(AF_i)\} \cup \{Ref(WS_i)\}$, wobei WS_i die Weltserver sind auf die die Agenten A_i und die Holonen H_i Zugriff haben.

Der folgende Abschnitt beschreibt die generische Umsetzung des MASg als Teil des PHMAS. Abschnitt 3.4 beschäftigt sich mit der probabilistischen holonischen Agentenarchitektur PH-Agent.

3.3 MASg - Das SimMarket-Grid für Multiagentensysteme

3.3.1 Allgemeine Anforderungen

Das SimMarket-Simulationssystem muss in der Lage sein, sehr viele Agenten gleichzeitig zu simulieren, die wiederum die Fähigkeit haben sollten, sich dynamisch zur Laufzeit auf ein Netzwerk von Computern zu verteilen, um Ressourcenengpässe auf einzelnen Rechnern zu vermeiden. Der parallele Zugriff von mehreren Benutzern auf das System, die an verschiedenen geografisch getrennten Filialen arbeiten, sollte möglich sein. Auch der mobile Zugriff über drahtlose Netzwerke (*UMTS, WLAN* etc.) mittels *PDAs* und *Tablet-PC* sollte prinzipiell realisierbar sein. Darüber hinaus soll das System derart verteilbar sein, dass Händler und Hersteller kooperativ an der Sortimentsoptimierung (*CM - Category Management* bzw. *TPM - Trade Promotion Management*) und der Nachbestellung (*CPFR - Collaborative Planning, Forecasting and Replenishment*) arbeiten können. So ist es denkbar, dass ein Teil des Simulationssystems beim Händler läuft und ein anderer beim Hersteller. In zukünftigen Anwendungsszenarien werden auch die Zulieferfirmen eine wichtige Rolle spielen, so dass die Supply Chain vom Hersteller zum Händler bis hin zum Endkunden letztlich lückenlos vom SimMarket-System abgedeckt werden wird. Im Rahmen der Einführung von *RFID*-basierten (*Radio Frequency Identification*) Logistiksystemen im Einzelhandel wird auch die Agententechnologie eine immer wichtigere Rolle spielen, da diese Konzepte ideale Schablonen für die Umsetzung eines solchen Systems sind (siehe [CACM05]). Die bei einem RFID-System anfallenden Transaktionsdaten der an einer Supply Chain beteiligten Unternehmen sind eine ideale Grundlage für das SimMarket-System. Das Agenten-Grid MASg ermöglicht eine einfache temporäre und partielle Kopplung mehrerer

Systeme, um die Kooperationen zwischen Unternehmen optimal zu unterstützen. Aber auch innerhalb eines Unternehmens und einer Filiale, abhängig von der existierenden IT-Infrastruktur, sind unterschiedliche Netzwerkstrukturen denkbar:

1. Sowohl Datenbank, als auch die Agenten und die grafischen Benutzeroberflächen (GUIs) laufen auf einem Rechner. Nutzt nur ein Anwender das System und ist der verwendete Rechner leistungsfähig genug, dann ist diese Konfiguration ausreichend.
2. Die Datenbank und die Agenten laufen auf einem Server, während der bzw. die Benutzer auf ihren Rechnern nur die Benutzeroberflächen starten. Wenn mehrere Benutzer deren Rechner weniger leistungsfähig sind, das System verwenden, ist dies die bevorzugte Architektur.
3. Sollten die Benutzer einen performanten Rechner haben, dann ist es auch möglich, die Datenbank auf einem Server zu installieren und sowohl die Agenten als auch die GUIs auf den Rechnern der Anwender laufen zu lassen.
4. Ein weiteres Szenario wäre, die Datenbank und die Agenten jeweils auf verschiedenen Servern laufen zu lassen, so dass die Benutzer wiederum nur die Benutzeroberflächen auf ihren Rechnern besitzen müssten. Diese Konfiguration wäre zu bevorzugen, falls die benötigten Ressourcen der Agenten denen eines Standard-PCs übersteigen.
5. Bei einem noch größeren Ressourcenbedarf ist es möglich, die Agenten auf mehrere Server zu verteilen bzw. diese autonom migrieren zu lassen.

Der einfache Zugriff auf heterogene Datenbanken und Data Warehouses spielte ebenfalls eine wichtige Rolle beim Design von MASg.

3.3.2 Architektur

MASg besteht aus drei Komponenten: dem Repository-Framework, dem Agenten-Framework und dem Basisagenten, die die folgenden Eigenschaften realisieren:

- Verteilbarkeit, Mobilität => klonen, kopieren, seriellisieren, deseriellisieren, initialisieren => *load balancing,*
- Autonomie, Nebenläufigkeit => Mehrbenutzerzugriff,
- Verwaltung: Suchen und Finden von Agenten, auch anhand von spezifizierten Fähigkeiten,
- Sicherheit,
- Initialisieren und Beenden von Agenten,
- Agentenkommunikation,

Das Repository-Framework ist der „Kopf“ aller Agenten-Frameworks und besitzt die IP-Nummern aller Agenten-Frameworks im MASg-Verbund. Jedes neue Framework und jeder Agent muss sich beim Repository anmelden, um in den Verbund aufgenommen zu werden. Es ist vorgesehen pro verfügbaren Rechner ein Agenten-Framework zu installieren, das dort wiederum für die Verwaltung der Agenten verantwortlich ist. Darüber hinaus kennt das Repository-Framework alle im System agierenden Agenten und deren Lokalitäten. Jeder Agent und jedes Framework identifiziert sich dabei über eine weltweit eindeutige Identifikationsnummer (GUID - Global Unique IDentification), die bei der Anmeldung von Agenten und Frameworks beim Repository-Framework von diesem vergeben wird. Das Repository-Framework bietet sowohl „White-Pages“- als auch „Yellow-Pages“-Dienste an. Über Ersteres kann man sich die Adresse von Frameworks und Agenten oder sogar die direkte Referenz auf einen Agenten über seine GUID geben lassen. Die „Yellow-Pages“-Dienste bieten Funktionen an, um beispielsweise die Referenzen aller Agenten eines bestimmten Typs zu erhalten. Migriert ein Agent von einem Rechner auf einen anderen, so wird dies sofort dem Repository-Framework von dem migrierenden Agenten mitgeteilt, so dass das Repository immer eine gültige Referenz aller Agenten besitzt.

Die Agenten-Frameworks sind für die lokale Verwaltung auf einem dem MASg angehörigen Rechner verantwortlich. Die lokalen Frameworks sind in der Lage neue Agenten zu initialisieren und nicht mehr gebrauchte zu terminieren. Ein Agent kann sich nicht selbst beenden, da er nicht weiß, ob er noch gebraucht wird oder nicht, d. h., ob andere Agenten oder Benutzer noch Referenzen auf ihn halten. Des Weiteren werden die Agenten-Frameworks für das Annehmen und das Verschicken von Agenten benötigt. Beschließt ein Agent auf einen anderen Rechner zu migrieren, sendet dieser ein Signal an sein lokales Agenten-Framework, welches daraufhin den Agenten vom aktiven in einen passiven Zustand versetzt, ihn seriellisiert und an das gewünschte Ziel-Framework verschickt. Das entgegennehmende Framework deseriellisiert den Agenten und startet ihn in einem neuen *Thread*[23]. Das verschickende Framework aktualisiert die Einträge im Repository-Framework und beendet den zurückgebliebenen passiven Agenten.

Der Basisagent realisiert einen generischen Agenten, der die Kommunikation und die Interaktion mit den Frameworks transparent umsetzt. Neue Agenten werden nach einfachen

[23] Ausführungsstrang innerhalb eines Prozesses, der früher auch *task* genannt wurde. *Threads* werden von vielen modernen Betriebssystemen angeboten und sogar von einigen Prozessoren (z. B. Pentium 4) hardwareseitig realisiert, da diese wesentlich schlanker sind als vollständige Prozesse.

Regeln von diesem Basisagenten abgeleitet und erben somit seine gesamte Funktionalität. Der Agentendesigner kommt mit den Anmelde- und Verwaltungsfunktionen des Basisagenten in keinerlei Kontakt, da diese transparent und generisch im Basisagenten realisiert sind. Der Basisagent besitzt einfache Regeln, die beschreiben unter welchen Umständen ein Agent auf einen anderen Rechner migriert. Diese Regeln können vom Designer bei Bedarf aktiviert und deaktiviert werden. Eine einfache Regel lautet beispielsweise:

Wenn die CPU-Auslastung im Zeitraum [t..l] im Durchschnitt über Wert X liegt, dann verschiebe dich auf einen Rechner mit weniger Auslastung.

Die Auslastungen von anderen Rechnern im MASg können von den Agenten entweder über die lokalen Frameworks oder über das globale Repository-Framework erfragt werden. Mit Hilfe dieser Regeln lassen sich bereits einfache dezentrale Load-Balancing-Algorithmen in den Basisagenten umsetzen. Das Verschieben von Agenten wird außerdem benötigt, falls einzelne Rechner aus dem Grid genommen werden sollen und infolgedessen, deren Frameworks abgeschaltet werden müssen. Das Beenden eines Frameworks wird erst dann endgültig durchgeführt, wenn die laufenden Agenten benachrichtigt wurden und auf andere Frameworks migriert sind.

Das Beenden eines Repository-Frameworks kann indes zwei Folgen haben: 1. Das gesamte MAS-Grid wird beendet, d. h., alle Agenten und Frameworks auf allen Rechnern werden beendet. 2. Die Funktion des Repository-Frameworks wird von einem anderen Agenten-Framework übernommen. Dies ist prinzipiell möglich, da jedes Agenten-Framework auch die Fähigkeiten eines Repository-Frameworks hat, die allerdings deaktiviert sind. Nur ein entsprechendes Flag bestimmt, ob ein Frameworks Repository ist oder nicht. Beim Beenden eines Repository-Frameworks kann somit das Flag bei einem anderen Agenten-Framework gesetzt werden. Dazu muss diesem Framework der Inhalt des aktuellen Repository übergeben werden. Außerdem müssen die laufenden Agenten des aktuellen Repository-Frameworks auf andere Rechner migrieren. Das wichtigste beim Verschieben von Agenten und der Repository-Funktionalität ist die Wahrung der Systemkonsistenz. Zugreifende Agenten und Benutzer müssen zu jedem Zeitpunkt gültige und eindeutige Referenzen auf alle Objekte des MASg bekommen können oder idealerweise zu jedem Zeitpunkt besitzen.

Das Starten von neuen Frameworks kann von zentraler Stelle remote auf allen verfügbaren Rechner auf denen ein Framework existiert erfolgen. Das Verwalten der Frameworks und aller laufenden Agenten kann ebenfalls von verschiedenen Stellen im Netzwerk durchgeführt werden, das heißt, Agenten und Frameworks können überwacht, beendet oder manuell verschoben werden.

Wird ein Agent für die Lösung eines Problem benötigt, so kann dieser auf dem lokalen Framework erzeugt werden oder bei schwachen Rechnern direkt remote auf einem anderen dem MASg angehörigen Rechner. Der Erzeugungsort eines Agenten kann sowohl von Systembenutzern als auch von agentenerzeugenden Systemen erwählt werden. Jeder Benutzer des Systems arbeitet ausschließlich mit seinen eigenen Agenten, die für ihn erzeugt worden sind. Der Zugriff auf Agenten anderer Benutzer ist aus Sicherheits- und Konsistenzgründen nicht erlaubt. Dadurch entfällt ein erheblicher Synchronisierungs- und Koordinierungsaufwand, der nötig wäre, um den Mehrfachzugriff zu ermöglichen.

Das MASg unterstützt unterschiedliche Arten von Autonomiemodellen:

1. Das klassische Modell, bei dem jeder Agent in einem eigenen Thread läuft. Dies kann allerdings bei massiven Multiagentensystemen mit mehreren tausend Agenten zu starken Performanzproblemen führen, da der Overhead der durch die Koordinierung (Ablaufplanung/Scheduling, Synchronisation, Kommunikation) entsteht exponentiell wächst.
2. Bestimmte Gruppen von Agenten werden in einem Thread zusammengefasst. Dies ist häufig sinnvoll, da dies das Problem aus 1. beinahe vollständig beseitigt.
3. Die Agenten, die einem Benutzer zugeordnet sind, werden in einem Thread zusammengefasst. So können alle Benutzer parallel auf dem System arbeiten, ohne sich gegenseitig zu beeinflussen, die Aufgabenstellungen eines Benutzers werden hingeben sequenziell abgearbeitet.
4. Die Kombination aus 2. und 3. bei der die Zusammenfassung der Agenten einerseits auf Benutzerebene und anderseits nach Aufgabenstellung der Agenten erfolgt.

Die Frameworks bzw. die Basisagenten ermöglichen die Kommunikation der Agenten untereinander und die Kommunikation von Benutzern mit Agenten. Der Basisagent bietet eine einheitliche Kommunikationsschnittstelle, über die er Kontakt zu anderen Agenten aufnehmen kann und über die anderen Agenten Kontakt mit ihm aufnehmen können. Die globale Kommunikationsinfrastruktur wird dabei von den Agenten-Frameworks realisiert. Die Basisagenten bzw. die Frameworks sind dabei nach allen Aspekten gegen unbefugte Angriffe und Abhörversuche von Außen abgesichert. Die Sicherheit von MASg wird durch die Maßnahmen Authentisierung, Autorisierung und Verschlüsselung realisiert.

3.3.3 Implementierungsanforderungen

Die folgenden Anforderungen, die während der theoretischen Konzeption definiert worden sind, müssen erfüllt werden und darüber hinaus praktischen Ansprüchen genügen:

- Die Anwendung muss *verteilt* ausführbar sein, d. h. im Idealfall sollte jedes Objekt auf einem beliebigen Rechner abgelegt und von dort transparent wie ein lokales Objekt verwendet werden können.
- *Multi-threading* sollte einfach und performant nutzbar sein.
- Die Implementierung von *Kommunikationsinfrastrukturen* zwischen (verteilten) Objekten sollte möglich sein und über *XML* erfolgen.
- Die verwendete Programmiersprache sollte *objektorientiert* sein, um Agenten intuitiv als gekapselte Objekte implementieren zu können.
- *Sicherheit***:** Da es sich bei SimMarket um ein Projekt handelt, bei dem wir mit sensiblen Daten namhafter Handelsunternehmen arbeiten, sollte der Zugriff für externe Nutzer beschränkt bzw. möglichst verhindert werden können.

Weitere Anforderungen, meist aus praktischen Gesichtspunkten, sind:

- Die Unterstützung von *Webservices*, um die Funktionalität von SimMarket plattform- und systemunabhängig verfügbar zu machen.
- Eine gute Anbindung an *Datenbanken* bzw. *Data Warehouses*, um auf die riesigen Datenmengen der Handelsunternehmen zugreifen zu können.
- Die Unterstützung *mobiler Endgeräte* wie z. B. Pocket PCs, um Anwendern einen mobilen Zugriff auf wichtige Daten zu ermöglichen.
- Die verwendete Technologie sollte eine *hohe Akzeptanz* in der Wirtschaft insbesondere bei den beteiligten Handelsunternehmen haben, da die SimMarket Software in deren Märkten zu Testzwecken eingesetzt werden soll.
- Insgesamt gesehen sollte die Entwicklungsplattform eine *hohe Performanz* in allen Belangen bieten, um möglichst viele Funktionen in Echtzeit realisieren zu können und um dem Benutzer ein komfortables Arbeiten zu ermöglichen.
- Dazu sollte die Plattform eine gut strukturierte und *performante GUI-Bibliothek* bieten, die sich auch visuell programmieren lässt.
- Der *Support* des Herstellers sollte gesichert sein. Die verwendete Plattform sollte eine sichere *Zukunft* haben.

Um diese Anforderungen zu erfüllen, war es nötig ‚State-of-the-Art'-Technologien zu verwenden. Zur Auswahl standen: das .NET Framework mit C#, Java und C++. Unsere Entscheidung viel nach einigen Tests und Versuchsreihen auf das *.NET Framework* von Microsoft und die Programmiersprache **C#**, da diese die oben genannten Anforderungen am Besten erfüllt.

Insbesondere die von Grund auf neu gestalteten .NET Bibliotheken, die Sprachunabhängigkeit von .NET und die durch Mono[24] gegebene Plattformunabhängig, die performanten GUI-Bibliotheken, das Visual Studio .NET, der intuitive Aufbau von C# und die Remoting-Fähigkeiten waren Punkte, die dazu geführt haben, dass wir uns für .NET/C# entschieden haben. Die Remoting-Funktionalität erlaubt es, jedes beliebige Objekt auf jeden beliebigen .NET Rechner zu transferieren und es von dort transparent wie ein lokales Objekt auszuführen, was die Entwicklung eines Agentenframeworks erheblich erleichtert hat.

3.4 PH-Agent: Probabilistische holonische Agentenarchitektur

3.4.1 Einleitung

Die wesentliche Herausforderung war die Entwicklung einer geeigneten Agentenarchitektur. Folgende Fragen galt es zu klären:

- Welche Repräsentationsformalismen sind geeignet, um das Abverkaufsverhalten von Artikeln abzubilden?
- Welche Formalismen können effizient als Wissensbasen der Agenten in die Praxis umgesetzt werden? Und vor allem auf wie?
- Wie werden die Wissensbasen mit Wissen gefüllt? Wie wird das Wissen über die Laufzeit des Agenten aktuell gehalten?
- Welche Inferenzarten gibt es für die ausgewählte Wissensrepräsentation bzw. welche wird für die Aufgabenstellung benötigt? Welches Wissen soll bzw. muss inferiert werden?
- Wie sieht eine geeignete Kontrollarchitektur aus, die die Wissensbasis und andere Komponenten in die Agenten integriert?
- Welche Art von Gruppenbildung wird benötigt und muss folglich von der Agentenarchitektur unterstützt werden?

Obwohl die Frage nach einer geeigneten Wissensrepräsentation im Vordergrund stand, beginnt dieses Kapitel mit der Gesamtarchitektur der Agenten. Genau genommen ist diese Architektur das Ergebnis des Entwicklungsprozesses, dennoch soll hier aus Gründen der Übersichtlichkeit und des Verständnisses die Architektur als Erstes beschrieben werden.

---

[24] http://www.mono-project.com

3.4.2 Agentenarchitektur

Bei der Frage nach einer geeigneten Wissensrepräsentation war wichtig, dass diese mit maschinellen Lernverfahren kompatibel sein muss. Die Datengrundlage wird aus einer Vielzahl von operativen und dispositiven Systemen eines Einzelhändlers extrahiert und aggregiert. Folglich ist das resultierende Datenaggregat nicht frei von falschen, unvollständigen und fehlenden Daten. Des Weiteren ist die Domäne der Artikelsimulation in ihrer Komplexität etwa mit einer Wetterprognose vergleichbar und schon deshalb nicht mit einem einfachen System modellierbar: Selten sind eindeutige lineare Trends und sich exakt wiederholende Kaufmuster in den Artikeldaten zu beobachten. Deshalb haben wir einen probabilistischen Ansatz gewählt, der mit schlechten und fehlenden Daten umgehen kann. Probabilistische Ansätze basieren auf einem Wahrscheinlichkeitsmodell, das jedes Ereignis und jede Regel mit einer Wahrscheinlichkeit P ausstattet. Aussagen des Modells haben nie absoluten Charakter, sondern sind immer mit einer Wahrscheinlichkeit verbunden, z. B. Ereignis X tritt mit einer Wahrscheinlichkeit P auf, wenn die Bedingungen C erfüllt sind. Weitere Details zum Thema Wissensbasis sind im nächsten Abschnitt zu finden.

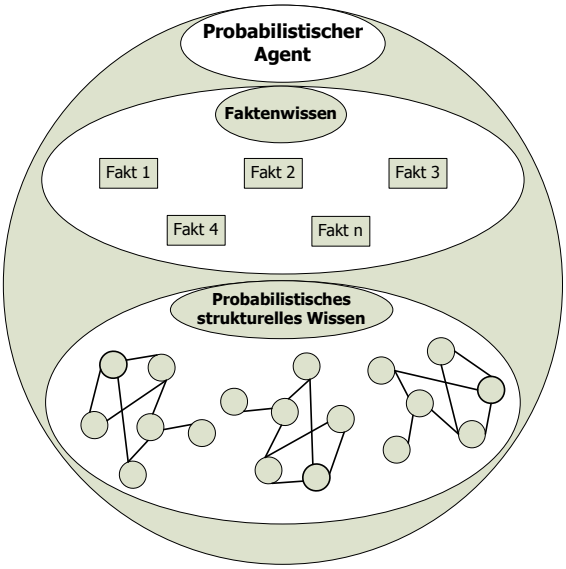


Abbildung 8: Probabilistische Agentenstruktur

Ein PH-Agent A = (F, S) hat deshalb eine Wissensbasis (KB, engl. *Knowledge Base*), die aus zwei Bestandteilen besteht: einer Komponente F für Faktenwissen und einer Komponente S für probabilistisches strukturelles Wissen. F speichert reines Faktenwissen ohne Relationen zwischen den Einträgen, z. B. den Preis eines Artikels, die Regalposition, den Hersteller etc. Einträge in F können, müssen aber nicht Konzepte in S sein. Die Wissenskomponente S: = $\{VN_i\}$ entspricht einer Menge von *Verhaltensnetzen* VN, bei denen

es sich um eine Menge von probabilistischen Netzen handelt, die auf den vergangenen Erfahrungen des Agenten basieren.

Darüber hinaus besitzt jeder Agent eine Menge von Diensten AS (*agent services*) und Handlungsmöglichkeiten AA (*agent actions*), die von anderen Agenten, Benutzern oder durch die Wissensbasis angestoßen werden können. Die Agentendienste können einfache Informationskanäle wie z. B. Webservices oder komplexere informationsverarbeitende Komponenten wie Cluster- oder Klassifikationsmodule sein. Durch Gruppenbildung, genauer gesagt durch Holonisierung, sind die Agenten in der Lage ihr Wissen und ihre Fähigkeiten zu bündeln. Die Holonisierungskomponente realisiert sowohl die *Head*-Funktionalitäten als auch die Verwaltung der Unteragenten und die Komposition von Agenten zu Holonen. Für verschiedene Arten von Agenten und Holonen und für unterschiedliche Komponenten innerhalb der Agenten werden unterschiedliche Aggregierungs- und Verschmelzungsmethoden gebraucht, die Bestandteil der Holonisierungskomponente sind (siehe Abschnitt 3.4.7).

3.4.3 Probabilistische Wissensbasis

Die probabilistische Wissensbasis speichert das strukturelle und mit Wahrscheinlichkeiten versehene Wissen eines Agenten. Bei der Erzeugung des Agenten muss dieser mit einem Basiswissen ausgestattet werden, um grundlegende Funktionalitäten realisieren zu können. Wird ein Agent für eine spezifische Aufgabe erzeugt, so muss der Erzeuger dafür sorgen, dass der Agent von Beginn an mit dem nötigen Wissen ausgestattet wird, damit dieser überhaupt eine Chance hat seine Aufgabe zu erfüllen. Des Weiteren muss ein Agent in der Lage sein, neues Wissen zu akquirieren. In regelmäßigen Abständen nimmt ein Agent über seine Sensoren neue Daten auf, die von ihm gefiltert und konvertiert werden müssen, um sie schließlich als neues Wissen in seine Wissensbasis abzulegen. In der Domäne der Artikelsimulation wird aus der Artikelabverkaufshistorie eine initiale Wissensbasis folgender Art erzeugt.

Definition: Die Wissenskomponente $S := \{VN_i\}$ entspricht einer Menge von *Verhaltensnetzen* VN (engl. *behaviour network*), die als Tupel $VN := (B(D_{[t,l]}), D_{[t,l]}, D_{>l}, DK)$ definiert werden, wobei $B(D_{[t,l]})$ eine Menge von probabilistischen Netzen B darstellt, die auf den vergangenen Erfahrungen des Agenten basieren. Der Datensatz $D_{[t,l]}$ entspricht den vergangenen Sensordaten des Agenten, die bereits mittels Data Mining in Netzen gespeichert worden sind, $D_{>l}$ sind die gegenwärtigen Sensordaten, die noch in kein Netz integriert worden sind, aber für die aktuelle Aufgabe des Agenten genutzt werden kann und DK (engl. *domain knowledge*) ist das Domänenwissen (des Erzeugers), das in die Agenten integriert worden ist.

Jeder Datensatz eines Verhaltensnetzes entspricht einem Zeitintervall [t, l]. Folglich haben die Netze eine spezifische Zeitkonfiguration $T_{VN} = \{t, l, k, b\}$, wobei t den Beginn des Sammelns von Daten markiert, l dem letzten Zeitpunkt entspricht, von dem sensorische Daten in dem Netzwerk enthalten sind, k der Anzahl der Messungen der Welt entspricht und b die Zeitbasis darstellt, d. h. der Länge des Zeitintervalls zwischen den Messungen.

Die Realisierung der Verhaltensnetze könnte prinzipiell durch jede geeignete Form von probabilistischen Netzen erfolgen. In SimMarket haben wir uns nach eingehender Analyse (siehe Kapitel 3.5) verschiedener Modelle für die Theorie der Bayes'schen Netze entschieden, die im Folgenden eingeführt werden soll. Anschließend werden die Bayes'schen Netze sukzessiv zu Verhaltensnetzen, wie sie in SimMarket verwendet werden, ausgebaut.

3.4.3.1 Bayes'sche Netze

Ein Bayes'sches Netz (BN) ist ein Wissensrepräsentationsformalismus der KI (siehe z. B. [RusNor03], [KorNic04], [Stephenson00]), der es erlaubt, unsicheres Wissen zu modellieren und darauf Inferenzmethoden anzuwenden. BN sind grafische Modelle, deren Knoten Zufallsvariablen (diskrete oder kontinuierliche) und deren Kanten Abhängigkeiten zwischen diesen repräsentieren. BN modellieren (un-)bedingte Abhängigkeiten zwischen Variablen, die beim Vorliegen von neuen Informationen in Form von probabilistischen Annahmen mit Inferenzmechanismen aktualisiert werden können. Am folgenden Beispiel soll der Aufbau und die Funktionsweise BN eingeführt werden. Es stammt von Lauritzen und Spiegelhalter aus dem Jahre 1988 [LauSpi88] und heißt *Asia*.

Asia (siehe Abbildung 9) ist ein kleines Bayes'sches Netz, das die Wahrscheinlichkeit berechnet, mit der ein Patient an Tuberkulose, Lungenkrebs oder Bronchitis erkrankt ist, auf Basis verschiedener Indikationen, wie zum Beispiel, ob der Patient raucht oder sich kürzlich in Asien aufgehalten hat. Kurzatmigkeit (Dyspnoea) kann eine Folge von Tuberkulose, Lungenkrebs, Bronchitis oder einer Kombination dieser Krankheiten sein oder aus anderen Gründen auftreten. Ein Besuch in Asien erhöht das Risiko von Tuberkulose, während Rauchen ein bekannter Risikofaktor für Lungenkrebs und Bronchitis ist. Das Ergebnis einer einfachen Brustuntersuchung mit Röntgenstrahlung lässt normalerweise nicht zwischen Lungenkrebs und Tuberkulose unterscheiden, ebenso wenig lässt sich das Vorhandensein von Dyspnoea direkt aus dem Röntgenbild diagnostizieren. Erfahren wir, dass ein Patient Raucher ist, dann verändern wir unsere Annahmen bzgl. seines Lungenkrebs- und Bronchitisrisikos, lassen jedoch die Annahme bzgl. Tuberkulose unverändert (d. h., dass das Ereignis ‚Tuberkulose' bei einer gegebenen leeren Menge von Variablen bedingt unabhängig von ‚Rauchen' ist). Angenommen wir erhalten ein positives Ergebnis der Röntgenstrahlenuntersuchung (‚X-Ray'), dann beeinflusst dies unsere Annahmen bzgl. ‚Tuberkulose' und ‚Lungenkrebs', aber nicht unsere Annahmen bzgl. ‚Bronchitis' (das

bedeutet, dass ‚Bronchitis' bei gegebener Variable ‚Raucher' bedingt unabhängig von ‚X-Ray' ist). Wissen wir allerdings zusätzlich, dass der Patient an Kurzatmigkeit leidet, dann verändert das Ergebnis der Röntgenstrahlenuntersuchung dennoch unsere Annahmen bezüglich Bronchitis (d. h., dass unter den Bedingungen ‚Raucher' und ‚Dyspnoea', eine Bronchitis' nicht bedingt unabhängig von ‚X-Ray' ist).

Ein Bayes'sches Netz beschreibt also die kausalen und diagnostischen Zusammenhänge von Ereignissen und Zuständen. Eine Kante von einem Knoten A zu einem Knoten B bedeutet, dass B bedingt abhängig von A ist, A also eine direkte Wirkung auf B hat. Umgekehrt kann man aber auch sagen, dass die Wahrscheinlichkeit von A umso größer ist, je höher die Wahrscheinlichkeit von B ist. Vorhandenes Wissen über einen Sachverhalt kann in Form von Evidenzen in das Netz eingegeben werden und mit Hilfe eines Inferenzalgorithmus über das Netz propagiert werden. Setzt man beispielsweise eine Evidenz bei dem Knoten ‚Raucher', kann man die revidierte Wahrscheinlichkeit für Lungenkrebs aus dem Netz auslesen. Umgekehrt kann aber auch eine Evidenz bei dem Knoten ‚Dyspnoea' gesetzt werden, um die möglichen Gründe für Kurzatmigkeit zu erfahren. Es lassen sich also sowohl kausale aus auch diagnostische Fragen an das Netz stellen.

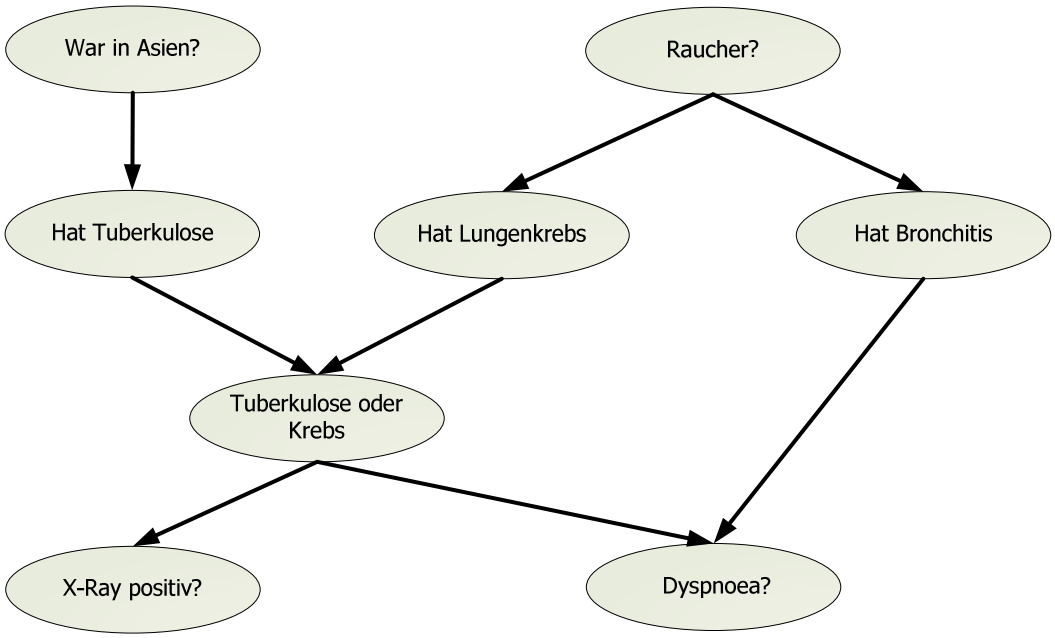


Abbildung 9: Bayes'sches-Netz-Beispiel „Asia"

Nehmen wir an, dass wir einen Datensatz D haben, der Messwerte (bzw. Fälle oder „Cases") $C_1^W,...,C_k^W$ einer Welt W zum Zeitpunkt t ∈ [1..k] enthält, die aus einer Menge von Werten bestehen, die einer Menge von Variablen $X_1,\ldots,X_n$, zugewiesen sind, wobei $X_i \in \Re$ ist. Z. B. könnte der Datensatz bzw. Weltzustand C_i die Beobachtungen (Sensorinput) eines Agenten

bezüglich der Welt, in der er während eines bestimmten Zeitabschnitts [t, l] „existiert“, darstellen. Jetzt soll aus diesen Rohdaten ein Modell der Welt des Agenten generiert werden, das es dem Agenten erlaubt, auf der Basis seiner Erfahrungen in der Vergangenheit Annahmen über die Zukunft zu machen und infolgedessen rationale Entscheidungen zu treffen. Gewünscht ist ein Modell, dass bei einer gegebenen Situation X (= ein Datensatz) die Wahrscheinlichkeit P für bestimmte Zielvariablen $X_i \ldots X_j \in X$ angibt. Eine Zielvariable kann sowohl informativen Charakter haben und weitere Informationen über die Welt liefern, so kann beispielsweise aus dem Vorliegen bestimmter Symptome die Wahrscheinlichkeit für verschiedene Krankheiten berechnet werden, als auch für eine bestimmte Aktion stehen und die Wahrscheinlichkeit für das Gelingen dieser repräsentieren.

Folgendes Beispiel soll die Funktionsweise illustrieren: Ein Agent in Form eines autonomen Roboters kann beispielsweise durch einen entsprechenden Knoten im Netz den Erfolg einer geplanten Vorwärtsbewegung berechnen. Ist der Weg versperrt würde der Erfolgsknoten eine Wahrscheinlichkeit von 100% auf dem Zustand ‚misslingt’ berechnen. Ist der Weg steinig aber passierbar käme vielleicht ein Wert von 60% für ‚misslingt’ und 40% für ‚gelingt’ heraus. Diese Werte ergeben sich aus den Erfahrungen des Agenten in der Vergangenheit und verdeutlichen, dass man in vielen Domänen, wie auch in der Supermarktsimulation, mit binären Aussagen nicht arbeiten kann.

Bayes’sche Netze sind probabilistische Modelle, die auf dem Wahrscheinlichkeitskalkül von Fermat und Pascal aus dem 17.ten Jahrhundert basieren. Wie die meisten Teilgebiete der modernen Mathematik wird die Wahrscheinlichkeitstheorie mengentheoretisch formuliert und axiomatisch aufgebaut. Die axiomatische Formulierung der Theorie geht auf Andrei N. Kolmogorov zurück, der in den 1930er-Jahren in Berlin [Kolmogorov33] die folgenden drei Axiome formulierte:

Sei U das Universum aller möglichen Ereignisse und $X \subseteq U$ ein Ereignisraum, der sich aus Teilmengen von U zusammensetzt und P das Wahrscheinlichkeitsmaß $P: X \rightarrow [0,1]$.

Axiom 1: $P(U) = 1$

Die maximale Wahrscheinlichkeit eines (sicheren) Ereignisses ist per Definition auf 1 festgelegt.

Axiom 2: Für alle $X \subseteq U$ gilt $0 \leq P(X) \leq 1$

Für jedes Ereignis $X \subseteq U$ ist die Wahrscheinlichkeit eine reelle Zahl zwischen 0 und 1. Negative Werte sind nicht möglich.

Axiom 3: Für alle X, Y $\subseteq$ U gilt, wenn X $\cap$ Y = $\varnothing$, dann P(X $\cup$ Y) = P(X) + P(Y)

Die Wahrscheinlichkeit einer Vereinigung abzählbar inkompatibler Ereignisse entspricht der Summe der Wahrscheinlichkeiten. Inkompatible Ereignisse sind disjunkte Mengen X, Y,…

Alle Funktionen über einem Feld von Untermengen von U, die den obigen Axiomen gehorchen, sind Wahrscheinlichkeitsfunktionen. Aus den Axiomen ergeben sich unmittelbar einige Folgerungen:

Eigenschaft 1: P(U\X) = 1 - P(A)

Aus der Additivität der Wahrscheinlichkeiten disjunkter Ereignisse folgt, dass komplementäre Ereignisse komplementäre Wahrscheinlichkeiten haben.

Eigenschaft 2: P($\varnothing$) = 0

Daraus folgt, dass das unmögliche Ereignis, die leere Menge, die Wahrscheinlichkeit null hat. Eine wichtige Erweiterung des Wahrscheinlichkeitskalküls, die aus den Axiomen folgt, ist das folgende Theorem:

Theorem 1: Für alle X,Y $\subseteq$ U gilt P(X $\cup$ Y) = P(X) + P(Y) - P(X $\cap$ Y)

Die Vereinigung von nicht disjunkten Mengen X, Y kann nicht durch einfache Addition der Wahrscheinlichkeiten berechnet werden, da dadurch – intuitiv gesprochen – der Schnitt (X$\cap$Y) doppelt gezählt werden würde. Deshalb muss der Schnitt einmal von der Summe abgezogen werden.

Das Konzept der *bedingten Wahrscheinlichkeit* ist unablässig für die sinnvolle Verwendung des Wahrscheinlichkeitskalküls. Es wird per Definition eingeführt:

Definition 1: Bedingte Wahrscheinlichkeit

$$P(X \mid Y) = \frac{P(X \cap Y)}{P(Y)}$$

P(X|Y) ist die Wahrscheinlichkeit des Eintretens eines Ereignisses X unter der Bedingung, dass ein Ereignis Y vorher eingetreten ist oder eintreten wird. Wenn die Wahrscheinlichkeit P(Y) = 0 ist, dann ist die bedingte Wahrscheinlichkeit nicht definiert. Aus dieser Formel leitet

sich die Verbundwahrscheinlichkeit $P(X \cap Y)$ ab, d. h. die Wahrscheinlichkeit, dass X und Y gemeinsam auftreten. $P(X \cap Y)$ schreibt man häufig auch P(X,Y).

Definition 2: *Verbundwahrscheinlichkeit* oder *Kettenregel (engl. chain rule):*

$$P(X \cap Y) = P(X \mid Y) \cdot P(Y)$$

Sind X und Y unabhängig, dann gilt:

$$P(X \cap Y) = P(X) \cdot P(Y) \Rightarrow P(X \mid Y) = P(X)$$

Die Verbundwahrscheinlichkeit lässt auch auf den multivariaten Fall $P(X_1,X_2,\ldots,X_n)$ ausdehnen. Man erhält dann den verallgemeinerten Ausdruck:

$$\begin{aligned} P(X_1, X_2, ..., X_n) &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2, X_1)...P(X_n \mid X_{n-1}, ..., X_1) \\ &= P(X_1)\frac{P(X_1, X_2)}{P(X_1)}...\frac{P(X_n, ..., X_1)}{P(X_{n-1}, ..., X_1)} \end{aligned}$$

Definition 3: *Unabhängigkeit*

$$X \perp Y \equiv P(X|Y) = P(X)$$

Aus der Definition der Verbundwahrscheinlichkeit ergibt sich die Definition für Unabhängigkeit. Wirft man beispielsweise einen Würfel zweimal, dann sind diese beiden Ereignisse unabhängig. Die Wahrscheinlichkeit, dass im ersten Wurf eine Sechs kommt, ist genauso hoch wie im zweiten Wurf nämlich 1/6. Wird die Wahrscheinlichkeit eines Ereignisses von einem vorher eingetretenen Ereignis beeinflusst, spricht man von *abhängigen* Ereignissen. Die *bedingte Unabhängigkeit* generalisiert dieses Konzept auf den Fall, dass X und Y unabhängig sind beim Eintreten eines dritten Ereignisses Z.

Definition 4: *Bedingte Unabhängigkeit*

$$X \perp Y \mid Z \equiv P(X|Y, Z) = P(X|Z)$$

Bedingte Unabhängigkeit ist der Fall, wenn ein Ereignis Z uns bereits alles über X verrät was auch Y tun würde oder sogar mehr. Kennt ein Agent Z, ist das Lernen von Y überflüssig. Die Ausnutzung dieser Eigenschaft ist ein wesentlicher Bestandteil der probabilistischen Wissensbasis der Agenten.

Aus dieser Wahrscheinlichkeitstheorie leitete Thomas Bayes das Bayes-Theorem ab:

Theorem 2: *Bayes-Theorem*

$$P(h \mid e) = \frac{P(e \mid h)P(h)}{P(e)}$$

Hierbei ist P(h) die A-Priori-Wahrscheinlichkeit für eine Hypothese h und P(e|h) für eine Evidenz e unter der Bedingung, dass die Hypothese h auftritt. Dieser Wert wird durch P(e) normalisiert, so dass sich die bedingten Wahrscheinlichkeiten aller Hypothesen zu 1 summieren. Das Bayes-Theorem ist deshalb so nützlich, weil häufig die Berechnung von P(Wirkung | Ursache) einfacher ist, als die Berechnung von P(Ursache | Wirkung).

Beispiel: Sei die Ursache gleich dem Ereignis, das eine Email eine Spam-Mail ist und die Wirkung, dass bestimmte Wörter in dieser Mail auftauchen. Bei einer vorhandenen Testmenge von Spam- und Nicht-Spam-Mails lässt sich die Wahrscheinlichkeit von P(Wörter = x | Spam = ja) einfach berechnen (durch Zählen zutreffender Mails). Meisten ist man allerdings am Umgekehrten interessiert, der Wahrscheinlichkeit P(Spam = ja | Wörter = x) das eine eingehende Mail eine Spam-Mail ist, wenn bestimmte Wörter vorkommen. Das Theorem von Bayes ermöglicht genau die Berechnung dieser Information.

P(h|e) bezeichnet man auch als *Posterior-Wahrscheinlichkeit*, spricht der Wahrscheinlichkeit, die sich ergibt, wenn man die vorliegenden Evidenzen einbezieht.

Mit dieser Wahrscheinlichkeitstheorie könnte man bereits eine einfache Wissensbasis eines Agenten modellieren, indem man die Wahrscheinlichkeiten aller Kombinationen dem Agenten bekannten Ereignissen in einer vollständigen Wahrscheinlichkeitsverteilung speichert.

Solch eine Verteilung hätte die Form:

$P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$ oder kompakter $P(X_1,\ldots,X_n)$.

Mit Hilfe der Kettenregel lassen sich aus einer Wahrscheinlichkeitsverteilung gemeinsame Wahrscheinlichkeiten berechnen:

$$\begin{aligned} P(X_1, X_2, ..., X_n) &= P(X_1)P(X_2 \mid X_1)P(X_3 \mid X_2, X_1)...P(X_n \mid X_{n-1}, ..., X_1) \\ &= \prod_i P(X_i \mid X_1, ..., X_{i-1}) \end{aligned}$$

Damit kann man die Wahrscheinlichkeit eines bestimmten Zustandes C aus dem Datensatz D berechnen, indem ein beliebiger Wert v_1 der Variablen X_1 und ein beliebiger Wert v_2 der

Variablen X_2 zugewiesen wird usw. Mit solch einer Verteilung wäre ein Agent bereits in der Lage, die Erfolgswahrscheinlichkeit einer bestimmten Aktion bzw. Handlung in der Welt zu prognostizieren, die auf Erfahrungen der Vergangenheit basiert. Leider ist diese Art von gemeinsamer Wahrscheinlichkeitsverteilung schon bei sehr geringer Anzahl an Variablen und Zuständen auf Grund ihrer exponentiell wachsenden Größe praktisch nicht verwendbar.

Eine Lösung dieses Dilemmas bieten Bayes'sche Netzwerke, die D ebenfalls modellieren können und im praktischen Einsatz deutlich langsamer wachsen als vollständige Wahrscheinlichkeitsverteilungen.

Definition *Bayes'sches Netz***:**
Bei einem Bayes'schen Netz (BN) – auch Belief-Netz genannt – handelt es sich um einen gerichteten azyklischen Graphen (*DAG - engl. Directed Acyclic Graph*) G=(V, E), wobei jeder Knoten V_i einer Variable $X_i \in X_1,\ldots,X_n$ entspricht, die eine finite Menge von sich gegenseitig ausschließenden Zuständen $s_1,\ldots,s_j$ haben kann. Eine Menge von gerichteten Kanten $E_i = (V_x,V_y) \in E_1 \ldots E_n$ verbindet die Menge der Knoten V_i miteinander und modelliert die Abhängigkeiten der Variablen untereinander. Dabei darf es im Graphen keinen zyklischen Pfad der Form $A_1 \Rightarrow \ldots \Rightarrow A_n$ mit $A_1 = A_n$ geben. Jeder Knoten ist mit einer Wahrscheinlichkeitsverteilung (*CPT – engl. Conditional Probability Table*) $P(X_i | \prod_{Xi})$ verbunden, wobei $\prod_{Xi}$ die Menge der Elternknoten von X_i in G ist.

Zusammenfassend ist ein BN ein Tupel B = (G, PD), wobei G der Graph und PD eine Menge von bedingten Wahrscheinlichkeitstabellen (CPT) ist. Bayes'sche Netze modellieren eine gemeinsame Wahrscheinlichkeitsverteilung (engl. *Joint Probability Distribution*), die über alle Variablen $X_1,\ldots,X_n$ folgendermaßen berechnet wird:

$$P(X_1,\ldots,X_n) = \prod_{i=1}^{n} P(X_i \mid \Pi_{Xi}).$$

Die prinzipielle Idee Bayes'scher Netze in Bezug auf vollständige Wahrscheinlichkeitsverteilungen ist, nur bedingte Wahrscheinlichkeiten zu kodieren und unabhängige und bedingt unabhängige Beziehungen zu eliminieren. Jede Variable X_i mit gegebenen Eltern Π_{Xi} ist von ihren Nachkommen unabhängig und von allen anderen Knoten – außer den Elternknoten – mindestens bedingt unabhängig. Man unterscheidet drei Arten von bedingten Unabhängigkeiten:

1. kausale Ketten
2. gemeinsame Ursachen und
3. gemeinsame Effekte

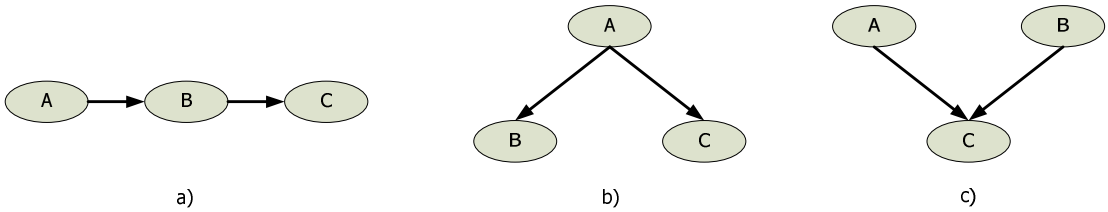


Abbildung 10: Die drei Arten von bedingten Unabhängigkeiten

In einer *kausalen Kette* gilt: $P(C|A \wedge B) = P(C|B)$, d. h. die Wahrscheinlichkeit von C ist bei gegebenen B die gleiche wie bei gegebenen A und B.

Beispiel: Sei A = *Person ist Raucher*, B = *Person hat Krebs* und C = *Person hat Dyspnoea*. Weiß man bereits, dass eine Person Krebs hat, dann ändert die Information *Person ist Raucher* nichts mehr an der Wahrscheinlichkeit, dass die Person unter Dyspnoea leidet.

Sei B eine *gemeinsame Ursache* für A und C, dann gilt: $P(C|A \wedge B) = P(C|B) \equiv A \perp C|B$. Z. B. ist Krebs eine bekannte Ursache für die Symptome Dyspnoea und einen positiven Röntgenstrahlentest. Liegt keine Evidenz für Krebs vor, so erhöht die Information das eine der beiden Symptome vorliegt die Wahrscheinlichkeit für die Krankheit und infolgedessen auch die Wahrscheinlichkeit für das Vorliegen des anderen Symptoms. Ist bereits bekannt, dass eine Person Krebs hat, verändert die Information über einen positiven Röntgenstrahlentest nichts an der Wahrscheinlichkeit für Dyspnoea.

Ein *gemeinsamer Effekt* wird durch ein Netz mit v-Struktur repräsentiert und liegt vor, wenn ein Effekt zwei Ursachen hat. In diesem Fall gilt: $P(A|C \wedge B) \neq P(A|C) \equiv \neg(A \perp C|B)$, d. h., die Eltern sind bedingt abhängig, da A und C im Allgemeinen unabhängig von einander sind. Erst wenn eine Evidenz über B vorliegt, besteht eine Abhängigkeit zwischen A und C. Sei B = *Person hat Krebs*, A = *Person ist Raucher* und C = *Person lebt in verseuchtem Gebiet* und die Information vorhanden das B gilt, dann erhöht das zusätzliche Wissen um die Tatsache, das die Person Nichtraucher ist, die Wahrscheinlichkeit, dass die Person in einem verseuchten Gebiet lebt deutlich.

Diese Konzepte werden durch das Konzept der *d-Separation* auf Mengen von Knoten verallgemeinert. Generell ist es möglich bei einer gegebenen Menge X von Knoten und einer Menge Y zu bestimmen, ob diese bei einer gegebenen Menge von Evidenzknoten E unabhängig sind. Dazu benötigen wir das Konzept der d-Separation (von engl. *direction-dependent separation*) [KorNic04].

Definition *(Ungerichteter) Pfad*: Ein Pfad zwischen zwei Knotenmengen X und Y ist jede Sequenz von Knoten zwischen einem Mitglied von X und einem von Y, so dass jedes

adjazente Paar von Knoten mit einer Kante (unabhängig von der Richtung) verbunden ist und kein Knoten mehrfach in dieser Sequenz auftaucht.

Definition *Blockierter Pfad*: Ein Pfad ist bei gegebener Knotenmenge E blockiert, wenn auf dem Pfad ein Knoten Z existiert für den mindestens eine der folgenden Bedingungen zutrifft:

1. Z ist in E und Z hat eine eingehende und eine ausgehende Kante auf dem Pfad (Kette).
2. Z ist in E und beide Pfadkanten sind ausgehende (gemeinsame Ursache).
3. Weder Z noch irgendein Nachfolger von Z ist in E und beide Pfadkanten sind eingehende Kanten nach Z (gemeinsamer Effekt).

Definition *d-Separation*: Eine Menge von Knoten E *d-separiert* zwei andere Mengen X und Y von Knoten, wenn jeder Pfad von einem Knoten in Menge X zu einem Knoten in Menge Y durch E *blockiert* wird.

Falls X und Y durch E d-separiert werden, dann sind X und Y bei gegebenem E *bedingt unabhängig*. Diese Eigenschaft macht man sich u. a. beim Lernen von BN zu Nutze, um den Speicherbedarf im Vergleich zu einer vollständigen Wahrscheinlichkeitsverteilung stark zu reduzieren.

Im Folgenden sollen nun einige Erweiterungen von Bayes'schen Netzen vorgestellt werden, die für die Modellierung der Wissensbasen von Agenten besonders relevant sind und auch – meist in modifizierter Form – im SimMarket-System umgesetzt worden sind.

3.4.3.2 Dynamische Bayes'sche Netze

Bayes'sche Netze (BN) modellieren Abhängigkeiten zwischen Variablen zu einem bestimmten Zeitpunkt oder innerhalb eines Zeitintervalls. Zeitliche Abhängigkeiten zwischen Variablen werden durch BN nicht expliziert repräsentiert. Eine Möglichkeit die zeitliche Anhängigkeit einer Variablen X_t von einem vergangenen oder zukünftigen Zustand zu modellieren besteht darin, eine oder mehrere neue Variablen einzuführen, die die Variable X in einer anderen Zeit repräsentiert, z. B. X_{t-1} und X_{t+1}. Diese Modellierungstechnik wird durch die Theorie der dynamischen Bayes'schen Netze (DBN) generalisiert. Somit handelt es sich bei DBN immer noch um BN, die allerdings explizit temporale Abhängigkeiten von Variablen modellieren.

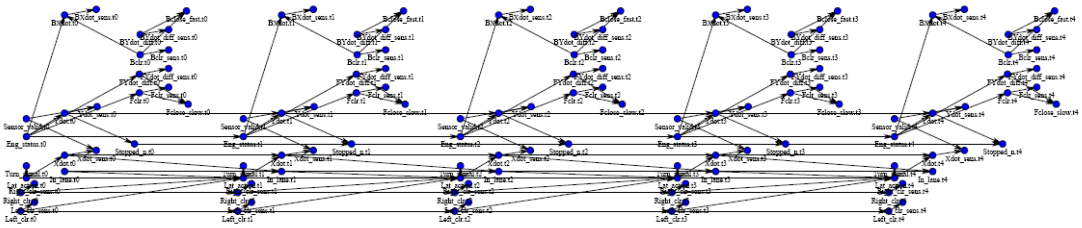


Abbildung 11: Beispiel für ein dynamisches Bayes'sches Netz (BATnetwork)

Angenommen eine Domäne besteht aus *n* Zufallsvariablen $X = \{X_1,\ldots,X_n\}$ wovon jede durch einen Knoten in einem BN repräsentiert wird. Möchte man nun explizit den zeitlichen Verlauf jeder Variable X_i modellieren, muss für jeden Zeitschritt *t* ein Knoten X_i^t eingeführt werden. Ein DBN mit drei Zeitscheiben hätte somit mindestens die folgenden Knoten:

- Aktueller Zeitpunkt: $\{X_1^t, X_2^t, ..., X_n^t\}$
- Vorheriger Zeitpunkt: $\{X_1^{t-1}, X_2^{t-1}, ..., X_n^{t-1}\}$
- Zukünftiger Zeitpunkt: $\{X_1^{t+1}, X_2^{t+1}, ..., X_n^{t+1}\}$

Ein Schritt in der Zeit wird *time-slice* (Zeitscheibe) genannt. Kanten zwischen Knoten innerhalb einer *time-slice* $X_i^T \rightarrow X_j^T$ bezeichnet man als *intra-slice*. Üblicherweise ist die Struktur eines Netzes $\forall T$ identisch, wobei eine Strukturänderung über die Zeit prinzipiell nicht ausgeschlossen ist. Die Kanten, die zwei *time-slices* miteinander verbinden nennen sich *inter-slice*-Kanten oder temporale Kanten. Dabei sind Kanten zwischen gleichen Knoten $X_i^T \rightarrow X_i^{T+1}$ und unterschiedlichen Knoten $X_i^T \rightarrow X_j^{T+1}$ möglich. In den meisten Fällen hat der aktuelle Wert einer Variablen X_i zum Zeitpunkt t auch einen Einfluss auf den folgenden Zeitpunkt t+1, so dass Kanten der Art $X_i^T \rightarrow X_i^{T+1}$ fast immer benötigt werden. Prinzipiell kann der Wert einer Variablen zu einem Zeitpunkt t jede Variable der nächsten *time-slice* t+1 beeinflussen. Würde man die Abhängigkeiten in der Praxis tatsächlich vollständig modellieren bekäme man schnell Komplexitätsprobleme. Zum Glück ist das in der Realität nicht nötig, da vorhandenes Domänenwissen für eine Reduzierung der Kantenanzahl genutzt werden kann. Die spezielle Struktur von DBN kann darüber hinaus bei der Implementierung von Inferenzverfahren für Optimierungen genutzt werden. Da DBN prinzipiell immer noch BN sind, kann jeder Inferenzalgorithmus für BN auch für DBN genutzt werden. Trotz spezieller Optimierungen in den Inferenzverfahren haben DBN häufig mit der Komplexität der Modelle zu kämpfen, da es sich bei jeder *time-slice* um eine (leicht modifizierte) Kopie des Ursprungsnetzes handelt.

In SimMarket verfolgen wir deshalb die Idee, zeitliche Abhängigkeiten möglichst nicht als naives DBN zu modellieren (was aber dennoch möglich ist), sondern indirekt in einzelne Knoten zu kodieren, z. B. Delta-Knoten (siehe dazu Kapitel 4.5.2). Jede Menge von Knoten eines Netzes, also auch temporale Ketten der Form $X_i^{t-m} \rightarrow ... \rightarrow X_i^t \rightarrow X_i^{t+o}$ lassen sich immer in einem einzelnen Knoten durch entsprechende Zustände zusammenfassen. Die entsprechende Kodierung kann bereits bei der Erzeugung der Trainingsmenge durchgeführt werden. Dies vereinfacht die Modellierung und verringert die Komplexität des gesamten Modells, da man sich im Vorfeld mehr Gedanken über die temporalen Abhängigkeiten macht und nicht versucht alle Abhängigkeiten „automatisch" per Lernverfahren zu ermitteln. Außerdem werden keine Kopien von allen Knoten für jede *time-slice* gemacht, die unter Umständen unnötig sind, da sie unabhängig von den temporalen Knoten sind. Ein weiterer Vorteil dieses Vorgehens ist, dass man dadurch auch Abhängigkeiten zu weiter zurück oder weiter in der Zukunft liegenden time-slices als nur die benachbarten modellieren kann. Dies wird bei DBN, schon auf Grund der Komplexität, im Allgemeinen ausgeschlossen. DBN eignen sich besonders bei der theoretischen Betrachtung von BN und bei Vergleichen mit verwandten Modellen, die ebenfalls temporale Abhängigkeiten modellieren können, z. B. Kalmanfilter und Hidden-Markov-Modelle (siehe Kapitel 3.5).

3.4.3.3 Entscheidungsnetze

In diesem Abschnitt sollen die BN erweitert werden, um die Entscheidungsfindung besser zu unterstützen. Dazu werden den BN zwei neue Knotentypen – *Entscheidungsknoten* (engl. *decision nodes*) und *Nützlichkeitsknoten* (engl. *utility nodes*) – und zwei neuen Kantentypen – *Informationskanten* (engl. *information arcs*) und *Präzedenzkanten* (engl. *precedence* oder *no-forgetting arcs*) – hinzugefügt und damit die Netze zu so genannten *Entscheidungsnetzen (EN) / Decision Networks* – auch als *Influence Diagrams* bekannt – ausgebaut [KorNic04]. Die sich ergebende Struktur kann als eine Kombination aus *Entscheidungsbäumen* und Bayes'schen Netzen angesehen werden.

Die „normalen" Knoten eines EN werden Zufallsknoten (engl. *chance nodes*) genannt und entsprechen den Knoten eines Bayes'schen Netzes. Die Eltern dieser Knoten können Entscheidungsknoten oder Zufallsknoten sein. In den meisten Implementierungen wird dieser Knotentyp als Oval dargestellt. Entscheidungsknoten werden in der Regel als Rechtecke, Nützlichkeitsknoten als Rauten dargestellt. Die Werte eines Entscheidungsknoten entsprechen den Handlungsalternativen zwischen denen ein Entscheider abwägen muss. Ein Entscheidungsknoten kann einen Zufallsknoten als speziellen Elternknoten haben, um auszudrücken, wann die Evidenz des Elternknotens bei der Entscheidungsfindung verfügbar sein muss. Die Kanten zwischen diesen Knoten sind die so genannten Informationskanten. Mit einer Informationskante ist eine Entscheidungstabelle verbunden, die angibt, unter

welchen Bedingungen des Elternknotens welche Entscheidung optimal ist. Diese Tabelle – auch *Policy* genannt – kann recht einfach berechnet werden [KorNic04].

Wenn ein Entscheidungsnetzwerk nur eine Entscheidung modelliert, existiert auch nur ein Entscheidungsknoten, soll dagegen eine Sequenz von Entscheidungen gefunden werden, dann kann ein Entscheidungsknoten andere Entscheidungsknoten als Elternknoten in der Reihenfolge der zu treffenden Entscheidungen haben. Entscheidungsknoten werden durch Präzedenzkanten miteinander verbunden. Nützlichkeitsknoten – auch *Wertknoten* (engl. *value nodes*) genannt – enthalten eine Funktion, die die Nützlichkeit eines Zustandes aus der Sicht des Agenten beschreibt. Die Elternknoten sind Variablen, die die Situation des Agenten beschreiben und einen direkten Einfluss auf den Nutzen haben – das beinhalten auch Entscheidungsknoten. Jeder Nützlichkeitsknoten ist mit einer Nutzentabelle verknüpft, die für jede mögliche Instanziierung der Elternknoten inklusive möglicher gewählter Aktionen den Nutzen beschreibt. Existieren im Netz mehrere Nützlichkeitsknoten, dann ist der Gesamtnutzen definiert als die Summe der Werte der einzelnen Nützlichkeitsknoten.

Die in SimMarket verwendeten Netze wurden bisher nicht mit Entscheidungs- und Nützlichkeitsknoten ausgestattet, da die Funktionalität von Entscheidungsnetzen auch mit normalen Knoten und nachgelagerten Funktionen, die die aktuelle Situation bewerten, erreicht werden konnte. Dennoch ist ein Entscheidungsnetz für viele Modellierungsprobleme ein elegantes Konzept, das im Projekt weiter verfolgt werden sollte.

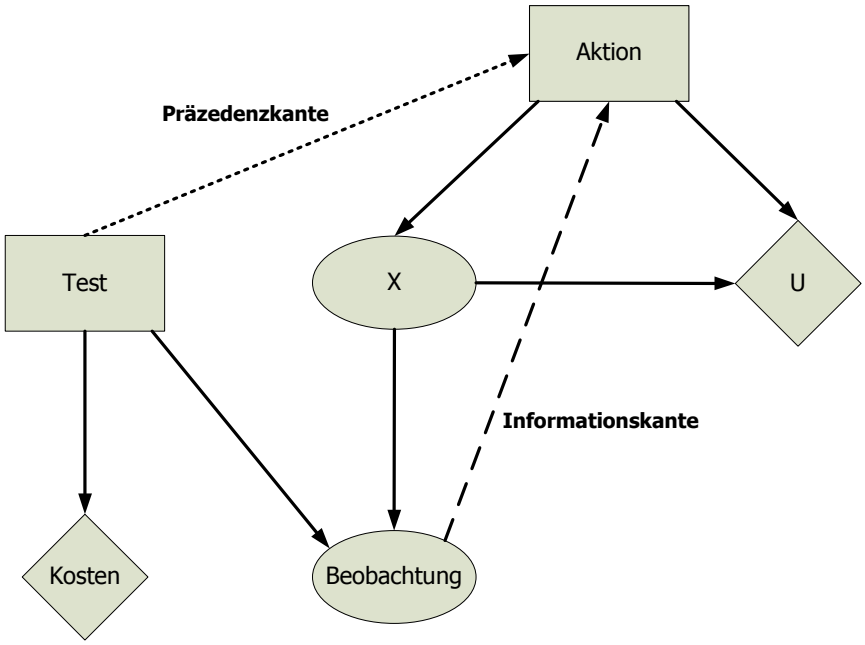


Abbildung 12: Ein Entscheidungsnetz

3.4.3.4 Objektorientierte Bayes'sche Netze

Ein *objektorientiertes Bayes'sches Netz (OBN)* ist ein Entscheidungs- oder normales Bayes'sches Netz, das zusätzlich zu den bereits bekannten Knoten so genannte *Instanzknoten* kennt. Ein Instanzknoten repräsentiert eine Instanz von einem anderen Netzwerk, dass in ein bestehendes Netz als Subnetz integriert werden kann. Im Kontext der objektorientierten Programmierung würde man einen Instanzknoten als *Klasse* bezeichnen. Wie diese, kann eine Instanz wiederum andere Instanzen beinhalten, so dass man mit Hilfe von OBN zu einer hierarchischen Beschreibung von Domänen kommen kann. Die Verwendung von OBN erleichtert einerseits die Modellierung von komplexen Problemstellungen und erhöht anderseits die Wiederverwendbarkeit von Teilkomponenten einmal erstellter Netze. In vollständigen OBN-Modellen kommen zu den Instanzknoten noch die Konzepte *Unterklassen* und *Vererbung* (von Apriori-Wahrscheinlichkeiten und CPTs) hinzu, wie dies aus objektorientierten Programmiersprachen bekannt ist.

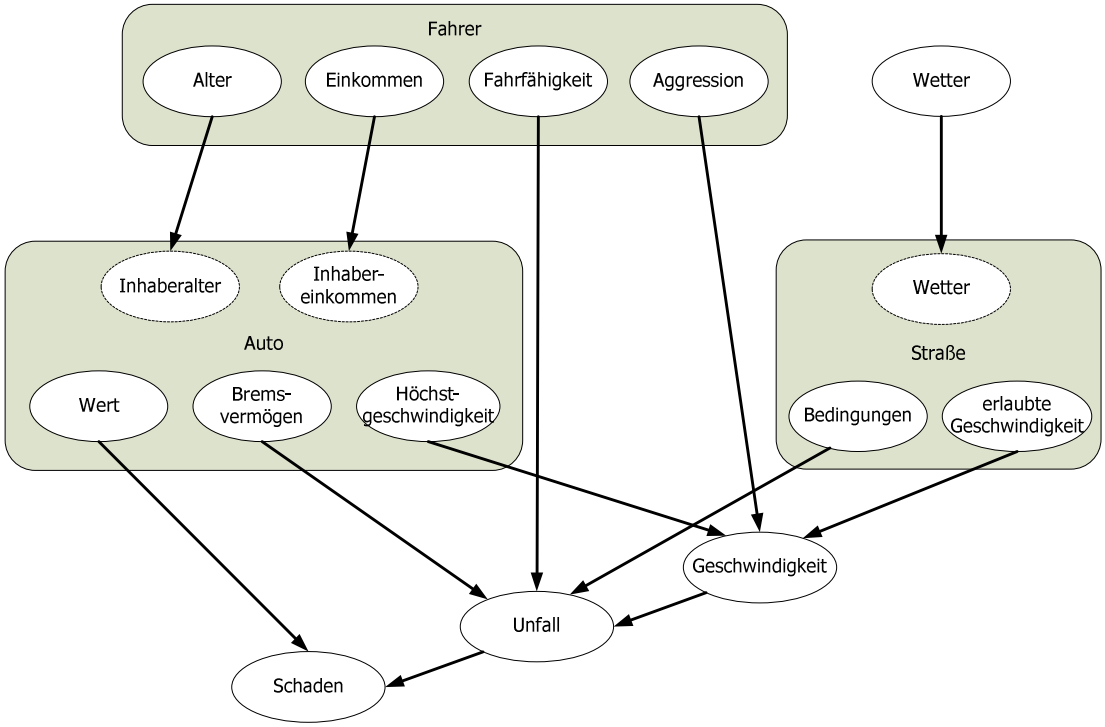


Abbildung 13: Ein objektorientiertes Bayes'sches Netz

In SimMarket haben wir ein ähnliches Konzept von hierarchischen Netzen integriert. Der große Unterschied besteht darin, dass in SimMarket ein Netz immer Bestandteil eines Agenten ist. Die hierarchische Modellierung findet auf der Agentenebene durch das Zusammenführen von Agenten zu Gruppen oder Holonen statt. Deshalb passt die

Bezeichnung *agentenbasierte Bayes'sche Netze (ABN)* besser auf die von uns entwickelten Modellierungskonzepte. Im Gegensatz zu OBN, bei denen eine Instanz nur eine Teilmenge von Knoten mit ihren Kanten ist, besitzen ABN spezielle Interfaceknoten, die die Verknüpfungsmöglichkeiten der Teilnetze mit anderen Netzen definieren. Da in SimMarket nur Agenten verschmolzen werden und keine einzelnen Netze können zudem jederzeit neue Relationen aus den Wissensbasen der Agenten abgeleitet und in die Metanetze integriert werden. Dadurch entsteht emergentes Wissen, das einen Holonen letztlich mächtiger werden lässt als die Summe der Einzelagenten.

3.4.4 Generierung der Agenten

Eine Herausforderung ist das Generieren der Agenten in der konkreten Anwendungsdomäne. Die Leistungsfähigkeit eines Agenten hängt ganz wesentlich vom Umfang und der Qualität seiner Wissensbasis ab, die während der Generierung mit Wissen aus einer gegebenen Datenbasis gefüllt werden muss. Dabei ist die Qualität der Datengrundlage entscheidend, da der beste Algorithmus aus einer schlechten Datenbasis kein zufriedenstellendes Ergebnis erzeugen kann. Die hier präsentierten Verfahren sind speziell für die Anwendung auf sehr große Datenmenge hin optimiert worden und bieten damit eine ideale Basis, um Agenten auf alle Art von transaktionalen Daten großer Datenbanken oder Data Warehouses zu erzeugen.

Ein Agent wird mit den folgenden Schritten generiert:

1. Problem identifizieren und spezifizieren.
2. Anzahl und Typen von Agenten spezifizieren, die für das Lösen des Problems nötig sind.
3. Abbildung von Aufgaben auf einzelne Agenten.
4. Festlegung der Konfiguration eines jeden Agenten und Zuweisung einer initialen Datenbasis, die zum Lösen der zugewiesenen Aufgaben benötigt wird.
5. Extraktion der Wissensbasen der Agenten aus initialen Datenbasen und Initialisierung der Agenten.
6. Start und Überwachung der Agenten, evtl. Zuweisung von neuen oder modifizierten Aufgaben oder Einleitung von Adaptionsverfahren, falls neue ihn betreffende Daten in einer Datenbasis vorliegen, die der Agent nicht selbstständig überwachen kann.

In diesem Kapitel soll vor allem der Punkt 5 behandelt werden. Die anderen Punkte werden am konkreten Beispiel der Artikelsimulation in Kapitel 4 aufgezeigt.

Das Problem des Erlernens eines Bayes'schen Netzes kann folgendermaßen definiert werden: Bei gegebener Trainingsmenge D als Beispiel einer Instanz der Welt W soll ein Netz

B = (G, PD) gefunden werden, das D am besten entspricht. D wird einer relationalen Datenbank entnommen und besitzt N Spalten. Jede Zeile entspricht einer gemeinsamen Beobachtung von N Variablen. Diese Variablen können beispielsweise N Sensoren eines Agenten entsprechen. Je nach Grad der Vorverarbeitung der Sensorsignale kann eine solche Variable einem einfachen Messwert (z. B. Abstand zum nächsten Objekt) oder einem abstrakteren Konzept (z. B. spezifische Handlung eines anderen Agenten) entsprechen. Die Anzahl der Zeilen von D ist die Größe der Trainingsmenge. Um ein Netzwerk aus einer solchen Trainingsmenge zu extrahieren, benötigt man drei Dinge: erstens eine Bewertungsfunktion/Metrik, die die Qualität eines Netzes in Bezug auf den gegebenen Datensatz D misst, zweitens einen heuristischen Suchalgorithmus, der den Raum der möglichen Netze (mit allen möglichen Topologien) durchläuft und idealerweise die beste globale Lösung liefert (also die Struktur eines Netzes lernt) und drittens eine Methode zum Erlernen der bedingten Wahrscheinlichkeitstabellen (CPT). Für jeden dieser drei Punkte gibt es unterschiedliche Ansätze, die gewisse Vor- und Nachteile haben.

Die detaillierte Vorgehensweise beim Lernen eines Bayes'schen Netzes aus einer gegebenen Datenbasis D ist die Folgende:

1. *Aufbereitung der Daten*, so dass die Zeilen des Datensatzes jeweils einem Lernfall und die Spalten den Variablen (Knoten) des Netzes entsprechen.
2. *Diskretisierung* der kontinuierlichen Daten der Variablen in eine endliche Menge von Zuständen (siehe hierzu [Schwaiger06]).
3. *Lernen der Struktur* des Netzes. Idealerweise repräsentiert die resultierende Struktur die realen kausalen Zusammenhänge. Dies muss nicht zwangläufig der Fall sein, da BN zwar eine gemeinsame Wahrscheinlichkeitsverteilung repräsentieren, nicht aber automatisch Kausalitäten, die den menschlichen Vorstellungen von der Domäne entsprechen. Dem Lernen der Struktur geht meistens eine Art Korrelationsanalyse voraus, die den Suchraum vorab einschränkt. Für das Strukturlernen wird eine heuristische Suche und eine Metrik zum Bewerten der Netzqualität benötigt.
4. *Lernen der bedingten Wahrscheinlichkeiten* (CPTs).
5. *Evaluierung* des resultierenden Netzes (siehe Kapitel 5) anhand von verschiedenen Gütekriterien mit eventueller Wiederholung der Schritte 2-5.

Das Erlernen Bayes'scher Netze ist sogar bei vollständiger Datengrundlage ein NP-hartes Problem ([ChiGeiHec94], [Chickering96]), so dass neben der Netzqualität, die Effizienz das wichtigste Gütekriterium für ein Lernverfahren ist.

Für das Lernen der bedingten Wahrscheinlichkeiten gibt es einerseits Methoden, die eine vollständige Trainingsmenge D voraussetzen wie z. B. den *CI-Algorithmus* von Verma und Pearl [VerPea90] und den *PC-Algorithmus* von Sprites [SprGlySch93], und anderseits

Methoden, die auch aus unvollständigen Datensätzen bedingte Wahrscheinlichkeiten lernen können, z. B. der *EM-Algorithmus* von Dempster [DemLaiRub77] und der *Gibbs-Sampler* von Geman und Geman [GemGem84]. Lernverfahren für unvollständige Daten sind speziell für praxisnahe Anwendungen von großer Bedeutung, da es oft vorkommt, dass einzelne Messwerte in der Datenbasis nicht erfasst sind oder als fehlerhaft identifiziert worden sind.

Für die heuristische Suche einer geeigneten Netztopologie lassen sich die bekannten Suchverfahren der KI verwenden bzw. entsprechend anpassen z. B. *genetische Algorithmen, Greedy Search, Hill-Climbing, Simulated Annealing, Tabu Search* oder *Metropolis Search* [RusNor03]. Das letzte Verfahren liefert als Besonderheit nicht nur die Netzstruktur, sondern ein vollständiges Netz mit bedingten Wahrscheinlichkeiten. Es ist ein approximatives Verfahren, dass auf der *Markov-Chain-Monte-Carlo-Suche (MCMC)* basiert [KorNic04]. Ein weiteres Verfahren, das sowohl die Struktursuche, als auch das Lernen der Wahrscheinlichkeiten kombiniert ist, der strukturelle EM-Algorithmus (*SEM-Algorithmus*) von Friedman [Friedman98]. Für SimMarket wurde ein eigenes Verfahren entwickelt, dass in [Schwaiger06] beschreiben wird.

Als Letztes gilt es, eine Metrik auszuwählen, die die Qualität eines gelernten Netzes oder auch einer einzelnen hinzugefügten Kante bewerten kann. Der erste Versuch in diese Richtung stammt von Cooper und Herskovit [CooHer91] mit ihrem *K2* Lernverfahren und der dazu gehörigen Metrik. Dieses Verfahren könnte man als *brute force* bezeichnen und ist deshalb in der Praxis nicht ohne weitere Verbesserungen anwendbar. In SimMarket verwenden wir zur Qualitätsbewertung der Netze eine Metrik aus der Gruppe der MDL-Bewertungsfunktionen (*MDL* von „*Minimum Description Length*" bzw. minimale Beschreibungslänge) verwendet. Die *MDL-Metriken* gehen auf Jorma Rissanen [Rissanen78] zurück und sind von der *Minimum Message Length* (*MML*) von Wallace und Boultan [WalBou68] aus dem Jahre 1968 abgeleitet. Die MDL-Bewertungsfunktion hat ihre Wurzeln im *Universal-Coding*, das zum Ziel hat, ein Modell M vom Datensatz D zu finden, das verwendet werden kann, um eine kompaktere Version von D zu produzieren. Die *Gesamtbeschreibungslänge* von M ist die Größe der komprimierten Version von D, die durch die MDL-Bewertungsfunktion minimiert wird. Im Bereich der Bayes'schen Netze balanciert die MDL-Funktion die Komplexität eines Netzes mit der Akkuratheit der Darstellung von D in einem Netz B aus. Es existieren verschiedene MDL-Metriken in der Literatur von denen die bekanntesten von Lam/Bacchus [LamBac93] und von Joe Suzuki [Suzuki96] stammen. Die Lam/Bacchus-Variante ist äquivalent [Friedman97] zu Schwarz' *Bayesian Information Criterion* (*BIC*) aus dem Jahr 1978 [Schwarz78]. In SimMarket verwenden wir eine modifizierte Variante [Friedman97] der Lam/Bacchus-MDL-Metrik ([LamBac93], [FrieGol02], [HecGeiChi95]), in der eine ähnliche Idee steckt, wie in Suzukis MDL-Metrik, nämlich die Größe des Datensatzes D in der Funktion mitzuberücksichtigen. Dies impliziert, dass die Codelänge im Gegensatz zur Lam/Bacchus-Variante effizienter repräsentiert wird.

$$\mathrm{MDL}(G) = \sum_{V_i} \left(\underbrace{\sum_{X_i, \Pi_{X_i}} \left(P(X_i \mid \Pi_{X_i}) \log_2 \frac{P(X_i \mid \Pi_{X_i})}{P(X_i) P(\Pi_{X_i})} \right)}_{F_1} - \underbrace{\mid B_{V_i} \mid \frac{\log_2 k}{2}}_{F_2} \right)$$

$|BVi|$ ist die Summe der Anzahl der Wahrscheinlichkeiten der CPT der V_i's und k ist die Anzahl der Datensätze in D. Der erste Teil F_1 der Formel maximiert die Akkuratheit des Netzes, während der zweite Teil F_2 ein „Strafwert" ist, der übergroße und dazu „übertrainierte" (engl. *overfitted*) Netze verhindert. Der große Vorteil der MDL-Metrik ist, dass sich der Gesamtwert des Graphen aus der Summe der MDL-Scores der einzelnen Kanten berechnen lässt. Dadurch lässt sich sehr einfach überprüfen, ob das Verändern einer einzelnen Kante die Netzqualität erhöht. Dies vereinfacht die Entwicklung eines heuristischen Suchalgorithmus zum Generieren von Netzstrukturen ganz erheblich.

Das aus dem Lernverfahren resultierende BN modelliert die Erfahrung und das Wissen eines probabilistischen Agenten betreffend seiner Sensordaten während des Zeitintervalls [t,1]. Zum Lernen von Bayes'schen Netzen siehe auch das Tutorial von David Heckerman [Heckerman96].

3.4.5 Inferenzen in probabilistischen Agenten

Nun interessiert uns, wie ein Agent seine Weltsicht (belief) revidiert, wenn neue Informationen (z. B. neue Sensordaten) vorliegen. Das Aktualisieren von BN beim Vorhandensein von neuen Informationen bezeichnet man als *Propagieren* (von Evidenzen), *Inferenz* ziehen, *Schlussfolgern* oder *Meinungsrevision* (engl. *belief updating*). Aus wahrscheinlichkeitstheoretischer Sicht entspricht eine Propagierung in einem BN der Berechnung der A-Posteriori-Wahrscheinlichkeitsverteilung für eine Menge von Anfrageknoten bei gegebenen Evidenzen (Beobachtungen). Das Inferieren von Evidenzen kann man sich bildlich als Fluss der neuen Information durch das gesamte Netzwerk vorstellen, bei dem die bedingten Wahrscheinlichkeitstabellen (CPT) aller von den neuen Informationen betroffenen Knoten aktualisiert werden. Der Informationsfluss ist dabei keineswegs durch die Richtung der Kanten beschränkt, sondern verläuft grundsätzlich bidirektional.

3.4.5.1 Inferenzarten

Ein Bayes'sches Netz ist die komprimierte Form einer gemeinsamen Wahrscheinlichkeitsverteilung einer Menge von Variablen. Es ist nun möglich für eine

beliebige Untermenge von Variablen und gegeben Evidenzen die A-Posteriori-Wahrscheinlichkeiten zu berechnen. Auf Grund der semantischen Unterschiede der Knoten, die sich durch die Struktur eines Netzes ergeben, unterscheidet man vier verschiedene Arten von Inferenzen in Bayes'schen Netzwerken:

1. *Diagnostische Interferenz*: Ist eine Evidenz, die von der Wirkung zur Ursache propagiert wird. Der Informationsfluss verläuft also entgegengesetzt zur Kantenrichtung im Netz. Beispiel: Ein Arzt stellt fest, dass ein Patient Dyspnoea hat und möchte wissen, wie wahrscheinlich es ist, dass der Patient Krebs hat. Durch diese Evidenz (und auf Grund der Struktur des Netzes) steigt auch die Wahrscheinlichkeit, dass der Patient ein Raucher ist (siehe Abbildung 14).
2. *Kausale oder prädikative Inferenz* heißt, dass Beobachtungen über Ursachen vorliegen und dadurch Wahrscheinlichkeiten von Effekten verändert werden. Beispiel: Erzählt ein Patient seinem Arzt, dass er Raucher ist und erhöht somit die Wahrscheinlichkeit, dass er Krebs hat (oder bekommen wird) noch bevor irgendein Symptom auftritt. Zusätzlich erhöht diese Information auch die Wahrscheinlichkeit, dass bestimmt Symptome auftreten werden, wie z. B. Kurzatmigkeit.
3. *Zwischenkausale Interferenz* ist das Schlussfolgern über verschränkte Ursachen eines gemeinsamen Effektes. Angenommen es gibt genau zwei Ursachen für einen bestimmten Effekt, z. B. „Raucher" und „Umweltverschmutzung" als gemeinsame Ursache für „Krebs". Obwohl Raucher und Umweltverschmutzung voneinander unabhängig sind, wirkt sich die Information Patient hat Krebs auf die Wahrscheinlichkeiten beider Knoten aus. Kommt nun noch die Information hinzu, dass der Patient Raucher ist, hat dies dennoch einen Effekt auf den Knoten Umweltverschmutzung. Die Wahrscheinlichkeit für diesen Knoten wird sinken, da man nun weiß, dass der Patient raucht. Mit anderen Worten, eine alternative Ursache für Krebs wurde durch eine Erklärung ausgeschlossen.
4. *Kombinierte Interferenz:* Da man bei BN völlig frei ist wo und welche Evidenzen man eingibt, ist auch eine beliebige Kombination der ersten drei Inferenzarten möglich. Z. B. liegen eine diagnostische und eine kausale Inferenz vor, wenn ein Patient das Symptom der Kurzatmigkeit aufweist und darüber hinaus bekannt ist, dass er Raucher ist. Die Wahrscheinlichkeit, dass er Krebs hat, ergibt sich aus der kausalen und der diagnostischen Inferenz.

3.4.5.2 Evidenzarten

Neue Informationen lassen sich mittels Evidenzen in die Knoten eines BN eingeben und über das Netz propagieren. Im obigen Netzbeispiel handelt es sich bei dem Knoten „Raucher" um

eine binäre Variable. Es kann natürlich auch Knoten geben, die mehr als zwei Zustände haben, z. B. der Knoten „Preis“ in einem Netz zur Modellierung von Artikelabverkäufen kann so viele Zustände haben, wie es unterschiedliche Preise für diesen Artikel gegeben hat. Eine Evidenz, die den Knoten „Preis“ auf einen bestimmten Preis festsetzt, z. B. „Preis“=1,99 wird *Hartevidenz* oder *spezifische Evidenz* genannt. Bei Hartevidenzen ist der exakte Zustand einer Variable bekannt.

Oft ist jedoch der genaue Zustand einer Variable nicht bekannt. Eine Evidenz bezeichnet man als *negative Evidenz,* wenn die Evidenz keinen exakten Zustand einer Variablen festlegt, sondern nur bestimmte Zustände ausschließt, z. B. wenn die Variable Y den Wert $Y=y_1$ *oder* $Y=y_2$ hat, aber auf keinen Fall die Werte $y_3,\ldots,y_n$ oder die Evidenz besagt, dass Y *nicht* den Wert y_1 sondern irgendeinen anderen hat.

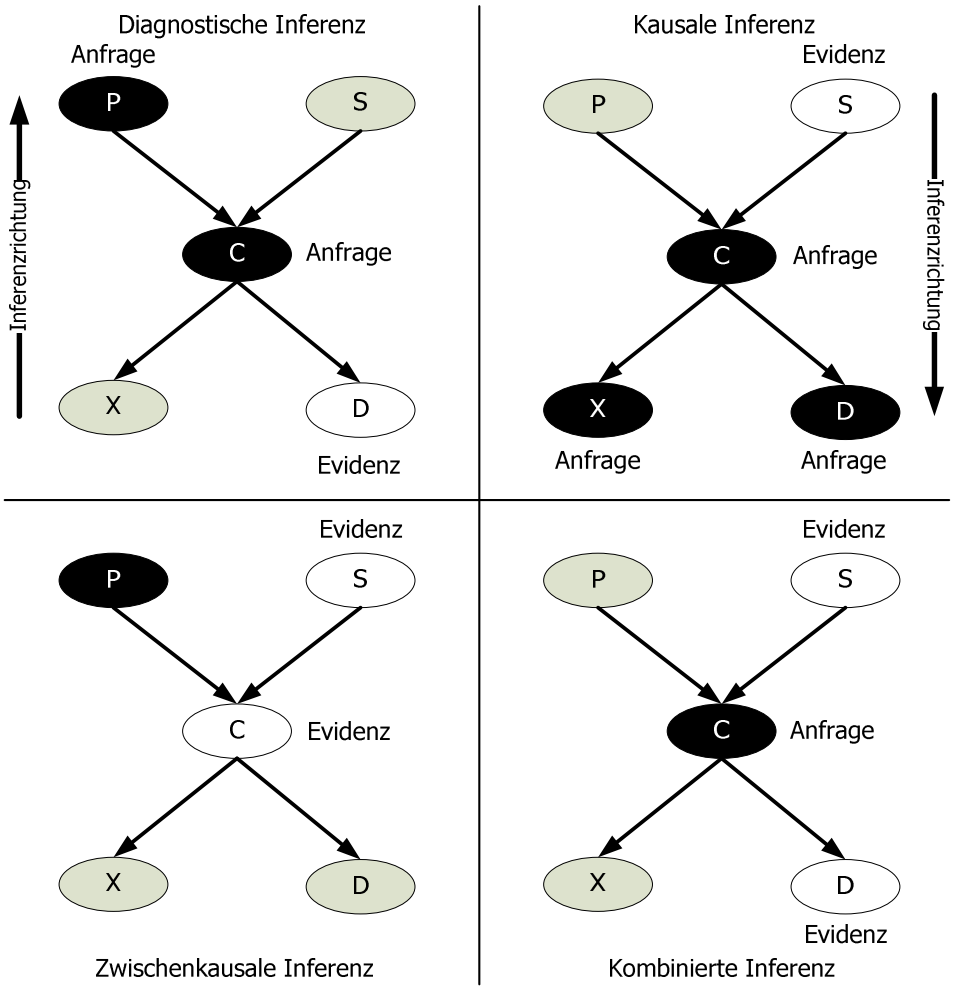


Abbildung 14: Inferenzarten

Des Weiteren kann eine Evidenz auch in Form einer Wahrscheinlichkeitsverteilung vorliegen, wenn die Evidenz selbst wiederum Unsicherheit beinhaltet, z. B., wenn ein Arzt nur zu 80% sicher sein kann, dass der Röntgenstrahlentest zur Erkennung von Krebs positiv ausfällt, wenn tatsächlich Krebs vorliegt. In dem Fall kann er eine Evidenz in Form einer *virtuellen Evidenz*, auch *Likelihood-Evidenz* genannt, der folgenden Form: 80% R-Test = positiv, 20% R-Test = negativ in das Netz eingeben. Diese unsichere Evidenz wird, vereinfacht gesagt, auf die vorliegende Apriori-Verteilung draufgerechnet und entspricht somit einer Gewichtung der Apriori-Verteilung auf der Basis einer neuen aber unsicheren Evidenz.

Die Standardliteratur zum Thema Bayes'sche Netze kennt nur diese drei Evidenzarten. Vomlel, Kim und Valtora [ValKimVom01] führen eine neue Evidenzart, die *Softevidenzen*, ein. Ähnlich wie bei virtuellen Evidenzen liegt als Eingabe eine Wahrscheinlichkeitsverteilung vor, die aber im Gegensatz zur virtuellen Evidenz nicht als Unsicherheit gegenüber den Apriori-Wahrscheinlichkeiten zu verstehen ist, sondern vielmehr als exakte Information, d. h. dass die gegebene Wahrscheinlichkeitsverteilung für eine Variable vollständig ersetzt. Ein Beispiel: Der Arzt aus unserem Beispiel weiß nach der Durchführung des Tests bei einem Patienten, dass der Röntgentest eine Wahrscheinlichkeit von 90% für eine Erkrankung des Patienten ausgegeben hat (und damit eine zehn prozentige Wahrscheinlichkeit für eine Nichterkrankung). Die A-Priori-Wahrscheinlichkeit wird in diesen Fall nicht mit der neuen Verteilung gewichtet, sondern auf die neue Verteilung festgelegt und soll sich auch nach dem Propagieren und dem Setzen von anderen Evidenzen nicht mehr ändern, da sich der Arzt bzgl. der gegebenen Verteilung sicher ist. Das Thema Softevidenzen wird ausführlich in Kapitel 4 behandelt.

3.4.5.3 Inferenzalgorithmen

Um die oben genannten Inferenzen in einen Bayes'schen Netzwerk ziehen zu können, d. h. die Posterior-Wahrscheinlichkeit für eine Menge von Anfrageknoten bei gegebenen Evidenzen zu berechnen, benötigt man geeignete Inferenzalgorithmen, die möglichst mit allen Evidenzarten umgehen können. In den letzten zwanzig Jahren wurden zahlreiche Verfahren entwickelt, die für verschiedene Netzarten geeignet sind und sehr unterschiedliche Performanz besitzen. Grundsätzlich gibt es zwei große Klassen von Inferenzverfahren: *exakte* und *approximative* Algorithmen. Generell muss an dieser Stelle gesagt werden, dass sowohl die Laufzeit der exakten, als auch der approximativen Algorithmen im Allgemeinen exponentiell mit der Komplexität der Netze wächst. Deshalb sind viele existierende Verfahren nie über den experimentellen Status hinausgekommen. Im Folgenden möchte ich die bewährtesten Verfahren inklusive meiner eigenen Verbesserungen vorstellen.

Der prinzipielle Vorgang der Propagierung bei vorliegenden Evidenzen ist für alle Verfahren ähnlich: Nach der Absorption der Evidenzen durch die CPTs der betroffenen Knoten werden eine oder mehrere Nachrichten mit Restwahrscheinlichkeiten generiert, die anschließend an benachbarte Knoten geschickt werden, wo sich der Vorgang wiederholt bis das gesamte Netz aktualisiert ist. Das Verschicken der Nachrichten geschieht dabei auch gegen die Richtung der Kanten im Netz, was dazu führen kann, dass in beliebigen Bayes'schen Netzen Nachrichten in Zyklen kreisen könnten. Der Grund ist, dass BN beliebige gerichtet azyklische Graphen (DAG) – auch *multiply-connected network* genannt – sein können, die bei nicht Beachtung der Kantenrichtung Zyklen enthalten können. Deshalb sind die bekannten Verfahren zum Propagieren nur für *Polytrees* – auch *Wälder* genannt (engl. *forest*) – zulässig. In einem Polytree darf jeder Knoten nur maximal einen Elternknoten, aber im Gegensatz zu einem *Baum* mehrere Wurzelknoten besitzen. Dadurch kann in einem Polytree nur maximal ein Pfad $X \rightarrow \ldots \rightarrow Y$ zwischen X und Y existieren. Selbst ein ungerichteter Polytree kann keine Zyklen enthalten. Deshalb umgehen die exakten Inferenzalgorithmen das Problem von Zyklen, indem der DAG eines beliebigen BN zunächst in einen Polytree konvertiert wird. Die eigentlichen Message-Pathing-Algorithmen basieren alle auf Polytrees.

3.4.5.4 Die Klasse der Junction-Tree-Verfahren

Aus der Klasse der Junction-Tree-Verfahren wurde von mir im Rahmen dieser Arbeit ein vollständiges Inferenzsystem für probabilistische Netze entwickelt, das den Kern des gesamten Simulationssystems in SimMarket bildet.

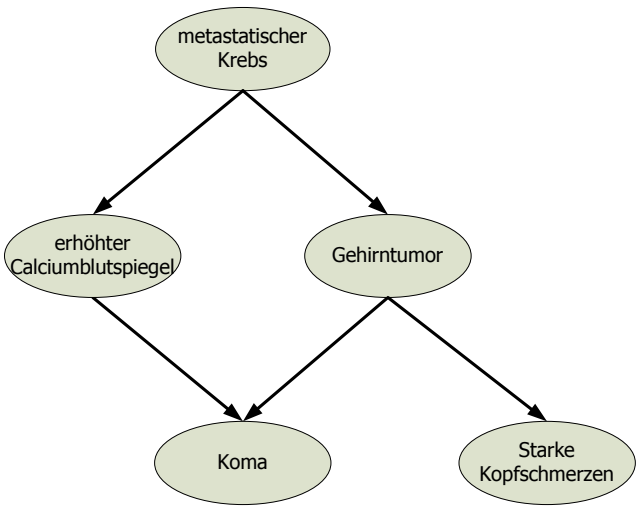


Abbildung 15: Ein Beispiel für ein Bayes'sches Netz mit fünf Variablen

In diesem Inferenzsystem kommen zahlreiche Algorithmen zum Einsatz, die häufig verbesserte Versionen der Originalvarianten darstellen oder sogar völlige Neuentwicklungen sind.

Zur Veranschaulichung der Funktionsweise des Inferenzsystems dient das Beispiel von Spiegelhalter [Spiegelhalt86] und Cooper [Cooper84] (Abbildung 15).

> *Metastatischer Krebs* ist eine mögliche Ursache für einen *Gehirntumor* und kann eine Erklärung für einen *erhöhten Kalziumblutspiegel* sein. Diese beiden Symptome können wiederum der Grund sein, warum ein Patient ins *Koma* gefallen ist. *Starke Kopfschmerzen* können ebenfalls eine Folge eines *Gehirntumors* sein.

In dem zugehörigen Bild sieht man, dass es sich hierbei nicht um einen Polytree, sondern um einen DAG handelt. Um die Propagierung ohne Zyklen zu ermöglichen, wird folgendes Verfahren angewandt: Alle mehrfachen Pfade von einen Knoten X zu einem Knoten Y werden durch Zusammenlegen der Knoten auf den Pfaden zwischen X und Y zu einem einzelnen Knoten beseitigt, so dass nur noch ein Pfad übrig bleibt. Führt man dies für alle Problemfälle durch, erhält man einen Polytree. Alle neuen Knoten, die durch Zusammenlegen von anderen Knoten entstanden sind, erhalten eine neue CPT, die durch Multiplikation der CPT der ursprünglichen Knoten berechnet wird.

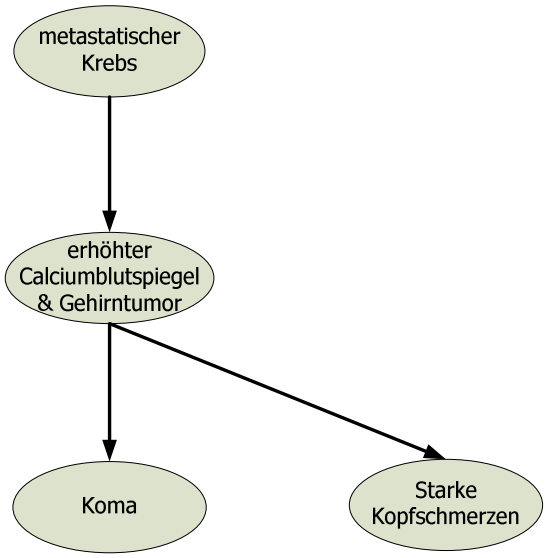


Abbildung 16: Polytree des obigen Beispiels

Der folgende Algorithmus erzeugt aus einem beliebigen BN einen Baum aus so genannten *Cliquen*. In einer Clique ist jeder Knoten mit jedem anderen Knoten durch eine Kante verbunden. Der resultierende Baum wird als *Junction-Tree* (manchmal auch *Join-Tree*)

bezeichnet. Da das *Junction-Tree-Verfahren* sogar einen Baum erzeugt und nicht nur einen Polytree, vereinfacht sich der anschließend anzuwendende Message-Passing-Algorithmus erheblich. Der Junction-Tree-Algorithmus in SimMarket basiert auf einer verbesserten Fassung von Neapolitan [Neapolitan89] und umfasst die folgenden Schritte:

1. Das Bayes'sche Netz wird im ersten Schritt moralisiert. Ein Netz ist moralisiert, wenn zwischen allen Elternknoten $P_1,\ldots,P_n$ eines Knotens X eine Kante existiert, folglich bilden X und $P_1,\ldots,P_n$ nach der Moralisierung eine Clique. Nach der *Moralisierung* werden aus den gerichteten, ungerichtete Kanten. Der resultierende Graph wird *moralisierter Graph* (engl. *moral graph*) genannt.
2. Im zweiten Schritt wird der Graph trianguliert. Ein Graph ist trianguliert, wenn jeder Zyklus der Länge > 3 mindestens eine *Sehne* (engl. *chord*) hat, so dass jeder Zyklus einen Unterzyklus aus nur drei Knoten besitzt. Das Ergebnis ist ein *triangulierter Graph*.
3. Im nächsten Schritt werden im triangulierten Graphen maximale Cliquen gesucht. Maximale Cliquen sind Teilgraphen, die durch jedes Hinzufügen eines weiteren Knotens den Status einer Clique verlieren würden.
4. Die gefundenen maximalen Cliquen werden nun in korrekter Abfolge (siehe unten) zu einem Baum, den Junction-Tree, zusammengesetzt.
5. Die neuen CPTs der Cliquen werden aus den CPTs der zugehörigen Einzelknoten berechnet.

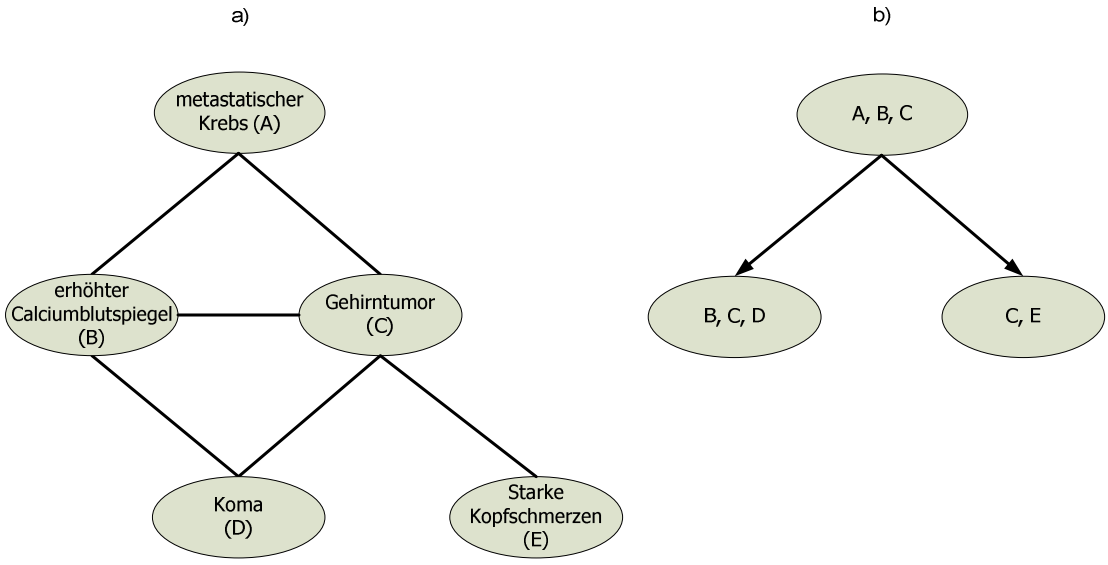


Abbildung 17: a) moralisierter Graph; b) resultierender Junction-Tree

Im obigen Bild sieht man den moralisierten Graph und den erzeugten Junction-Tree. Der moralisierte Graph bedarf keiner Triangulierung, der er bereits trianguliert ist.

Die Moralisierung ist einfach und kann nur auf eine Art und Weise durchgeführt werden. Für die Triangulierung dagegen gibt es zahlreiche Algorithmen. Ein Bayes'sches Netz ist optimal trianguliert, wenn die Größen der Cliquen-CPTs minimal sind. Das Finden einer optimalen Triangulierung ist NP-vollständig [ArnCorPro87] und bedarf deshalb einer guten Heuristik, die es in der Praxis auch meistens gibt ([ShoGei97]). In SimMarket verwende ich den folgenden Triangulierungsalgorithmus ([Stephenson00]):

1. Als Erstes müssen die Knoten des Graphen nummeriert werden. Dazu verwenden wir die *Maximum-Cardinality-Search (MCS)*:
    a. Einem belieben Knoten wird die 1 zugewiesen.
    b. Für jede folgende Nummer wird derjenige unnummerierte Knoten ausgewählt, der die meisten bereits nummerierten Knoten als Nachbarn besitzt. Bei einem Unentschieden wird ein beliebiger Knoten aus der Menge mit den meisten nummerierten Nachbarn gewählt.
2. Führe den folgenden Algorithmus von Knoten n bis Knoten 1 aus:
    a. Ermittle alle Knoten, die eine niedrigere Nummer besitzen als der aktuelle Knoten und adjazent zu diesem sind. Berücksichtige dabei auch während des Algorithmus neu hinzugefügt Kanten.
    b. Verbinde die ermittelten Knoten miteinander.
    c. Wiederhole a. mit dem nächsten Knoten.

Das Finden eines optimalen Junction-Trees ([BecGei96], [JenJen94]) hängt vor allem von der verwendeten Triangulierung, aber auch von dem Algorithmus, der aus den gefundenen maximalen Cliquen einen Junction-Tree erzeugt, ab. Als Nächstes benötigt man ein Verfahren, das in dem triangulierten Graphen alle maximalen Cliquen findet. Dazu verwenden wir die Methode von [Golumbic80]. Zuvor muss allerdings die Nummerierung der Knoten mit dem MCS-Algorithmus wiederholt werden, da während der Triangulierung neue Kanten hinzugefügt werden und dadurch die Nummerierung nicht mehr korrekt ist.

```
Gegeben: Graph G = (V, E)
Gegeben: Eine (durch MCS) sortierte Liste L = v1,…,vn mit absteigenden
Nummern der Knoten V von G
  Initialisiere: H = 1; S[1,…,n] = [0,…,0]; Cliquen = []

  foreach(Knoten v in L)
    if (v hat Nachbarn)
        if (irgendein Nachbar von v steht in L vor v)
            X = erster auftretender Nahbar von v in L
            u = das letzte austretende Element in X
            S[u] = der größere Wert von S[u] und |X| - 1
            if (S[v] < |X|)
                Cliquen = Cliquen + X, v
                H = der größere Wert von H und 1 + |X|
        else
            break
    else
        Cliquen = Cliquen + v
  return Cliquen, H
```

Anschließend müssen die gefundenen Cliquen zu einem Junction-Tree zusammengebaut werden. Wir verwenden dazu die Methode von Pearl [Pearl88]:

1. Die Liste der Cliquen wird nach dem höchst nummerierten Knoten (auf der Basis der Maximum-Cardinality-Suche) in den Cliquen sortiert.
2. Jede Clique wird mit derjenigen Clique verbunden, die in der sortierten Liste vor ihr steht und die meisten gleichen Knoten besitzt.

Als Letztes müssen die CPTs der Cliquen des Junction-Trees aus den originalen CPTs der einzelnen Knoten initialisiert werden. Jede Clique besitzt, wie einzelne Knoten des Graphen auch, eine CPT ψ, die das Kreuzprodukt aller möglichen Werte ihrer Knoten darstellt. Sei nun Clique_i zusammengesetzt aus r_i Knoten $X_1,...,X_{ri}$ wobei jeder Knoten $S_{r_i}^t$ Werte (Zustände) besitzt. Daraus folgt, dass es

$$\prod_{r=1}^{r_i} s_r^t$$

verschiedene Kombinationen von Variablenbelegungen über alle r_i Variablen in Clique_i gibt. Sei ψ_i die CPT ψ der Clique_i und ψ_{ij} die *j*ste mögliche Variablenbelegung der Clique_i. Dann lässt sich eine initiale Variablenbelegung aller ψ_i folgendermaßen berechnen:

foreach (Knoten v in V)
    Weise v einer einzelnen Clique zu, in der v mit seinen Eltern vorkommt
for (i = 1; i <= n; i++) (n = Anzahl an Cliquen)
    for (j = 1; j <= J; j++) (J = Anzahl möglicher Variablenbelegungen der $Clique_i$)
        Initialisiere ψ_{ij} mit dem Wert 1
    for (j=1; j <= J; j++)
        foreach (Knoten v_l zugewiesen der $Clique_i$)
            $\psi_{ij} = \psi_{ij} \cdot P(v_{lj} \mid Eltern(v_l)_j)$

Da eine Variable in mehreren Cliquen vorkommen kann, wird ihre Wahrscheinlichkeitsverteilung beim Initialisieren nur für eine Clique verwendet.

Auf den resultierenden Junction-Tree kann nun der weiter unten präsentierte Message-Passing-Algorithmus zum Propagieren von Evidenzen angewendet werden.

3.4.5.5 Konditionierungsmethode

Die zweite Methode, um ein Multiply-Connected-Netzwerk in einen Polytree umzuwandeln, die ich hier kurz skizzieren möchte, ist in der BN-Literatur recht unbekannt und wird in keiner mir bekannten Software angeboten. Dennoch ist diese Methode sehr interessant und soll hier erwähnt werden.

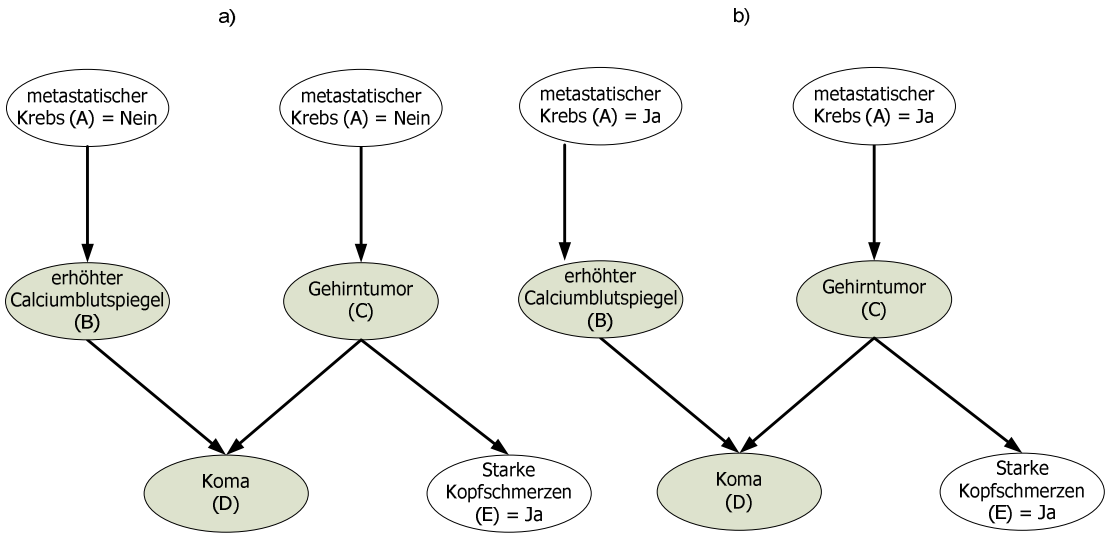


Abbildung 18: Das BN durch Konditionierung in zwei Polytrees zerlegt

Die Idee der Konditionierungsmethode (Schlussfolgern durch Annahmen) basiert auf der Tatsache, dass sich die Konnektivität eines BN durch das Instanziieren von ausgewählten

Gruppen von Variablen ändern lässt. Beispielsweise bewirkt das Instanziieren der Variable A mit einem beliebigen Wert (z. B. A = 0) das der Pfad B-A-C blockiert wird und ein Polytree entsteht. Auf den Rest des Netzwerks lässt sich nun der Message-Passing-Algorithmus für Polytrees anwenden. Um eine Evidenz e im obigen Beispielnetz zu propagieren, setzt man die Variable auf A = 0 und propagiert unter dieser Annahme. Anschließend wiederholt man dies mit A = 1 und mittelt die beiden Ergebnisse, wobei diese noch mit den Posterior-Wahrscheinlichkeiten P(A=1|E=1) und P(A=0|E=1) gewichtet werden müssen. Das Propagieren lässt sich auch in einem Schritt durchführen, indem man die Knoten beim Message-Passing (siehe unten) zwei (oder mehr) Nachrichten gleichzeitig bearbeiten lässt. Für jeden möglichen Zustand einer zu konditionierenden Variablen wird entweder eine eigene Nachricht benötigt oder die Größe der Nachrichten wächst mit der Anzahl der zu konditionierenden Knoten und deren Zustände entsprechend an. Im schlechtesten Fall kann die Nachrichtengröße somit exponentiell wachsen. Praktische Erfahrungen haben allerdings gezeigt, dass Zyklen recht selten sind. Zudem wird die Konditionierung hinfällig, wenn auf einen Knoten des Zyklus' eine Evidenz gesetzt wird. Bleiben dennoch Zyklen übrig, so sollte man den Knoten konditionieren, der die wenigsten Zustände hat oder, noch besser, einen Knoten konditionieren, der mehr als einen Zyklus beseitigt, da er in der Schnittmenge mehrerer Zyklen liegt. Die konkrete Vorgehensweise ist die Folgende:

1. Alle Zyklen im ungerichteten Graphen ermitteln, die nach dem Setzen von Evidenzen übrig geblieben sind.
2. Suchen einer minimalen Menge von Knoten, die nach einer Konditionierung alle Zyklen beseitigen würden. Eine Knotenmenge ist minimal, wenn die Größe bzw. die Anzahl der Nachrichten minimal ist. Dazu muss der folgende Ausdruck minimiert werden:

$$\sum_{i=1}^{n} \#s_{v_i} \cdot Z_{v_i} ,$$

wobei Knoten $v \in M$ ist, der Menge aller an den Zyklen beteiligten Knoten, $\#s_{v_i}$ die Anzahl der Zustände von v_i ist und Z_{v_i} die Anzahl an Zyklen ist, die durch das Konditionieren von v_i beseitigt werden würden.

Diese Art von Schlussfolgern durch Annahmen ist dem Menschen nicht unbekannt. Häufig gelingt es uns nicht die Wahrscheinlichkeit eines Ereignisses oder des Erfolgs einer Handlung abzuschätzen. Eine unbewusste Vorgehensweise ist, eine hypothetische Annahme zu machen, um dadurch die Schätzung zu vereinfachen. Anschließend negiert man diese Annahme und überprüft, ob das Ergebnis stark abweichend ist.

3.4.5.6 Message-Passing in Polytrees

Der *Message-Passing-Algorithmus* (*MPA*) ist dafür zuständig neue Informationen in Form von Evidenzen über das gesamte Netz zu propagieren, um die Wahrscheinlichkeiten aller betroffenen Knoten zu aktualisieren. Die bekannten Algorithmen funktionieren ohne weitere Maßnahmen nur in Polytrees, da sonst zu propagierende Wahrscheinlichkeiten in Zyklen kreisen könnten. Der bekannteste MPA stammt von Pearl [Pearl88] und wird in den meisten Software-Tools für BN verwendet. Peot und Shachter [PeoSha91] und Zweig [Zweig98] bieten in ihren Arbeiten verbesserte Varianten dieses Algorithmus an. Auch wir verwenden eine modifizierte Variante von Pearls originalem Algorithmus, die für die Verwendung in Junction-Trees optimiert wurde.

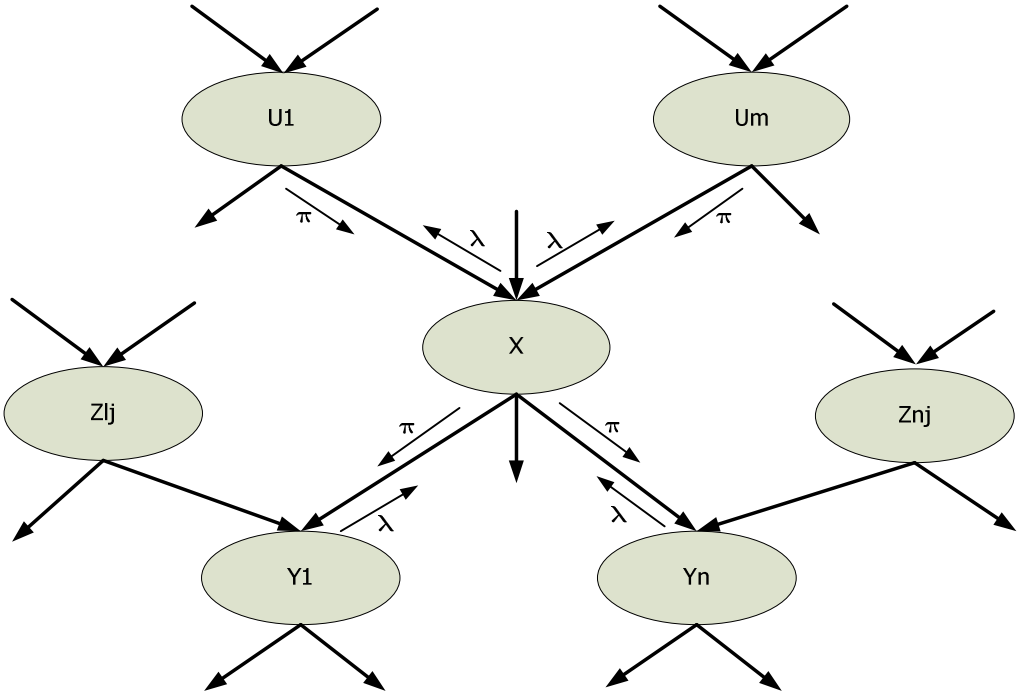


Abbildung 19: Message-Passing in Polytrees

Die Vorgehensweise des Algorithmus ist die Folgende:

Gegen sei eine Menge E_v von Knoten v für die eine Evidenz auf einen Zustand s_v gesetzt wurde. Nun werden zunächst alle Cliquen aktualisiert, die einen Knoten aus der Menge E_v besitzen. Aus den CPTs diese Cliquen werden alle Werte X entfernt, für die nicht gilt ($v=s_v$). Anschließend wird propagiert, um die Änderungen im Netz zu verteilen. Das Propagieren besteht aus zwei Phasen: Die erste Phase beginnt an allen Blättern des Baumes. Für jedes Blatt wird eine so genannte *λ-Nachricht* generiert, die die Änderungen dieser Clique, die für den Elternknoten relevant sind, beinhaltet. Die Elternknoten sammeln alle *λ*-Nachrichten von

allen Kinderknoten und aktualisieren damit ihre CPT, die in diesem Zusammenhang auch *Potenzial* genannt wird. Als Nächstes generieren diese Knoten ebenfalls λ-Nachrichten und schicken diese wiederum an ihre Elternknoten. Die erste Phase der Propagierung (*CollectEvidence* genannt) endet an der Wurzel des Baumes. Nun beginnt die zweite Phase (*DistributeEvidence* genannt). Nachdem der Wurzelknoten alle λ-Nachrichten verarbeitet hat, generiert er so genannte π-Nachrichten, die er an all seine Kinderknoten schickt. Analog zur ersten Phase wird dieses Verfahren rekursiv wiederholt, bis alle Blätter π-Nachrichten empfangen und verarbeitet haben. Durch das zweistufige Vorgehen wird sichergestellt, dass sich alle Evidenzen bei allen Knoten im Graphen auswirken, wenn zwischen den Knoten ein Zusammenhang besteht.

Abbildung 19 zeigt die allgemeine Form des Message-Passing in Polytrees. Verwendet man den Junction-Tree-Algorithmus, vereinfacht sich dieses Verfahren wie oben beschrieben, da pro Knoten nur maximal eine λ-Nachrichten generiert werden muss. Der von mir entworfene MPA sieht im Detail folgendermaßen aus:

Definitionen:

$S_c = Clique_c \cap Parent(Clique_c)$

E_v = Evidenzknoten, für diese Knoten wurde eine Evidenz gesetzt

$PS_c = S_c \setminus E_v$

$PR_c = Parent(Clique_c) \setminus (S_c \cup E_v)$

$R_c = Clique_c \setminus S_c$

λ_c = Menge von λ-Nachrichten, die aus Potenzialen ψ bestehen

π_c = Menge von π-Nachrichten, die aus Potenzialen ψ bestehen

$P(c)$ = das Potenzial einer Clique

Propagate()

CollectEvidence(root)

DistributeEvidence(root)

Berechne neue Wahrscheinlichkeitsverteilungen für alle Knoten v

foreach (Variable v in $Clique_c$)

$$P(v = v') = \sum_{w \in W} P(Clique_c = w)$$

```
CollectEvidence( Clique c )
  if ( Childs(c) ≠ ∅ )
        CollectEvidenceRec( c )
  else
        if ( P( c ) = 0, ∀i)
              exit „Inconsistent data!“


CollectEvidenceRec( Clique c )
  if (Childs(c) = ∅ )
        λ_prop( c )
  else
        if ( #λc = #Childs( c )
              if ( c ≠ root)
                    λ_prop( c )
              else
                    if ( P( c ) = 0, ∀i)
                          exit „Inconsistent data!“
        else
              foreach ( k in Childs( c ) )
                    CollectEvidenceRec( k )
              CollectEvidenceRec( c )


DistributeEvidence( Clique c )
  if ( Childs(c) ≠ ∅ )
        if ( c = root )
              π_prop( c )
              foreach ( k in Childs( c ) )
                    DistributeEvidence( k )
        else
              if (πc ≠ ∅)
                    P( c ) * πc
                    πc = ∅
              π_prop( c )
              foreach ( k in Childs(c) )
```

DistributeEvidence(k)

else

if ($\pi_c \neq \varnothing$)

$P(c) * \pi_c$

$\pi_c = \varnothing$

λ_prop(Clique c)

if ($R_c \setminus E_v = \varnothing$)

if (Parent(c) $\neq \varnothing$)

$\lambda_{Parent(c)} * \lambda_c$

if ($Nodes_c \setminus E_v \neq \varnothing$)

$\lambda_c = 1$

else

if ($PS_c = \varnothing$)

if (Parent(c) $\neq \varnothing$)

$$\lambda_{Parent(c)} \cdot \sum \lambda_c$$

$$\lambda_c = \frac{\lambda_c}{\sum \lambda_c}$$

else

$$\lambda_{Parent(c)} \cdot \sum_{PS_c} \lambda_c$$

$$\lambda_c = \frac{\lambda_c}{\sum_{PS_c} \lambda_c}$$

π_prop(Clique c)

foreach (k in Childs(c))

if($PS_k \neq \varnothing$)

if ($PR_k = \varnothing$)

$\pi_k = P(c)$

else

$$\pi_k = \sum_{PS_k} P(c)$$

SetEvidenz(Knoten v, Variable s)
Update: PS, PR, E_v

foreach ($Clique_c$)
$Clique_c = Clique_c \setminus v$
$P(Clique_c) = P(Clique_c)$, mit Werten X für die gilt (v = s)

Propagate()
Normalisiere Wahrscheinlichkeitsverteilungen für alle Knoten

3.4.5.7 Approximative Inferenz durch stochastische Simulation

Exakte Inferenzalgorithmen wie der Junction-Tree-Algorithmus sind NP-hart und können daher bei größeren Netzen schnell unpraktikabel werden. Wie viele Knoten mit exakten Methoden, z. B. dem Junction-Tree-Verfahren handhabbar sind, lässt sich nicht pauschal sagen, da die Komplexität wesentlich von der Anzahl der Zustände der Knoten und von dem Verknüpfungsgrad abhängt. Erfahrungsgemäß sind Netze mit 50 und mehr Knoten mit exakten Verfahren problemlos handhabbar. Alles was weit darüber hinausgeht, lässt sich nur approximativen Inferenzalgorithmen berechnen. Ein Ansatz für approximative Inferenzen für mehrfach verbundene Netzwerke sind stochastische Simulationen, die auch unter der Bezeichnung *Monte-Carlo-Verfahren* bekannt sind. Monte-Carlo-Verfahren (MC) sind randomisierte Sampling-Algorithmen, deren Genauigkeit auf der Anzahl der Samples, die generiert werden, beruht. MC-Verfahren werden heutzutage in vielen Bereichen der Informatik, wo exakte Verfahren nicht zur Verfügung stehen oder auf Grund der Komplexität nicht praktikabel sind, angewendet, z. B. beim Optimierungsverfahren *Simulated Annealing*. Im Falle Bayes'scher Netze wird die stochastische Simulation für die Berechnung der A-Posteriori-Wahrscheinlichkeit eines oder mehrerer Anfrageknoten verwendet und ersetzt damit exakte Inferenzverfahren. Dazu wird approximativ eine große Anzahl von Wahrscheinlichkeitsverteilungen generiert und zu einer Verteilung kombiniert, die nach dem *Gesetz der Großen Zahlen* mit steigender Anzahl an Samples zu der exakten Verteilung konvergiert. Obwohl die approximativen Verfahren aus theoretischer Sicht immer noch NP-hart sind, konvergieren diese in der Praxis schnell genug, um bei großen Netzen den exakten Inferenzverfahren überlegen zu sein. In der Literatur existieren viele unterschiedliche Verfahren, die alle gewisse Vor- bzw. Nachteile haben, wie z.B. *Direct Sampling*, *Logic Sampling* (*Rejection Sampling)*, *Variational Methods*, *Loopy Propagation*, *Likelihood Weighting (Importance Sampling)* und der *Markov-Chain-Monte-Carlo-Algorithmus (MCMC)* ([KorNic04], [RusNor03], [GemGem84]).

Für SimMarket wurde eine spezielle Variante des MCMC, genauer gesagt des *Gibbs-Sampler*, entwickelt. Zudem wurde das *Likelihood-Weighting-Verfahren* in das Simulationssystem integriert.

Der Gibbs-Sampler, wie er aus der Literatur bekannt ist, wurde in SimMarket durch eine Simulated-Annealing-Routine (SAR) ergänzt, die die Ergebnisse deutlich verbessert.

In der Praxis hat sich gezeigt, dass die approximativen Verfahren deutlich weniger Speicher verbrauchen als der Junction-Tree-Algorithmus und das die Komplexität der Inferenzberechnung erheblich langsamer mit der Komplexität des Netzes wächst. Allerdings hat sich auch gezeigt, dass die Vorteile dieser Verfahren erst bei sehr großen Netzen zum Tragen kommen. Bei Netzen mit weniger als ca. 50 Knoten ist der in SimMarket integrierte und hoch optimierte Junction-Tree-Algorithmus schneller. Die approximativen Verfahren sind schneller, wenn sich die Kosten des Sampling in großen Netzen amortisieren. Das Likelihood-Weighting-Verfahren, das sich in der Praxis als das bessere Verfahren herausgestellt hat, benötigt mindestens 10^4 Sampling-Iterationen, um akzeptable Ergebnisse zu liefern. Eine Anzahl von $6*10^4$ Iterationen hat sich empirisch als guter Wert erwiesen. Eine Iterationsanzahl von mehr als $2*10^5$ führte in keinem Fall zu einer signifikanten Verbesserung der Ergebnisse. Diese Aussagen beziehen sich auf die in SimMarket verwendete Netzstruktur der Artikel- und Warengruppenagenten und können nicht für beliebige Netze verallgemeinert werden. Die Güte der Approximationen wurde durch einen Vergleich mit dem exakten Junction-Tree-Verfahren ermittelt. Dabei hat sich herausgestellt, dass die Güte des Likelihood-Weighting-Verfahrens mit der Anzahl der Evidenzen abnimmt.

3.4.6 Adaption der Agenten

Die Wissensbasis des Agenten wird im Normalfall aus der gesamten vorliegenden Datenmenge, die für diesen Agenten relevant ist, extrahiert. Während der Laufzeit nimmt ein Agent kontinuierlich neue Daten über seine Sensoren auf, die er für eine zielgerichtete Planung im Hinblick auf seine Ziele einsetzt und die in der Wissensbasis des Agenten abgelegt werden sollten. Da es nicht sinnvoll ist, den in den vorangegangenen Abschnitten geschilderten Extraktions- und Lernprozess bei jeder neuen Sensorinformation auf ein Neues anzuwenden, benötigen wir ein adaptives Integrationsverfahren für neues Wissen, das mit den folgenden Schwierigkeiten umgehen kann.

Es ist nicht auszuschließen, dass der Agent Daten sammelt, die sich gegenseitig widersprechen oder im Widerspruch zu bereits gelernten Fakten stehen. Deshalb ist es sinnvoll einen Mechanismus in die Agenten zu integrieren, der neues Wissen bei der Adaption der Wissensbasis stärker gewichtet als altes Wissen und den Agenten mit der Zeit altes Wissen „vergessen“ lässt.

Das Problem der Adaption lässt sich folgendermaßen formalisieren:

Definition **Adaption**: Gegeben sei ein probabilistischer Agent *A* mit einem Verhaltensnetz *VN* und eine Bewertungsfunktion *BF*, dann versteht man unter Adaption die Modifikation aller probabilistischer Netze $B(D_{[t,l]})$, so dass der Gütewert von *BF* hinsichtlich der neuen Sensordaten $D_{>l} \cup D_{[t,l]}$ unter Berücksichtigung des Domänenwissens *DK* optimiert wird.

Je nach Informationsgehalt der neuen Sensordaten müssen mehr oder weniger Teile eines Verhaltensnetzes angepasst werden. Enthalten die neuen Sensordaten viele neue Informationen oder stehen neue Fakten im Widerspruch zu vorhandenem Wissen, sind größere Modifikationen an den Verhaltensnetzen nötig, als nur bei wenigen neuen Informationen. Der Maßstab, ob und wie umfassend ein Netz geändert werden muss, wird mit Hilfe der Bewertungsfunktion BF ermittelt. Wir verwenden in SimMarket den bereits beschriebenen MDL-Score. Die BF gibt einen Wert an, wie gut das aktuelle Netz die Datenbasis D modelliert. Kommen neue Daten $D_{>l}$ zur Datenbasis $D_{[t,l]}$ hinzu verringert sich in der Regel der Wert von BF. Ein Adaptionsalgorithmus beschreibt einen Weg, wie der Wert von $BF(D_{>1} \cup D_{[t,1]})$, unter Berücksichtigung des bereits im Netz integrierten Domänenwissens DK, optimiert werden kann, ohne das Netz vollständig neu zu lernen. Auch die Annahme, dass neues Wissen womöglich „exakter“ ist, als altes Wissen muss von einem Adaptionsalgorithmus bedacht werden. Da die meisten Umgebungen, in denen Agenten agieren, höchst dynamisch sind, folgt, dass neuere Sensordaten meist ein genaueres Bild der Welt liefern als altes Wissen, das bereits in den Verhaltennetzen gespeichert ist. Das Ziel der Adaption ist es den Modifikationsaufwand der Netze möglichst gering zu halten. Deshalb wird bei der Adaption nach einem minimalen Teilbereich des Netzes gesucht, der durch möglichst kleine Modifikationen den Wert von BF auf den gewünschten Wert optimiert.

Das Gewichten von alten und neuen Daten bzw. das Vergessen von Wissen lässt sich prinzipiell auch in ein vollständiges Lernverfahren integrieren. Da allerdings der Lernaufwand exponentiell mit der Datenmenge wächst, möchte man vollständiges Lernen vermeiden und stattdessen die Funktion BF durch partielles Lernen optimieren.

Das Problem der Adaption wird in vier Teilschritte zerlegt:

Die einfachste Art der Adaption geschieht durch das Aktualisieren der bedingten Wahrscheinlichkeitstabellen (CPT). Das bekannteste Verfahren in dieser Klasse ist die aHugin-Methode von Spiegelhalter und Lauritzen [SpiLau90] bzw. von Olesen, Lauritzen und Jensen [OleLauJen92]. Es kann sowohl mit vollständigen als auch mit unvollständigen Daten umgehen. Die Gewichtung der Vergangenheit und der Gegenwart wird durch die Definition so genannter *Fading-Tabellen*, die die Gewichte beinhalten, realisiert. Dieses Verfahren kann ein Netz nur im Rahmen der bereits vorhandenen Struktur adaptieren, bei größeren Datenänderungen ist dieses Verfahren nicht mehr anwendbar.

Bei Artikelagenten müssen beispielsweise häufig neue Verkaufspreise berücksichtigt werden, die durch neue Zustände im Netz repräsentiert werden müssen. Es muss also eine Adaption auf der Ebene der Zustände stattfinden. Dazu wird ein neuer Zustand s bei Knoten v eingefügt und die CPTs von v und seinen Kindern *Childs(*v*)* angepasst. Diese müssen nicht vollständig neu gelernt werden, sondern können durch geschicktes Hinzufügen einer neuen Dimension erweitert werden, die dann die Wahrscheinlichkeiten der neuen Kombinationen von Zuständen, insbesondere des neuen Zustands s, enthält. Da für viele Kombinationen zu diesem Zeitpunkt keine Informationen vorliegen, werden diese Kombinationen mit Nullwahrscheinlichkeiten aufgefüllt. Diese Art von Adaption, bei der nur einzelne Zustände eines Knotens geändert werden, ist in der Literatur bisher nicht bekannt, lässt sich aber mit dem oben beschriebenen Verfahren bewerkstelligen.

Die Adaption der Struktur eines Netzes gestaltet sich dagegen schwieriger, da es sich hierbei nicht um einen kontinuierlichen Prozess handelt. Erst wenn es hinreichend viele Veränderungen in den Daten gibt, muss sich die Struktur ändern, d. h., eine Kante fällt weg, kommt hinzu oder die Richtung ändert sich. Die Schwierigkeit besteht darin, zu erkennen, wann welche (möglichst minimale) Änderung den Wert von BF signifikant verbessert. Dazu gibt es zwei grundsätzliche Ansätze:

1. man sammelt neue Sensordaten und überprüft in regelmäßigen Abständen, ob eine Strukturadaption nötig ist, oder
2. man generiert mehrere alternative Modelle, die nach einem Model-Averaging-Konzept gemäß ihrer A-Posteriori-Wahrscheinlichkeiten im Inferenzprozess gewichtet werden.

Eine ausführliche vergleichende Diskussion der verschiedenen Verfahren findet man bei [RouSan99]. Aus den Strukturadaptionsverfahren ergibt sich automatisch ein Vorgehen, um die zu verwendenden Knoten bzw. Einflussfaktoren, die das Verhalten eines Agenten beeinflussen, im Netz zu ermitteln und zu adaptieren. Führt nach der Adaption keine Kante (beliebiger Richtung) zu einem Knoten oder zu einem Teilnetz, so können diese Knoten gelöscht werden, da sie keinen Einfluss auf das Verhalten des Agenten ausüben. Umgekehrt sollte in regelmäßigen Abständen überprüft werden, ob neue Knoten die Qualität (also den Wert von BF) des Netzes verbessern. Dazu fügt man testweise neue noch nicht existierende Knoten dem Netz hinzu und berechnet den Wert von BF. Ergibt sich eine signifikante Verbesserung, sollte der neue Knoten im Netz bleiben.

Lernverfahren, die mit *versteckten Variablen* (engl. *hidden variables*) [Friedman97] arbeiten, können ebenfalls für die Adaption genutzt werden. Diese Verfahren nehmen an, es existiert ein weiterer nicht bekannter Einflussfaktor, der noch nicht im Netz modelliert ist und fügen einen anonymen Knoten testweise in das Netz ein. Mit einem erweiterten *EM-Algorithmus* (*Expectation-Maximisation*) wird anschließend der Wert von der versteckten

Variable ermittelt. Hat diese neue Variable tatsächlich einen signifikanten Einfluss auf andere Variablen, wird diese beibehalten.

3.4.7 Holonisierung

Eine der essenziellen Eigenschaften der probabilistischen Agenten in SimMarket ist ihre Fähigkeit sich zu Holonen zusammenzuschließen. Wie in Kapitel 3.1.1 beschrieben, versteht man unter Holonisierung den Prozess der Verschmelzung von mehreren Agenten zu einem Holonen. Je nach Anforderung können die Agenten vollständig oder partiell miteinander verschmelzen. Eine ausführliche Abhandlung der verschiedenen Holonenformen ist in [Schillo04] und [Schwaiger06] zu finden.

In vielen Domänen ist das Generieren von präzisen globalen Modellen ein NP-vollständiges Problem. Deshalb muss oft zwischen weniger detaillierten Modellen, die zwar einfach zu generieren sind, aber meistens zu ungenau sind und komplexen Modellen, die sehr exakt sind, aber in der Praxis nur schwer oder gar nicht erzeugt werden können, abgewogen werden. Einen Lösungsansatz bietet das Konzept der Holonenbildung:

Das globale Modell einer Domäne wird gewonnen, indem viele Agenten kleinere lokale Teilmodelle der Domäne repräsentieren z. B. von einzelnen Artikeln, Kunden etc. und diese erst bei Bedarf zu größeren globalen Modellen mittels Holonisierung verschmolzen werden. Dabei werden zusätzlich neue Relationen zwischen den Teilmodellen aufgedeckt.

In [GerSieVie99] wird eine formalisierte Einführung von Holonen und eine abstrakte Beschreibung des Holonisierungsprozesses präsentiert. Im Rahmen dieser Arbeit wird das Holonenkonzept erstmalig auf eine probabilistische Agenten-Architektur angewandt und durch einen strukturbasierten Holonisierungsalgorithmus konkretisiert, der sich sowohl für vollständiges als auch für partielles Verschmelzen von Agenten verwenden lässt. Das Verfahren beschreibt, wie zwei Verhaltensnetze von zwei verschiedenen Agenten zu einem Metanetz verschmolzen werden, das dann als Wissensbasis des resultierenden Holonen fungiert. In dem holonischen Metanetz bleiben die ursprünglichen Netze als identifizierbare Teilnetze erhalten.

Für die Holonisierung ist es notwendig die verwendeten Bayes'schen Netze sukzessiv zu Verhaltensnetzen zu erweitern und mit semantischen Annotationen anzureichern. Ein Konzept der Welt W eines Agenten wird durch einen Knoten oder ein Teilnetzwerk von Knoten in einem Verhaltensnetz repräsentiert. Die Bedeutung, die ein Knoten V_i oder Teilnetz vor einer Verschmelzung der Agenten hatte, wird durch eine oder mehrere Referenzen $V_i \leftarrow \{Ref(A_1),\ldots,Ref(A_n)\}$ auf die ursprünglichen Einzelagenten während der Fusion erhalten. Zusätzlich kann ein Knoten V_i ein *Tag* $st_i \in DK$ haben, das die Semantik dieses einzelnen Knotens $V_i \leftarrow st_i$ angibt, z. B. erhält ein Knoten mit dem Namen "Abverkaufsmenge" die semantische Bezeichnung *Einheiten*.

Die Struktur eines Bayes'schen Netzes drückt nicht implizit kausale Abhängigkeiten aus, sondern nur bedinte Abhängigkeiten. Um ein Netz für menschliche Benutzer interpretierbarer zu machen, sollten diese Abhängigkeiten kausalen Beziehungen entsprechen. Außerdem hat sich in der Praxis gezeigt, dass kausale Strukturen häufig die Komplexität der Netze minimieren. Um die gewöhnliche Struktur eines Netzes während der Fusion aufrecht zu erhalten und die Komplexität so klein wie möglich zu halten, führen wir daher weitere *Tags* ein, die die Kausalität eines Knotens $V_i \leftarrow ct_i \in DK$ modellieren.

Jeder Knoten kann mit den folgenden Attributen belegt werden, die dabei helfen die Schnittstellen der Agenten untereinander zu definieren: {*Inout, Input, Output, Inner*}. Diese Flags beschreiben ausschließlich die Relationen zwischen Knoten von unterschiedlichen Agenten und nicht zwischen Knoten innerhalb eines Agentennetzes. *Input-Knoten* bedeutet, dass der Knoten ein Input-Knoten für ein Teilnetz eines anderen Agenten ist, d. h., dass eingehende Kanten von anderen Agenten-Teilnetzen in diesem Knoten münden dürfen. *Output-Knoten* bezeichnen das Gegenteil von Input-Knoten, sie dürfen nur ausgehende Kanten zu anderen Agenten-Teilnetzen besitzen (oder beliebige andere Kanten zu Knoten desselben Agenten). *Inout* bedeutet, dass dieser Knoten sowohl ein Input- als auch ein Output-Knoten ist. *Inner-Knoten* sind innere Knoten eines Teilnetzes und besitzen nur Kanten zu und von Knoten innerhalb eines Agentennetzes, während Verbindungen zu anderen Agentennetzen nicht erlaubt sind.

Alle Kanten E_i von G besitzen sich gegenseitig ausschließende Kantentypen $ET \subseteq DK$, die definieren, ob es sich um *vorgeschlagene*, *verbotene* oder eine *zwingende* Kante handelt. Diese Kantentypen werden benutzt, um das partielle Domänenwissen des Benutzers zu modellieren und das initiale Lernen der Netze zu unterstützen. Falls ein Benutzer eine Beziehung zwischen zwei Knoten vermutet, aber nicht zu 100% sicher ist, kann er eine vorgeschlagene Kante einsetzen. Vorgeschlagene Kanten werden mit dem *MDL-Wert* überprüft und nur eingesetzt, wenn sich dadurch die Gesamtbewertung des Netzes erhöht. Verbotene Kanten werden benutzt, um den Suchraum des Algorithmus zum Strukturlernen einzuschränken und offensichtlich falsche Kanten zu verbieten. Zwingende Kanten werden in jedem Fall eingesetzt, selbst wenn die Gesamtbewertung sich verringert.

Zusätzlich zu den Kantentypen kann der Benutzer spezifizieren, ob eine Kante gerichtet oder ungerichtet ist. Eine ungerichtete Kante sollte nur verwendet werden, wenn der Benutzer die Kausalität zwischen zwei abhängigen Knoten nicht kennt. Da ein Bayes'sches Netz nur gerichtete Kanten erlaubt, testet das Lernverfahren, welche Kantenrichtung ein akkurateres Netzwerk ergibt und fügt diese ein.

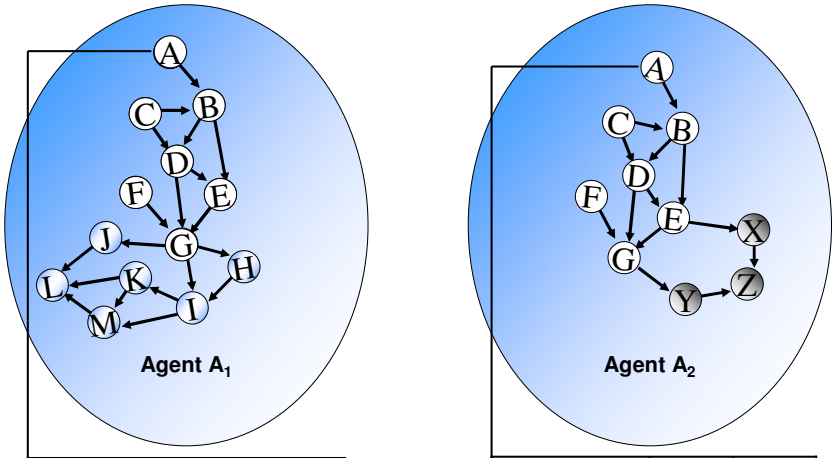
Das Lernen initialer Verhaltensnetze wird ausführlich in [Schwaiger06] behandelt. Der folgende Verschmelzungsalgorithmus verwendet im dritten Schritt eine speziell für die Holonisierung erweiterte Variante des in [Schwaiger06] beschriebenen Lernverfahrens mit Domänenwissen.

Auf Grund der speziellen Semantik und des Domänenwissens der in SimMarket verwendeten Netze sprechen wir im Folgenden nicht mehr allgemein von Bayes'schen Netzen sondern von Verhaltensnetzen. Der Begriff Verhaltensnetz steht für ein probabilistisches Netzwerkmodell mit Eigenschaften wie z. B. objektorientiert, hierarchisch und Repräsentation von Semantik und Domänenwissen (siehe auch 3.5.1).

Im Folgenden beschreibe ich das konkrete Verschmelzungsverfahren, bei dem die Agenten Teile ihres Wissens an einen „Klon" abgeben, der das Wissen der Agenten zu einem Metanetz fusioniert und dabei neues Wissen ableitet.

3.4.7.1 Finden maximaler gleicher Teilnetze

Abbildung 20 zeigt zwei Agenten mit probabilistischen Netzwerken, die in einen Holon verschmolzen werden sollen. Zunächst müssen alle maximalen gleichen Teilnetze gefunden werden, da diese Teile nicht verschmolzen werden müssen. Gleiche Knoten kommen je nach Netzstruktur relativ häufig vor, z. B. bei Artikelagenten die Zeitknoten *Tag, Wochentag, Monat* etc. Die Information, welche Knoten identisch sind, wird ebenfalls benötigt, um zu wissen, wo ungleiche Knoten angeknüpft werden.



$P(A_1\|B_1,C_1)$	$A_1 = 0$	$A_1 = 1$
$B_1 = 0, C_1 = 0$	0.2	0.8
$B_1 = 0, C_1 = 1$	0.5	0.5
$B_1 = 1, C_1 = 0$	0.1	0.9
$B_1 = 1, C_1 = 1$	0.43	0.57

Bedingte Wahrscheinlichkeitstabelle A_1

=

$P(A_2\|B_2,C_2)$	$A_2 = 0$	$A_2 = 1$
$B_2 = 0, C_2 = 0$	0.2	0.8
$B_2 = 0, C_2 = 1$	0.5	0.5
$B_2 = 1, C_2 = 0$	0.1	0.9
$B_2 = 1, C_2 = 1$	0.43	0.57

Bedingte Wahrscheinlichkeitstabelle A_2

Abbildung 20: Finden maximaler gleicher Teilnetze

Wenn keine Knoten gleich sind, werden die beiden Netze als getrennte Teilnetze behandelt. In späteren Verarbeitungsschritten können allerdings neue Kanten zwischen den

beiden Teilnetzen hinzugefügt werden und daraus ein einzelnes Gesamtnetz resultieren. Um alle maximalen gleichen Teilnetze zu finden, beginnen wir bei den Wurzelknoten und vergleichen sie miteinander.

Definition Gleichheit von Knoten
Zwei Knoten V_i, V_j sind gleich, wenn die bedingten Wahrscheinlichkeitstabellen (CPT) und die eingehenden Nachbarn $Pa(V_i) = Pa(V_j)$ gleich sind, wobei $Pa(V_i) = V_1,..,V_n$ alle Elternknoten von V_i bezeichnet. Das bedeutet, dass alle möglichen Pfade ($V_i \rightarrow V_{i+1} \rightarrow \ldots \rightarrow V_n$ = Wurzelknoten) zu Wurzelknoten gleich sein müssen. Der Grund hierfür ist, dass eine unterschiedliche Wahrscheinlichkeitsverteilung (CPT) von $Pa(V_i)$ nach der Kompilierung des Netzes zu unterschiedlichen CPT von V_i führen kann.

Falls zwei Wurzelknoten mit gleichen CPTs gefunden werden, fährt der Algorithmus rekursiv mit den ausgehenden Nachbarn dieser Knoten fort. Dieses Verfahren wird für alle gleichen Wurzelknoten und bislang unbesuchten Knoten wiederholt. Wir verwenden verschiedene Farben für die Knoten, um zu kennzeichnen, welche Knoten im Verlaufe der rekursiven Funktion bereits *visited*, welche noch *unvisited*, welche *equal* und welche *taken* sind.

Definition des Operators ==
Der binäre Operator == ist *wahr genau dann, wenn* die CPT zweier Knoten gleich sind. Der Operator testet also nicht die volle Gleichheit von BN-Knoten, wie oben definiert.

Nach dem ersten Durchlauf der Schleife in *FindMaxEqualNets(G_B)* repräsentieren die Knoten mit der Farbe *taken* ein maximales gleiches Subnetz. Nach n Durchläufen hat man alle disjunkten maximalen gleichen Subnetze der Graphen G_A und G_B gefunden.

FindMaxEqualNets(G_B)
foreach $root_{GB}$ in $roots_{GB}$
if $root_{GA} == root_{GB}$
$root_{GB}.NodeColour \leftarrow taken$
FindEqualChildNodesComingFromParent()
$roots_{GB} \leftarrow unvisited\ roots_{GB}$
$equalSubNets \leftarrow \{taken\ nodes_{GB}\}$
else
$root_{GB}.NodeColour \leftarrow visited$
$roots_{GB} \leftarrow roots_{GB} \setminus \{root_{GB}\}$
return *equalSubNets*

Nachdem gleiche Wurzelknoten gefunden sind, wird die Methode *FindEqualChildNodesComingFromParent()* aufgerufen, um alle Kindknoten zu prüfen. Kindknoten sind gleich, wenn ihre CPT und alle möglichen Wege zu Wurzelknoten gleich sind (*AllPathsToRootEqual(equalNode)*). Wenn gleiche Knoten gefunden werden, wird die Methode wieder für alle Kindknoten aufgerufen:

```
FindEqualChildNodesComingFromParent()
  foreach child_GB in childs_GB
      if child_GA == child_GB
          equalNode ← child_GA
      child_GB.NodeColour ← visited
      if (equalNode ≠ ∅) AND (AllPathsToRootEqual(equalNode))
          child_GB.NodeColour ← taken
          FindEqualChildNodesComingFromParent()
```

Die Boole'sche Funktion *AllPathsToRootEqual()* prüft die Gleichheit für alle Knoten für alle Pfade zu Wurzelknoten. Falls gleiche Elternknoten gefunden werden, wird die Methode *FindEqualChildNodesComingFromChild()* aufgerufen, um auch alle gleichen Kindknoten dieses Elternknotens zu finden. *FindEqualChildNodesComingFromChild()* ist der Methode *FindEqualChildNodesComingFromParent()* sehr ähnlich, mit dem Unterschied, dass wir prüfen müssen, ob ein Kindknoten vorher besucht wurde, da wir von einem Kinderknoten kommen und keine Zyklen durchlaufen möchten.

Die Laufzeit für das Finden von allen maximalen gleichen Teilnetzen bei konstanten CPT-Größen ist linear $O(n+m)$ mit der Anzahl an Knoten, da jeder Knoten der beiden zu vergleichenden Netze nur einmal „angefasst" werden muss. Allerdings hat der Operator == exponentielle Laufzeit, da die Größen der CPT exponentiell mit der Anzahl der Zustände z der zugehörigen Knoten X_i und ihrer Elternknoten $\pi(X_i)$ wachsen. Die Größe E_{CPT} einer CPT für den Knoten X_i mit z Zuständen und *n* Knoten beträgt:

$$E_{CPT}(X_i) := z_{X_i} \cdot z_{\Pi_1(X_i)},\ldots,\cdot z_{\Pi_{n-1}(X_i)} \leq \underbrace{\max(z_{X_i})^n}_{:=C}$$

Da die Laufzeit des Verfahrens auch von der Größe der CPT abhängt, ist dieses im schlimmsten Falle wie das Wachstum der CPT exponentiell $O(n+m \cdot C)$.

```
AllPathsToRootEqual(node)
  foreach parent_GB in node.Parents_GB
      if parent_GB.NodeColour == unvisited
          parent_GB. NodeColour ← visited
          if parent_GA == parent_GB
              if AllPathsToRootEqual (parent_GB)
                  parent_GB.NodeColour ← taken
                  FindEqualChildNodesComingFromChild()
                  return true
              else return false
          else return false
      else return true
```

3.4.7.2 Das Verschmelzen von probabilistischen Netzwerken

Nach der Identifizierung aller maximal gleichen Teilnetze und damit aller gleicher Knoten $V_E=\{V_1,\ldots,V_{i-1}\}$ können wir die ungleichen Teile des Netzwerks $V_{UE} = \{V_i,\ldots,V_n\}$ des Agenten A_2 identifizieren (siehe Abbildung 21). Bevor diese Teile mit dem Netzwerk G_1 von Agent A_1 verbunden werden, wird ein Klon CL_1 von A_1 produziert, d.h., es wird eine exakte Kopie von A_1 erstellt. Damit wir auch der Datensatz D_1 nach D_{CL} geklont.

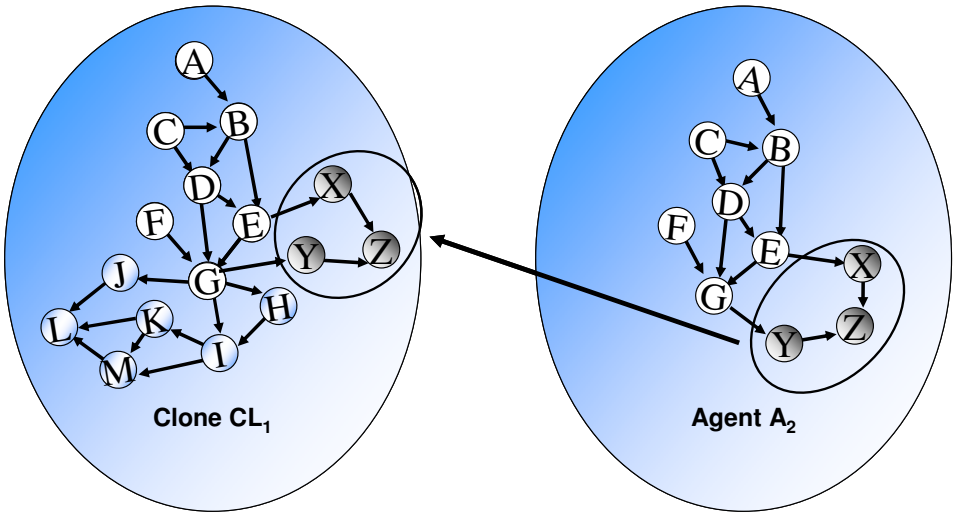


Abbildung 21: Das Verschmelzen von probabilistischen Netzwerken

Für alle V_{UE} in A_2 werden Kopien im Netzwerk von CL_1 erzeugt. Dann werden Kopien aller Kanten zwischen V_{UE} zwischen den neuen Knoten in CL_1 hinzugefügt. Diese Kanten weisen immer von V_E nach V_{UE}. Die umgekehrte Richtung V_{UE} nach V_E kann nicht

vorkommen, da sonst die Knoten von V_{UE} Eltern von V_E wären. Dies ist aber gemäß der Definition von Gleichheit der Teilnetzen nicht möglich, da alle Eltern von V_E auch in V_E sein müssen. Nach dem Verschmelzungsprozess muss keine CPT aktualisiert oder neu gelernt werden. Alle Knoten in V_E sind unbeeinflusst, weil keine neuen Kanten auf Knoten in V_E weisen. Die CPTs in V_{UE} müssen auch nicht aktualisiert werden, da keine neuen Kanten zu Knoten in V_{UE} hinzugefügt worden sind.

Im nächsten Schritt verschmelzen wir die Rohsensordaten der Vergangenheit $D_{[t,l]}$ und die noch nicht integrierten Sensordaten $D_{>l}$ der Agenten A_1 und A_2 miteinander. Die Verschmelzung von Datensätzen ist relativ einfach, da alle Knoten der Netze als einzelne Spalten in D repräsentiert werden. Die Datensätze werden verschmolzen, indem die Spalten der ungleichen Knoten in $D_2 \in A_2$ zu den entsprechenden Datensätzen in $D_{CL} \in CL_1$ hinzugefügt werden.

Natürlich ist es möglich, dass keine gleichen Knoten gefunden werden. In diesem Fall wird das Netzwerk von A_2 zu CL_1 als isoliertes Teilnetz hinzugefügt. Im nächsten Schritt der Holonisierung können neue Relationen entdeckt und damit neue Kanten hinzugefügt werden.

Für die Laufzeit in diesem zweiten Schritt des Verfahrens gilt die gleiche Aussage wie für den ersten Schritt: Da die Netze im schlimmsten Fall exponentiell $O(C)$ mit der Anzahl an Zuständen und Knoten wachsen, gilt dies natürlich auch für das Klonen eines Netzes bzw. Teilnetzes. Bei konstant großen Netzen ist dieser Schritt ebenfalls in konstanter Zeit $O(1)$ durchführbar.

3.4.7.3 Entdecken und Integrieren von Abhängigkeiten

Im nächsten Schritt wird eine Suche nach Abhängigkeiten zwischen den verschmolzenen Agenten gestartet. Dazu verwende ich eine modifizierte Variante des Lernverfahrens für initiale Netze aus [Schwaiger06]. Diese sieht wie folgt aus:

Begonnen wird mit einer *Kreuzvalidierung* zwischen dem alten und dem neuen Teil des Netzwerkes. Für alle möglichen Kanten zwischen *Output-Knoten* $\cup$ *InOut-Knoten* und *Input-Knoten* $\cup$ *InOut-Knoten*, welche zwischen den holonischen Teilnetzen liegen, wird das *mutual weight* (siehe [Schwaiger06]) berechnet, falls die Kante nicht durch eine eventuell vorhandene *verbotene Kante* ausgeschlossen wurde. Beginnend mit der Kante mit dem größten Gewicht werden alle Kanten nacheinander in CL_1 eingefügt und die CPT des Zielknotens neu berechnet. Erhöht sich der MDL-Wert durch diese Hinzunahme, wird die Kante beibehalten, andernfalls wieder verworfen.

LernHolonDependencies()
foreach *inNodes* = CL_1.*InputNodes* ∪ CL_1.*InoutNodes*
foreach *outNodes* = CL_1.*OutputNodes* ∪ CL_1.*InoutNodes*
if (*inNode* ∈ H_i ⇒ *outNode* ∉ H_i)
∧ ($E_{(inNode, outNode)}$ ∉ CL_1.*ForbiddenEdges*)
newEdge = (*inNode, outNode, Proposed, Directed*)
CalculateMutualWeight(newEdge)
edgeList.Add(newEdge)
edgeList.Sort()
foreach *newEdge* in *newEdges*
InsertEdge(newEdge, CL_1*)*
LearnCPT(newEdge.EndNode)
if *MDLScore(*CL_1*)* decreases
Remove(newEdge)
LearnCPT(newEdge.EndNode)

Auf diese Weise entsteht aus zwei oder mehr Agenten ein Holon, der mehr Wissen besitzen kann, als die Summe der beiden Ursprungsagenten und dessen Struktur für viele Aufgaben effizienter ist, da erstens mehr Wissen vorhanden ist und zweitens Kommunikationskosten zwischen Einzelagenten sinken. Bei diesem Prozess emergiert neues Wissen, wenn zwei unterschiedliche Agenten, die allerdings in einer gewissen Relation zueinander stehen müssen, verschmolzen werden (siehe Abbildung 22). Dies entspricht exakt der Definition von *Emergenz*.

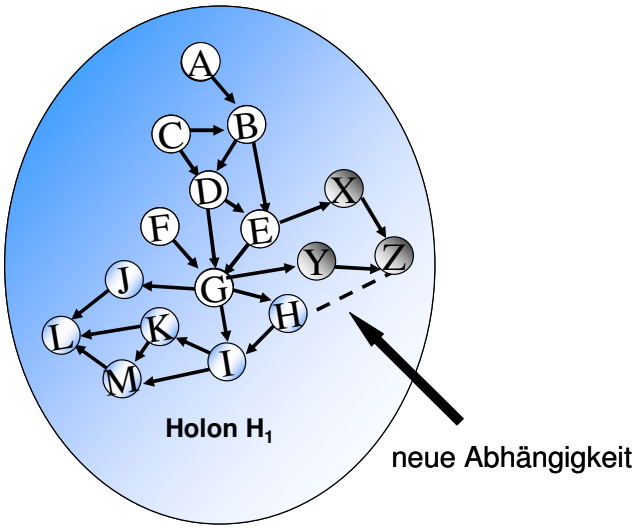


Abbildung 22: Abgeleitete Abhängigkeiten

Ein Holon existiert im Allgemeinen nur temporär, bis das aktuell vorliegende Ziel erreicht wurde und anzunehmen ist, dass in Kürze keine weitere Verwendung des Holon nötig wird. Die Hauptanwendung dieses Verfahren liegt in der Abhängigkeitsanalyse (siehe Kapitel 5.2) von Artikeln, bei der zwei Artikelagenten zu einem Holon verschmolzen werden, um die Wechselwirkungen zwischen diesen simulieren zu können. Eine Evaluierung des Verfahrens ist in Kapitel 5.3.3 zu finden.

Die Laufzeit dieses letzten Schrittes wird vor allem durch die Laufzeit des *mutual weight* bestimmt. Diese ist auch wiederum exponentiell. Vernachlässigt man die Laufzeit des *mutual weight*, liegt eine Laufzeit von $O(2nm)$ vor, da im schlimmsten Fall jeder Knoten n_i ($n = \#n_i$) eines Netzes A mit jedem Knoten m_j ($m = \#m_j$) des zu verschmelzenden Netzes B für jeweils beide Kantenrichtungen getestet werden muss. Des Weiteren ist dieser Schritt von der Laufzeit der MDL-Berechnung und von dem verwendeten Lernverfahren für die Aktualisierung der CPT abhängig. Beide können im schlimmsten Fall wiederum exponentielle Laufzeiten besitzen. Daraus folgt eine Komplexität von $O(n \cdot m \cdot C)$. Die Komplexität des gesamten Verschmelzungsverfahrens ergibt sich aus der Komplexität der drei Schritte:

$$O(n+m \cdot C) + O(C) + O(n \cdot m \cdot C) = O(nmC)$$

Angenommen $m \leq n$ dann folgt, dass die Gesamtkomplexität $O(n^2C)$ beträgt. Im anderen Falle gilt eine obere Schranke von $O(m^2C)$.

3.5 Verwandte Modelle

In diesem Abschnitt sollen zu Bayes'schen Netzen (BN) verwandte bzw. alternative Wissensrepräsentationsformalismen vorgestellt und diskutiert werden. Bevor der Entschluss, BN für die Verhaltensnetze zu verwenden, gefallen ist, wurden diese Verfahren in einem Auswahlverfahren näher analysiert.

3.5.1 Anforderungen

Gesucht war ein Modell, das sowohl für die Analyse, als auch für Simulationen und auch für Prognosen geeignet ist. In der Vergangenheit hatte sich gezeigt, dass viele klassische Data-Mining-Projekte nicht den gewünschten Erfolg gebracht haben, da die Verfahren schlecht für Prognose geeignet waren und somit nur schwer in wertschöpfende Entscheidungen umgesetzt werden konnten. Insbesondere sollte unser Modell auch für die klassischen Data-Mining-Verfahren Clustering, Klassifikation und Ähnlichkeitsanalyse geeignet sein (siehe dazu [Schwaiger06]). Dies ist ebenfalls mit den Verhaltensnetzen realisiert und darüber hinaus im

Vergleich zu den klassischen Verfahren stark verbessert worden. Die konkreten Anforderungen sind:

- **quantitativ**: Das Modell soll exakte quantifizierte Aussagen generieren können. Reine qualitative Ansätze wie z. B. Qualitative probabilistische Netze [RusNor95] können lediglich Tendenzen angeben und keine konkreten Werte.
- **multivariat**: Zu simulierende Zielvariablen sollen von beliebig vielen Einflussfaktoren abhängig sein. Univariate Verfahren betrachten dagegen nur eine einzige Zeitreihe/Kurve.
- **nicht linear**: Alle Abhängigkeiten zwischen Variablen sollen nicht linear modellierbar sein. Die Einschränkung auf lineare Modelle ist in der Praxis nicht anwendbar, da in den seltensten Fällen lineare Prozesse vorliegen.
- **beliebige Simulationszeiträume**: Das Modell soll sowohl sehr kurze, bis hin zu einzelnen Ereignissen simulieren können, als auch sehr langfristige Simulationen durchführen können.
- **mehrere Zielvariablen**: Es sollen gleichzeitig mehrere Zielvariablen in einem Modell simulierbar sein, da die Komplexität der Anwendungsfälle nicht durch eine einzelne Variable ausgedrückt werden kann (Mehr-Gleichungssystem).
- **kontinuierliche & diskrete Variablen**: Das Modell soll sowohl diskrete, als auch kontinuierliche Variablen als Input und auch als Output verarbeiten können.
- **beliebige Bedarfsschwankungen**: Man unterscheidet drei Arten von Bedarfsschwankungen: 1. saisonale, 2. sporadische z. B. durch Aktionen und 3. Trends. Für exakte Prognosen müssen alle drei Arten durch ein Modell bewältigt werden können.
- **interpretierbar**: Ein Modell sollte zu jedem Zeitpunkt intuitiv und interpretierbar bleiben. Das erhöht die Wartbarkeit, sichert die Korrektheit und schafft Vertrauen beim Anwender. Außerdem ist es eine zwingende Voraussetzung für den nächsten Punkt:
- **Domänenwissen**: Bei den Anwendern vorhandenes Expertenwissen sollte problemlos in die Modelle einfließen können, um die Ergebnisse sukzessive verbessern zu können.
- **maschinell lernbar**: Trotzdem ist es bei der Modellierung von komplexen Domänen unumgänglich, dass das Modell mittels Data Mining automatisch aus Rohdaten gewonnen werden kann. Eine reine manuelle Erstellung von Modellen ist bei komplexeren Zusammenhängen unpraktikabel und kann nicht zu korrekten quantifizierten Ergebnissen führen. Insbesondere sollten Abhängigkeiten zwischen Variablen automatisch erkannt und quantifiziert modelliert werden. Idealerweise drückt diese Abhängigkeit eine Kausalitätsbeziehung aus.

- **adaptierbar**: Wird ein Modell aus vielen Rohdaten gelernt, ist es häufig unpraktikabel dieses bei jedem neu hinzukommenden Datensatz vollständig neu zu lernen. Deshalb sollte ein Modell an neue Datensätze adaptierbar sein. Ein weiterer Pluspunkt ist häufig, dass man neues Wissen stärker gewichten kann als altes Wissen und dadurch die aktuelle Entwicklung der Welt exakter im Modell wiedergegeben wird. Teilweise ist es sogar erwünscht altes Wissen vollständig zu „vergessen“, um den Umfang des Modells gleich bleibend zu halten.
- **simulierbar**: Ziel ist es, ein Modell zu finden, mit dem nicht nur einfache Prognosen möglich sind, sondern beliebige Was-wäre-wenn-Szenarien simuliert werden können.
- **beliebige Zeitbasis**: Die Größe der diskreten Simulationsschritte sollte frei wählbar sein, um beispielsweise tageweise, wochenweise oder monatsweise Simulationen zu ermöglichen.
- **diagnostisch**: Da das Modell auch für Analyse der Vergangenheit verwendet werden soll, muss dieses nicht nur prognostische, sondern auch diagnostische Anfragen beantworten können. D. h. ermitteln, warum eine bestimmte Situation eingetreten ist und damit auch, unter welchen Umständen eine solche in Zukunft noch einmal eintreten könnte.
- **Data-Mining-fähig**: Wie bereits oben erwähnt, soll das Modell auch für die klassischen Data-Mining-Anwendungen Clustering, Klassifikation und Ähnlichkeitsanalyse geeignet sein.
- **verteilbar**: Sehr komplexe Simulationsaufträge sollten auf ein Rechner-Grid verteilbar sein.
- **holonisierbar**: Die auf Multiagentenebene bewährte Technik der Holonenbildung muss auf die Wissensbasen der Agenten, in diesem Fall die Verhaltennetze, abbildbar sein, um effizient die Interdependenzen zwischen Agenten modellieren zu können.

3.5.2 Graphical Belief Models / Dempster-Shafer-Theorie

Graphical Belief Models (GBM) wurden parallel zu *Bayes'schen Netzen* (BN) entwickelt und basieren im Gegensatz dazu auf der *Dempster-Shafer-Theorie* (DST) und nicht auf dem klassischen Wahrscheinlichkeitskalkül (Kapitel 3.4.3). Eine ausführliche Darstellung von GBM und einen Vergleich mit BN findet man in [Almond95].

Die DST ist ein alternativer Formalismus zur Behandlung von Unsicherheit. Im Gegensatz zu BN die auf dem Bayes-Theorem passieren, basiert die DST auf der Modellierung von Unwissenheit, d.h., statt der Wahrscheinlichkeitsfunktionen werden die von [Dempster66] eingeführten BELIEF-Funktionen benutzt, die die Unsicherheit über den

Wahrheitsgehalt einer Aussage repräsentieren. Die DST unterscheidet zwischen der Wahrscheinlichkeit P(A) eines Ereignisses A und dem Wissen über dieses Ereignis dem BELIEF BEL(A). GBM sind das Dempster-Shafer-Pendant zu BN und werden ebenfalls durch einen Graphen repräsentiert, dessen Struktur Unabhängigkeitsannahmen darstellt. Statt bedingte Wahrscheinlichkeiten $P(V_i|Pa_1,..,Pa_j)$ mit einer Apriori-Verteilung P_i werden BELIEF-Funktionen $BEL(V_i|Pa_1,..,Pa_j)$ definiert.

Die DST lässt sich am besten anhand eines Beispiels veranschaulichen:

Sei A die Aussage „Artikel X ist ein Hochpreisartikel“, dann gilt im klassischen Wahrscheinlichkeitskalkül $P(A) + P(\neg A) = 1$. Angenommen ein Kunde K kennt diesen Artikel X aber gar nicht, dann folgt:

- Es wäre falsch zu behaupten K glaubt an diese Aussage $P(A) = 1$.
- Es wäre auch falsch zu behaupten K glaubt nicht an diese Aussage $P(\neg A)$.
- Korrekt wäre $P(A) = 0 = P(\neg A)$.

Die Basisidee der DST ist, dass man zwischen dem Glauben BEL(A) an eine Aussage A und ihrer Plausibilität PL(A) unterscheidet. Der BELIEF BEL(A) ist definiert als die Summe der Wahrscheinlichkeiten aller Evidenzen e, die A implizieren:

$$BEL(A) = \sum_{\{B \subseteq A\}} e(B)$$ „Summe der Ereignisse, die dafür sprechen“

BEL(A) kann auch als untere Schranke der Wahrscheinlichkeit, dass der Ausgang eines Experiments in A liegt, angesehen werden. Die Plausibilität PL(A) gibt die Wahrscheinlichkeit an, dass A nicht widerlegt wird und entspricht somit einer oberen Schranke der Wahrscheinlichkeit, dass ein Ereignis in A liegt.

$PL(A) = 1 - BEL(\neg A)$ „Summe der Ereignisse, die nicht dagegen sprechen“

Auch für diese Art der Repräsentation von Wahrscheinlichkeit wurden Axiome – ähnlich den Kolmogorov-Axiomen aus Kapitel 3.4.3.1 – formuliert [Almond95]. Zu der *Dempster-Shafer-Theorie* gehören Operatoren, die es erlauben mit den BELIEF-Funktionen auf ähnliche Weise zu arbeiten wie mit Standardwahrscheinlichkeiten und die sich zu einem Propagierungsalgorithmus für *Graphical Belief Models* verallgemeinern lassen.

Da in SimMarket die Modelle aus empirischen Daten gewonnen werden, also objektive Wahrscheinlichkeiten gewonnen werden, ist die Unterscheidung von Belief-Wahrscheinlichkeiten für uns nicht relevant und wir haben uns für Bayes’sche Netze entschieden.

3.5.3 (Künstliche) Neuronale Netze

(*Künstlichen) Neuronalen Netzen (kurz KNN)* [RusNor03] sind von biologischen neuronalen Netzen inspiriert, stellen aber in der Regel keine reale Simulation dieser dar. Neuronale Netze sind eine Realisierung des *Konnektionismus*, der im Gegensatz zum *Konstruktivismus* ein System als Wechselwirkung vieler vernetzter, einfacher Einheiten versteht. Der Konstruktivismus beruht auf der Hypothese, dass Systeme durch schrittweises Zerlegen algorithmisierbar und vollständig symbolisch beschreibbar sind. Beim Konnektionismus entsteht Wissen aus der Interaktion einer großen Anzahl von Einheiten z. B. Neuronen. Daher ist häufig – wie bei Neuronalen Netzen – die interne Funktionsweise eines konnektionistischen Modellsystems nicht nachvollziehbar.

KNN können nicht lineare Funktionen aus einer vorliegenden Datenmenge lernen und sind – kurz gesagt – universelle Funktionsapproximierer, die überall dort eingesetzt werden können, wo abstrakte Muster (engl. *pattern matching*) in Datenmengen erkannt werden sollen z. B. Bild-, Gesicht- und Schrifterkennung, Sprachgenerierung und Klangerzeugung, Data Mining und Zeitreihenanalyse (z. B. Wetter, Aktien etc.).

In den letzten Jahren ist das Gebiet der Neuronalen Netze stark gewachsen und hat zu einer unüberschaubaren Menge unterschiedlicher Arten von Netzen geführt. Diese Tatsache zusammen mit den folgenden weiteren Einschränkungen, die für die meisten Neuronalen Netzarten gelten, haben dazu geführt, dass wir uns gegen Neuronale Netze entschieden haben.

1. Das Trainieren von KNN ist in der Regel ein hoch dimensionaler, nicht linearer Optimierungsprozess. Dabei steht man oft vor dem Problem, dass lokale von globalen Optima nicht unterschieden werden können und somit sehr lange Laufzeiten nötig sind, um optimale Lösungen zu finden. Mit wachsender Anzahl der Einflussfaktoren, Ausgabevariablen und Neuronen in den Zwischenschichten wachsen die Laufzeiten vieler Lern- und Inferenzverfahren exponentiell an.
2. Die Auswahl der passenden Netzart und die Modellierung der Netztopologie ist sehr aufwendig und erfordert sehr viel Erfahrung auf diesem Gebiet, zumal jedes Problem andere Strukturen, Lernverfahren, Bewertungsfunktionen und passende Training- und Testdaten erfordert.
3. KNN neigen dazu die Trainingsdaten zu exakt zu repräsentieren (Overfitting). Infolgedessen liefern die Netze, angewandt auf neue Daten, keine zufrieden stellenden Ergebnisse. Allgemein gültige Strategien, um eine ausreichende Generalisierung der Netze zu erreichen existieren nicht.
4. Domänenwissen über Kausalitäten, Korrelationen, bedingte (Un-)Abhängigkeiten und anderes Vorwissen kann nicht ohne weiteres in KNNs integriert werden.
5. Ist ein KNN einmal gelernt, ist die interne Funktionsweise nicht mehr nachvollziehbar und die Ergebnisse schlecht interpretierbar (Blackbox-Verhalten).

Dadurch fehlt oft das Vertrauen der Anwender in ein System, dass KNNs verwendet.

3.5.4 Markov-Modelle

Am bekanntesten und für praktische Anwendungen am relevantesten sind die so genannten *Hidden-Markov-Modelle* (kurz *HMM*), die ihren Ursprung in diskreten *Markovketten* haben.

3.5.4.1 Markovketten

Bei Markovketten handelt es sich um stochastische Prozesse, die die folgende Eigenschaft besitzen: Ist der jetzige Augenblick eines Prozesses bekannt, dann lassen sich Prognosen über den zukünftigen Verlauf des Prozesses nicht durch zusätzliche Daten der Vergangenheit verbessern. Die formale Definition einer Markovkette n-ter Ordnung lautet:

$$P(X_{t+1} = x_{t+1} \mid X_t = x_t, X_{t-1} = x_{t-1,\ldots}, X_0 = x_0) = P(X_{t+1} = x_{t+1} \mid X_t = x_{t,\ldots}, X_{t-n+1} = x_{t-n+1})$$

Diese Eigenschaft nennt man auch die Gedächtnislosigkeit oder *Markov-Eigenschaft*. In den meisten praktischen Anwendungen werden nur Markovketten 1er-Ordnung verwendet, die immer eine Anfangverteilung *u* und Übergangswahrscheinlichkeiten besitzen, deren Größe von der Anzahl der Zustände des Zustandsraumes abhängen. Die Übergangswahrscheinlichkeiten werden durch die folgende Matrix beschrieben:

$$p_{ij}(t) := P(X_{t+1} = j \mid X_t = i);\ i,j = 1,\ldots,m$$

Sind die Wahrscheinlichkeiten unabhängig vom Zeitpunkt t ($p_{ij} = p_{ij}(t)\ \forall t$), so spricht man von *homogenen* Markovketten. Bei homogenen Markovketten erster Ordnung kann die Wahrscheinlichkeit, in n Schritten vom Zustand i in den Zustand j überzugehen, mit der n-ten Potenz der Übergangsmatrix berechnet werden:

$$p_{ij}^{n} := M^{n}(i, j)$$

Eine Markovkette ist *stationär*, wenn die Wahrscheinlichkeit, sich in einem bestimmten Zustand zu befinden, unabhängig von t ist. Das bedeutet formal:

$$\pi_j^* := \lim_{n \to \infty} u(i) p_{ij}(n)$$

Die stationären Zustände stellen sicher, dass sich das System nach einiger Zeit einpendelt.

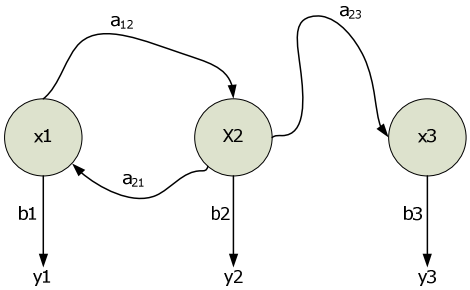
Da bei Markovketten jeder Zustand des Models mit einem deterministisch-beobachtbaren Ereignis korrespondiert, können diese Modelle nur bereits beobachtete Werte annehmen. Dies ist für die meisten Anwendungen zu restriktiv.

3.5.4.2 Hidden-Markov-Modelle

Hidden-Markov-Modelle sind temporale probabilistische Modelle, die häufig für die Modellierung von Zeitreihen verwendet werden, so z. B. für die Bild-, Sprach- und Handschriftenerkennung. HMM werden durch zwei Zufallsprozesse beschrieben, wobei es sich beim ersten Prozess um eine – wie oben definiert – Markovkette mit Zuständen $X = x_1,\ldots,x_n$ und Übergangswahrscheinlichkeiten $A = \{a_{ij}\}$ handelt, deren Zustände allerdings von außen nicht direkt sichtbar (*hidden*) sind. Der zweite Zufallsprozess erzeugt stattdessen für jeden Zeitpunkt der Kette beobachtbare Ausgangssymbole $Y = y_1,\ldots,y_m$ auf der Basis einer Wahrscheinlichkeitsverteilung $B = \{b_{ij}\}$ (Emissionswahrscheinlichkeit). Die versteckten Zustände beschreiben den wahren Zustand des modellierten Systems, wobei die Anzahl der beobachtbaren Zustände abweichend von der Anzahl versteckter Zustände sein kann. Ein HMM benötigt die Menge der Anfangswahrscheinlichkeiten $\pi = \pi_1,\ldots,\pi_n$, wobei π_i die Wahrscheinlichkeit angibt im ersten Zeitschritt im Zustand x_i zu sein. Da die verstecken Zustände eine Markovkette bilden, erfüllen diese folglich die Markov-Eigenschaft (meistens erster Ordnung). Daraus folgt, dass die gemeinsame Wahrscheinlichkeitsverteilung einer Sequenz von Zuständen X und Beobachtungen O folgendermaßen berechnet werden kann:

$$P(x_1,...,x_t,o_1,...,o_t) = P(x_1)P(o_1 \mid x_1)\prod_{t=2}^{T} P(x_t \mid x_{t-1})P(o_t \mid x_t)$$

In Kurzform lautet die formale Definition eines HMM $\lambda = (A, B, \pi)$.



Es bedeuten:

- x - (verborgene) Zustände des Markovmodells
- a - Übergangswahrscheinlichkeiten
- b - Emissionswahrscheinlichkeiten
- y - (sichtbare) Ausgabesymbole

Abbildung 23: Hidden-Markov-Modell

Drei grundlegende Problemstellungen sind typisch:

1. Es existiert ein vorgegebenes HMM und man möchte die Wahrscheinlichkeit einer speziellen Ausgabesequenz bestimmen. Dies kann mit dem *Forward-Algorithmus* gelöst werden.
2. Es ist ein vollständiges HMM gegeben und man möchte die wahrscheinlichste Sequenz der versteckten Zustände für eine gegebene Ausgabesequenz bestimmen. Lösbar mit dem *Viterbi-Algorithmus*.
3. Gegen ist nur die Ausgabesequenz und man möchte die wahrscheinlichsten Parameter eines HMM bestimmen, die zu der Ausgabesequenz geführt haben. Dieses Problem ist mit dem *Baum-Welch-Algorithmus* lösbar.

Der Vorteil eines HMM ist, dass alle Operationen mit einfachen Matrixberechnungen realisierbar sind. Sie finden daher vor allem in Echtzeitanwendungen wie z. B. Spracherkennung Anwendung. Der Nachteil von HMM ist, dass ein System durch eine Variable X mit n Zuständen repräsentiert wird, dass heißt, dass alle Zustände und vor allem alle Zustandskombinationen in diese eine Variable kodiert werden müssen. Bei komplexen Systemen werden die benötigten Matrizen rasch sehr groß.

Jedes HMM lässt sich durch ein dynamisches Bayes'sches Netz (DBN) mit einer Zustandsvariablen und einer Evidenzvariablen repräsentieren. Genauso kann jedes DBN durch ein HMM realisiert werden. Dazu müssen allerdings alle Variablen des DBN in eine einzige kodiert werden. Folglich wachsen die korrespondierenden Matrizen exponentiell mit der Anzahl an Variablen und deren Anzahl an Zuständen. DBN/BN haben gegenüber HMM den Vorteil, dass sie „sparse" sind, also Lücken im (temporalen) probabilistischen Modell ausnutzen und nur bedingt unabhängige Beziehungen darstellen.

3.5.5 Kalmanfilter

Kalmanfilter (siehe [RusNor03]) sind stochastische Zustandsschätzer für dynamische Systeme, die dafür entwickelt worden sind den Zustand (z. B. Position und Geschwindigkeit) eines physikalischen Systems auf der Basis von teils redundanten und verrauschten Messungen zu schätzen. Wie bei der Regressionsanalyse geht es darum, den mittleren quadratischen Fehler zu minimieren, was als Inferenz in temporalen probabilistischen Modellen formuliert wird, bei denen das Übergangsmodell die physikalische Bewegung des gemessenen Systems und das Sensormodell den Messprozess darstellen. Der Vorteil von Kalmanfilter ist deren iterative Berechenbarkeit, die sie für Echtzeitanwendungen prädestiniert. Kontinuierlich wird der vorhergesagte Wert mit dem tatsächlich nachträglich gemessenen Ausgangswert verglichen und die Differenz linear gewichtet für die Verbesserung des Modells genutzt (Prädiktor-Korrektor-Struktur).

Im Laufe der Zeit wurden zahlreiche Varianten der Kalmanfilter entwickelt. Die Wichtigsten sind dabei: *Erweitere Kalmanfilter (EKF), Switching Kalmanfilter (SKF), Sigmapunkt-Kalmanfilter (SPKF), unscented Kalmanfilter (UKF), central difference Kalmanfilter (CDKF), Kalman-Bucy-Filter (KBF)* und *Partikelfilter*.

Kalmanfilter gehören zu den verbreitetsten Algorithmen für die Zustandsschätzung von linearen und nicht linearen Systemen. Sie werden unter anderem für die Flugzeugnavigation, für Tracking-Systeme im automobilen oder militärischen Bereich und für PLL-Filter in Radios, Funkgeräten, Computern etc. verwendet.

Kalmanfilter können ebenfalls wie HMM durch dynamische Bayes'sche Netze repräsentiert werden. Die entsprechenden DBN benötigen kontinuierliche Variablen mit bedingten linearen Gauß'schen Verteilungen. Umgekehrt kann jedoch nicht jedes DBN durch einen Kalmanfilter dargestellt werden, da die aktuelle Zustandsverteilung bei Kalmanfiltern immer eine einzelne multivariate Gaußverteilung ist. DBN können dagegen beliebige Verteilungen repräsentieren. Hochgradig nichtlineare KF können dies zwar auch, aber im Gegensatz zu DBN wird die Handhabung und Realisierung von Kalmanfiltern umso aufwändiger, je nichtlinearer ein System ist. DBN kommen dagegen mit beliebiger „Nichtlinearität" bei gleich bleibendem Aufwand zurecht.

3.5.6 Fazit

Das Ergebnis der Evaluierung dieser Kerntechnologien, die für die Wissensbasen der Agenten infrage kommen könnten, war, dass Bayes'sche Netze (BN) die meisten der oben aufgelisteten Anforderungen erfüllen. So existieren für BN maschinelle Lernverfahren, sie können gleichzeitig mehrere Zielvariablen mit nicht linearen Abhängigkeiten modellieren und sind im Gegensatz zu Neuronalen Netzen gut interpretierbar und können sowohl prognostische als auch diagnostische Anfragen verarbeiten. Zusätzlich existieren Adaptionsverfahren, um die Netze beim Eintreffen von neuen Datensätzen zu aktualisieren.

Da auch Bayes'sche Netze nicht alle Anforderungen zufrieden stellend erfüllen, müssen diese in einigen Punkten weiter entwickelt werden. Die Integration von Domänenwissen und semantischen Knotenannotation ist mit herkömmlichen BN nicht ohne weiteres möglich. Zudem müssen die Netze um Holonisierung und Simulationsfähigkeit erweitert werden.

4 Modellierung und Simulation von Artikeln

4.1 Einleitung

In den vorangegangenen Kapiteln wurden die theoretischen und technologischen Grundlagen vorgestellt, die die Basis des SimMarket-Systems bilden. Wir haben insbesondere gesehen, wie aus sehr großen Datenmengen, die in Datenbanken oder Data Warehouses gespeichert sind, automatisiert probabilistische holonische Agenten generiert werden und diese mit dem Multiagenten-Grid MASg für verschiedene Domänen verwendet werden können. Im vorliegenden Kapitel soll nun gezeigt werden, wie diese Techniken im SimMarket-System eingesetzt worden sind und an welcher Stelle die bekannten Verfahren von mir weiterentwickelt wurden.

Das erklärte Ziel des SimMarket-Projektes ist u. a. die Entwicklung eines neuartigen *Category-Management-Systems* mit dem Sortimentsverantwortliche ihre Entscheidungsprozesse modellieren und optimieren können. Die Aufgaben eines Sortimentsverantwortlichen reichen dabei vom Festlegen von Preisen und Werbeaktionen, über die Platzierungsoptimierung, bis hin zum Ein- und Auslisten von Artikeln. Diese Prozesse sollen durch das im Folgenden beschriebene Artikelsimulationssystem unterstützt werden. Dieses System ist in der Lage auf der Basis von Realdaten die Auswirkungen von Preisänderungen und Aktionen auf den Umsatz und den Ertrag eines Unternehmens vorab zu simulieren, um den Händler in seinen Entscheidungsprozessen zu unterstützen.

Die Verwendung des Simulationssystems soll folgendermaßen ablaufen. Zunächst bekommt der Sortimentsverantwortliche einen detaillierten Überblick über die Ist-Situation seines Sortiments. Die Erfassung des Ist-Zustandes wird durch zahlreiche Analysefunktionen, die Kennzahlen berechnen und *Reports* erstellen, unterstützt. In einer speziellen Ansicht kann der Benutzer mit Hilfe dieser Informationen neue zukünftige Szenarien für beliebige Artikel definieren. Diese Szenarien beinhalten seine möglichen Entscheidungen bezüglich des Sortiments für die Zukunft. Für jeden selektierten Artikel wählt er den Zeitraum der Aktion und die Art der Aktion aus. Darüber hinaus kann er unabhängig von Promotionen Preisänderungen für Artikel festlegen. Auf diese Weise kann er beliebig viele alternative Szenarien definieren und anschließend simulieren. Die simulierten Auswirkungen werden vom System aggregiert, (vor-)interpretiert und dem Benutzer anschließend präsentiert. Die simulierten Szenarien können hinsichtlich einer gewählten Zielgröße wie Umsatz oder Ertrag bewertet und sortiert ausgegeben werden. Der Sortimentsverantwortliche kann nun den beschriebenen Prozess bei Bedarf wiederholen, um seine Entscheidungen zu optimieren.

4.2 Gesamtarchitektur von SimMarket

Um die Gesamtarchitektur von SimMarket zu beschreiben, soll das System aus den drei Sichten agentenorientiert, objektorientiert und serviceorientiert betrachtet werden.

4.2.1 Agentenorientierte Sicht

Jede messbare Entität eines Supermarktes wird durch einen Agenten im System repräsentiert und je nach Eigenschaft wird diese als vollständiger probabilistischer holonischer Agent (PH-Agent) oder nur mit Teilen der Funktionalität modelliert. Alle Agenten werden mit Hilfe des Multiagenten-Grid MASg erzeugt und verwaltet. Die folgenden Agenten sind realisiert worden:

- *Artikelagenten*: Die Artikelagenten umfassen sowohl das reine Faktenwissen über die Artikel wie z. B. Verkaufspreise, Hersteller, Warengruppe etc. als auch aus Abverkaufsdaten extrahiertes strukturelles probabilistisches Wissen.
- *Warengruppenagenten*: Mit Hilfe von Warengruppenagenten bildet man die Warengruppenhierarchie eines Supermarktes auf die Agentenpopulation ab. Warengruppenagenten kennen ihre Unteragenten d. h., ihre untergeordneten Warengruppen oder Artikel und können sowohl Warengruppen- als auch Artikelagenten als Subagenten besitzen. Dabei kann ein Agent zu mehreren Warengruppenagenten gehören. Der Warengruppenagent kapselt Funktionalitäten, die auf Mengen von Agenten angewendet werden müssen z.B. Korrelationsanalysen und leitet Simulationensaufträge an die Subagenten weiter.
- *Kunden- und Kundengruppenagenten*: Alle identifizierbaren Kunden eines Marktes werden ebenfalls als Agenten modelliert. Dadurch lassen sich differenziertere Simulationen mit einzelnen Kundengruppen durchführen (siehe [Schwaiger06]).
- *Metaagenten*: Metaagenten kapseln für jeden Anwender des Systems alle Agenten und Gruppenagenten einer Kategorie. Der Artikelmetaagent dient als Einstieg in die Agentenhierarchie, um auf einen Artikelagenten zuzugreifen. Die Metaagenten sind die einzigen Agenten im System, die neue Agenten erzeugen dürfen. Dadurch besitzt jede Kategorie von Agenten einen Single-Point-Of-Creation. Dadurch wird vermieden, dass ein Agent mehrfach im System existieren kann. Dies ist insbesondere wichtig, da ein Agent zu mehreren Gruppenagenten gehören darf. Will eine Instanz auf einen Agenten über einen Gruppenagenten zugreifen, gibt der Gruppenagenten die entsprechende Referenz zurück, wenn er diese bereits besitzt. Anderenfalls fordert er diese beim Metaagenten an. Der Metaagent wiederum schaut in seinem globalen Verzeichnis nach, ob der Agent bereits existiert und liefert die Referenz zurück. Es besteht nämlich die Möglichkeit, dass der Agent

bereits durch die Verwendung in einem anderen Gruppenagent erzeugt worden ist. Existiert der Agent noch nicht, erzeugt der Metaagent diesen, trägt ihn in das globale Verzeichnis ein und gibt die Referenz an den Antragssteller zurück.

- *SimProgAgent*: Der SimProgAgent (Simulations- und Prognose-Agent) initiiert und kontrolliert Simulationen und wertet diese zu Prognosen und komplexeren *Reports* aus. Im SimProgAgent werden Simulationsaufträge verwaltet, die wiederum aus mehreren Szenarien bestehen können. Diese Aufträge werden an die zuständigen Agenten verteilt, die durch Simulation auf die definierten Szenarien mit ihrem individuellen Verhalten reagieren. Der SimProgAgent sammelt die Ergebnisse der an der Simulation beteiligten Agenten und aggregiert diese zu einem *Report*. Des Weiteren hat der SimProgAgent ein Evaluationsframework, das automatisiert aus einer Datenbasis Szenarien generiert und diese den Agenten zur Simulation übergibt. Die Agenten erhalten dagegen keine Informationen über den Zeitraum der generierten Szenarien, sondern dürfen nur auf Daten bis zum Beginn des ersten Testszenarios zugreifen. Die Ergebnisse der Simulationen werden anschließend vom Evaluationsframework mit den realen Ergebnissen verglichen und Gütemaße (siehe Kapitel 2.4.3) berechnet.
- *Kassenbon(gruppen)agenten*: Die Kassenbonagenten modellieren neben Kunden- und Artikelagenten eine weitere Sicht auf die Transaktionsdaten, die in SimMarket die zentrale Faktentabelle darstellt. Die Kassenbonagenten kapseln alle Funktionen und Analysen, die rein auf Kassenbons durchgeführt werden könnten, z. B. Warenkorbanalysen mit Assoziationsregeln. Kassenbonagenten können genauso wie Kunden- und Artikelagenten klassifiziert und geclustert werden. Die resultierenden Gruppen werden durch Kassenbongruppenagenten modelliert.
- *Umwelt- und Eventagenten*: Die Umwelt- und Eventagenten repräsentieren die externen, nicht beeinflussbaren Einflussfaktoren, die in irgendeiner Weise auf die zu simulierenden Ereignisse wirken. Diese Agenten können beispielsweise das Wetter repräsentieren oder die ökonomische Lage des Landes, die Kaufkraft der Bevölkerung, die Lage an den Weltbörsen, Feiertage, Saisons, nationale Ereignisse, wie Winter-/Sommer-Schlussverkauf etc.

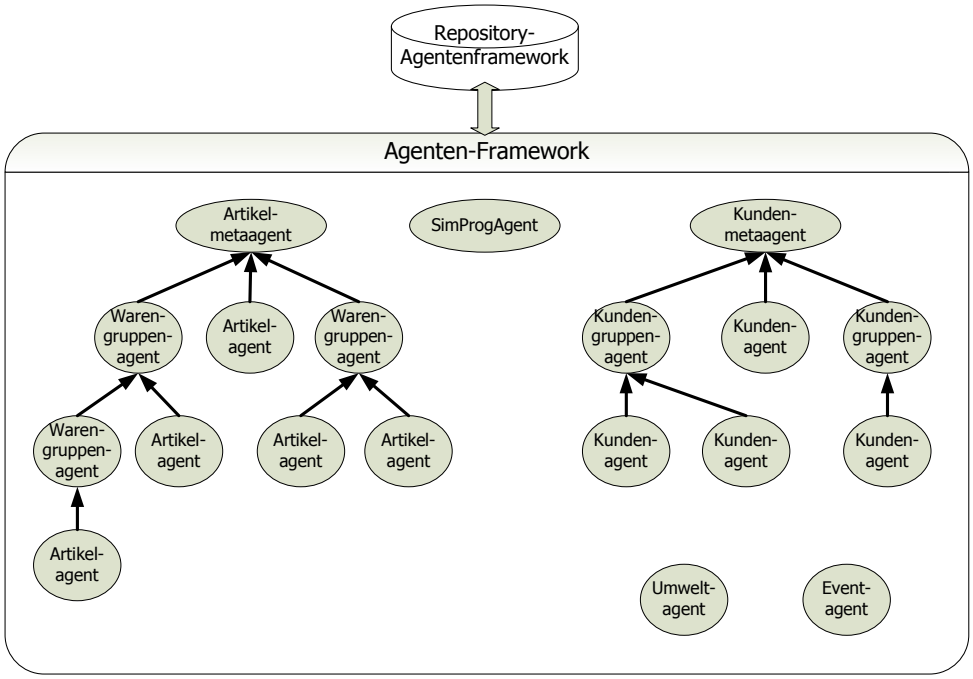


Abbildung 24: Die SimMarket-Agentenumgebung

Alle Agenten melden sich nach ihrer Initiierung bei ihrem zuständigen MASg-Framework an und werden zusätzlich im MASg-Repository registriert. Möchte ein GUI-Client beispielsweise auf einen Agenten zugreifen, kann er sich über das MASg-Repository den zuständigen Metaagenten geben lassen und diesen nach dem gewünschten Agenten fragen. Theoretisch kann auf die Agenten auch direkt über das Framework mit Hilfe der global eindeutigen Identifikationsnummer mit der jeder Agent beim Repository registriert ist zugegriffen werden. Der Nachteil bei dieser Vorgehensweise ist, dass der Benutzer nur den gewünschten Agenten erhält, wenn dieser bereist erzeugt worden ist. Das Repository selbst erzeugt keine Agenten. Der Grund dafür ist, dass die Logik für das Erzeugen von Agenten stark domänenabhängig ist und somit nur von dem zugehörigen Metaagenten durchgeführt werden kann. Das MASg-Framework für sich alleine genommen ist vollkommen domänenunabhängig. Domänenspezifische Funktionen werden erst ab der Ebene der Metaagenten implementiert. Da für jeden Benutzer des SimMarket-Systems eigene Metaagenten erzeugt werden, verteilt sich die Last auf alle im System vorhandenen Metaagenten. Ohne diese Agenten würden alle Anfragen direkt über das Framework laufen, was zu einem Engpass führen könnte.

4.2.2 Objektorientierte Sicht

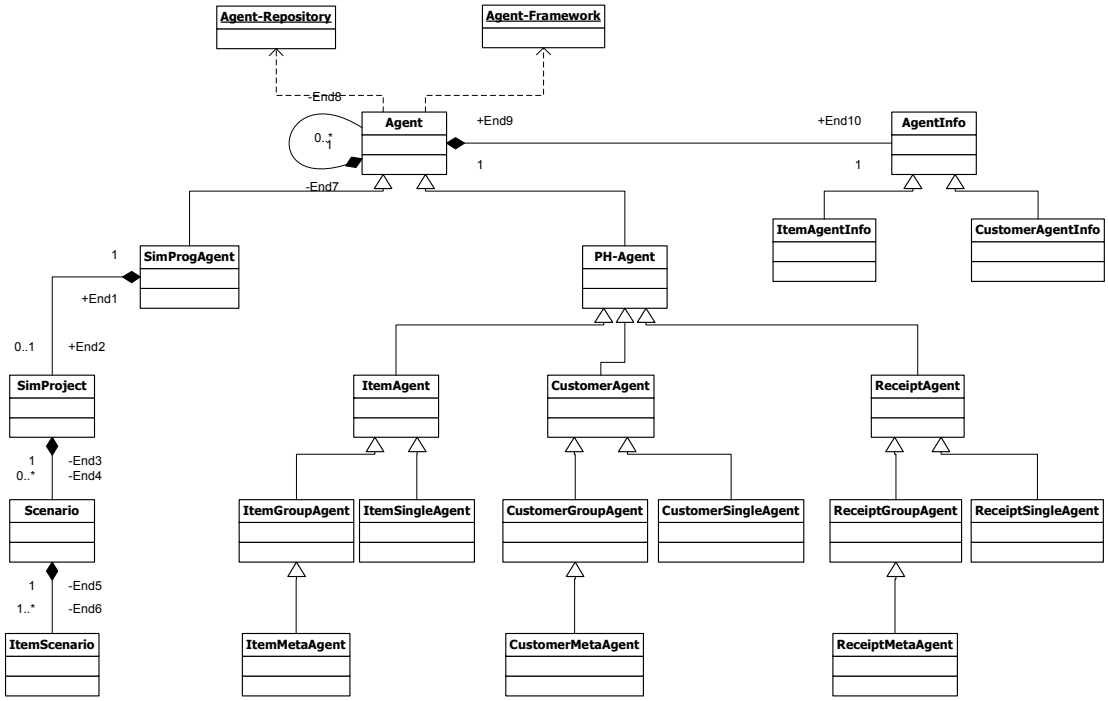


Abbildung 25: Klassenhierarchie der Agenten

Alle Agenten in SimMarket erben von der Klasse *Agent*, die vom Agenten-Framework MASg angeboten wird. *Agent* realisiert alle Basisfunktionen eines Agenten und stellt zusätzlich die Konnektivität zum Agenten-Framework und –Repository her. Durch die rekursive Eigenschaft der Klasse *Agent* ist es möglich, dass sowohl die Gruppenagenten als auch die Einzelagenten von der Klasse *Agent* erben. Die Klasse *PH-Agent* erweitert die Basisklasse *Agent* zu einem probabilistisch-holonischen Agenten (siehe 3.4). Diese Klasse kapselt die probabilistische Wissensbasis mit den Verhaltensnetzen und die Fähigkeit, auf der Basis der Verhaltensnetze Holonen zu bilden (siehe 3.4.3 bis 3.4.7). Darüber hinaus besitzt *PH-Agent* generische Funktionen, um Simulationen (siehe 4.6), Clustering und Klassifizierung (siehe [Schwaiger06]) durchführen zu können. Die Klassen *ItemAgent*, *CustomerAgent* und *ReceiptAgent* kapselt jeweils Funktionen und Attribute, die sowohl von Einzel- als auch Gruppenagenten verwendet werden können, aber auf die Anwendungsgebiete Artikel, Kunden oder Kassenbons beschränkt sind. Da prinzipiell jeder *Agent* die Fähigkeiten eines Metaagenten besitzt, können die spezifischen Metaagenten von ihren jeweiligen Gruppenagenten abgeleitet werden.

Der *SimProgAgent* verwaltet die Projekte der Benutzer des SimMarket-Systems. Ein *Project* besteht aus mehreren *Scenario*-Objekten, die einen zu simulierenden Sachverhalt

repräsentieren. Ein *Scenario* besteht aus beliebig vielen *ItemScenarios*, die eine Konfiguration eines Artikels beschreiben. Eine Konfiguration ist eine Menge von *Action*-Objekten, die beispielsweise eine Preisänderung oder eine Bewerbung darstellen. Für jeden zu simulierenden Artikel darf ein *Scenario* nur ein zugehöriges *ItemScenario* enthalten. Möchte ein Benutzer alternative *ItemScenarios* simulieren, so wird ein weiteres *Scenario*-Objekt erzeugt. Mit Hilfe der *Scenario*-Objekte können verschiedene Artikel und Situationen, aber auch alternative Konfigurationen bzgl. eines Artikels simuliert werden. Ein Benutzer kann mehrere *Projects* anlegen und in der SimMarket-Datenbank speichern, um diese zu einem späteren Zeitpunkt wieder verwenden zu können. Es kann aber nur eine Instanz eines *Project*-Objektes gleichzeitig in SimMarket geladen werden. *Project*, *Scenario* und *ItemScenario* enthalten unterschiedliche Simulationskonfigurationen wie Lernintervalle, Simulationsintervalle und Art der Simulation. Manche Einstellungen existieren nur auf einer Ebene, andere dagegen auf mehreren. In diesem Fall können Einstellungen von der höheren Ebene geerbt werden, wenn auf einer tieferen Ebene die Einstellungen nicht gesetzt worden sind. Auf allen Ebenen eines Projektes werden zusätzlich ihre jeweiligen Ergebnisse festgehalten, wenn bereits eine Simulation durchgeführt worden ist. Die Ergebnisse werden auf der nächsthöheren Ebene aggregiert und wenn möglich Prognosegütemaße berechnet (siehe 2.4.3). Die Ergebnisse können entweder mit den Szenarien oder getrennt von diesen für weitere Auswertungen in der Datenbank oder in eine Datei im *XML-Format* gespeichert werden. Zusätzlich zur Verwaltung des aktuellen Projektes ist der *SimProgAgent* für die Simulation verantwortlich. Der SimProgAgent simuliert allerdings nicht selbst, sondern delegiert nur die Szenarien zu den jeweiligen Artikel- und Warengruppenagenten, von denen jeder in der Lage ist, eine Simulation über sein eigenes Verhalten durchzuführen. Da die Agenten zudem beliebig über das MASg-Framework verteilbar sind, erhalten wir so eine verteilte Simulationsarchitektur, bei der jeder Agent von sich aus entscheiden kann, z. B. bei Ressourcen-Engpässen, den Rechner zu wechseln. Nach einer Simulation sammelt und aggregiert der SimProgAgent die Ergebnisse und wertet diese aus. Die in Abbildung 25 dargestellten *AgentInfo*-Objekte werden im nächsten Abschnitt erläutert.

4.2.3 Serviceorientierte Sicht

Serviceorientiertes Programmieren ist eine neuere Technik der Softwareentwicklung, die wegen der wachsenden Komplexität heutiger Softwaresysteme entwickelt wurde. Bei komplexen Systemen besteht einerseits die Gefahr, dass einzelne Klassen zu groß werden und anderseits, dass die Anzahl der Klassen unüberschaubar wird.

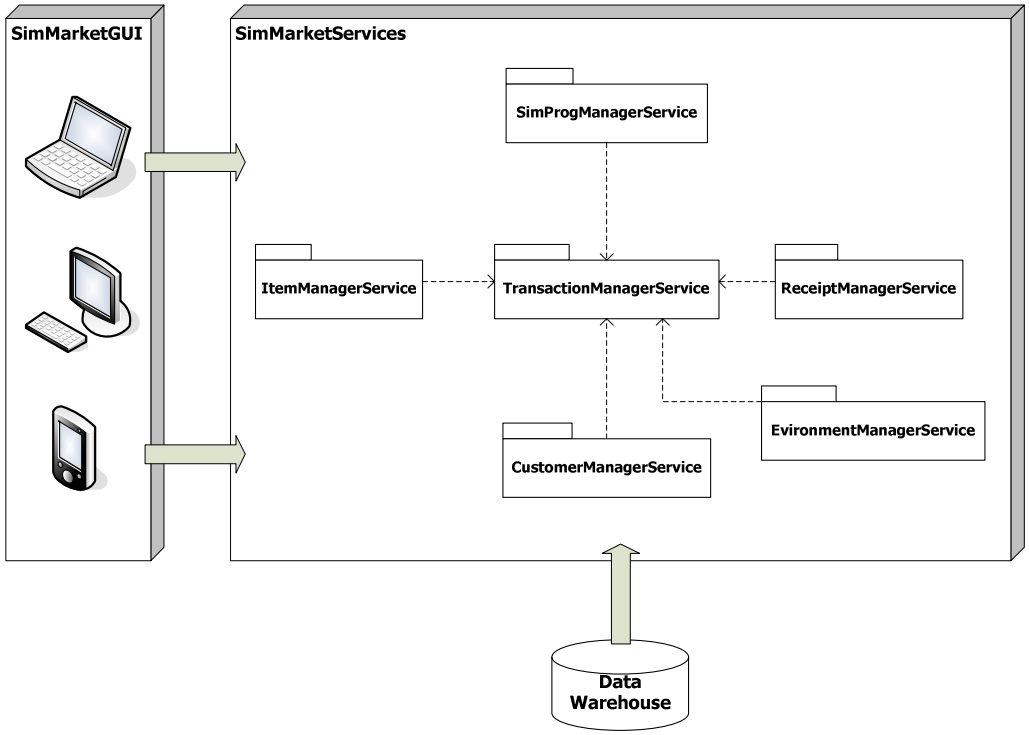


Abbildung 26: Serviceorientierte SimMarket-Architektur

Diese Probleme werden besonders deutlich, wenn verschiedene Systeme miteinander gekoppelt werden sollen. Bei so genannten Legacy-Systemen erfordert die Integration von n Applikationen die Entwicklung von n(n-1) Schnittstellen, die wiederum häufig Änderungen an der Codebasis erfordern. Hinzu kommen immer neuere Anforderungen an die Systeme wie Verteilbarkeit, Sicherheit und Nebenläufigkeit, die nicht ohne weiteres in althergebrachte Systeme integriert werden können. Serviceorientierte Architekturen adressieren diese Probleme, indem sie eine neue Abstraktionsebene in Softwaresysteme einführen.

Multiagentensysteme sind in vielen Punkten mit serviceorientierten Architekturen verwandt, trotzdem können auch bei MAS ähnliche Komplexitätsprobleme auftauchen wie bei herkömmlichen Systemen. Wie wir selbst feststellen mussten, kann die Größe einzelner Agentenklassen Probleme bei der Wartbarkeit und der Wiederverwendbarkeit von Codeteilen bereiten. Ein erster Schritt zu serviceorientierten Architekturen ist die Entwicklung von *Komponenten*. *Komponentenorientiertes Programmieren* steht in der Abstraktionshierarchie zwischen dem objektorientiertem und dem serviceorientiertem Programmieren. Eine Komponente hat bereits sehr viele Eigenschaften eines Agenten bzw. eines Dienstes, ist aber eine Ebene tiefer anzusiedeln.

Beispiele für Agentenkomponenten in SimMarket sind z. B. die Wissensbasis und die Simulationskomponente. Da sich Komponenten nach außen hin wie normale Objekte

verhalten, fehlen ihnen immer noch wichtige Eigenschaften, um die Interoperabilität moderner Systeme zu verbessern. Komponenten können nicht ohne weiteres verteilt und über ein Netzwerk auf sie zugegriffen werden. Serviceorientierte Architekturen dagegen bieten unterschiedlichste Ressourcen anderen Teilnehmern über ein Netzwerk als unabhängige *Dienste* (engl. *services*) an, die – wie im Falle von *Web Services* – über eine herstellerübergreifende Schnittstelle angesprochen werden können. Da diese Dienste zwischen unterschiedlichen Technologien (z. B. .NET und Java) interoperabel sind, sind solche Komponenten sehr gut wieder verwendbar. Da auf Dienste transparent über ein Netzwerk zugegriffen werden kann, sind sie prinzipiell auch verteilbar.

Entwickelt man Dienste mit den gleichen Eigenschaften wie Komponenten, sind Dienste auch miteinander kombinierbar. Dadurch können neue Dienste durch die Komposition von mehreren vorhandenen Diensten erzeugt werden. Der größte Nachteil von Diensten ist deren Zustandslosigkeit, die in einigen Fällen nicht gewünscht bzw. nicht machbar ist. An dieser Stelle können wir nun wieder auf das altbewährte Agentenparadigma zurückreifen. In SimMarket ist ein Agent eine Komposition überschiedlichster Komponenten und Dienste. Jeder Agent wiederum ist selbst ein Dienst der allerdings nicht zwingend zustandslos ist. In SimMarket unterscheiden wir in der serviceorientierten Architektur zwischen externen und internen Diensten. Die externen Dienste dienen vor allem Dritten, die bereits ein eigenes System besitzen und SimMarket als Erweiterung integrieren möchten. Die internen Dienste dienen uns selbst für die Entwicklung neuer Komponenten und für die Anpassung des Systems an einen neuen Kunden. Durch die serviceorientierte Architektur ist es uns möglich durch die Bearbeitung von XML-Konfigurationsdateien ein völlig neues System zusammenzustellen. XML ist die Grundlage der Schnittstellenbeschreibung und der Interaktion aller Dienste in SimMarket. Des Weiteren wird in SimMarket zwischen Remote-Diensten und Nicht-Remote-Diensten unterscheiden. Die Remote-Dienste liegen grundsätzlich auf einem Server und werden ausschließlich durch *CallByReference* angesprochen. .NET bietet mit den *.NET Remote Services* einen einfach zu verwendenden Mechanismus, um auf beliebige Objekte transparent über ein Netzwerk zu zugreifen, als handelt es sich um lokale Objekte. Diese Art von Dienst wird in SimMarket genutzt, wenn wenig Zugriffe auf die Remote-Objekte stattfinden, die angestoßenen Berechnungen allerdings derart rechenintensiv sind, dass sie sich nicht auf dem Client, sondern nur auf einem besonders leistungsfähigen Server ausgeführt werden können. Liegt ein Objekt ohnehin auf dem Server und handelt es sich dabei um ein sehr speicherintensives Objekt, ist es ebenfalls sinnvoll dieses auf dem Server zu belassen und es remote anzusprechen. Gegen die Verwendung von Remote Services spricht die schlechte Performanz eines Zugriffes auf ein solches Objekt (siehe dazu die von mir betreute Diplomarbeit [Schlicker06]). Finden sehr viele Zugriffe auf ein solches Objekt statt, ist es günstiger das komplette Objekt zunächst auf den Client zu kopieren – also per CallByValue aufzurufen – und es anschließend lokal

anzusprechen, als sehr oft remote auf dieses zuzugreifen. In der Praxis hat sich der Mittelweg aus beiden Varianten als der effizienteste herauskristallisiert. Wie in Abbildung 25 zu sehen besitzt jeder Agent ein *AgentInfo*-Objekt, das bei Bedarf seriellisiert und verschickt werden kann. Dieses Objekt kapselt alle wichtigen und vor allem häufig angefragten Informationen des Agenten. Der Agent selbst ist als .NET Remote Service implementiert und verschickt nur das AgentInfo-Objekt CallByValue. Jeder Agent ist also ein verteilbarer Dienst, der von jeder Komponente im SimMarket-Netzwerk angesprochen werden kann. Agenten sind dagegen keine Web Services und bieten somit auch keine Schnittstelle nach außen an. Externe Schnittstellen werden zurzeit erst entwickelt und werden auf einer höheren Ebene angesiedelt sein (siehe dazu Abbildung 26).

Abbildung 26 zeigt die internen Dienste von SimMarket auf der höchsten Ebene. Bei dem verwendeten Schema handelt es sich um eine Sternarchitektur mit dem *TransactionManagerService* als zentralsten Dienst. Der *TransactionManagerService* kapselt alle Funktionen, die direkt und ausschließlich auf die Faktentabelle – im konkreten Fall die Transaktionsdaten der Händler – des Data Warehouse zugreifen. Die anderen Dienste liegen sternförmig um diesen zentralen Dienst und stellen jeweils eine andere Sicht auf die Faktentabelle dar. Damit entspricht die Architektur der Dienste dem Datenschema des zu Grunde liegenden Data Warehouse (siehe Anhang D). *ItemManagerService*, *ReceiptManagerService*, *SimProgManagerService*, *CustomerManagerService* und *EnvironmentManagerService* sind unterschiedliche Dimensionen, die über der Transaktionstabelle und den entsprechenden Dimensionstabellen definiert sind. Beispielsweise erhält man die Abverkaufszahlen eines Artikels, indem man den *ItemManagerService* befragt, der u. a. die Dimension Artikelabverkaufszahlen repräsentiert. Diese Dimension wird dann rekursiv durch die korrespondierenden Warengruppen- und Artikelagenten in Unterdimensionen aufgeteilt, um z. B. die Abverkaufszahlen eines bestimmten Artikels zu erhalten.

4.3 Artikel- und Warengruppenagenten

Jeder Artikel- und Warengruppenagent soll in der Lage sein, das Abverkaufsverhalten der repräsentierten Entität eigenständig mittels Simulation zu ermitteln. Deshalb existiert keine zentralisierte Simulationsroutine, sondern jeder Agent ist in der Lage auf der Basis seiner probabilistischen Wissensbasis Simulationen durchzuführen. Durch die Verwendung des MASg-Frameworks kann jeder individuell simulierende Agent zwecks Load-Balancing auf ein Netzwerk von Rechnern – genauer gesagt auf ein Rechner-Grid – verteilt werden. Darüber hinaus besteht die Möglichkeit die Agenten in einer minimalisierten Form als persistente autonome Prozesse für Überwachungsaufgaben auf dem Grid zu halten. Die Überwachung von zuvor definierten Schwellenwerten ist ein hilfreiches Feature, um bei

Abweichungen die verantwortlichen Mitarbeiter umgehend mit einem Report zu benachrichtigen, automatische Nachbestellungen einzuleiten oder auch Mails/Mahnungen an Zulieferer zu verschicken. Des Weiteren können die Agenten genutzt werden, um im Idle-Betrieb des Rechners nach versteckten Korrelationen im Sortiment zu suchen und wiederkehrende Simulationen und Reports im Voraus zu berechnen. Dabei adaptieren sich die Agenten kontinuierlich an die sich periodisch ändernde Datengrundlage (Transaktionsdaten, Sortimentsänderungen etc.).

4.3.1 Architektur

Jeder Artikel- und Warengruppenagent wird mit Hilfe des MASg-Frameworks erzeugt und enthält sowohl reines Faktenwissen als auch probabilistisches Wissen. Da die Informationen eines Agenten aus Datenbanken stammen, können diese jederzeit beendet und aus den Daten der DB neu erstellt werden kann. Auf der Basis dieser Daten bieten die Agenten ein breites Spektrum an Diensten an, beispielsweise die Simulations- und Evaluationsfunktionalität, die Substitutionsanalyse und verschiedene Arten der Korrelationsanalyse. Durch diese Architektur wird eine serviceorientierte Struktur nicht nur nach außen, sondern auch intern bis hinunter auf die Agentenebene realisiert.

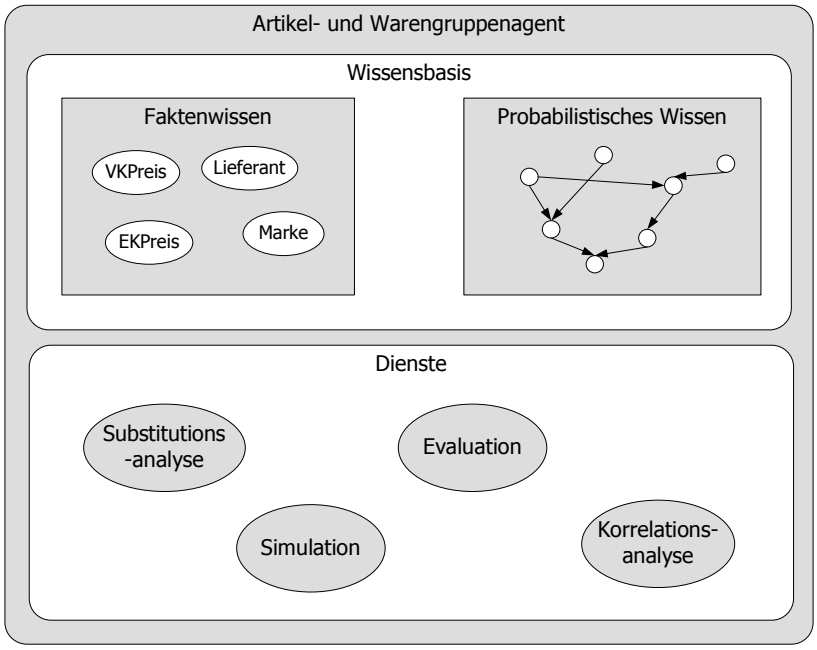


Abbildung 27: Architektur der Artikel- und Warengruppenagenten

Da alle Agenten des MASg-Frameworks die Holonen-Funktionalität haben, unterscheiden sich die Warengruppenagenten nur durch die angebotenen Dienste und

Informationen von den Artikelagenten. Jeder Agent kann prinzipiell als Gruppenagent (einfache Form eines Holons) fungieren, da das Agentenobjekt nach dem Composite-Entwurfsmuster entworfen worden ist. Ein Warengruppenagent kapselt alle Informationen und Dienste bzgl. einer Menge von Artikeln. Angefangen von einfachen Kennzahlen wie Umsatz und Ertrag einer Warengruppen, bis hin zu vollständigen Korrelationsanalysen über die gesamte Warengruppe.

4.3.2 Generierung

Alle Artikel- und Warengruppenagenten werden ausschließlich vom *ItemMetaAgent* erzeugt. Nach der Erzeugung durch den Metaagenten melden sich die Agenten beim Agent-Repository an und bekommen eine globale eindeutige ID-Nummer (GUID). Durch den *Single-Point-Of-Creation* wird verhindert, dass ein Agent unnötigerweise mehrfach an unterschiedlichen Stellen erzeugt und die Konsistenz des Systems zerstört wird. Der Metaagent ist die einzige Stelle im System, die immer eine gültige Referenz auf den erzeugten Agenten hat. Das Verschieben eines Agenten geschieht zwar über das Agenten-Framework, dieses teilt aber unmittelbar dem verantwortlichen Metaagent die neue Referenz mit. Ursprünglich war geplant, die Funktionalität der Metaagenten in den Agenten-Frameworks oder sogar im Agent-Repository zu realisieren. Es war aber zu befürchten, dass diese Komponenten zu einem Engpass im System werden würden, da jeder Zugriff auf einen beliebigen Agenten über diese zentralen Stellen erfolgt wäre. Um die Verteilbarkeit des Systems und die Dezentralität zu erhöhen, wurden die Metaagenten eingeführt. Diese können wie alle anderen Agenten auch im Grid verteilt werden.

Selbstverständlich werden nicht immer alle möglichen Agenten im System erzeugt sondern nur die, die gerade benötig werden. Des Weiteren haben die Agenten drei verschiedene Initialisierungszustände:

1. Ein Agent wird nur als AgentInfo erzeugt und erhält die wichtigsten statischen Informationen, wie z. B. Artikelnummer, GUID und Gruppenzugehörigkeiten. Ein AgentInfo-Objekt wird erzeugt, sobald der Agent in einer Liste auftaucht. Zum Beispiel, wenn ein Warengruppenagent erzeugt wurde und seine Subagenten in der GUI aufgelistet werden sollen.
2. In der zweiten Stufe wird der eigentliche Agent erzeugt und einfaches Faktenwissen gespeichert. Dies geschieht, wenn ein Artikel zwecks genauerer Analyse ausgewählt wird. Dieser Agent kann bereits einfache Aufträge, wie z.B. Korrelationsanalysen, ausführen.
3. Erst in der letzten Stufe wird bei Bedarf die probabilistische Wissensbasis initialisiert. Erst wenn ein Dienst angesprochen wird, der für die Berechnung die

probabilistische Wissensbasis benötigt, z.B. für eine Simulation, wird diese erzeugt.

Die drei Stufen können nach Gebrauch schrittweise wieder rückgängig gemacht werden. Nach längerem Nichtgebrauch der probabilistischen Wissensbasis kann diese getrennt beendet werden. Wann, welcher Agent in welcher Form erzeugt wird, entscheidet sich dynamisch auf Grund der angeforderten Informationen und Dienste. Nach Verwendung der Agenten können diese beendet werden. Die genaue Entscheidung darüber ist abhängig vom Nutzerprofil des Anwenders und von den verfügbaren Systemressourcen.

4.4 Simulations- und Prognoseagent

Der Simulations- und Prognoseagenten (Klasse *SimProgAgent*) hat die Aufgabe das aktuelle Projekt eines Benutzers zu verwalten, Simulationen vorzubereiten, die Erzeugung benötigter Agenten anzustoßen, Simulationsaufträge zu verteilen und zum Schluss die Ergebnisse zu sammeln und aufzubereiten.

Möchte ein Benutzer eine Simulation durchführen kann er sich zunächst entscheiden, ob er ein vorhandenes Projekt laden möchte oder ein neues angelegt werden soll. Im Projekt (Klasse *SimProject*) werden verschiedene globale Einstellungen des Benutzers verwaltet.

Um eine Simulation zu definieren legt der Benutzer eines oder mehrere Szenarien (Klasse *Scenario*) an. Szenarien bestehen aus Artikelszenarien (Klasse *ItemScenario*), die eine oder mehrere Maßnahmen (Klasse *Action*) bzgl. eines Artikels wie z. B. Preisänderungen und Promotionen enthalten. Enthält ein Szenario mehrere Artikel bzw. Artikelszenarien werden diese während der Simulation zu einem Holon verschmolzen. Alle Aktionen eines Artikelszenarios beziehen sich auf ein festzulegendes Zeitintervall. Zwei Szenarien können die gleichen Artikel haben und den gleichen Zeitraum simulieren, aber beispielsweise mit unterschiedlichen Preisen versehen sein oder auch völlig unterschiedliche Artikel und Zeiträume simulieren. Jedes Szenario besitzt ein eigenes Lern- und Simulationsintervall. Das Lernintervall gibt den Zeitraum an, der als Lernbasis für die Verhaltensnetze der Agenten verwendet werden soll. Das Simulationsintervall ergibt sich normalerweise aus dem minimalen Startzeitpunkt und dem maximalen Endpunkt aller Aktionen aller Artikelszenarien.

Hat der Benutzer ein Projekt vollständig definiert und die Simulation gestartet, wählt der SimProgAgent die passende Strategie. Auf Grund der Einstellungen im Projekt veranlasst der SimProgAgent die Erzeugung der richtigen Agenten und teilt die globale Simulation auf diese auf. Nach Beendigung aller verteilten Simulationen sammelt der SimProgAgent die Ergebnisse ein und aggregiert diese zu einem Gesamtergebnis.

4.4.1 Evaluationsframework

Die meisten Klassen der Projektverwaltung besitzen eine abgeleitete Evaluationsversion (siehe Abbildung 28). Diese Klassen werden statt der normalen Klassen verwendet, wenn eine Evaluation durchgeführt werden soll. Dies ist immer dann möglich, wenn über den Simulationszeitraum die realen Abverkaufswerte bereits bekannt sind, z. B., weil man einen Zeitraum in der Vergangenheit simuliert. Auch in diesem Fall dürfen die Verhaltensnetze für eine realistische Evaluation nur die Daten vor dem Simulationszeitraum als Trainingsmenge verwenden, obwohl die Daten über den Simulationszeitraum vorliegen.

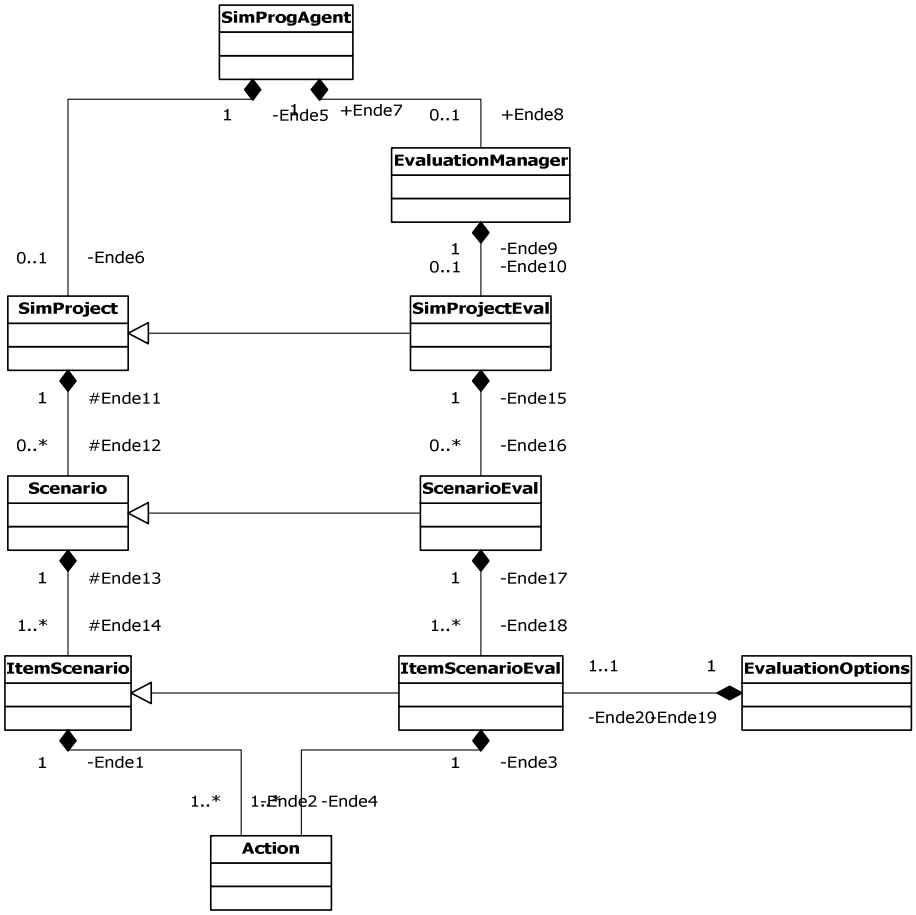


Abbildung 28: Klassenhierarchie der Simulations- und Evaluationsklassen

Evaluationsklassen haben die Funktion, die Simulationsergebnisse mit den realen Daten anzureichern und Prognosegütemaße zu berechnen. Dazu werden verschiedene Maße auf die simulierten und realen Datenreihen angewendet. Anschließend werden diese Maße auf Szenario- und Projektebene mittels Durchschnittsbildung aggregiert.

4.4.1.1 Evaluationsmanager

Der Evaluationsmanager ist die Hauptkomponente des Evaluationsframeworks. Neben der bereits beschriebenen Möglichkeit manuell erstellte Szenarien zu evaluieren, besitzt der Evaluationsmanager die Fähigkeit automatische Evaluationen auf großen Mengen von Warengruppen und Artikeln durchzuführen. Dazu wählt man zunächst eine Menge von Artikel und/oder Warengruppen, die in der Evaluation berücksichtigt werden sollen, per Drag & Drop aus. Dabei muss man entscheiden, ob jeder Artikel in ein neues Szenario eingefügt werden soll oder mehrere in einem zusammengefasst und somit während der Simulation zu einem Holonen verschmolzen werden. Nach der Definition der Constraints startet der Evaluationsmanager eine Routine, um in den Vergangenheitsdaten für jeden Artikel nach Situationen zu suchen, die die definierten Constraints erfüllen. Für jede gefundene und ausgewählte Situation wird ein korrespondierendes Artikelszenario mit den in der Vergangenheit stattgefundenen Aktionen angelegt und dem passenden Szenario zugeordnet. Anschließend wird eine Simulation mit Evaluation durchgeführt, die Ergebnisse aufbereitet und angezeigt.

Auf Grund der Komplexität der Simulationsdomäne und der Mehrdimensionalität der Modelle werden zahlreiche Parameter benötigt, um eine automatische Evaluation zu definieren. Diese sind notwendig, um alle Fassetten einer Simulation testen zu können. Die Einstellungen unterteilen sich in szenariobezogene und artikelszenariobezogene Parameter. Die Parameter der Szenarien sind im Einzelnen:

- Zeitbasis: In welcher zeitlichen Granularität soll die Simulation ablaufen, tageweise, wochenweise usw. Dementsprechend wird die Datenbasis für das Lernen der Verhaltensnetze aufbereitet.
- Art des Suchreitraumes:
    1. Man definiert einen Gesamtzeitraum, in dem passende Szenarien gesucht werden sollen. Der Lernzeitraum wird über das eingestellte Lernverhalten definiert (s. u.).
    2. Man legt getrennt einen Lernzeitraum und einen Suchzeitraum fest. Der Lernzeitraum beschreibt einen minimalen Zeitraum, der für das Lernen verwendet werden soll. Im Suchzeitraum wird wie in 1. nach Szenarien gesucht. Das Ende des Lernintervalls wird vor der Simulation ebenfalls wie bei 1. auf einen Tag vor dem Beginn der ersten gefundenen Aktion festgelegt.
- Lernverhalten: Gibt an, ob das Lernintervall maximiert oder minimiert werden soll. In der Regel ist die Maximierung sinnvoller.

- Alle Artikel im gleichen Zeitraum simulieren: Alle Aktionen aller Artikel im Szenario müssen sich im gleichen Zeitraum befinden. Diese Option ist für die Holonenbildung nötig.
- Alle möglichen Szenarien finden: Im Normalfall wird immer nach einem passenden Gesamtszenario gesucht. Mit dieser Option werden alle passenden Szenarien gefunden und simuliert. Das kann allerdings dazu führen, dass das Lernintervall der klein wird und damit die Ergebnisse ungenauer. Außerdem sind die Ergebnisse schwerer vergleichbar, da für alle Artikel unterschiedliche viele und unterschiedliche lange Szenarien gefunden werden.

Die Parameter der Artikelszenarien sind:

- Aktionspräferenz: Gibt die Art der Aktionen an die gefunden und damit evaluiert werden sollen.
- Anzahl bestimmter Evidenzen: Gibt die Art und Anzahl der Evidenzen an, die gesetzt werden sollen. Gefundene Aktionen resultieren in unterschiedlichen Arten von Evidenzen, die für die Simulation im Verhaltensnetz gesetzt werden müssen. Mit dieser Option können die Aktionen dementsprechend gefiltert werden.
- Evidenzen durchgängig setzen: Diese Option gibt an, dass bei mehreren gefundenen Aktionen für einen Artikel die Zeiträume zwischen diesen Aktionen mitsimuliert werden sollen.

Alle Einstellungen können getrennt für alle Szenarien bzw. Artikelszenarien gesetzt werden. Es gibt aber die Möglichkeit, die Einstellungen eines Objektes auf alle anderen zu übernehmen.

4.5 Verhaltensnetze

Im aktuellen Kapitel werden die drei Stufen Datenpräparation, Modellierung der Verhaltensnetze und Anwendung der Netze für die Simulation von Artikelabverkaufsverhalten behandelt.

4.5.1 Extraktion

Das Ziel des Extraktionsprozesses ist eine Tabelle der folgenden Form:

Name	Datentyp	Größe	Null	Beschreibung
Datum	DateTime	8	Nein	Startdatum des beobachteten Intervalls
InVariable 1..N	Double	12,4	Nein	Messbare Einflussfaktoren, die das Einkaufsverhalten beeinflussen. Jede Variable bekommt eine eigene Spalte.
OutVariable 1..M	Double	12,4	Nein	Gemessene Zielvariablen, die prognostiziert werden sollen. Jede Variable bekommt eine eigene Spalte.

Tabelle 2: Data-Mining-Tabelle

Eindeutiger Schlüssel: Datum

Diese Data-Mining-Tabellen (DM-Tabellen) bilden die Lernbasis für die Verhaltensnetze von Artikel- und Warengruppenagenten und können für unterschiedliche Zeitbasen erstellt werden. Jede Zeile dieser Tabelle entspricht einem Lernfall und kann sowohl einen Tag als auch einen größeren Zeitraum wie z. B. eine Woche repräsentieren. Das Datum bezeichnet immer den Beginn des Zeitraumes eines Lernfalles. Das oben angegebene Format entspricht dem Format, das von den Verhaltensnetzen erwartet wird und nicht unbedingt dem gespeicherten in der Datenbank. Um zu einer Tabelle dieser Form zu kommen, wurde bereits beim Import der Daten ein Bereinigungs- und Konvertierungsprozess durchgeführt, bei dem die Daten auf Konsistenz, Korrektheit und Vollständigkeit geprüft werden. Da die Verhaltensnetze das Abverkaufsverhalten der Artikel aus diesen Tabellen lernen, können sich Fehler fatal auf die Simulationsgüte auswirken. Deshalb wurden von mir in SimMarket u.a. die folgenden Maßnahmen zur Qualitätssicherung getroffen:

1. Primary Keys für alle Tabellen definiert, um redundante Daten zu vermeiden.
2. Foreign Keys definiert, um die Konsistenz und Vollständigkeit der Daten sicherzustellen.
3. Nicht benötigte und fehlerhafte Daten wurden gelöscht.
4. Die Rohdaten der Händler wurden in ein neues Schema (siehe Anhang D) überführt, das Konsistenz, Vollständigkeit und Korrektheit durch das Zusammenführen von mehreren Ausgangstabellen in eine Zieltabelle sicherstellt.
5. Manuelle Stichproben von allen Daten.

Die Spalte „Datum“ fließt in der Regel nicht direkt in die Netze ein, sondern dient zur als eindeutiger Schlüssel für jede Zeile. Da die Verhaltensnetze durch das zu Grunde liegende Wahrscheinlichkeitskalkül ausschließlich auf Fließkommazahlen arbeiten, müssen alle Datenspalten den Datentyp Float/Double haben. Zwar gehen die Daten der DM-Tabelle nicht direkt in ein Netz ein, sondern werden vorher weiter transformiert, dennoch sollte die

notwendige Konvertierung in Double-Werte bereits in der Datenbank stattfinden. Theoretisch wäre es aber möglich eine entsprechende Komponente vor das Lernen der Netze zu setzen, die beliebige Datentypen in entsprechende Double-Werte abbilden kann.

Für eine weitere Verdeutlichung der Form der DM-Tabellen hier eine konkrete Ausprägung einer solchen Tabelle:

Datum	Wochentag	Tag	Monat	Maßnahme	Preis	Menge
21.04.2005	4,0	21,0	4,0	0,0	1,99	0,0
…						
28.06.2005	2,0	28,0	6,0	1,0	1,39	69,0

Tabelle 3: Konkretes Beispiel für eine DM-Tabelle

Die Tabelle beschreibt die Abverkaufsmenge, den Verkaufspreis und die durchgeführten Maßnahmen eines Artikels für jeden Verkaufstag zwischen dem 21.04.2005 und dem 28.06.2005 eines Artikels. Diese Tabelle muss lückenlos für jeden Tag und jede Spalte gefüllt sein. Insbesondere die Tage, an denen der Artikel nicht verkauft worden ist, müssen enthalten sein. Ausnahmen sind nicht verkaufsoffene Tage wie Sonntage oder Feiertage. Ist ein Artikel nicht eingelistet, so sollte diese Tabelle ebenfalls keine Einträge für diese Zeiträume enthalten, obwohl es in diesen Zeiträumen dennoch zu Abverkäufen dieses Artikels kommen kann, da der Artikel evtl. noch auf Lager liegt und Restbestände abverkauft werden. Dieses Abverkaufsverhalten kann aber nicht als repräsentativ für diesen Artikel gelten. Daten von temporären Auslistungen würden die Simulation nur verfälschen.

Tabellen mit unvollständigen Daten werden mit speziellen Lernverfahren wie dem EM- und dem SEM-Algorithmus ([DemLaiRub77], [Friedman97], [Friedman98]) behandelt. Diese Verfahren ermitteln die fehlenden Werte aus den Vorhandenen bei der Generierung der Netze.

Im obigen Beispiel wurden zu den bereits beschriebenen Einflussfaktoren zusätzlich drei Zeitvariablen hinzugenommen. Dadurch lässt sich das Abverkaufsverhalten abhängig vom Wochentag, vom Tag des Monats und dem Monat beschreiben.

Nun fehlt noch ein letzter Schritt, um die Daten für das Lernen von Verhaltensnetzen zu verwenden, die *Diskretisierung*. Da die Verhaltensnetze nur mit beschränkten Mengen an Zuständen pro Knoten praktikable betrieben werden können, müssen die Werte der zumeist kontinuierlichen Variablen in den DM-Tabellen vor der Generierung der Netze diskretisiert werden. Bei der Diskretisierung geht es darum eine Menge von lückenlosen und überschneidungsfreien Intervallen zu finden, in die die kontinuierlichen Werte einer Variable abgebildet werden können, so dass der Erwartungswert der Wahrscheinlichkeitsverteilung unter allen denkbaren Bedingungen möglichst nahe an der Realität liegt.

Auf diese diskretisierten DM-Tabellen kann nun ein Lernverfahren (siehe 3.4.4) für Verhaltensnetze angewendet werden. Die Knoten des resultierenden Netzes entsprechen einer 1:1-Abbildung der Spalten der DM-Tabelle, d. h., mit Ausnahme der Datumsspalte würde jede Spalte als ein Knoten im Netz modelliert werden. Durch Anwendung eines Struktur- und CPT-Lernalgorithmus könnte man ohne weiteres Zutun ein Verhaltensnetz extrahieren. Da man aber in der Regel über umfassendes Domänenwissen verfügt, beschreibt das nächste Kapitel wie Verhaltensnetze vor dem Lernprozess mit Domänenwissen modelliert werden können.

4.5.2 Modellierung

Trotz der Mächtigkeit der Verhaltensnetze werden nicht alle Anforderungen durch diese erfüllt, sondern erst durch speziell entwickelte Modellierungstechniken, die ich in diesem Abschnitt darstellen möchte.

Normale Bayes'sche Netze besitzen nur Knoten mit einer endlichen Anzahl diskreter Zustände. Dagegen sind viele zu modellierende Variablen wie die Absatzmenge und der Umsatz eines Artikels kontinuierlich. Daraus folgt: 1. Die Lernmenge muss zunächst für alle Variablen partitioniert werden (siehe [Schwaiger06]) und 2. die Ausgabe der diskreten Knoten muss wieder in einen metrischen Wert umgerechnet werden. Die Ausgabe eines Knotens in einem BN hat die Form $P_{X1},\ldots,P_{Xn}$ wobei P_{Xn} die Wahrscheinlichkeit des Zustandes n des Knotens X ist. n repräsentiert dabei ein Werteintervall [v,..,w] mit v <= w, das dem Anwender die Bedeutung des Intervalls verdeutlicht. Die Intervalle können beispielsweise für Verkaufspreise (VKPreise), Absatzmengen oder auch Wochentage stehen. In den Verhaltensnetzen werden diese Werte benutzt, um den Erwartungswert (EW) der Wahrscheinlichkeitsverteilung eines Knotens zu berechnen. Der EW ist die mit der Wahrscheinlichkeit P gewichtete Summe der Mittelwerte MW aller Zustände n eines Knotens X:

$$EW(X) = \sum_{\forall i \in n} P(i) \cdot MW(i)$$

Dabei gilt es zu beachten, dass ein Intervall [v, w] prinzipiell zwei verschiede Mittelwerte hat:

1. $MW_1 = (v + w / 2)$ und
2. $$MW_2 = \frac{\sum_{\forall d \geq v, d < w, d \in Tr(X)} d}{\#d}$$

der reale Mittelwert, der sich aus den Datenpunkten d der Trainingsmenge Tr ergibt, die in das Intervall [v, w] des Zustandes n eines Knotens X bei der Diskretisierung fallen. Wir verwenden letzteren Mittelwert, um einen realistischeren Erwartungswert zu bekommen.

Der EW gibt somit den durchschnittlichen Wert einer Variablen pro Zeiteinheit unter den mittels Evidenzen eingegebenen Bedingungen an. So gibt beispielsweise der EW(Menge) die durchschnittliche Abverkaufsmenge eines Artikels pro Zeitbasis (i. d. R. Tag oder Woche) des Netzes an. Der EW kann für alle Knoten ausgerechnet werden, da Voraussetzung ist, dass alle Variablen als Doublewerte in die Netze einfließen. Einige EW wie z. B. der des Knotens „Wochentag“ haben allerdings keine sinnvolle Bedeutung.

In der Theorie der Bayes’schen Netze sind prinzipiell alle Knoten vom gleichen Typ, in dem Sinne, dass es bspw. keine Unterscheidungen zwischen Input- und Outputknoten gibt, da Evidenzen auf alle Knoten gesetzt werden können. Auch enthalten die Knoten standardmäßig keine Semantik, die auf eine bestimmte Bedeutung oder Funktion der Knoten hindeutet. Für die Modellierung und die anschließende Interpretation kommt man jedoch nicht umhin, die Knoten in semantische Kategorien einzuteilen. Dies hilft bei der Modellierung von Domänenwissen und lässt sich für Performanz-Optimierungen bei der Propagierung von Evidenzen verwendet. Insgesamt gibt es vier Ebenen.

Auf der ersten Ebene unterscheiden wir die drei Kategorien:

- **Zeitknoten:** Sie modellieren alle zeitlichen Einflüsse, die auf eine zu modellierende Entität wirken können, z. B. Wochentag und Monat. Genau genommen handelt es sich bei den Zeitknoten auch um Einflussfaktoren, die aber für die Prognose und Simulation eine besondere Bedeutung haben.
- **Einflussfaktoren:** Alle Variablen, die einen Einfluss auf die zu modellierende Entität haben könnten und primär gesetzt und nicht prognostiziert werden sollen, obwohl dass bei der Verwendung von BN prinzipiell egal ist. Beispiel: Aktionen und Verkaufspreise bei Artikelagenten.
- **Zielvariablen:** Alle Variablen, die prognostiziert werden sollen. Dennoch können auch diagnostische Anfragen an diese Knoten gestellt werden (Welche Situation muss eintreten, um einen bestimmten Wert einer Zielvariable zu erreichen?), z. B. Absatzmenge und Umsatz.

Die nächste Ebene ist stark von der Fragestellung abhängig, die man modellieren möchte. Für die korrekte Interpretation der Ergebnisse eines vorhandenen Netzes ist das Verständnis dieser Ebene sehr wichtig. Wir unterscheiden die folgenden Typen von Variablen, die durch einen Knoten repräsentiert werden.

- **Absolute Variablen** modellieren die realen Werte von absoluten Messgrößen / Kennzahlen, z. B. Absatzmenge, VKPreis, Aktionen etc.
- **Relative Variablen** sind immer Vergleichsmaße, die den Wert beispielsweise eines Artikel immer in Relation zu anderen Dimensionen wie z. B. anderer Zeitraum, Artikel, ausdrücken. Relative Variablen können kategorisch sein, wie z. B. Preiskategorie, die den VKPreis eines Artikels relativ zu den anderen Artikeln der Warengruppe in eine Kategorie z. B. niedrig-, mittel- oder hochpreisig einordnet. Oder auch kontinuierlich und bspw. die prozentuale Veränderung gegenüber einer Baseline angeben. Die Baseline kann für verschiedene Dimensionen wie Zeit, eine oder mehrere Warengruppen berechnet werden, um die prozentuale Veränderung eines Artikels bezüglich der Vergangenheit oder der gesamten Warengruppe zu ermitteln.
- **Differenzielle Variablen** haben immer einen fixen Bezugspunkt und drücken den aktuellen Unterschied zu diesem aus. Zum Beispiel wie viele Tage sind seit der letzten Promotion vergangen? Diese Arten von Knoten sind sehr nützlich, um Effekte nicht nur während eines bestimmten Ereignisses, sondern auch danach simulieren zu können, z. B. Absatzeinbruch nach einer längeren Promotionsperiode (mit Preissenkung).

Auf der nächsten Ebene unterscheiden wir:

- **kategorische Variablen** (umfasst auch binäre Variablen) und
- **kontinuierliche Variablen**

Diese Kategorie bestimmt, ob und welche Art von Diskretisierung nötig ist und entscheidend zur Güte der Simulation beiträgt. Dabei muss immer zwischen der möglichst exakten Rekonstruktion der Vergangenheit und einer Generalisierung für zukünftige Anfragen abgewogen werden. Insbesondere starke Schwankungen in einer Zeitreihe erfordern intelligente Diskretisierungsverfahren, die z.B. wie in SimMarket mit einer vorangehenden Glättung der Daten arbeiten. Bei kategorischen Variablen hat man naturgemäß keinen Informationsverlust, da man die Kategorien direkt auf Zustände abbilden kann.

Auf der letzten Ebene muss dann schließlich die exakte Bedeutung eines Knotens festgehalten werden. Dies ist für eine (automatische) Interpretation / Erklärungskomponente und, wie wir später noch sehen werden, für die Simulation nötig. Dazu wird jedem Knoten ein so genannter *ContentType* zugeordnet. Beispiele dafür sind:

- Day
- Month

- Week
- Weekday
- SalesPrice
- Promotion
- SalesAmount
- Turnover
- PriceCategory etc.

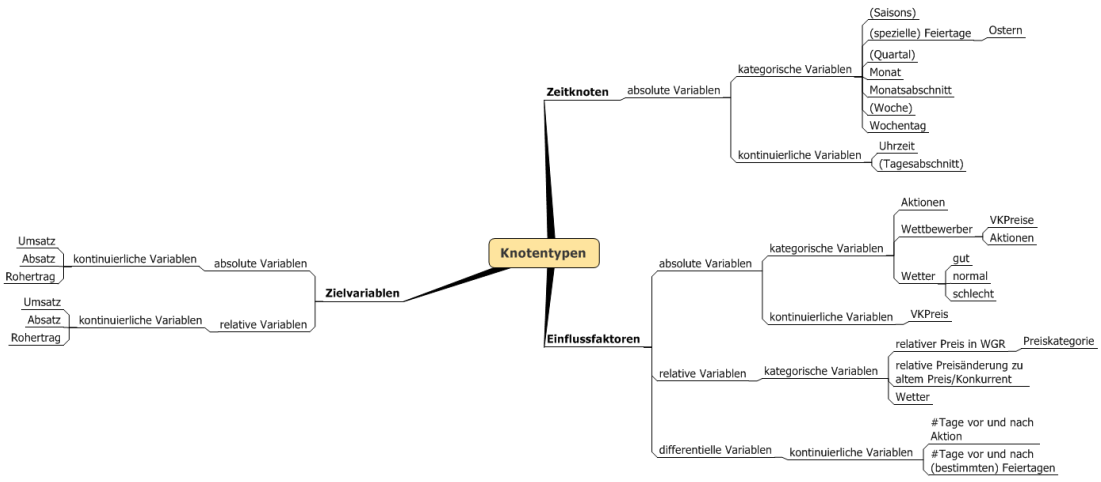


Abbildung 29: Knotentypen eines Verhaltensnetzes

Nachdem wir gesehen haben, wie man Domänenwissen in unterschiedliche Knotentypen kodieren kann und welche Arten von Variablen in BN verwendet werden können, widmen wir uns nun der Topologie der Netze.

Bei der Modellierung der Topologie eines BN gilt es folgendes zu beachten: Bei den Kanten eines Netzes muss man zwischen der wahrscheinlichkeitstheoretischen Bedeutung und der semantischen Bedeutung unterscheiden. Obwohl es sich bei BN um azyklische Graphen mit gerichteten Kanten handelt, wirken Evidenzen, die in ein Netz eingegeben werden, in beide Richtungen. Auch drückt die Pfeilrichtung nicht unbedingt eine Kausalität im Sinne A ist eine Ursache für Wirkung B, aus. Eine Kante steht nur für eine bedingte Abhängigkeit im wahrscheinlichkeitstheoretischen Sinne und wirkt zudem noch in beide Richtungen. Genau genommen beschreibt eine Kante A→B die Zustände von B unter allen möglichen Zuständen von A, d. h. die Zustände von B werden auf alle Kombinationen von A und B aufgesplittet. Wird eine Evidenz bei A gesetzt, schränkt dies den Raum der Möglichkeiten von B ein und verändert dadurch die Wahrscheinlichkeitsverteilung von B. Umgekehrt gilt das Gleiche. Daraus folgt, dass jeder weitere Knoten z. B. eines Einflussfaktors, der eine Kante auf B hat, eine detailliertere Betrachtung von B unter den

Bedingungen der Elternknoten zulässt. Allerdings schränkt man mit jedem neuen Elternknoten, für den eine Evidenz gesetzt wird, den Wertebereich von B stark ein, eventuell mehr als man möchte (Stichwort Overfitting). Dies kann insbesondere dann ärgerlich sein, wenn man auf B selbst noch eine Evidenz setzen möchte (und A und B beispielsweise Kanten nach C haben). Ist B durch A aber bereits zu stark eingeschränkt, dann kann man die gewünschte Evidenz auf B nicht mehr setzen. Wie kommt man nun zu einer sinnvollen Netztopologie?

1. Komplett manuell vorgeben,
2. automatisch mittels Lernverfahren erstellen lassen oder
3. eine Kombination aus 1. und 2.

Das in SimMarket implementierte Verfahren ist in der Lage sowohl mit vorgegebenem Domänenwissen umzugehen, als auch automatisch Netze ohne Vorwissen zu generieren. Das Domänenwissen kann in Form von Kantenattributen und durch ein weiteres Knotenattribut dem Lernverfahren mitgegeben werden. Dazu kann der Benutzer folgende Kantentypen vorgeben:

- *forbidden edges*: Eine verbotene Kante zwischen zwei Knoten bedeutet, dass dort auf keinen Fall vom Lernverfahren eine Kante eingefügt werden darf, auch dann nicht, wenn die Bewertungsfunktion eine Qualitätsverbesserung des Netzes ermittelt.
- *forced edges*: Eine erzwungene Kante wird unter allen Umständen eingefügt, auch dann, wenn die Bewertungsfunktion keine Qualitätsverbesserung oder sogar einen negativen Wert auf Grund eines hohen Penality-Wertes des Netzes ermittelt.
- *proposed edges*: Vorgeschlagene Kanten werden auf jeden Fall priorisiert durchgetestet, aber nur bei einer signifikanten Qualitätssteigerung eingefügt.

- *directed*: Die Kante wird in der angegebenen Richtung getestet bzw. eingefügt. Kann benutzt werden, um die Lesbarkeit des Netzes zu erhöhen, insbesondere um bekannte Kausalitäten in der Netzstruktur auszudrücken.
- *undirected*: Ist man sich nicht über die Wirkungsrichtung sicher oder ist diese irrelevant, dann kann eine Kante als ungerichtet an den Lernalgorithmus übergeben werden. Dieser testet beide Richtungen durch und fügt immer die mit der besten Bewertung ein.

Die Netz-Topologie kann zudem durch die folgenden Knotenattribute mit Domänenwissen beeinflusst werden:

- *input node*: Ein Knoten, in den nur Kanten rein gehen dürfen.
- *output node*: Ein Knoten, der nur ausgehende Kante besitzen darf.
- *inout node*: Ein Knoten, der sowohl eingehende, als auch ausgehende Kanten bekommen darf.
- *inner node*: Ein Knoten, der keine weiteren Kanten während der Holonisierung bekommen darf.

Diese Knoten-Attribute werden an zwei Stellen in SimMarket verwendet: einerseits beim Lernen eines einzelnen Netzes und anderseits bei der Holonisierung. Bei der Erstellung eines Einzelnetzes werden diese Attribute analog zu den Kantenattributen interpretiert. Das Attribut *inner node* ist in diesem Falle allerdings nicht sehr sinnvoll, da der Knoten immer isoliert bleiben würde und somit für das Netz keine Relevanz besäße. Dieses Attribut wird für die Holonisierung verwendet, um innere Knoten, die keine weiteren Verbindungen aufbauen dürfen, zu definieren. Die Knotenattribute beschreiben, welche Knoten in welcher Art und Weise bei einer Holonisierung mit Knoten von anderen Agenten verknüpft werden dürfen? Die Knotentypen *input node, output node* und *inout node* bezeichnen wir als *interface nodes*.

Die Modellierung eines vollständigen Verhaltensnetzes erfolgt zusammenfassend durch die folgenden Schritte:

1. Datensichtung: Welche Daten stehen mir zur Verfügung, um eine Entität zu modellieren und welchen Umfang haben diese? Sind diese vollständig, korrekt und aussagekräftig? Wie muss ich die Daten aufbereiten, um zu einer Data-Mining-Tabelle zu kommen?
2. Fragestellung: Welche Fragestellungen will ich mit meinem Modell behandeln können?
3. Knotentypen: Welche Variablentypen kann ich aus den Daten generieren und auf welche Weise muss ich diese vorab transformieren? Welche Typen brauche ich für meine Fragestellungen? Welche Knoten übernehme ich in das Netz?
4. Diskretisierung: Die Diskretisierung der Knoten entweder manuell vornehmen oder einen Algorithmus auswählen.
5. Domänenwissen: Möglichst viel Domänenwissen in Form von Kanten- und Knotenattributen in das Netz eingeben, evtl. Kausalitäten modellieren und den Suchraum mittels verbotener Kanten etc. einschränken.
6. Lernalgorithmus: Lernverfahren auswählen und dessen Parameter festlegen. Das Netz mit dem Lernverfahren vervollständigen und das Ergebnis sichten.
7. Evaluierung: Die Qualität des Netzes kann mit unterschiedlichen Metriken gemessen werden. Entweder im Vergleich zu der modellierten Lernmenge oder

idealerweise schon in Hinblick auf die Fragestellung z. B. Prognosegüte. Bei Bedarf die Schritte 3-7 mit veränderten Einstellungen wiederholen.

4.5.3 Holonisierung

Mit den bis hierhin beschriebenen Methoden werden bereits viele Anforderungen an ein Simulationssystem erfüllt. Ein bisher außer acht gelassenen Einflussfaktors erweist sich noch als schwierig – die Wechselwirkungen zwischen einzelnen Artikeln. Wird ein Artikel beworben oder im Preis gesenkt wirkt sich dies immer auch auf andere Artikel des Sortimentes aus. Es kann sein, dass einige Artikel „mitgezogen“ werden, da die Bewerbung den Fokus der Verbraucher auf die gesamte Warengruppe gelenkt hat und ihnen diese Kategorie von Artikeln schmackhaft gemacht hat (*Pulleffekt*) oder, dass andere Artikel durch *Cross-Selling-Effekte* verstärkt mitgekauft werden. Umgekehrt kann die Bewerbung eines Artikels auch den Abverkauf von anderen Produkten kannibalisieren (*Kannibalisierungseffekt*). Im schlimmsten Fall werden die Gewinne des Promotionsartikels durch die kannibalisierten Artikel aufgezehrt. Dies verdeutlicht, wie schwierig es ist mit einer Bewerbung einen Gewinn zu machen und vor allem wie schwierig es ist den Erfolg einer geplanten Aktion im Vorfeld zu prognostizieren. Die Bewerbung eines Artikels (mit Preissenkung) führt erst einmal zu einem geringeren Rohertrag pro Artikel, da eine Promotion immer mit Kosten verbunden ist und eine Preissenkung die Gewinnspanne verringert. Die verringerte Gewinnspanne muss durch einen Mehrverkauf des Artikels ausgeglichen werden oder im Idealfall sogar den Gesamtgewinn erhöhen. Dabei müssen alle zeitlichen Faktoren wie Feiertage, Saisons etc. und die Aktionen der Mitbewerber berücksichtigt werden.

Will man zu einer Gesamtoptimierung eines Sortimentes und der geplanten Aktion kommen, müssen alle parallel laufenden Aktionen des Marktes, alle Pull-, Kannibalisierungs- und Cross-Selling-Effekte in der Planung bedacht werden. Um all diese Effekte in einer Simulation berücksichtigen zu können, verwenden wir in SimMarket das von mir entwickelte Holonisierungsverfahren (siehe Kapitel 3.4.7).

Um einen Warengruppenholon zu bilden, werden wie bisher die einzelnen Artikel als probabilistische Artikelagenten modelliert und bei Bedarf zu einem Holon zusammengefügt. Bei Vorliegen der Einzelagenten bedarf es keines großen Aufwandes um einen Holonen zu erzeugen, da das lokale Wissen in den Einzelagenten erhalten bleibt und nur einige wenige Kanten hinzukommen. Der Suchraum kann zudem durch Domänenwissen erheblich eingeschränkt werden. Neu gelernt werden müssen lediglich die CPTs von Knoten in die eine neue interdependente Kante mündet. Holonen sind also ein effektiver Mechanismus um aus bereits vorliegendem lokalem Wissen weiteres und vor allem globales Wissen abzuleiten. Holonen müssen dazu *on-demand* und *temporär* erzeugt werden, wobei globales Wissen

emergiert. Die Qualität einer Artikelsimulation wird verbessert, da nun nicht nur die Preisänderung des Artikels selbst in die Simulation mit einbezogen wird, sondern auch die Preisänderungen von anderen korrelierenden Artikeln. Durch die Erstellung von virtuellen Warengruppen können ganz neue Arten von Gruppensimulationen durchgeführt werden. So lässt sich ein Holon auch für eine Korrelations- und Substitutionsanalyse verwenden, bei der die wechselseitige Wirkung von Artikel direkt analysiert wird. Welche Artikel korrelieren miteinander und vor allem welchen quantifizierten Einfluss üben die Artikel aufeinander aus? Daraus lässt sich schließen, welche Artikel von welchen anderen substituiert werden. Dies ist insbesondere für eine Einlistung- und Auslistungsanalyse nötig. So erhält man durch die Holonenbildung nicht nur eine verbesserte Simulation, sondern kommt zu ganz neuen Analysemöglichkeiten.

Die holonischen Verhaltennetze sind aber nicht nur ein Mechanismus für verbesserte Simulationen und erweiterte Analysemöglichkeiten, sondern dienen auch der Organisation und somit der besseren Lesbarkeit der Netze. Da die Holonenbildung ein hierarchisches Konzept ist, lassen sich damit beliebige hierarchische und objekt- bzw. agentenorientierte Netze visuell darstellen. Somit sind die probabilistischen Verhaltensnetze auch mit der Funktionalität von objektorientierten und hierarchischen Bayes'schen Netzen (siehe [RusNor03], Kapitel 3.4.3.4) vergleichbar. Da die Verhaltensnetze direkt aus relationalen Datenbanken unter Berücksichtigung der dort vorliegenden Relationen gelernt werden können, subsumieren sie außerdem *Probabilistic Relational Models (PRS)* (siehe [FrGeKoPf99]).

4.6 Simulation

4.6.1 Der Simulationsprozess

Es gibt drei Möglichkeiten eine Simulation zu definieren und durchzuführen: 1. durch (manuelles) Erstellen eines Szenario-Objektes, 2. durch eine Anfrage in Form eines **MSL**-Ausdrucks (***M**ulti-dimensional **S**imulation **L**anguage*) und 3. durch Angabe eines Optimierungsziels. Bei optimierenden Fragestellungen werden keine Szenarien oder konkrete Anfragen vorgegeben, sondern es soll ein möglichst optimales Szenario gefunden werden, das eine oder mehrere vorgegebene Kennzahlen optimiert. Dazu wird durch *Heuristiken* zielgerichtet eine Menge von Szenarien erzeugt, die dann in einer Simulation getestet werden. Das Ergebnis einer solchen Simulation ist das beste Szenario der erzeugten Menge. Vor einer Simulation werden alle Szenarien intern in einen oder mehrere *MSL-Ausdrücke* umgewandelt. Bei *optimierenden Simulationen* können auch direkt MSL-Ausdrücke generiert werden. MSL ist eine von mir entwickelte *mehrdimensionale Simulationssprache*, die im nächsten Abschnitt behandelt werden wird.

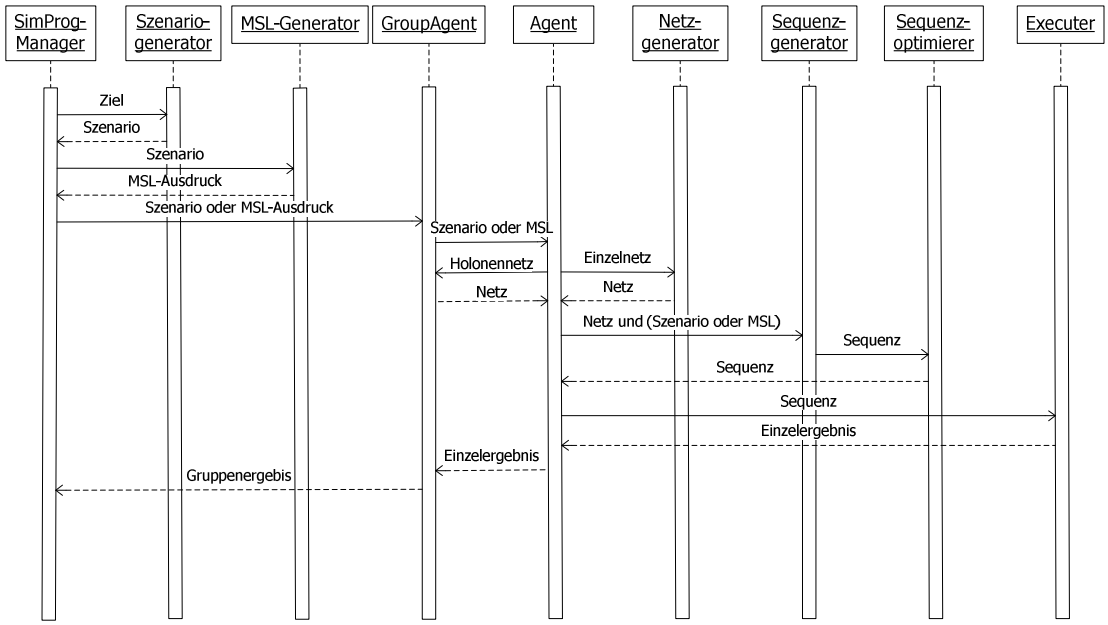


Abbildung 30: Der Simulationsprozess

Der Simulationsprozess, als Teil des Simulations- und Prognosemanagers, kann also entweder ein Szenario, einen MSL-Ausdruck oder ein Ziel als Input entgegennehmen. Ziele werden an einen *Szenariogenerator* übergeben, der – wie beschrieben – eine Menge von Szenarien oder MSL-Ausdrücken generiert. Bekommt der Simulationsprozess ein Szenario, wählt dieser die passenden Agenten aus, die für die Simulation nötig sind und fordert diese von den Meta-Agenten an. Die Szenarien werden an die entsprechenden Agenten, Artikel- oder Warengruppenagenten, weitergeleitet. Diese wiederum entscheiden, welche Verhaltensnetze für die Durchführung der Simulation gebraucht werden. Je nach Fragestellung erzeugt der Agent ein oder mehrere Einzelnetze oder ein Holonennetz. Das Szenario und die Netze werden anschließend an einen *MSL-Generator* weitergegeben. Dieser erzeugt aus den Szenarien und passend zu den Verhaltensnetzen einen – in seltenen Fällen mehrere – MSL-Ausdrücke. Bei Simulationen, die direkt als MSL-Ausdruck definiert worden sind, entfällt der letzte Schritt. MSL-Ausdrücke werden durch einen *Parser* und *Interpreter* (*Sequenzgenerator*) in eine *Sequenz von Netzkonfigurationen* $Sq = \{Z_1,\ldots,Z_n\}$ übersetzt. Eine Sequenz besteht aus einer sortierten Menge an Zuständen, d. h. einer Kombination verschiedener Evidenzen $Z_i = \{Ev_1,\ldots,Ev_j\}$. Vor dem Ausführen einer Sequenz wird diese durch einen Sequenzoptimierer hinsichtlich Ausführungsgeschwindigkeit optimiert. Dazu werden die Zustände sortiert, sodass die Ausführungskosten AK, dargestellt durch den folgenden Ausdruck, minimiert werden:

$$AK = \sum_{i=1}^{n-1} \#(Z_i \cap Z_{i+1})$$

Die optimierte Sequenz wird anschließend an den *Executer* übergeben. Bevor der Executer die Sequenz tatsächlich ausführt, wird die Sequenz daraufhin analysiert, welches Inferenzverfahren am geeignetesten ist und, ob abhängig vom gewählten Verfahren die Netzstruktur optimiert werden kann oder sogar muss (siehe beispielsweise BigClique-Algorithmus). In SimMarket verwende ich folgendes Auswahlverfahren:

1. Es liegt kein Netz oder Polytree vor und es existiert nur ein Query-Node => Naive-Bayes-Verfahren verwenden.
2. Es liegt ein Polytree ohne Zyklus vor => verwende einfaches Message-Passing von Pearl ohne Junction Tree.
3. Es liegt ein Netz mit Zyklus der Länge 3 vor => Sonderfall, der in Pearls Algorithmus (siehe [Pearl88]) eingebaut wird.
4. Es liegt ein Netz mit einem Zyklus der Länge > 3 vor => Es muss der Junction-Tree-Algorithmus oder die Konditionierungsmethode verwendet werden. Sekundärstrukturen werden erst bei tatsächlichem Bedarf generieren, da es seien kann, dass die vorliegenden Evidenzen das Netz sowieso ausreichend konditionieren und damit zyklenfrei machen. Falls das nicht der Fall ist, wird ermittelt, ob die Konditionierungsmethode oder das JT-Verfahren günstiger (bzgl. Zeit und Speicher) ist und, ob beide Verfahren überhaupt möglich sind (genug Speicher?). Nur im allerletzten Fall wird ein Junction Tree (temporär) als Sekundärstruktur erzeugt.
5. Liegt zusätzlich zu den Bedingungen in 4. eine oder mehrere Softevidenzen vor, so muss der Junction Tree mit Hilfe des BigClique-Algorithmus erstellt werden.

In einer erweiterten Version des Auswahlverfahrens werden auch approximative Inferenzverfahren berücksichtigt.

Erst nach der Auswahl eines Verfahrens kann das Verhaltensnetz kompiliert werden, d. h., die für das Inferenzverfahren benötigte Sekundärstruktur erstellt werden. Schließlich führt der Executer die optimierte Sequenz auf der Sekundärstruktur aus. Dabei werden nacheinander die Zustände des Netzes gesetzt und propagiert. Der Übergang von einem Zustand zum nächsten erfolgt nur durch die Änderung der unterschiedlichen Evidenzen und nicht durch vollständiges Löschen und Setzen aller Evidenzen. Das Inferieren in den Verhaltensnetzen kann durch Einführen spezieller Query-Nodes weiter beschleunigt werden. Durch MSL-Ausdrücke werden bereits die Query-Nodes eindeutig spezifiziert, so dass das Inferenzverfahren nicht vollständig über das gesamte Netz laufen muss, sondern nur bis die

Query-Nodes aktualisiert worden sind. Dadurch spart man im Idealfall 50% der Rechenzeit, falls die Query-Nodes in der Wurzel des Junction Trees stecken.

Nun gilt es ein geeignetes Modell für die gegebene Simulation zu erzeugen. Für jeden Agenten und für jeden Anwendungsfall wird eine Netzkonfiguration, also im Wesentlichen das Domänenwissen, über eine *Verhaltensnetzbeschreibungssprache* (***BDL*** *– Behaviour Network Description Language*) definiert und als Datei in einer Datenbank abgelegt. Neben dem Domänenwissen über die Struktur des Netzes werden außerdem der Lernprozess und die Simulation vorkonfiguriert.

Die Schritte der Simulation:

1. Netzkonfiguration
    - Domänenwissen
        - Kanten
            - Knotentupel (A, B)
            - Interpretation des Knotentupels: directed, undirected
            - Einschränkung des Suchraumes des Lernalgorithmus: forbidden, proposed, forced
        - Knoten
            - Einflussfaktoren/Dimensionen festlegen (damit auch minimale Zeitdimension / Zeitbasis)
            - Dimensionshierarchie (ohne Hierarchie: ContentType)
            - Einschränkung des Suchraumes des Lernalgorithmus: Input, Output, InOut, Inner
            - Diskretisierung
    - Lernkonfiguration
        - Lernintervall
        - Lernverfahren + Parameter
        - Art der Data-Mining-Tabelle
            - ein Eintrag pro Zeitbasis
            - kaufaktbasierend
            - andere Aggregationen
    - Simulationseinstellungen
        - Simulationsverfahren

Die Netzkonfiguration wird vor einer Simulation geladen und daraus ein noch nicht gelerntes Basisnetz erzeugt. Alle Parameter des Netzes können vor dem Lernprozess zur Laufzeit durch

manuelle Ergänzungen oder durch Anpassungen an das aktuelle Simulationsszenario überschrieben werden. So wird in der Regel das Lernintervall dem Simulationsszenario angepasst. Das Erzeugen eines Netzes unterteilt sich in die folgenden Schritte:

- Laden der Konfigurationsdatei
- Lernen des Netzes auf Basis des Domänenwissens und der Data-Mining-Tabelle
- Kompilierung des Netzes

Daraus folgt, dass das Ändern des Domänenwissens und der Lernkonfiguration vor dem Lernprozess erfolgen muss, dagegen werden die Simulationseinstellungen erst vor der Kompilierung festgelegt. Das Lernverfahren nutzt das Domänenwissen, um 1. den Suchraum einzuschränken und damit den Prozess zu beschleunigen und 2. um die Qualität des Netzes zu erhöhen. Ist ein Netz bereits vollständig und eindeutig in BDL beschrieben, muss der Lernprozess nur noch die Wahrscheinlichkeiten der CPTs ermitteln.

Die Kompilierung des Netzes wird erst dann durchgeführt, wenn eine Simulationsanfrage vorliegt, da sich die Sekundärstruktur auf Grund des Simulationsszenarios ändern kann. Andernfalls könnte es passieren, dass das Netz neu gelernt bzw. kompiliert werden muss.

2. Konfiguration des Simulationsprozess:
    - Simulationsprozess lässt sich über Szenarien mit Zeitintervallen oder direkt als MSL definieren.
    - Zeitbasis = über welchem Zeitknoten/-dimension soll simuliert werden.
    - Über welche anderen Knoten soll noch simuliert werden. Die letzten beiden Punkte sind nur bei Szenariodefinitionen wichtig, da dies in MSL-Ausdrücken explizit drin steckt.
    - (optional) Definition von Query-Nodes, um Inferenzmechanismen zu beschleunigen. Bei MSL Ausdrücken ist dies wiederum nicht nötig, da die Query-Nodes durch die Anfrage in Form von Rows, Columns etc. definiert werden.

Die von mir im Rahmen von SimMarket verwendeten Dimensionen / Einflussfaktoren sind:

Verwendete Dimensionen:

- Zeit = Dimension
    - Tag des Monats = Level
    - Abschnitt des Monats (z. B. Drittel oder Viertel)
    - Wochentag
    - Monat = Level => Jan, Feb, März … Members

- Simulationsentität
  - Artikel(-gruppen)
- Preise
  - Verkaufspreise
    - brutto
    - netto
    - brutto, rabattiert
  - Preis pro Mengeneinheit
  - relative Preise
    - Preiskategorien
    - Preis pro Mengeneinheit
  - Einkaufspreise
- Maßnahmen
  - Maßnahmenkategorien 1..A
- Zielvariablen
  - Abverkaufsmengen
  - Umsatz
  - Ertrag

Weitere mögliche Dimensionen sind:

- Rabatte
  - Rabattkategorien (kundenindividuelle Aktionen) 1..R
- Wetter
  - Wetterkategorien 1..W
- Konkurrenz
  - Preise (s. o.)
  - Maßnahmen (s. o.)
- Ökonomische Kennzahlen
  - Kaufkraft
  - Anzahl Arbeitslose
  - Wachstum des Bruttoinlandsproduktes
  - Bevölkerungswachstum

Für jede Dimensionsart muss definiert werden, ob und welche Aggregationen zulässig sind.

4.6.2 Simulationssprache MSL

Für ein vollständiges Simulationssystem benötigen wir noch die folgenden Komponenten:

1. Eine **Verhaltensnetzbeschreibungssprache** mit der insbesondere Domänenwissen vormodelliert werden kann,
2. eine **simulationsbasierte Anfragesprache**, um Wissen aus den instanziierten Simulationsmodellen ableiten zu können.

Ein standardisiertes Format, um prediktive Modelle zu beschreiben, bietet die *Data Mining Group (DMG)* mit der XML-basierten *Predictive Model Markup Language* (*PMML*) an. Die Mitglieder der DMG haben sich zum Ziel gesetzt, gemeinsam Data-Mining-Standards zu definieren. Mitglied sind alle bekannten und großen Hersteller von Data Mining Software[25]. Leider kommt PMML zurzeit nicht infrage, da Bayes'sche Netze oder ähnliche Modelle in der aktuellen Version nicht unterstützt werden. In SimMarket verwenden wir mit der *BDL (Behaviour Network Description Language)* ein eigenes Format.

Der interessantere Teil der genannten Komponenten ist die simulationsbasierte Anfragesprache, die nicht nur einfache Konfigurationen der Netze abfragen, sondern zudem komplexe Simulationen mit den Verhaltensnetzen ermöglichen soll. Dazu habe ich die ***M**ulti-dimensional **S**imulation **L**anguage* kurz *MSL-Sprache* entwickelt, wobei die n-dimensionale Struktur der Verhaltensnetze die Anforderungen an eine adäquate Anfragesprache definierte. Die mehrdimensionale Anfragesprache MDX (**M**ulti-**D**imensional E**X**pressions) von Microsoft ist eine mit SQL verwandte Sprache, um auf mehrdimensionale Daten z. B. Cubes in Data Warehouses zuzugreifen. MDX ist somit eine geeignete Grundlage für eine mehrdimensionale simulationsbasierte Anfragesprache.

Zunächst galt es, die geeigneten syntaktischen Konzepte zu isolieren und die Nomenklatur an die Verhaltensnetze anzupassen. Anschließend mussten die noch fehlenden Elemente hinzugefügt und die Semantik definiert werden.

---

[25] Mitglieder sind u. a. IBM, KXEN, Magnify, Microsoft, MicroStrategy, National Center for Data Mining, Oracle, Prudential Systems, Salford Systems, SAS, SPSS, Statsoft, Angoss Software, Insightful, NCR, Quadstone, Urban Science und SAP.

Abbildung der **MDX-Nomenklatur** auf die Verhaltensnetze:

- Member: Zustand eines Knotens, z. B. hat der Knoten Wochentag die Members Montag, Dienstag usw.
- Level: Hierarchiestufe der Knotenkategorien. Auf der untersten Ebene ein Member.
- Dimension: Oberklasse von Knoten z. B. hat die Dimension Zeit die Level Jahr, Quartal, Monat. Dimensionen können beliebig viele Level haben.
- Axis: Dimensionen der Rückgabewerte einer Query, wobei die ersten 5 einen Namen haben:
  - Columns / Spalten
  - Rows / Zeilen
  - Pages / Seiten
  - Chapters / Kapitel
  - Sections / Abschnitte
- Measure: Wahrscheinlichkeit eines Zustandes (=atomar) oder Erwartungswert (= Aggregation von Mengen)
- Dimensionalität: Eine Eigenschaft einer Menge von Level oder Members, die beschreibt von welcher Dimension und in welcher Reihenfolge diese sind.
- Tupel: Eine Menge von Members mit unterschiedlicher Dimensionalität, d. h. Members von unterschiedlichen Dimensionen. Tupel werden mit runden Klammern () gekennzeichnet.
- Set: Eine Menge von Tupeln mit gleicher Dimensionalität, d. h., sie haben die gleichen Dimensionen in der gleichen Reihenfolge. Sets werden durch geschweifte Klammern { } gekennzeichnet.
- Funktion: Eine Funktion kann Sets, Tupels, Members, Levels, Dimensionen oder Werte zurückgeben.
- Slice: Mit Slice bezeichnet man eine beliebige Teilmenge einer mehrdimensionalen Datenstruktur. Beispielsweise erhält man aus einem 3-dimensionalen Cube einen zwei-dimensionalen Slice (Tabelle), indem man eine Dimension durch das Festlegen einer Eigenschaft einschränkt.

Beispiele:

Die Definition der Syntax der Simulationssprache MSL ist in Anhang A zu finden. Die folgenden Beispiele sollen die Anwendung der Sprache verdeutlichen, sie umfassen die wichtigsten Anwendungsfälle.

Simulation eines einfachen Wertes:

```
Simulate
      { [products].[product1].[quantity].exp } On Columns
With
      [BNet]     //Verhaltensnetz
Where
      ( [products].[price].[1,99] ) //einschränkende Dimension
```

Simulation der Menge für alle Preise bei einer Promotion:

```
Simulate
      { ([products].[product1].[quantity].exp } On Columns,
      { [products].[price].members} On Rows
With
      [BNet]
Where
      ( [products].[action].[yes] )
```

Simulation der Menge über die Zeit (Wochentage):

```
Simulate
      { [products].[product1].[quantity].exp } On Columns,
      { [time].[weekday].members } On Rows
With
      [BNet]
```

Simulation der Menge über die Zeit (Wochentage) für alle Artikel:

```
Simulate
      { [time].[weekday].members } On Columns,
      { [products].members }
      *                                        //CrossJoin()
      { [measures].[AmountExp] } On Rows
With
      [BNet]
```

Simulation der Menge über die Zeit (Wochentage) für zwei bestimmte Artikel mit unterschiedlichen Preisen:

```
Simulate
      { [time].[weekday].members } On Columns,
```

```
      { ([products].[product1].[name],
      [product].[product1].[price].[exp] := '1,99' ),
      ([products].[product2].[name],
      [product].[product2].[price].[exp] := '2,39' ) }
      *
      { [measures].[AmountExp] } On Rows    //Measures
With
      [BNet]
```

Semantik von MSL

Der Block <resultDimensions> beschreibt die (mehrdimensionalen) Rückgabewerte der Query. In den meisten Fällen werden nur ein oder zwei Dimensionen, z.B. Abverkaufsmengen für alle Preis- und Promotionskombinationen, abgerufen, da die Ergebnisse in der Regel direkt an die grafische Benutzeroberfläche übergeben werden und höhere Dimensionen schlecht darstellbar sind. Ein Verhaltensnetz kann man sich auch wie eine mehrdimensionale Datenstruktur ähnlich eines Cubes vorstellen. Durch die in <resultDimensions> definierten Sets werden die vorhandenen Dimensionen eingeschränkt. Im Falle der Verhaltensnetze wird dazu ein Inferenzmechanismus benötigt, der die Auswirkungen der Einschränkungen durch Evidenzen auf die anderen Teile des Netzes berechnet. Das Setzen von Evidenzen ist auf verschiedene Weisen möglich. Normalerweise geschieht es bei den Verhaltensnetzen über die Angabe eines Erwartungswertes. Entspricht der Wert zufällig einem realen Mittelwert eines Intervalls, wird eine Hartevidenz gesetzt. In allen anderen Fällen wird eine Softevidenz gesetzt. Liegt der Wert unter dem bisherigen Minimum oder Maximum wird eine Extrapolationsevidenz gesetzt.

```
([products].[productX].[price].[exp] := '2,39' )
```

Diese Syntax ist die bevorzugte Methode um Evidenzen zu setzen. Der Benutzer muss sich in diesem Fall keine Gedanken über die Evidenzart machen. Die folgenden Methoden sind optional.

Setzen einer Hartevidenz auf einen Zustand über Indizes:

```
([products].[productX].[price].[states].[indices].[2])
```

Setzen einer Hartevidenz auf einen Zustand über den realen Mittelwert:

```
([products].[productX].[price].[states].[realavg].[1,99])
```

Setzen einer Wahrscheinlichkeitsverteilung für einen Knoten:

```
([products].[productX].[price].[distr] := '(1,0,0,0)' )
```

Im Falle der Artikelagenten haben die Verhaltensnetze standardmäßig die Dimensionen [time] und [products]. Letzter wird vor allem für Holonennetze benötigt, um auf die Artikelteilnetze zugreifen zu können. Nach der Auswahl eines Artikels über [products].[produktX] befindet man sich bereits auf der Knotenebene, auf der die bekannten Knotenwerte zugreifbar und manipulierbar sind. So beispielsweise der Erwartungswert und die Wahrscheinlichkeitsverteilung:

```
[products].[productX].[price].[exp]
[products].[productX].[price].[distr]
```

Wie oben zu sehen ist, kann man über den Level „[states]" auf die Zustände eines Knotens zugreifen. Damit kann man je nach Statement über eine direkte Referenz oder über den ‚:='-Operator eine Evidenz setzen (Wahrscheinlichkeit auf 100%). Das Auslesen der Erwartungswerte und Wahrscheinlichkeiten geschieht über Measures wie z. B. [measures].[AmountExp].

Der Funktionsumfang von MSL umfasst einige verhaltensnetzspezifische Funktionen, um auf Werte des Netzes zugreifen zu können. So lässt sich von einem Zustand die Wahrscheinlichkeit ([state].prop), der Durchschnitt ([state].avg), der reale Durchschnitt ([state].realavg), die linke Intervallgrenze ([state].lborder) und die rechte ([state].rborder) abrufen. Alle Member eines Levels lassen sich über [level].members abrufen. Die Beschreibung einer Menge (Set) kann durch *Filter* eingeschränkt werden. Mit filter (<set> , <searchCondition>) lässt sich die selektierte Menge durch boolesche Ausdrücke einschränken z. B. filter ({ [products].[price].members}, [prices] > 1,99). Mit Hilfe von *Ranges* kann man gezielt Teile einer Dimension als Menge definieren z. B. { [products].[product1] : [products].[product5] }.

Die wichtigsten Funktionen für die Beschreibung von komplexen Simulationen von MSL sind * (CrossJoin) und # (NonEmptyCrossJoin). Diese Operatoren veranlassen, dass aus zwei Mengen alle Konfigurationen abgeleitet und durchsimuliert werden. Wobei CrossJoin alle Kombinationen, auch falsche und unvollständige mit Platzhaltern für unmögliche Kombinationen ausgibt und NonEmptyCrossJoin nur korrekte Kombinationen simuliert.

Der *With*-Teil eines MSL-Ausdrucks bezeichnet das für die Simulation zu verwendende Verhaltensnetz.

4.6.3 Simulationsalgorithmen

Das Ergebnis des Simulationsprozesses ist eine Sequenz $Sq = \{Z_1,\ldots,Z_n\}$ von Netzzuständen (siehe Kapitel 4.6.1). Die Zustände haben die Eigenschaft, dass Z_i immer unabhängig von allen Z_{i+j} für alle j ist. Diese Bedingung ist nicht automatisch erfüllt, da beispielsweise eine Bewerbung eines Artikels in Zustand Z_i immer Auswirkungen auf die Zustände $Z_{>i}$ hat. Die Unabhängigkeit wird durch eine geeignete Konvertierung der Inputdaten bzw. durch das Einführen neuer Variablen erreicht. Bei obigem Beispiel wird eine Variable ‚Anzahl Tage nach einer Promotion' eingeführt. Das Lernverfahren ermittelt die Auswirkung dieser Variable auf alle anderen im Netz und beseitigt somit die Abhängigkeit aufeinander folgender Zustände durch das Einführen einer neuen Variable. Daraus folgt nun, dass ein geeigneter Simulationsalgorithmus alle Zustände $Z_i = \{Ev_1,\ldots,Ev_j\}$ nacheinander und unabhängig von einander setzen kann. Solch ein Algorithmus muss die folgenden drei Dinge realisieren:

1. Eine Methode, um vorliegende Evidenzen in Form von Potenzialänderungen in ein Verhaltensnetz zu integrieren,
2. ein Inferenzverfahren, um die Potenzialänderungen über das Netzwerk zu propagieren und
3. ein optimierendes Verfahren, das vorgegebene Zielwerte eines Netzes annähert.

Punkt zwei habe ich in SimMarket durch den Junction-Tree-Algorithmus (Kapitel 3.4.5) realisiert. Punkt 1 und 3 bedürfen einer genaueren Betrachtung.

In der Theorie der Bayes'schen Netze versteht man unter einen Zustand Z_i eine Konfiguration mit $\{X_1 = x_1, X_2 = x_2,\ldots,X_n = x_n\}$, d. h., die Variablen des Netzes werden auf einen bestimmten Wert festgelegt, z. B. wird die Variable ‚Wochentag' auf ‚Wochentag = Montag' festgelegt. In einen Netzzustand müssen nicht alle Variablen festgelegt werden. Den Vorgang, eine Variable X_i auf einen Wert x_i festzulegen, bedeutet anders ausgedrückt, dass die Wahrscheinlichkeit von X_i im Zustand x_i auf 1 bzw. 100% festgelegt wird. Deshalb bezeichnet man diese Art von Evidenz auch als *Hartevidenzen*. Das Setzen von Evidenzen lässt sich auf das Setzen einer beliebigen Wahrscheinlichkeitsverteilung verallgemeinern. Einer Variablen X_i wird nicht ein konkreter Zustand x_i sondern eine beliebige Verteilung $Distr_{Xi}(P(x_1), P(x_2),\ldots,P(x_n))$ zugewiesen, wobei $P(x_1) + P(x_2) +\ldots+ P(x_n) = 1$ ergeben muss. Für eine gegebene Wahrscheinlichkeitsverteilung $Distr_{Xi}$ existieren zwei Arten der Interpretation, als:

1. *virtuelle Evidenz*: Bei der virtuellen Evidenz wird eine Wahrscheinlichkeitsverteilung $Distr_{Xi}$ für eine Variable X_i anstelle einer festen Zuordnung $X_i = x_i$ gegeben, weil die beobachteten Ereignisse als unsicher angesehen werden. Eine gegebene Verteilung $Distr_{Xi}$ drückt die Unsicherheit einer

Evidenz für X_i aus. Bildlich gesprochen entspricht $Distr_{Xi}$ einer Gewichtung der aktuellen Verteilung von X_i. Je höher die Wahrscheinlichkeit eines Zustandes x_i ist, desto höher ist auch das Gewicht der Information in der gegeben Verteilung $Distr_{Xi}$. Nach dem Integrieren von $Distr_{Xi}$ in das Netz wird diese Information über den restlichen Teil propagiert. Die resultierende Verteilung von X_i ergibt sich aus der ursprünglichen Verteilung, der Evidenzverteilung und aus dem propagierten Potenzial aus anderen Bereichen des Netzes.

2. *Softevidenz*: Im Gegensatz zu einer virtuellen Evidenz drückt $Distr_{Xi}$ keine Unsicherheit aus, sondern das konkrete Wissen über eine veränderte Verteilung von X_i, d. h., die Verteilung von X_i soll nun exakt der gegeben Verteilung $Distr_{Xi}$ entsprechen. Beipeilweise ist bekannt, dass ein bestimmtes Ereignis nur freitags oder samstags eintritt, dann soll die Verteilung der Variable ‚Wochentag' folgendermaßen aussehen: $Distr_{Wochentag} = (0, 0, 0, 0, 0.5, 0.5, 0)$.

Je nach Evidenzart müssen unterschiedliche Integrationsverfahren angewendet werden.

4.6.3.1 Integration von Hartevidenzen

Das Setzen einer oder mehrerer Hartevidenzen P(X|e) ist in Kombination mit dem Junction-Tree-Algorithmus eine sehr einfache Operation. Das Integrieren von Hartevidenzen wird mit der folgenden Prozedur durchgeführt:

```
foreach(Clique_i , die Variablen mit Evidenzen e enthält )
{
φ_Clique_i = φ_Clique_i * φ_e (wobei φ_e die Hartevidenz ist);
}
Propagate();
NormalizeProbabilities();
```

Propagate() propagiert mit dem JT-Algorithmus die Evidenzen über das Netz und stellt die Konsistenz her. Nun können wir die Wahrscheinlichkeit von X **und** e mit

$$P(X,e) = \sum_{Clique_i \setminus \{X\}} \phi_{\text{Clique}_i} \text{ berechnen.}$$

Um P(X|e) zu erhalten, müssen die Potenziale des Netzes normalisiert werden (NormalizeProbabilities()):

$$P(X \mid e) = \frac{P(X,e)}{P(e)} = \frac{P(X,e)}{\sum_X P(X,e)}$$

Um die Propagierungsgeschwindigkeit und den Speicherverbrauch zu optimieren, wird eine spezielle Kompressionstechnik für Hartevidenzen, das so genannte *Evidence Shrinking*, verwendet. Wird eine Hartevidenz Ev auf einen Zustand Z_{xi} gesetzt, dann bedeutet das mathematisch gesehen, dass in allen Cliquen-CPTs, in denen der Knoten X_i vorkommt, die Fälle mit $X_i = x_i$ mit null multipliziert werden. Je mehr Hartevidenzen gesetzt werden, desto mehr Nullwerte enthalten die Cliquen-CPTs. Da Nullwerte beim Propagieren keine Auswirkungen haben, werden unnötige Berechnungen durchgeführt und Speicher verbraucht. *Evidence Shrinking* vermeidet das unnötige Berechnen mit Nullwerten, indem Cliquen-Strukturen beim Setzen von Hartevidenzen neu organisiert werden. Dimensionen, die auf Grund einer Hartevidenz nur Nullwerte enthalten, werden aus den CPTs herausgeschnitten. Dieses Vorgehen erfordert allerdings eine sehr flexible Datenstruktur für die Darstellung der CPTs. Zu berücksichtigen ist dabei außerdem, dass die Reorganisation ebenfalls mit Kosten verbunden ist. Insbesondere das Löschen von Evidenzen wird erschwert, da sehr einfach Dimensionen aus einer geeigneten Datenstruktur gelöscht, aber nicht unbedingt wieder hinzugefügt werden können. Je nach Datenstruktur kann das bedeuten, dass die CPTs nach dem Löschen von Evidenzen neu berechnet oder die alten Werte zwischengespeichert werden müssen. Dies kann die Vorteile vom Evidence Shrinking wieder aufheben. Das trifft insbesondere zu, wenn im Durchschnitt sehr wenige Evidenzen[26] gesetzt werden.

Die von mir für die Repräsentation von CPTs entwickelte Datenstruktur *CPTree* basiert auf Bäumen, bei denen jede Ebene einer Dimension der CPT entspricht. Die Blätter enthalten die tatsächlichen Wahrscheinlichkeiten. Die Reihenfolge der Dimensionen muss für alle CPT gleich, um konsistente Operationen zwischen CPT durchführen zu können. Die Baumstruktur erlaubt es sehr einfach Null-Dimensionen, die sich durch Hartevidenzen ergeben, herauszuschneiden. Diese Operation geschieht durch das Umbiegen von Pointern von Null-Dimensionen auf die darauf folgende Dimension. Dabei werden die Fälle der Null-Dimension zu einem zusammengefasst.

---

[26] Da bei der Artikelsimulation sehr viele Evidenzen gesetzt werden, hat sich in der Praxis gezeigt, dass das *Evidence Shrinking* bis zu 7-mal schneller ist (Artikel-Holonen-Netz mit 9 Artikeln => ca. 30 Knoten).

4.6.3.2 Integration von Softevidenzen

Die Integration von Softevidenzen ist mit Junction-Trees leider nicht ohne weiteres möglich, da das Propagieren im Junction-Tree aus zwei Schritten besteht: 1. aus CollectEvidence() und 2. aus DistributeEvidenz(). Absorbiert man die Information einer Softevidenz und propagiert man diese um die Konsistenz des Netzes wieder herzustellen, wird die absorbierte Evidenz nicht nur im Netz verteilt, sondern auch durch Potenzialänderungen an anderen Stellen des Netzes nachträglich verändert. Die durch eine Softevidenz festgelegte Verteilung eines Knotens wird durch das Propagieren unerwünschterweise verändert. Prinzipiell lässt sich eine Softevidenz jedoch mit einer generalisierten Form des Algorithmus für Hartevidenzen in ein Netz integrieren:

1. Wähle ein Cluster C aus, dass X enthält.
2. Setze $\phi_{C_X} = \frac{\phi_{C_X} \cdot \phi_{E_X}}{\phi_{E_{old}}}$,

wobei ϕ_{C_X} das Potenzial einer Clique C ist, die den Knoten X enthält. ϕ_{E_X} ist die zu integrierende Softevidenz und $\phi_{E_{old}}$ die Wahrscheinlichkeitsverteilung von X vor der Evidenz.

Das Problem ist nun, dass man die absorbierte Evidenz über das Netz propagieren muss, ohne die Verteilung selbst zu ändern. Valtorta, Vomlel und Kim geben in [ValKimVom01] zwei Verfahren an, um die Problematik des Junction-Tree-Algorithmus mit Softevidenzen zu lösen.

Iterative Modifikation des Junction-Tree-Algorithmus

Als erstes werden alle Hartevidenzen integriert und normal propagiert. Die sich ergebende Verteilung nennen wir Q, mit dem folgenden iterativen Algorithmus aktualisiert wird.

```
WHILE NOT converged
{
      FOR (j = 0; j < k; j++)
      {
            Q(C_lj) = P(C_lj) · P(O_j) / Q(O_j) ;
            JT.Propagate();
      }
}
```

Jede gegebene Softevidenz $P(O_j)$, $j = 1,\ldots,k$ wird in eine Clique C_{lj}, die O_j enthält mit der *Iterative Proportional Fitting Procedure* (*IPFP*) absorbiert und anschließend mit dem Junction-Tree-Verfahren propagiert.

Die Konvergenz dieses Verfahrens ergibt sich aus den allgemeinen Ergebnissen über die Konvergenz von IPFP, die in [Csi75], [Jir95], [Hab74] und [Vom99] zu finden sind. Wenn eine Lösung existiert, dann konvergiert das Verfahren gegen diese. Existiert keine Lösung, können zwei Dinge passieren, entweder das Verfahren alterniert zwischen zwei Lösungen oder das CPT-Potenzial geht durch Multiplikationen mit Nullwerten gegen null.

Dieses Verfahren ist sehr einfach zu implementieren und kann um einen vorhandenen Junction-Tree-Algorithmus gelegt werden. Es ist im Vergleich zu dem zweiten Verfahren sehr Platz sparend, hat aber das Problem der extremen Laufzeit. Für das einmalige Setzen einer Menge von Softevidenzen müssen wiederholt die Evidenzen absorbiert und die Propagierung ausgeführt werden, bis ein hinreichend genaues Ergebnis erreicht wird. Je länger das Verfahren läuft desto genauer wird das Ergebnis (*anytime*). Allerdings gibt es Spezialfälle, in denen das Verfahren bereits nach ein oder zwei Iterationen konvergiert.

Der BigClique-Algorithmus

Der große Nachteil des Junction-Tree-Verfahrens in Kombination mit Softevidenzen ist, dass die Propagierung nach jeder Absorption von Softevidenzen für das gesamte Netz durchgeführt werden muss. Eine Lösung für dieses Problem bieten Valtorta, Vomlel und Kim mit dem BigClique-Algorithmus in [ValKimVom01] an. Dieses Verfahren modifiziert den Junction-Tree derart, dass eine Propagierung nicht nötig ist, da alle Knoten mit Softevidenzen in einer Clique zusammengefasst werden:

1. Baue einen Junction-Tree, so dass alle Variablen, für die Softevidenzen gesetzt werden sollen, in einer (großen) Clique C_1 sind. Die Variablen dürfen zusätzlich auch in anderen Cliquen auftauchen. C_1 muss die Wurzel des Junction-Trees sein.
2. Setze alle Hartevidenzen und propagiere diese mit dem herkömmlichen Verfahren.
3. Absorbiere alle Softevidenzen in die BigClique C_1 mit dem im nächsten Abschnitt stehenden Verfahren.
4. Propagiere die Änderungen des Cliquen-Potenzials durch Aufrufen von DistributeEvidence() auf die unteren Ebenen des Baums.

Im 4. Schritt ist kein CollectEvidence() nötig, da keine Evidenzen in andere Cliquen absorbiert und die Hartevidenzen bereits im 2. Schritt propagiert worden sind. Valtorta, Vomlel und Kim zeigen sogar, dass in Schritt 2 nur CollectEvidence() aufgerufen werden muss, da die Potenzialänderungen von Hartevidenzen und Softevidenzen im 4. Schritt gemeinsam verteilt werden können. Die Frage, die Valtorta, Vomlel und Kim offen lassen, ist

wie man einen Junction-Tree mit einer BigClique erstellt. Ich benutze dazu das folgende Verfahren:

1. Moralisiere (siehe 3.4.5.4) den Graphen wie bisher.
2. Füge zwischen alle Knoten mit Softevidenzen ungerichtete Kanten hinzu. Dies stellt sicher, dass alle Knoten mit Softevidenzen in einer Clique landen.
3. Trianguliere den Graphen.
4. Baue den Junction-Tree mit dem bekannten Verfahren zusammen.
5. Nun muss die BigClique noch Wurzelkoten im Junction-Tree werden. Dazu werden alle Kanten, die von der BigClique zur Wurzel führen, in der Richtung umgedreht.

Mit diesem BigClique-Tree wird sichergestellt, dass alle betroffenen Knoten in der Wurzel-Clique sind und für den IPFP kein Propagate() in jedem Schritt durchgeführt werden muss. Der Nachteil dieses Verfahrens ist die mögliche Größe der BigClique. Dieses Verfahren kann man als Brute-Force-Methode ansehen, da der Vorteil einer komprimierten Darstellung der globalen Wahrscheinlichkeitsverteilung mittels Bayes'scher Netze teilweise aufgegeben wird. Dieses Verfahren ist in der Regel deutlich schneller als das Erste, allerdings ist der Platzbedarf deutlich höher. Zur Erinnerung, die Größe einer CPT wächst exponential mit der Anzahl der darin enthaltenen Knoten und ihrer Zustände. So kann es schnell passieren, dass der BigClique-Algorithmus unpraktikabel wird.

Absorption der Softevidenzen in die BigClique

Unter Absorption versteht man den Anpassungsprozess der gemeinsamen Wahrscheinlichkeitsverteilung $P(C_1)$, so dass diese mit den gegebenen Softevidenzen für die Variablen $A \subseteq C_1$ konform gehen, wobei $A = \{A_1, A_2, \ldots, A_k\}$. Sei $Q(C_1)$ die gemeinsame Wahrscheinlichkeitsverteilung nach der Absorption. Dann gilt $\forall i$

$$\sum_{C_1 \setminus A_i} Q(C_1) = P(A_i)$$

wobei $P(A_i)$ die Softevidenz auf A_i, $i = 1,\ldots,k$ ist. Die Absorption wird durch die Verwendung der Iterative *Proportional Fitting Procedure* (*IPFP*) realisiert. Jeder Zyklus besteht aus k Schritten, einer für jede Softevidenz. Die Formel lautet:

$$Q_{(0)}(C_1) = P(C_1)$$

$$Q_{(i)}(C_1) = \frac{Q_{i-1}(C_1) \cdot P(A_j)}{Q_{i-1}(A_j)}$$

wobei $j = (i - 1) \bmod k + 1$.

Auch mit dem BigClique-Algorithmus ändert sich nichts prinzipiell an den Konvergenzeigenschaften von IPFP. Bei einer konsistenten Eingabemenge konvergiert IPFP gegen die gegebenen Evidenzen. Bei einer inkonsistenten Eingabemenge ist keine Konvergenz sichergestellt. Dennoch ist es erwünscht, dass das Verfahren auch bei schwach konsistenten und inkonsistenten Evidenzen gegen möglichst „gute“ Verteilungen konvergiert. Dazu präsentiert Vomlel in seiner Arbeit [Vom99] fünf Modifikationen des IPFP, die er gegeneinander testet:

1. Conservative modification of IPFP with convex mixture.
2. Conservative modification of IPFP with log-convex mixture.
3. Method of iterative arithmetic averages.
4. Method of iterative geometric averages.
5. Generalized EM algorithm.

Fazit ist, dass der generalisierte EM-Algorithmus die besten Ergebnisse liefert, da er bei konsistenten Evidenzen die gleichen Ergebnisse liefert wie der IPFP und darüber hinaus bei inkonsistenten Daten am ehesten zur gewünschten Verteilung konvergiert. Beim *GEM-Algorithmus* handelt es sich um die generalisierte Variante des *EM-Verfahrens* (Expectation-Maximisation), das vor allem für das Lernen von Bayes’schen Netzen bei fehlenden Daten (engl. *missing values*) verwendet wird. Der Trick besteht darin, dass inkonsistente Evidenzen derart manipuliert werden, als wären sie aus unvollständigen Daten (engl. *incomplete data*) geschätzt worden. Andrej Kuklin hat im Rahmen der von mir betreuten Master Thesis [Kuk05] sowohl den IPFP, als auch den *Conservative modification of IPFP* implementiert. Er konnte keine signifikante Verbesserung in unseren Anwendungsfällen (Artikelsimulation) des zweiten Verfahrens gegenüber dem IPFP feststellen. Eine Evaluierung des GEMA in unserer Domäne steht noch aus.

4.6.3.3 Setzen von Erwartungswerten

Wie bereits erwähnt, können wir für jede Wahrscheinlichkeitsverteilung eines Knotens den *Erwartungswert* EW berechnen. Dies ist möglich, da nicht nur die Wahrscheinlichkeiten der Zustände, sondern auch die realen *Mittelwerte* der Intervalle, d. h., der Durchschnitt aller Lernfälle, die in dieses Intervall gefallen sind, in den Zuständen gespeichert werden. Kategorische Zustände wie z. B. die Wochentage werden intern in Double-Werte von 0,0 bis 6,0 abgebildet. Um alle Intervalle lückenfrei zu machen, werden diese immer folgendermaßen maximiert:

1. Vor der Maximierung muss gelten: $a_j \leq b_j$ und $b_j \leq a_{j+1}$ $\forall j$

2. Für alle Verteilungen mit Intervallen $[a_j, b_j]$ und $[a_{j+1}, b_{j+1}]$ für alle j = 0,..,n-1 wird a_0 auf –unendlich und a_{n-1} auf +unendlich gesetzt.
3. b_j wird auf a_{j+1} gesetzt, wobei $b_j < a_{j+1}$ interpretiert wird, wenn ein Vergleich mit Intervallgrenzen stattfindet.

Durch diese Maximierung können nun nicht nur beliebige Wahrscheinlichkeitsverteilungen gesetzt werden, sondern auch Evidenzen auf beliebige Werte v. Ein beliebiger Wert v_i eines Knotens X_i fällt nun immer exakt in ein Intervall. Diese Definition der Zustände garantiert, dass ein Netz für jeden beliebigen gegeben Wert einen gültigen Zustand besitzt. Allerdings erreicht man damit keine exakte Simulation von beispielsweise beliebigen Preisen, da jeder neue Preis einem vorhandenen Zustand zugeordnet werden würde. Man bekämme eine Prognose mit einem alten anstatt dem neuen Preis. Um auch unbekannte Werte korrekt zu simulieren, kann für jede Verteilung der Erwartungswert manipuliert werden. Eine gegebene Evidenz Ev_i mit einem Erwartungswert EW_i für einen Knoten X_i wird folgendermaßen interpretiert:

1. Fällt EW_i auf den Mittelwert $MW_{i,j}$ des j-sten Zustandes $x_{i,j}$ wird auf $x_{i,j}$ eine Hartevidenz gesetzt.
2. Ist $EW_i < MW_{i,0}$ wird eine Hartevidenz auf $x_{i,0}$ gesetzt, ist $EW_i > MW_{i,n+1}$ wird eine Hartevidenz auf $x_{i,\,n+1}$ gesetzt.
3. In allen anderen Fällen wird der Erwartungswert EW_i durch eine Verteilung $Distr_{EWi}$ angenähert. Liegt EW_i zwischen $MW_{i,j}$ und $MW_{i,j+1}$ erhalten die zugehörigen Zustände abhängig von den Entfernungen von $MW_{i,j}$, $MW_{i,j+1}$ zu EW_i anteilig positive Wahrscheinlichkeiten. Mit Hilfe der oben vorgestellten Softevidenzen wird eine Verteilung der Form

$$(0,...,0,\left(1-\frac{EW_i - MW_{i,j}}{MW_{i,j+1} - MW_{i,j}}\right),\left(\frac{EW_i - MW_{i,j}}{MW_{i,j+1} - MW_{i,j}}\right),0,...,0) \text{ gesetzt.}$$

Damit sind wir in der Lage beliebige Preise zwischen dem bisher vorgekommenen Minimum und Maximum zu interpolieren. Die Annäherung eines gegebenen Erwartungswertes kann natürlich mit unendlich vielen verschiedenen Verteilungen z. B. *Gaußverteilungen* etc. realisiert werden. Die hier vorgestellte Lösung ist eine einfache aber meiner Meinung nach sinnvolle Heuristik. Gerade bei Knoten auf deren Zustände eine Ordnung definiert werden kann z. B. bei Preisen erscheint es intuitiv sinnvoll nur den benachbarten Zuständen Wahrscheinlichkeiten zuzuordnen, da es unlogisch erscheint auch weiter entfernten Preisen (geringe) Wahrscheinlichkeiten zu vergeben.

Eine Fragestellung ist jetzt noch offen: Was macht man, wenn man einen Preis der kleiner als das bisherige Minimum oder größer als das bisherige Maximum ist, setzen möchte?

4.6.3.4 Extrapolationsevidenzen

Extrapolationsevidenzen (EP-Evidenz) kommen zum Einsatz, wenn Erwartungswerte EW_i außerhalb der vorliegenden Verteilung gesetzt werden sollen, d. h. $EW_i < \min\{MW_{i,j}\}$ oder $EW_i > \max\{MW_{i,j}\}$ für Knoten X_i und Zustände $x_{i,j}$ mit $i = 1,\ldots,l$ und $j = 0,\ldots,n-1$. Um dies zu realisieren, gibt es drei von mir entwickelte Verfahren:

1. Durch Extrapolation des gewünschten Erwartungswertes in der Lernbasis.
2. Nachträgliche Extrapolation des gewünschten Erwartungswertes mittels Simulation.
3. Durch Hinzufügen eines künstlichen Zustandes bei dem Knoten X_i, bei dem ein Erwartungswert $EW_i < \min\{MW_{i,j}\}$ oder $EW_i > \max\{MW_{i,j}\}$ gesetzt werden soll.

Verfahren eins lässt sich durch künstliches Hinzufügen benötigter Lernfälle zu der Datenbasis und Extrapolation der fehlenden Werte mit entsprechenden Verfahren erreichen. In vielen ähnlichen Anwendungsszenarien kommt häufig der sehr universell einsetzbare EM-Algorithmus zum Einsatz. Er hat sich als das beste Verfahren für die Behandlung von unbekannten und fehlenden Daten bewährt. Das EM-Verfahren wird dazu für die Extrapolation von unbekannten Werten erweitert, da in der Regel nur Werte interpoliert und nicht extrapoliert werden können. Dieses Verfahren hat den Nachteil, dass ein Netz beim Aufkommen einer entsprechenden Evidenz vollständig neu gelernt werden muss. Dieses Verfahren ist auch für die Realisierung von Softevidenzen geeignet.

Bei dem zweiten Verfahren wird ein bereits vorhandenes Netz vorausgesetzt. Dieses wird von seiner Struktur her nicht verändert, sondern selbst für eine Extrapolation verwendet. Dazu werden zunächst alle bekannten Evidenzen in das Netz eingegeben, d. h., die Dimensionen werden durch die vorhandene Information eingeschränkt. Nun werden nacheinander alle möglichen Kombinationen der verbliebenen Knoten, mit Ausnahme des zu extrapolierenden Knotens, generiert und gesetzt. Ist eine Konfiguration gesetzt worden, wird nun über alle noch möglichen Zustände des zu extrapolierenden Knotens mit Hartevidenzen iteriert. Dieses Verfahren wird nun für alle verbliebenen Netzkonfigurationen wiederholt. Daraus ergeben sich *n* Folgen von Erwartungswerten für den zu extrapolierenden Knoten. Aus dieser Menge an Werten kann nun der gewünschte Erwartungswert extrapoliert werden. Dazu muss entweder ein mehrdimensionales Extrapolationsverfahren wie z. B. die multiple Regression verwendet werden, oder die Menge der Folgen wird vorab durch Aggregation, z. B. mittels Durchschnittsbildung, in eine zweidimensionale Form gebracht. Die

zweidimensionale Folge der resultierenden Erwartungswerte des Knotens kann nun beispielsweise mit dem Least-Square-Verfahren (siehe unten) extrapoliert werden.

Das dritte – ebenfalls von mir entwickelte – Verfahren geht davon, dass bereits ein vollständiges Netz vorliegt. Dieses wird aber im Gegensatz zum zweiten Verfahren um einen neuen künstlichen Zustand erweitert. Abhängig davon, ob $EW_i < \min\{MW_{i,j}\}$ oder $EW_i > \max\{MW_{i,j}\}$ ist, wird links bzw. rechts der Verteilung ein neuer Zustand hinzugefügt. Wie bei dem BigClique-Algorithmus auch, muss diese Erweiterung vor der Kompilierung des Netzes, d. h. vor der Erzeugung des Junction-Trees, erfolgen. Als Folge der Erweiterung muss die CPT des Knotens und die CPTs aller Kinder dieses Knoten erweitert und extrapoliert werden. Da die CPTs in der SimMarket-Implementierung als Bäume (*CPTrees*) repräsentieren werden, ist die nachträgliche Erweiterung bereits gelernter CPT problemlos möglich.

Eine konkrete Ausprägung von Extrapolationsevidenzen nach dem dritten Verfahren sind die so genannten Least-Square-Evidenzen, die ihren Namen von dem Verfahren haben, dass angewendet wird, um die durch den neuen Zustand entstandenen Lücken in den CPTs zu extrapolieren. Durch das Hinzufügen eines neuen Zustandes entsteht in den betroffenen Cliquen für jede Kombination von Zuständen der Elternknoten und der eigenen Zustände eine Lücke. Der Wert dieser Lücke wird durch Anwendung des Least-Square-Verfahrens aus den vorhandenen Werten der Zustände des modifizierten Knotens extrapoliert. Wird nur eine EP-Evidenz gesetzt, entspricht das dem Least-Square-Verfahren im 2-dimensionalen Raum, welches allerdings für alle Dimensionen der Knoten-CPT wiederholt angewendet wird. Daraus folgt, dass zwar eine lineare Extrapolation durchgeführt wird, die aber auf einen n-dimensionalen Raum angewendet wird. Dadurch geht die nicht lineare Eigenschaft des Verhaltensnetzes nicht verloren. Anders ausgedrückt, das n-dimensionale nicht lineare Verhaltensnetz wird in viele 2-dimensionale nicht lineare Funktionen zerlegt, die dann mit einem linearen Verfahren extrapoliert werden.

Sollen mehrere nicht unabhängige EP-Evidenzen gleichzeitig gesetzt werden muss ein n-dimensionales Least-Square-Verfahren angewendet werden. Diese Verfahren sind in der Literatur unter der Bezeichnung *multiple Regression* bekannt. Wie bereits in 2.4.2 beschrieben, wird die multiple Regressionsanalyse auch für Prognosen verwendet. Hier können wir somit den neuen Ansatz der Verhaltensnetze mit einem bewährten klassischen Verfahren kombinieren. Die Vorteile der Verhaltensnetze bleiben dabei voll erhalten und gleichen zudem die Nachteile der Regressionsanalyse aus. Die positiven Eigenschaften der Regression werden für eine Verbesserung der Verhaltensnetze genutzt.

Das konkrete in SimMarket implementierte EP-Verfahren:

1. Selektiere alle zu extrapolierenden Slices aus einer CPT. Die vorhandenen Wahrscheinlichkeiten sind die (beobachteten) Werte der x-Achse. Die Durchschnittswerte der Knotenintervalle sind die Punkte der y-Achse.

2. Denormalisiere die Werte der selektierten Slices durch Multiplikation der Wahrscheinlichkeiten mit den Experience-Werten aus der Experience-Tabelle (siehe [Schwaiger06]), die beim Lernen des Netzes angelegt worden ist und stelle dadurch den initialen Wert der Wahrscheinlichkeiten her.
3. Wende das folgende Least-Square-Verfahren auf die Werte aller Slices an, um die fehlenden Werte berechnen zu können, d. h., gesucht wird eine Gerade (pro Slice), die durch die Werte der Wahrscheinlichkeiten gelegt wird und die Summe der quadrierten Abstände minimiert.
4. Berechne mit den gefundenen Geraden die fehlenden Werte der CPTs.
5. Normalisiere die Wahrscheinlichkeiten der Slices, so dass die Summe 1 ergibt.

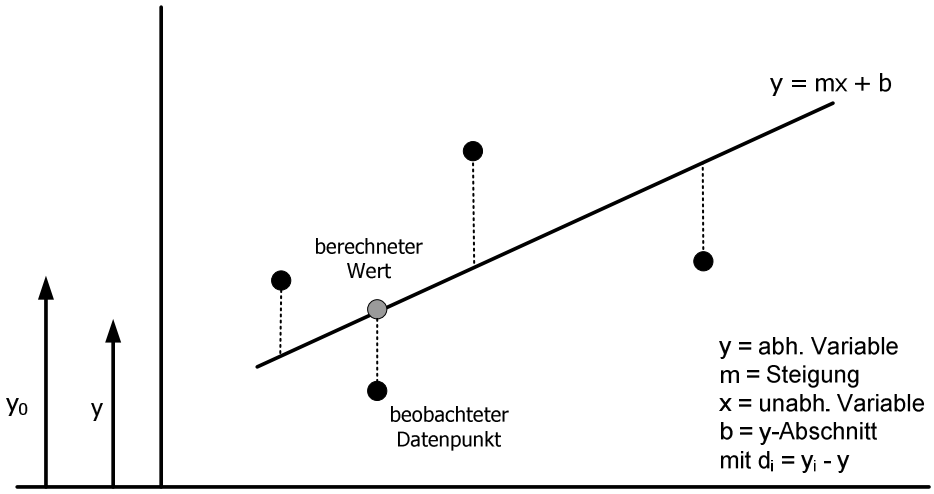


Abbildung 31: Das Least-Square-Verfahren

Das Least-Square-Verfahren, auch lineare Regression genannt, legt eine Gerade y = mx+b in die vorhandenen Datenpunkte y_i, so dass die Summe der quadrierten Abstände $\sum_i d_i^2$ mit $d_i = y_i - y$ der Datenpunkte von der Gerade y minimiert wird. Um die obige Summe zu minimieren, müssen die Konstanten m und b in der folgenden Gleichung

$$sum = \sum_{i=1}^{n} [y_i - (mx_i + b)]^2 ,$$

wobei n die Anzahl aller Punkte ist, derart ermittelt werden, dass *sum* minimiert wird.

Dies wird mit den folgenden beiden Formeln berechnet:

$$m = \frac{n \cdot (\sum_i x_i y_i) - (\sum_i x_i) \cdot (\sum_i y_i)}{n \cdot (\sum_i x_i^2) - (\sum_i x_i)^2},$$

$$b = \frac{\sum_i (y_i) - m \cdot (\sum_i x_i)}{n}$$

Ist die Gerade y=mx+b bekannt, kann jeder beliebige Wert auf der Geraden extrapoliert werden.

Als alternative Verfahren zu Least-Square können auch exponentielle oder quadratische Funktionen für eine Extrapolation der Wahrscheinlichkeiten verwendet werden.

4.6.3.5 Ausblick

Es existieren weitere Verfahren, um beliebige Werte als Evidenzen setzen zu können, die in weiteren Untersuchungen evaluiert werden sollten. Als Erstes wären *kontinuierliche Knoten* möglich, bei denen keine diskreten Werte in bedingten Wahrscheinlichkeitstabellen gespeichert werden, sondern Verteilungen in Form einer oder mehrerer Funktionen. Infolgedessen müssen bei kontinuierlichen Knoten keine Werte interpoliert werden, da immer der gesamte Wertebereich durch die Funktionen definiert ist. Es gibt Ansätze die CPTs durch *Künstliche Neuronale Netze (KNN)* zu repräsentieren, die für diese Anwendung besonders geeignet sind.

Weiterhin wäre es interessant die Kombination von Softevidenzen mit approximativen *Inferenzverfahren* zu untersuchen, da bei diesen Verfahren nicht die systembedingten Nachteile des Junction-Tree-Algorithmus auftreten, die nur durch Reorganisation mit einer BigClique umgangen werden können. Auch die Kombination mit anderen exakten Inferenzverfahren sollte näher evaluiert werden, z. B. mit der Konditionierungsmethode, da dort ebenfalls die genannten Probleme nicht auftreten.

4.6.4 Trendanalyse

Mit den bis hierhin beschriebenen Modellen und Verfahren sind bereits fast alle Anforderungen (siehe Abschnitt 3.5.1) an ein umfassendes Simulationssystem erfüllt. Die einzige Systemeigenschaft, die bisher noch nicht erfüllt ist, ist das Erkennen und Vorhersagen von Trends. Bisher handelt es sich bei dem Simulationsverfahren um ein nichtlineares Konstantmodell, dass keine Werte außerhalb des bisherigen Wertebereichs vorhersagen kann.

Bei den so simulierten Werten handelt es sich immer um den Erwartungswert, der auf der Basis der bisher vorgekommenen Ereignisse berechnet wird.

Die Analyse von Trends geschieht nicht in der Lern- und Modellierungsphase der Netze sondern wird als Teil des Simulationsprozesses durchgeführt. Eine Trendanalyse wird immer bezogen auf eine einzelne Variable, in unserem Fall immer die Abverkaufsmenge eines Artikels, durchgeführt. Das Ergebnis der Trendanalyse fließt in die Aufbereitung der Simulationsergebnisse ein. Das Verfahren besteht aus drei Schritten:

1. Nachdem ein Netz für eine Simulation erzeugt worden ist, wird nicht nur der in den Szenarien definierte Simulationszeitraum simuliert, sondern auch das gesamte Lernintervall. Das heißt, dass für jeden Lernfall in der Data-Mining-Tabelle durch Rekonstruktion der damaligen Situation ein simulierter Wert mit Hilfe des Netzes erzeugt wird. Es wird also sowohl Vergangenheit, Gegenwart als auch die Zukunft simuliert. Dadurch erhält man für jeden Lernfall mit einem realen Wert (der Abverkaufsmenge) auch einen simulierten.
2. Für jeden Vergangenheits- und simulierten Wert wird die absolute Differenz berechnet. Man erhält dadurch eine Menge von Punkten, die die Simulationsfehler darstellen. Auf diese Punktmenge wird ein einfaches klassisches Verfahren zur Schätzung einer Funktion wie z. B. die exponentielle Glättung oder das Least-Square-Verfahren angewandt. Dadurch erhält man eine Kurve des systematischen Fehlers der Simulation oder anderes ausgedrückt die Trendfunktion. Da die Funktion, die die Fehlerpunkte realisieren, bereits von allen Einflussfaktoren (engl. *Demand Influence Factors*) wie Saisons, Aktionen etc. bereinigt ist, kann und muss das angewandte Verfahren sehr einfach sein.
3. Die ermittelte Trendfunktion kann nun für die Korrektur von prognostizierten Werten verwendet werden. Dazu wird für jeden in der Zukunft simulierten Wert ein Trendfaktor mit Hilfe der Trendfunktion berechnet und der simulierte Wert damit angepasst.

4.7 Das SimMarket-System

Ich möchte nun einen Überblick über das SimMarket-System geben, indem zahlreiche Funktionalitäten realisiert wurden, die für verschiedene Anwendungsfälle notwendig sind. Die im Folgenden beschriebenen Hauptkomponenten sind der Artikel-, der Evaluationsmanager, der Simulations- und Prognosemanager und das BNetz-Tool, das die Verhaltensnetze aller Agenten realisiert und darstellt. Alle weiteren Komponenten sind in [Schwaiger06] ausführlich beschreiben.

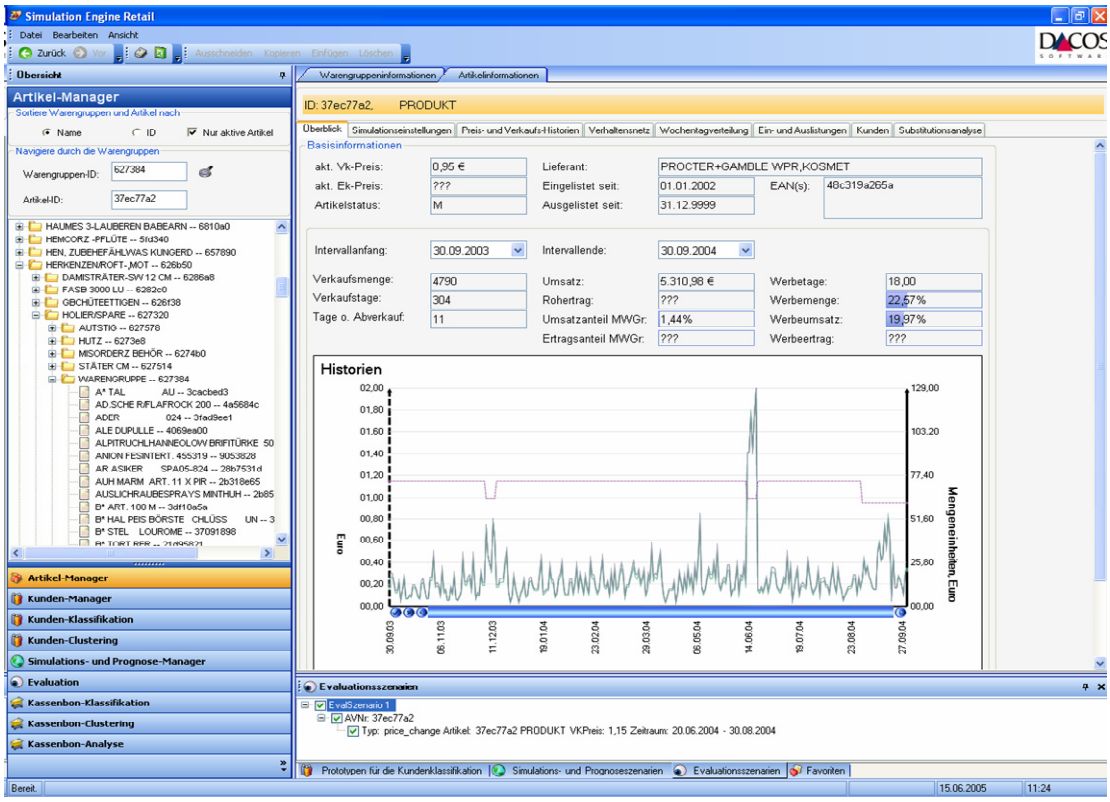


Abbildung 32: Das SimMarket-System

Abbildung 32 zeigt die grafische Benutzeroberfläche, wie sie für die Evaluierung der Konzepte dieser Arbeit (siehe Kapitel 5.3) verwendet wurde. Auf den Screenshots wurden, mit Rücksicht auf die Wünsche des Händlers, alle Ertragsrechnungen deaktiviert und die Bezeichner der Warengruppen und Artikel anonymisiert.

Die Oberfläche unterteilt sich in drei Hauptbereiche: die Auswahl der Hauptkomponenten links unten, ein von der gewählten Hauptkomponente abhängiges Navigationsfenster auf der linken Seite und der Informationsbereich auf der rechten Seite. Das Bedienkonzept orientiert sich dem aktuellen Standard Outlook 2003 und ist individuell an die Bedürfnisse des Benutzers anpassbar. Alle Bereiche der Oberfläche sind als andock- bzw. abdockbare Fenster, die ein- und ausblendet und beliebig in der Größe verändert werden können realisiert. Im rechten unteren Bereich können verschiedene Merklisten eingeblendet werden, z. B. Favoriten-, Szenarien- und Prototyplisten.

4.7.1 Der Artikelmanager

Der Artikelmanager bündelt alle Informationen und Sichten für die Planung und Konfiguration von artikelorientierten Analyse- und Simulationsszenarien. Nach Aktivierung

des Artikelmanagers erscheint im Navigationsfenster die vollständige Warengruppenhierarchie des Händlers, die per Maus bis auf die Artikelebene durchnavigiert werden kann. Durch Eingabe einer Warengruppennummer kann man die entsprechende Warengruppe (WGR) direkt anzuspringen. Durch Anwahl einer WGR erhält der Anwender zahlreiche Daten über diese Warengruppe, z. B. Umsatz, Ertrag, Anzahl der Artikel und die Wochentagsverteilung der Absatzmenge aller Artikel der WGR. Außerdem können statistische Daten über die Artikel der WGR tabellarisch angezeigt und sortiert werden. Unterhalb der untersten WGR, der so genannten Miniwarengruppe (MWGR), erscheint eine Liste aller Artikel, die ebenfalls angewählt werden können. Wie in Abbildung 32 zu sehen erhält der Benutzer zahlreiche Basisinformationen, die sowohl Stammdaten, als auch berechnete Kennzahlen umfassen. Die Kennzahlen werden online, bezogen auf das ausgewählte Intervall, aus der Datenbank, hauptsächlich aus den Transaktionsdaten der Kassen, berechnet. Die Zusammenstellung der Kennzahlen wurde mit den potenziellen Anwendern des Handelsunternehmens Globus St. Wendel in mehreren Workshops erarbeitet und sie umfassen alle wichtigen Daten, die auf einen Blick eingesehen werden müssen, z. B. Umsatz, Absatzmenge, Rohertrag, Anzahl an Promotionstag, prozentualer Anzahl der Abverkäufe während einer Promotion usw. Unter den Kennzahlen ist ein Diagramm, das den Verlauf der Absatzmenge, des Umsatzes, den Ertrags und der Verkaufspreise (VKPreis) grafisch darstellt. Die einzelnen Kurven können ein- und ausgeblendet werden. In weiteren Ansichten können die Wochentagsverteilung des angewählten Artikels und die Daten seiner Ein- und Auslistungen eingesehen werden. In der artikelbezogenen Kundensicht (Abbildung 33) wird die prozentuale Verteilung der Kundengruppen, die den ausgewählten Artikel gekauft haben, angezeigt. Alle identifizierbaren Kunden z. B. über EC-, Kredit- oder Kundenkarten werden auf der Basis von statischen Kennzahlen, von Klassifikationsalgorithmen oder Clusterings in eine disjunkte Typologie eingeordnet. Die Transaktionen dieser Kunden werden mit der zugehörigen Kundennummer annotiert, so dass für jeden Artikel alle Transaktionen mit Kundennummer und der entsprechenden Kundengruppe des Kunden extrahiert und infolgedessen die prozentuale Verteilung der Gruppen berechnet werden kann. Mit Hilfe dieser Ansicht kann der Sortimentsverantwortliche auf einen Blick das Käuferprofil der Kunden eines Artikels ermitteln und kann daraus schließen, welche Kundensegmente durch eine Änderung an diesem Artikel (z. B. durch eine Preiserhöhung) positiv oder negativ beeinflusst wird. Verfolgt der Sortimentsmanager eine gezielte Kundengruppenstrategie z. B. die Vergrößerung der Gruppe „Gelegenheitskäufer“, dann ist diese Sicht hilfreich bei der Identifizierung von Artikeln, die besonders bei Gelegenheitskäufern beliebt sind.

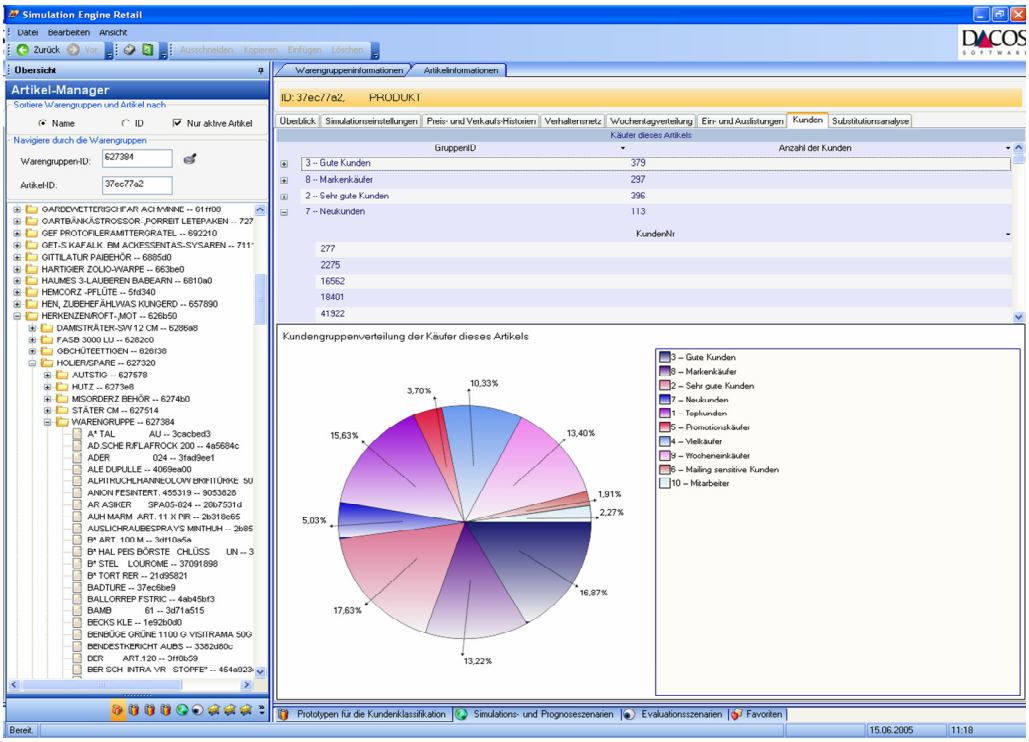


Abbildung 33: Kundengruppenverteilung eines Artikels

Möchte der Anwender ein Analyse- oder Simulationsprojekt beginnen, wechselt er in die Ansicht „Simulationseinstellungen“. Dort werden für den ausgewählten Artikel Szenarien für eine Analyse der Vergangenheit oder eine Prognose der Zukunft definiert, indem der Benutzer für ein beliebiges festzulegendes Zeitintervall eine Maßnahme (z. B. eine Preisänderung und/oder eine Bewerbung) einem Szenario zuordnet. Für jeden Artikel können beliebig viele Szenarien mit unterschiedlichen Maßnahmen definiert und anschließend durch Simulation miteinander verglichen werden. Die eigentliche Simulation wird im Simulations- und Prognosemanager durchgeführt.

Maßnahmen wie Preisreduzierung und Bewerbungen haben immer Auswirkungen auf andere Artikel des Sortiments. Einige Artikel werden durch Cross-Selling-Effekte besser abverkauft, andere werden durch den modifizierten Artikel kannibalisiert. Solche Effekte zwischen Artikeln können durch eine Abhängigkeitsanalyse ermittelt und anschließend in den Simulationen berücksichtigt werden. In der entsprechenden Ansicht kann eine solche Analyse in zwei Stufen durchgeführt werden: Als Erstes liefert eine Korrelationsanalyse zwischen dem gewählten Artikel und einer definierbaren Gruppe von anderen Artikeln erste Hinweise auf Abhängigkeitsbeziehungen zwischen Artikeln.

Korrelationsmaße sind für diese Fragestellung besonders gut geeignet, da sie nicht die (absolute) Ähnlichkeit von Abverkaufskurven wie z. B. Vektormaße ausdrücken, sondern nur

etwas über proportional ähnliche Veränderungen der Kurven zweier Artikel aussagen. Beispielsweise drückt eine Korrelation aus, dass, wenn Artikel A sich besser verkauft, Artikel B sich immer schlechter verkauft ohne absolute Quantitäten zu berücksichtigen. Da man mit den Korrelationsmaßen keine quantifizierten Prognosen durchführen kann und diese Kennzahlen sowieso über alle Ereignisse der Vergangenheit hinweg ermittelt werden, konkretisiert die zweite Stufe der Abhängigkeitsanalyse die Verdachtsmomente der ersten Stufe mit einer Verhaltensnetzanalyse. Dazu werden die am stärksten korrelierenden Artikel mittels Holonisierung (siehe Kapitel 3.4.7) zu einem Holon verschmolzen. Das Metanetz des Holonen kann nun für eine dedizierte Analyse von beliebigen Einflussfaktoren eines Artikels auf die Effektknoten der anderen Artikel genutzt werden. Beispielsweise kann man damit die Wirkung einer Preisänderung eines Artikels X auf alle anderen Artikel des Holonen quantifiziert an den Änderungen der Erwartungswerte der Mengenknoten aller Artikel ablesen.

4.7.2 Der Simulations- und Prognosemanager

Im Simulations- und Prognosemanager (SimProgManager) ist statt der Warengruppenhierarchie das aktuelle Projekt mit den definierten Szenarien im Navigationsbaum zu sehen. Durch Auswahl der Szenarien können die Einstellungen dieser im Hauptfenster eingesehen und unabhängig voneinander konfiguriert werden.

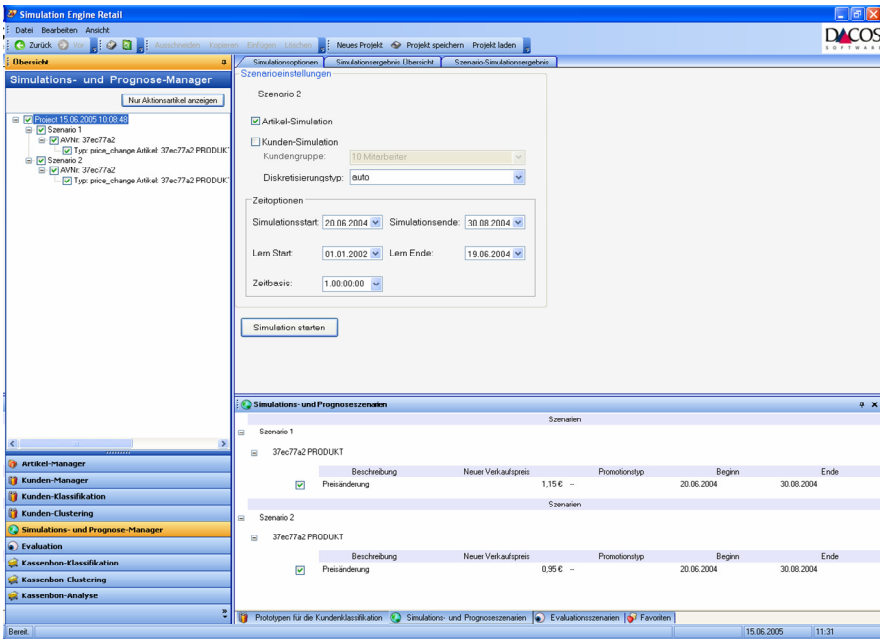


Abbildung 34: Konfiguration einer Simulation

Im Hauptfenster lassen sich der Simulationszeitraum, das Lernintervall der Verhaltensnetze, der Diskretisierungstyp für die Trainingsmenge und die Zeitbasis der Simulation verändern (siehe Abbildung 34). Die Zeitbasis wird in Tagen angeben und definiert die Größe eines Simulationsschrittes. Die Zeitbasis 1 für eine tageweise und die Zeitbasis 7 für eine wochenweise Simulation. Zusätzlich kann man im Hauptfenster eine Kundengruppe auswählen, um die Simulation auf diese eingeschränkte Menge von Kunden anzuwenden und somit deren Veränderungen des Kaufverhaltens ermitteln.

In Abbildung 35 ist das Ergebnis einer Artikelsimulation mit Absatzmenge, Umsatz und Ertrag tabellarisch für jedes Szenario dargestellt. Des Weiteren sieht man das Ergebnis des klassischen Prognoseverfahrens „exponentielle Glättung 2. Ordnung“ im Vergleich zu dem Simulationsergebnis. Dadurch wird dem Betrachter sofort verdeutlicht, dass der simulative Verhaltensnetzansatz den meisten klassischen Verfahren weit überlegen ist. Durch Aufklappen eines Szenarioeintrages kann man die Werte jedes einzelnen Simulationsschrittes detailliert einsehen. In weiteren Ansichten werden die Simulationsergebnisse grafisch in Form von Kurven dargestellt.

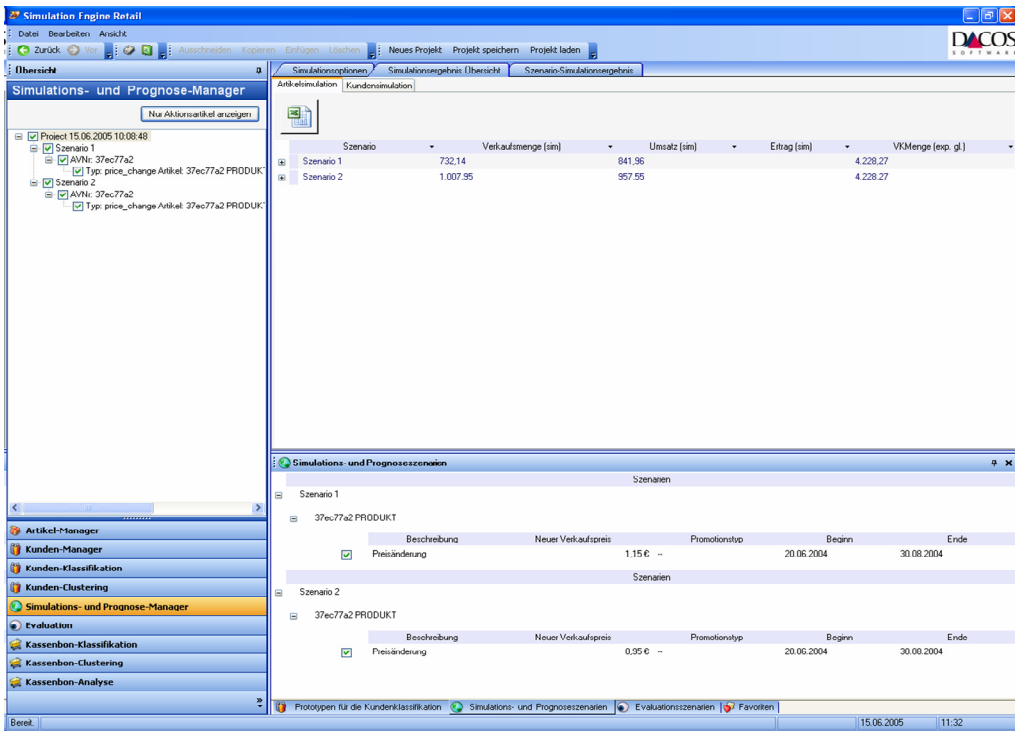


Abbildung 35: Ergebnis einer Simulation

4.7.3 Der Evaluationsmanager

Der Evaluationsmanager ist mit dem Simulations- und Prognosemanager vergleichbar, da man manuell definierte Szenarien simulieren und sich die Ergebnisse anzeigen lassen kann. Im Unterschied zum SimProgManager kann man mit ihm jedoch nur Szenarien der Vergangenheit simulieren. Dazu können beliebige Situationen der Vergangenheit in ein Szenario übernommen und nachsimuliert werden. Das simulierte Ergebnis wird anschließend mit den realen Werten aus der Datenbank abgeglichen und verschiedene Prognosegütemaße berechnet. Um zu einer realistischen Simulation zu kommen, werden den Agenten nur die Daten bis zum Beginn der Simulation zur Verfügung gestellt.

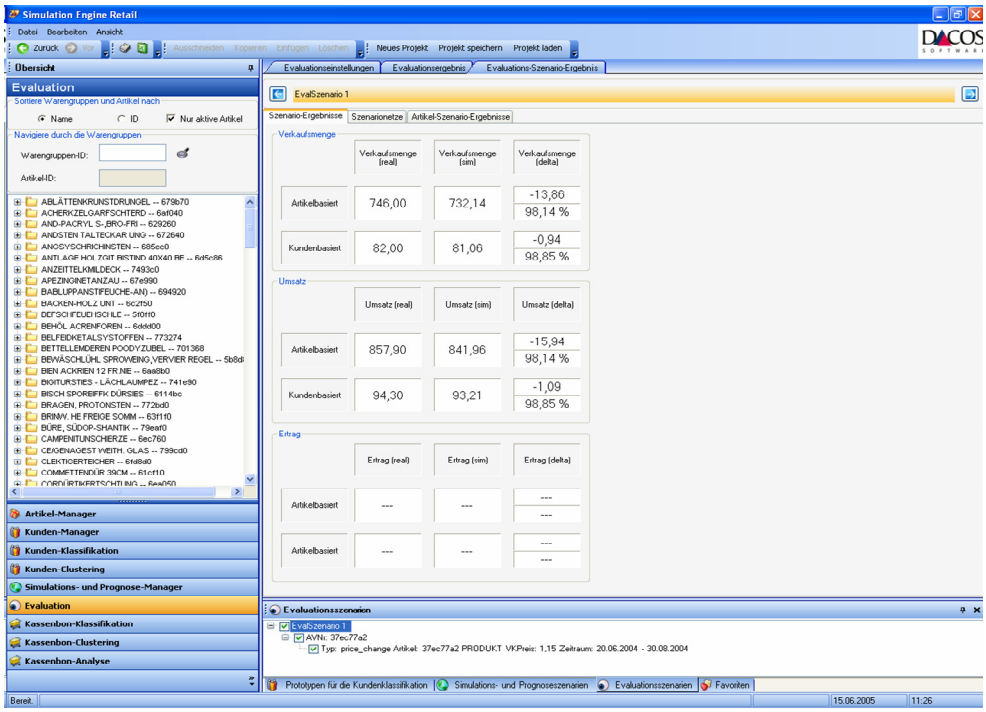


Abbildung 36: Ergebnisse einer Evaluation

Außer der manuellen Simulation hat der Evaluationsmanager auch eine Komponente für das automatische Erstellen von Evaluationsszenarien. In einer entsprechenden Ansicht lassen sich zahlreiche Bedingungen definieren, die ein Artikelszenario erfüllen muss, um dem aktuellen Projekt hinzugefügt zu werden. Zunächst wählt man eine Menge von Artikeln oder eine Warengruppe auf beliebiger Hierarchieebene aus und fügt sie dem Projekt zu. Wird eine WGR ausgewählt so werden alle Artikel, die zu dieser Gruppe gehören unabhängig von der Hierarchieebene dem Projekt hinzugefügt. Dabei kann man auswählen, ob die Artikel alle in ein Szenario zusammengefasst oder in unterschiedliche Szenarien verteilt werden sollen. Artikel, die im gleichen Szenario liegen, werden bei der Simulation automatisch zu einem Holon verschmolzen. Alternativ existiert auch eine Komponente, die aus der Menge aller

Artikel in der Datenbank die für eine Simulation geeigneten Artikel heraussucht und dem Projekt in Form von neuen Szenarien hinzufügt. Dazu können zahlreiche Qualitätsanforderungen und Einschränkungen an die Artikel und ihren Daten definiert werden:

- Minimale Länge des Abverkaufszeitraums,
- Maximale Abverkaufslücke,
- Anzahl an Preisinkonsistenzen zwischen der Artikelhistorie (siehe Anhang D Tabelle ItemHistory) und den Transaktionsdaten,
- Test auf Promotionsinkonsistenzen, d. h., es wurde eine Promotion durchgeführt aber die entsprechende Aktionsnummer nicht in der Artikelhistorie gesetzt,
- Auswahl der Hauptkategorie: Food oder Drogerie,
- Auswahl des Artikeltyps: Schnell- oder Langsamdreher, wobei die Definition von Langsamdreher ebenfalls in einem Feld eingestellt werden kann,
- Beschränkung auf eine Miniwarengruppe.

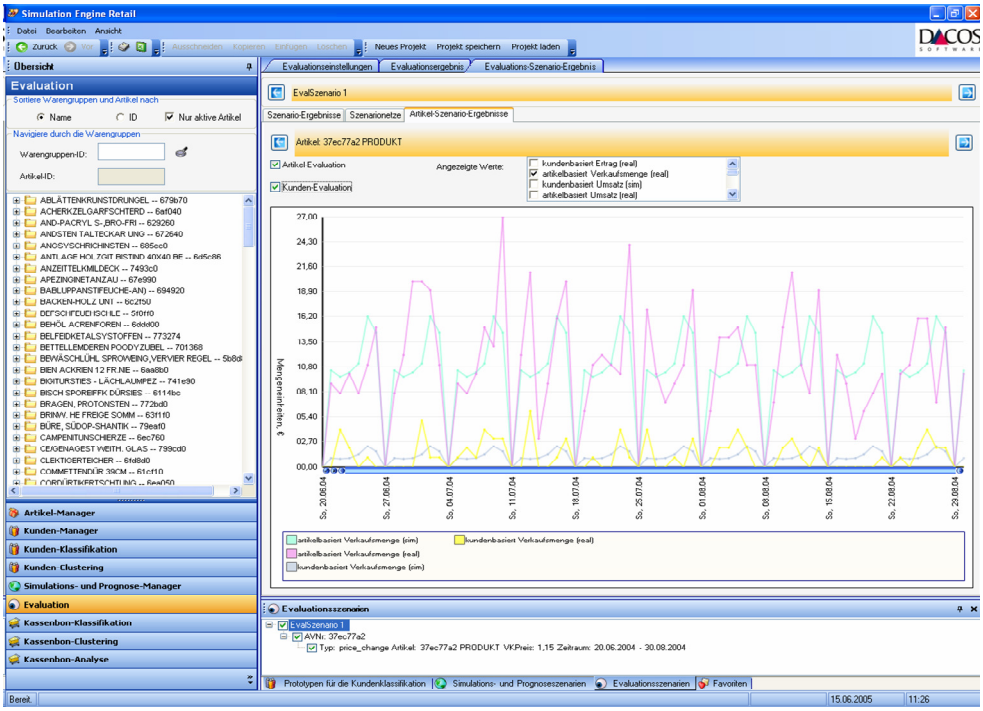


Abbildung 37: Grafische Darstellung von Evaluationsergebnissen

Nun müssen für die ausgewählten Artikel geeignete Artikelszenarios in der Vergangenheit, also in der Datenbank, gefunden und den Szenarios hinzugefügt werden. Folgende Bedingungen können für die Suche nach Szenarien definiert werden. Werden diese nicht manuell gesetzt, verwendet der Evaluationsmanager Default-Werte.

Einstellungen für die Suche nach Szenarien:

- Artikel- und/oder Kundensimulation (Kundengruppe)
- Such- und Lernintervall als Gesamtintervall oder
- Such- und Lernintervall getrennt voneinander,
- Maximale Anzahl an Simulationstagen,
- Suchstrategie: Minimierung oder Maximierung des Lernintervalls bei der Definition eines Gesamtintervalls,
- Zeitbasis der Simulation,
- Diskretisierungsalgorithmus für die Lerndaten.
- Flag: Simulationsintervall muss nicht direkt an das Lernintervall anschließen, d. h., es darf eine Lücke zwischen den Intervallen geben. Diese Option ist vor allem in der Kombination mit der folgenden Einstellung sinnvoll.
- Flag: Alle Artikel im gleichen Zeitraum simulieren, diese Option ist für eine Simulation mit Holonen Voraussetzung.
- Flag: Alle möglichen Szenarien erstellen, d. h., es wird nicht nur ein Artikelszenario gesucht, das auf die nachfolgenden Bedingungen passt, sondern alle.

Einstellungen für die Suche nach Artikelszenarien:

- Art der zu simulierenden Ereignisse:
    - Preisänderungen
    - Bewerbungen
    - Alle Arten
- Art und Anzahl der Evidenzarten, d. h. viele Situationen mit Änderungen an einem Produkt, die mit Hart- und/oder Softevidenzen simuliert werden können, gefunden werden sollen.
- Flag: Alle Evidenzen durchgängig setzen, d. h., es werden nicht nur die Zeiträume simuliert, für die ein passendes Artikelszenario gefunden worden ist, sondern auch die Intervalle dazwischen.

Nach Definition der Bedingungen sucht der Evaluationsmanager nach passenden Artikelszenarien in der Datenbank, fügt diese den entsprechenden Szenarien hinzu und startet die Evaluation. Das Ergebnis entspricht dem der manuellen Evaluation.

4.7.4 Das SimMarket-BNetz-Tool

Das SimMarket-BNetz-Tool ist die grafische Visualisierung und Konfigurationswerkzeug für die Verhaltensnetze der Agenten. Es besteht aus zwei Hauptkomponenten: der Graph- und der Diagrammansicht. Die Graphansicht (Abbildung 38) dient zur Darstellung und Manipulation der Topologie der aus den Realdaten extrahierten Verhaltensnetze und zeigt sowohl die Einfluss-, als auch die Effektknoten und deren ermittelten Abhängigkeiten zwischen einander an. Für eine bessere Übersichtlichkeit können die Knoten mit den Kanten beliebig gezoomt, verschoben und angeordnet werden. Des Weiteren kann in dieser Ansicht auf zahlreiche Optionen des Netzes über ein Kontextmenü zugegriffen werden. Knoten und Kanten können gelöscht, hinzugefügt, Knoten- und Kantenattribute geändert und die Data-Mining-Tabelle angezeigt werden.

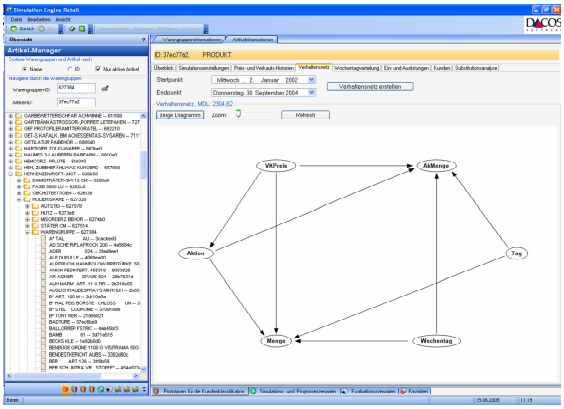


Abbildung 38: Graphansicht des BNetz-Tools

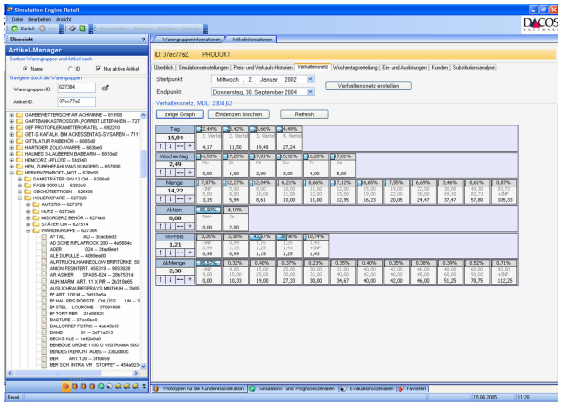


Abbildung 39: Diagrammansicht des BNetz-Tools

Die Diagrammansicht ist eine andere Darstellungsform der Verhaltensnetze und dient dazu, die relevanten Kennzahlen des Netzes anzuzeigen und erlaubt das Setzen und Propagieren von Evidenzen. Jede Zeile in dieser Ansicht entspricht einem Knoten in der Graphansicht, das erste Element enthält den Namen des Knotens und dessen Erwartungswert, der aus der rechts daneben stehenden Wahrscheinlichkeitsverteilung berechnet worden ist. Die Verteilung setzt sich aus einer Menge von Zuständen, die die zugehörige Variable annehmen kann, zusammen und entspricht den Werten der Variable in der Trainingsmenge, die im kontinuierlichen Fall auf eine handhabbare Anzahl von Intervallen diskretisiert werden. Der Knoten „Wochentag“ hat beispielsweise eine eingeschränkte Menge von Zuständen: „Montag“, „Dienstag“, „Mittwoch“ usw. und muss nicht diskretisiert werden, für den Knoten „Menge“ dagegen, gilt diese Aussage nicht. Ein Zustand besteht aus dem Intervall, das er repräsentiert, dem realen Mittelwert und der zugehörigen Wahrscheinlichkeit. Auch kategorische Variablen wie „Wochentag“ werden intern als Double-Werte

repräsentiert, so dass auch hier die Berechnung eines zugegebenermaßen nicht interpretierbaren Erwartungswertes möglich ist. Da alle Intervalle lückenlos sein müssen, werden Intervalle von kategorischen Variablen künstlich nach rechts bis an den Anfang des darauf folgenden Intervalls erweitert. Aus dem gleichen Grund wird der Anfang des ersten und das Ende des letzten Intervalls immer auf –unendlich bzw. +unendlich gesetzt. Diese Intervallausdehnungen ändern nichts an den Simulationsergebnissen, da der Erwartungswert immer auf der Basis des realen Mittelwerts (der Trainingsmenge) berechnet wird. Die Wahrscheinlichkeit eines Zustandes ergibt sich aus der Häufigkeit in der Trainingsmenge.

In der Diagrammansicht gibt es vier verschiedene Möglichkeiten Evidenzen in das Netz einzugeben:

1. Durch Anklicken eines oder mehrerer Zustände wird auf diesen eine Hartevidenz gesetzt.
2. Durch Anklicken des Knotennamens wird ein Fenster geöffnet, das die Eingabe einer Wahrscheinlichkeitsverteilung erlaubt, die dann mittels Softevidenz in das Netz integriert wird.
3. Durch Anwahl des aktuellen Erwartungswertes eines Knotens lässt sich in einem erscheinenden Fenster ein neuer Wert eingeben, der ebenfalls mit Hilfe einer Softevidenz angenähert wird.
4. Über einen speziellen Button lässt sich ein künstlicher Zustand hinzufügen, der dann mittels Extrapolationsevidenz in das Netz integriert wird.

Nach Eingabe einer Evidenz wird diese über das Netz propagiert und beeinflusst damit alle abhängigen Variablen im Netz, was sofort durch Veränderung der Erwartungswerte der anderen Knoten sichtbar wird.

5 Anwendungsfälle und Evaluierung

5.1 Einleitung

Die praktischen Erfahrungen und zahlreichen Gespräche mit Experten des Handels haben gezeigt, dass es in der Domäne der Artikelsimulation kein allgemeingültiges Modell, dass das Abverkaufsverhalten aller Artikel annähernd korrekt beschreibt, geben kann. Das Abverkaufsverhalten eines Artikels ist ebenso komplex und vielfältig wie die Menschen, die diesen Artikel kaufen und damit das Abverkaufsverhalten bestimmen. Die einzige Chance, die man bei der Modellfindung hat, ist die Nutzung der vorliegenden harten Fakten – die Daten des Artikels. Gelingt es aus diesen Daten ein individuelles Modell für einen Artikel zu generieren, hat man die Möglichkeit zu exakten quantifizierten Aussagen zu kommen. Folglich steht und fällt nicht nur die Qualität der Simulationsergebnisse mit der Datengrundlage sondern auch die Komplexität der Fragestellungen die beantwortet werden können. Je komplexer die zu beantwortenden Fragestellungen sind, desto komplexer und umfangreicher muss auch die benötigte Datengrundlage sein. Bekanntermaßen lassen sich mit einer unären Abverkaufshistorie bereits einfache Prognosen durch die Extrapolation der Abverkaufskurve realisieren. Die Realisierung einer komplexeren Artikelsimulation erfordert dagegen andere Anforderungen an die Daten. Das heißt folglich, dass nicht alle Anwendungsfälle mit allen Artikeln des Sortiments möglich sind, da bei vielen die Datengrundlage nicht ausreichend ist. Dies ist aber nur bedingt problematisch, da in der Regel komplexe Fragestellungen – beispielsweise eine Preisoptimierung – auch nur bei bestimmten Artikeln, bei denen sich der Preis oft ändert, erforderlich sind. Die Optimierung des Preises eines Artikels, der immer den gleichen Preis in der Vergangenheit hatte, ist nicht sinnvoll. Diese Tatsache erschwert allerdings die systematische Evaluation des Systems. Für eine Evaluation müsste man eigentlich das SimMarket-System auf alle Artikel eines Händlers anwenden und die möglichen Fragestellungen mit verschiedenen Parametern wie z. B. unterschiedlichen Preisen, Promotionen, Zeitintervalle etc. und Kombinationen davon, testen. Mann kann aber eine Evaluation nur durchführen, indem Situationen der Vergangenheit in der Simulation rekonstruiert und den Zugriff auf die Lerndaten bis zum Beginn des Simulationszeitraumes beschränkt werden, sodass das System über den eigentlichen Simulationszeitraum keine Informationen besitzt. Dies schränkt natürlich die Art der evaluierbaren Fragestellungen pro Artikel auf die tatsächlich in der Vergangenheit vorgekommenen Fälle ein. Des Weiteren sind auch in diesen Fällen komplexe Fragestellungen nur dann zulässig, wenn die verfügbare Datenmenge vor dem Simulationszeitraum hinreichend umfangreich ist. Und schließlich spielt auch die

Datenqualität bei der Evaluation eine große Rolle, da die Evaluationsergebnisse nur dann einen Rückschluss auf die Güte des Modells zulassen, wenn die Lerndaten korrekt, vollständig, plausibel und konsistent sind.

Für die Evaluation des *Globus*-Systems (siehe Kapitel 5.3) wurden zunächst alle sinnvollen Warengruppen aus den Bereichen Food und Drogerie herausgesucht. Diese beiden Gruppen decken die Sortimente der anderen Partner *dm-drogerie markt* und *tegut…* vollständig und auch einen Großteil des Kernsortiments von *Globus* ab. Non-Food-Gruppen wie z. B. Fashion und Baumarktartikel bedürfen einer anderen Modellierung und werden deswegen nicht berücksichtigt. Im ersten Schritt – der Artikelselektion – müssen die folgenden Aspekte berücksichtigt werden:

- Die Daten eines Artikels sollten 1. vollständig, 2. konsistent. 3. plausibel und 4. korrekt.
- Der Artikel ist mindestens 1 Jahre eingelistet und disponiert gewesen und
- hat sich über diesen Zeitraum "regelmäßig" abverkauft, d. h., der Artikel hat keine Intervalle > 30 Tage ohne Abverkauf. Damit werden reine Aktionsartikel oder "Ladenhüter", die extrem selten verkauft werden herausgefiltert.
- Die Historie des Artikels enthält keine Preis- oder Promotionsinkonsistenzen und besonders wichtig:
- es dürfen keine Promotionskennzeichnungen fehlen, wenn in Wirklichkeit eine Promotion stattgefunden hat.

Zusätzlich zu den oben genannten Bedingungen lassen sich noch die Kategorien Food oder Drogerie, der Typ Langsam- oder Schnelldreher inklusive Definition von Langsamdreher und die Miniwarengruppe für die Artikelselektion festlegen. Wenn eine ausreichend große und damit repräsentative Menge an Artikeln im ersten Schritt selektiert wurde, geht es im zweiten Schritte darum, aus den Vergangenheitsdaten Evaluationsszenarien zu extrahieren.

Als Erstes muss für die Suche eine Strategie ausgewählt werden: Entweder man definiert ein einzelnes Evaluationsintervall in dem nach einer zu definierenden Suchstrategie nach Szenarien gesucht wird oder man legt getrennt voneinander ein Lern- und ein Suchintervall fest. Für den ersten Fall kann man zwischen den Strategien Minimierung oder Maximierung des Lernintervalls wählen, wobei in der Regel Letzteres die sinnvolle Strategie ist. Für jedes Artikelszenario lassen sich vor der Suche die Art und Anzahl von Evidenzen angeben, d. h. die Anzahl von Situationsänderungen für die die festgelegten Evidenzen bei der Simulation benötigt werden. Die Art einer Änderung lässt sich zudem auf bestimmte Kategorien z. B. nur Preisänderungen oder nur Promotionen einschränken. Über Flags lässt

sich einstellen, ob die Zeiträume zwischen den gefundenen Szenarien ebenfalls mitsimuliert werden sollen, ob alle möglichen Szenarien oder nur die angegebene Anzahl erstellt werden sollen und, ob alle Artikel im gleichen Zeitraum simuliert werden sollen. Letzteres ist bei einer Simulation mit Holonen besonders wichtig. Des Weiteren lässt sich einstellen, ob das Simulationsintervall immer direkt an das Lernintervall anschließen muss oder, ob eine Lücke existieren darf. Eine Auswahl der Länge, die ein Szenario maximal haben darf, soll verhindern, dass zu große Unterschiede zwischen den Simulationsintervallen der Artikel entstehen und damit die Vergleichbarkeit der Prognosegüten unmöglich wird.

Für die Suche nach geeigneten Artikelszenarien für die Evaluation bei gegebener Evidenzart habe ich die folgenden Bedingungen zu Grunde gelegt:

- mit Hartevidenzen: Artikel hat mindestens 2 Intervalle mit gleichem Preis und mind. 2 verschiedene Preisintervalle (z. B. 1,99; 1,39; 1,99).
- mit Softevidenzen: Artikel hatte mindestens 3 (für Simulation 2) unterschiedliche Preise, wobei der 3. Preis zwischen den ersten beiden liegen muss.
- mit Extrapolationsevidenzen: Der 3. Preis muss außerhalb des Intervalls von Preis 1 und 2 liegen.
- mit Promotionen: Artikel hatte mind. 2 (gleiche) Promotionsarten (1 für Simulation).
- mit Deltaknoten: Mindestens 2 Wochen vor und 3 Wochen nach einer Promotion (insgesamt mind. 5 Wochen) müssen simulieren werden.

Für das Lernen der Agentenmodelle und die eigentliche Evaluation habe ich die folgenden Voraussetzungen festgelegt:

- War der Artikel während des Lernintervalls ausgelistet oder nicht disponiert dürfen die Daten von diesen Tagen nicht in die Lernmenge einfließen.
- Intervalle sollten so gewählt werden, dass sowohl tageweise als auch wochenweise sinnvoll simuliert werden kann, d. h., Szenarien sollen möglichst wochenweise definiert werden.
- Das Lernintervall sollte mindestens 90, besser 365 Abverkaufstage haben.
- Der Artikel sollte während des Simulationsintervalls nicht ausgelistet und durchgehend disponiert gewesen sein, wenn dies nicht im Szenario definiert ist, da dies die Evaluation erheblich verfälscht.
- Verkaufspreis-Gruppen, Verbund- und Sammelartikel sollten wie ein Artikel behandelt werden, da es sich dabei meistens um die gleichen Artikel nur mit unterschiedlichen „Geschmacksrichtungen“ handelt, die immer den gleichen Preis

haben und zur selben Zeit beworben werden. Zum Beispiel alle Jogurts der Marke Bauer.

Da einige Fehler erst nach einer Simulation korrekt erkannt werden können, wird im 4. Schritt eine erneute Filterung der Szenarien durchgeführt.

- Stellt sich bei der Simulation heraus, dass der reale Abverkauf immer gleich null war, so hat das ausgewählte Szenario gegen eine oben genannte Regel verstoßen und kann gelöscht werden.
- Werden nur so genannte „Inconsistent Cases“ simuliert, so kann man davon ausgehen, dass entweder bei der Szenarienselektion ein Fehler unterlaufen ist oder, dass die Datengrundlage fehlerhaft ist. Diese Fälle werden ebenfalls ausgefiltert.
- Dass gleiche gilt, wenn die Simulation nur Abverkaufsmengen gleich null ausgibt. So genannte Ladenhüter wurden bereits vorher ausgeschlossen, da diese nicht sinnvoll zu simulieren sind.

Mit dieser Systematik wurde das SimMarket-System mit den Daten des Partners Globus evaluiert und auf die folgenden Dimensionen angewendet.

Eine repräsentative Menge von *Artikeln* und *Szenarien* aus den Bereichen:

- Food
- Drogerie
- Schnelldreher
- Langsamdreher
- aus unterschiedlichen (Mini-)Warengruppen, mit
- Hart-,
- Soft- und
- Extrapolationsevidenzen und
- mit Holonenbildung, wobei die Holonenbildung durch eine Vorausgegangene
- Abhängigkeitsanalyse unterstützt wird.

5.2 Anwendungsfälle

Die technische Weiterentwicklung der SimMarket-Simulation-Engine und die zahlreichen Gespräche mit potenziellen Anwendern, Vertriebsgesellschaften und Wagniskapitalgebern führten zu immer neuen Anwendungsfällen. Neuerungen im System ließen neue Ideen auf Anwenderseite für vorhandene Probleme entstehen und umgekehrt die Wünsche und Probleme der Anwender gaben den Anstoß für neue technische Entwicklungen in SimMarket.

In Anhang E sind die gesammelten Anwendungsfälle, mit denen sich Sortimentsverantwortliche, Marketingmanager, Einkäufer, Kundenmanager und Filialleiter im Einzelhandel tagtäglich beschäftigen, detailliert aufgelistet. Die folgenden Kategorien von Anwendungsfällen haben wir daraus extrahiert und getestet.

5.2.1 Preissimulation

Der Anwender definiert für einen oder mehrere Artikel überschneidungsfrei beliebige Preise für beliebige Zeitintervalle in der Zukunft und erzeugt damit ein Szenario. Der Anwender kann durch die Verwendung von mehreren Szenarien alternative Konfigurationen gleichzeitig definieren und die Ergebnisse miteinander vergleichen. Bei der Simulation sollen für jeden Artikel individuell die Einflussfaktoren ermittelt und verwendet werden. Innerhalb eines Szenarios sollen sowohl die Wechselwirkungen zwischen den Artikeln als auch zwischen den unterschiedlichen Preisintervallen eines Artikels berücksichtigt und hinreichend exakt quantifiziert mitsimuliert werden. Das Ergebnis einer Simulation ist die Abverkaufsmenge, der Umsatz und der Rohertrag der simulierten Artikel innerhalb des Simulationsintervalls. Durch die manuelle oder auch automatische Erzeugung von alternativen Szenarien pro Artikel lassen sich die genannten Ergebnisse hinsichtlich einer Kennzahl z. B. Rohertrag optimieren.

5.2.2 Promotionssimulation

Der Anwender definiert für einen oder mehrere Artikel Promotionsaktionen für beliebige Zeitintervalle in der Zukunft und erzeugt damit ein Szenario. Durch die Definition mehrerer Szenarien kann der Anwender alternative Konfigurationen gleichzeitig erzeugen und die Ergebnisse miteinander vergleichen. Bei der Simulation sollen für jeden Artikel individuell die Einflussfaktoren ermittelt und verwendet werden. Innerhalb eines Szenarios sollen sowohl die Wechselwirkungen zwischen den Artikeln als auch zwischen den unterschiedlichen Promotionsintervallen und insbesondere mit Nichtpromotionsintervallen berücksichtigt und hinreichend exakt mitsimuliert werden. Das Ergebnis einer Simulation ist die Abverkaufsmenge, der Umsatz und der Rohertrag der simulierten Artikel und wenn möglich die Summe der Kosten aller Promotionen innerhalb des Simulationsintervalls. Durch die manuelle oder auch automatische Erzeugung von alternativen Szenarien pro Artikel lassen sich die genannten Ergebnisse hinsichtlich einer Kennzahl z. B. Rohertrag optimieren. Liegen Realdaten vor, sollen die Simulationsergebnisse evaluierbar sein.

5.2.3 Maßnahmenanalyse

Der Anwender wählt für einen oder mehrere Artikel eine oder mehrere Maßnahmen, z. B. Preisänderungen oder Promotionen der Vergangenheit, für ein beliebiges oder für das in der Vergangenheit zur Maßnahme zugehörige Zeitintervall und erzeugt damit ein Szenario (der Vergangenheit). Er kann auch mehrere Szenarien beschreiben und alternative Konfigurationen für Analysen gleichzeitig auswählen, um Ergebnisse miteinander zu vergleichen. Der Anwender erhält mit der Maßnahmenanalyse die Möglichkeit für jeden Artikel die Fragen zu stellen „Was wäre passiert, wenn ich statt x y gemacht hätte?“ oder „Unter welchen Umständen ist x passiert und y erreicht worden?“. Bei der Analyse sollen für jeden Artikel wenn möglich individuell die damaligen Einflussfaktoren ermittelt und verwendet werden. Innerhalb eines Szenarios können sowohl die (damaligen) Wechselwirkungen zwischen den Artikeln als auch zwischen den unterschiedlichen Maßnahmen berücksichtigt und hinreichend exakt mitsimuliert werden. Das Ergebnis einer Analyse ist die Abverkaufsmenge, der Umsatz und der Rohertrag der analysierten Artikel und wenn möglich die Summe der damaligen Kosten aller Maßnahmen innerhalb des Analysezeitraumes.

5.2.4 Abhängigkeitsanalyse

Die Abhängigkeitsanalyse zeigt Interdependenzen zwischen den Artikeln auf, die dann näher analysiert werden können. Die Simulationsroutine nutzt die Abhängigkeitsanalyse, um geeignete Holonisierungskandidaten zu ermitteln und dadurch die Simulationsergebnisse zu verbessern. Der Anwender wählt zunächst einen oder mehrere Artikel, zu denen abhängige Artikel gefunden werden sollen und eine Testmenge aus. Die Abhängigkeitsanalyse ermittelt in einem ersten Schritt die Korrelationen zwischen den gewählten Artikeln und gibt einen Wert aus, der zwar die Abhängigkeiten der Abverkaufsmengen ausdrückt, aber keinerlei Einflussfaktoren, zeitliche Bedingungen oder Wechselwirkungen zwischen Faktoren berücksichtigt. Zudem handelt es sich dabei um einen relativen Wert, der nicht für quantifizierte Aussagen verwendbar ist. Quantifizierbare Ergebnisse erhält der Anwender im zweiten Schritt, bei dem die am stärksten korrelierenden Artikel zu einem Holon verschmolzen werden. Mit Hilfe eines Holon kann der Anwender detaillierte Abhängigkeitsanalysen, z. B. eine Substitutionsanalyse, die Push-, Pull- und Kannibalisierungseffekte zwischen Artikel in Abhängigkeit von den Einflussfaktoren ermitteln kann, durchführen.

5.2.5 Kassenbonsimulation

Der Anwender definiert für einen oder mehrere Artikel Maßnahmen für beliebige Zeitintervalle in der Zukunft und erzeugt damit ein Szenario. Die Kassenbonsimulation ermittelt, wie sich Änderungen am Sortiment auf die Struktur der Kassenbons, z. B. durchschnittlicher Umsatz/Ertrag, Anzahl (verschiedener/beworbener) Artikel / Warengruppen, Wochentagsverteilung, Preiskategorien der Artikel etc. auswirken. Bei der Simulation sollen für jeden Artikel individuell die Einflussfaktoren ermittelt und verwendet werden. Innerhalb eines Szenarios sollen sowohl die Wechselwirkungen zwischen den Artikeln als auch zwischen den Maßnahmen berücksichtigt und hinreichend exakt mitsimuliert werden. Die Ergebnisse einer Simulation sind die oben genannten Kassenbonkennzahlen bzw. die Veränderung derer in Prozent innerhalb des Simulationsintervalls. Durch die manuelle oder automatische Erzeugung von alternativen Szenarien pro Artikel lassen sich die genannten Ergebnisse hinsichtlich einer Kennzahl z. B. Anteil einer bestimmten Warengruppe auf einem Bon optimieren.

5.2.6 Automatische Disposition

Die Anbindung des SimMarket-Simulationssystems könnte die Qualität von automatischen Dispositionssystemen auf Grund der komplexeren Simulationsmodelle signifikant verbessern und dadurch Leerstände und Überkapazitäten von Lagern vermeiden. Bei der automatischen Disposition geht es weniger darum Abverkaufszahlen möglichst exakt zu prognostizieren, sondern darum Bestellmengen zu ermitteln und Leerstände zu vermeiden. Dies erlaubt natürlich einen wesentlichen größeren Spielraum bei der Prognose, da z. B. Coladosen immer palettenweise bestellt werden müssen. Man muss somit „nur“ ermitteln, wann wie viele Paletten bestellt werden müssen. Dabei kalkuliert man immer ein großzügiges Sicherheitspolster ein. Das Wichtigste bei der automatischen Disposition ist, dass die Prognose für alle zu disponierenden Artikel zu jedem Zeitpunkt anwendbar und ausreichend performant für eine große Menge von Artikeln ist. SimMarket könnte insbesondere die Qualität von im Bedarf stark schwankenden Artikeln erheblich verbessern und damit bestehende Dispositionssysteme optimal ergänzen.

5.2.7 Listungsanalyse

In Gesprächen mit Sortimentsverantwortlichen wurde deutlich, dass vor allem die Auslistung von Artikel von besonderem Interesse ist und weniger die Einlistung. Das liegt daran, dass sich die Frage welche Artikel eingelistet werden sollen oft nicht stellt, da der Händler durch zahlreiche Constraints wie Verträge mit den Herstellern oder der Innovativität eines neuen Produktes gezwungen ist, dieses in sein Sortiment aufzunehmen. Viel wichtiger ist die Frage,

welcher Artikel ausgelistet werden soll, da der Platz im Regal in der Regel komplett ausgenutzt wird, so dass erst ein Artikel ausgelistet werden muss, bevor ein neuer eingelistet werden kann. Dennoch möchte der Sortimentsmanager natürlich wissen was passiert, wenn er Artikel x einlistet.

In der Listungsanalyse gilt es also drei Fälle zu unterscheiden:

1. Ein neuer Artikel wird eingelistet und ersetzt einen anderen, der gleichzeitig ausgelistet wird.
2. Ein neuer Artikel wird eingelistet ohne dass ein alter ausgelistet wird.
3. Ein vorhandener Artikel wird ausgelistet ohne dass ein neuer eingelistet wird.

Des Weiteren muss bedacht werden, ob die Änderung des Sortiments dauerhaft bzw. für einen längeren Zeitraum durchgeführt wird oder, ob es sich nur um eine temporäre Änderung z. B. eine Einlistung eines speziellen Aktionsartikels für den Zeitraum einer Promotion handelt. Im Falle einer dauerhaften Änderung ist Punkt 1 häufiger, bei temporären Listungen ist vor allem Punkt 2 und 3 häufiger. Dennoch können auch Szenario 2 und 3 bei dauerhaften Änderungen am Sortiment auftauchen, wenn der Händler eine veränderte Strategie bezüglich einer Warengruppe z. B. Verbreiterung, Vertiefung oder Straffung verfolgt.

Daraus ergeben sich vier verschiedene Anwendungsfälle der Listungsanalyse:.

1. Ein- und später auch Auslistung eines Aktionsartikels im Rahmen einer Promotionsplanung.
2. Substitution eines vorhandenen Artikels durch einen Neuen.
3. Dauerhafte Auslistung eines Artikels zwecks Straffung der Warengruppenzusammensetzung.
4. Dauerhafte Einlistung eines Artikels zwecks Verbreiterung oder Vertiefung des Sortiments.

Das prinzipielle Problem der Listungsanalyse ist, dass man nicht – wie bei den anderen Anwendungsfällen – auf eine Menge von Lernfällen für einen Artikel zurückgreifen kann, da ein Artikel meistens nur einmal ein- und einmal ausgelistet wird. Somit bedarf es einer viel größeren Abstraktion der Modelle, um Erfahrungen der Vergangenheit mit anderen Artikeln übertragbar auf neue Ereignisse zu machen.

Bei der Auslistung eines Artikels erhält man Rückschlüsse auf die Wirkung, indem man Ereignisse der Vergangenheit, die einer Auslistung ähnlich sind, analysiert. Beispielsweise kann man sich anschauen, wie der Kunde reagiert hat, als der Artikel z. B. durch Lieferprobleme nicht im Regal gestanden hat und deshalb nicht gekauft werden konnte.

Ein weiterer Anhaltspunkt ist der Zeitraum der Einlistung. Welche Kannibalisierungseffekte hat der Artikel durch seine Einlistung bei anderen Artikeln ausgelöst? Dabei gilt die Annahme, dass sich diese Effekte durch die Auslistung umkehren.

Die dritte mögliche Strategie besteht in der Analyse bereits ausgelisteter Artikel und deren damaligen Auswirkungen auf die Warengruppe. Durch eine Ähnlichkeitsanalyse der bereits ausgelisteten Artikel mit dem Auszulisteten, kann eine Prognose durch Gewichtung der Erfahrungen der Vergangenheit erzeugt werden.

Alle drei Möglichkeiten werden durch Modellierung der Listungsereignisse in einem Verhaltensnetz eines Warengruppenagenten realisiert und für Analysen verwendet. Der Warengruppenagent kann durch Holonisierung erzeugt werden. Zusätzlich benötigt er zwei Knoten in seinem Verhaltensnetz, die die Ein- und Auslistungen repräsentieren. Die Werte der zugehörigen Variablen können entweder einfache Flags sein mit der Bedeutung „ein Artikel wurde innerhalb des vergangenen Zeitraums [a, b] ein- bzw. ausgelistet“ oder die ID-Nummern der ein- bzw. ausgelisteten Artikel darstellen, um genauer Rückschlüsse auf die Auswirkungen einer speziellen Aktion ziehen zu können. Letztere Variante bietet sich an, wenn der Anwender in der Lage ist die Ähnlichkeit der zu vergleichenden Artikel manuell zu definieren.

Eine automatisierte Ähnlichkeitsanalyse wird in einigen Fällen nicht möglich sein, da z. B. über einzulistende Artikel noch keinerlei Daten vorliegen. Durch manuelle Definition von Ähnlichkeiten können die Erfahrungen von besonders ähnlichen Artikeln auf andere übertragen werden. Prinzipiell gilt, dass ein Artikel eher ausgelistet werden kann, wenn mehrere „ähnliche“ Artikel in der Warengruppe vorhanden sind, da dann die Kunden eine akzeptable Ausweichmöglichkeit haben.

Ist ein Artikel dagegen „einzigartig“ im Sortiment, könnte sich der Kunde dazu veranlasst sehen den Händler zu wechseln oder zumindest dieses Produkt woanders zu kaufen. Die Einzigartigkeit eines Artikels lässt sich mit einer Korrelationsanalyse und mit Hilfe der Holonisierung prüfen. Durch Holonisierung lassen sich potenzielle Abhängigkeiten exakter prüfen, da die Analyse auf bestimmte Fragestellungen eingeschränkt werden kann. So kann man beispielsweise testen, ob die Kunden bei einer Preiserhöhung des auszulistenden Artikels verstärkt zu anderen gegriffen haben.

5.3 Ergebnisse

Im Folgenden präsentiere ich die Ergebnisse für verschiedene Anwendungsfälle. Zunächst wird eine automatische Massenevaluation, die die Qualität der Simulation auf möglichst vielen Artikeln und Szenarien ermittelt, präsentiert. Anschließend zeige ich, wie man die Simulation für die Bewertung von alternativen Szenarien, die beispielsweise von einem Sortimentsmanager erstellt worden sind, genutzt werden kann. Als drittes wird die

Anwendung des Verschmelzungsalgorithmus für Holonen im praktischen Einsatz demonstriert.

5.3.1 Automatische Evaluation

Für die automatische Massenevaluation wurden die Daten des Projektpartners Globus (St. Wendel) von knapp drei Jahren einer Filiale verwendet. Die folgende Statistik zeigt die Größe der verwendeten Datenmengen und soll einen Eindruck der zu Grunde liegenden Komplexität vermitteln:

- Zeitraum: 01.01.2003 – 17.09.2005
- Anzahl der aktiven (abverkauften) Artikel: > 150.000
- Anzahl Transaktionen: > 100.000.000
- Anzahl Kassenbons: > 10.000.000
- Anzahl aktive MWGR: > 3000
- Durchschnittliche Größe der MWGR (# ak. Art.) ~ 50
- Anzahl aktiver Artikel in
  - Drogerie: > 10.000
  - Food: > 25.000
- Anzahl Aktionen: ca. 500
- Durch. Anzahl beworbener Artikel pro Aktion: ca. 250

Für die automatische Evaluation wurden die folgenden Warengruppen berücksichtigt, die einen Großteil des Sortiments von Globus darstellen und zudem die Sortimente der anderen Partner *dm-drogerie markt* und *tegut…* fast vollständig abdecken.

Food	Drogerie
610, 615, 620, 623, 624, 625, 627, 628, 629, 632, 635, 640, 641, 642, 643	644, 645, 650

Tabelle 4: Die evaluierten Warengruppen[27]

[27] Die Zahlen 610, 615 etc. stehen für die Warengruppennummern der höchsten Hierarchieebene bei Globus.

Um untersuchen zu können, wie das Simulationssystem bei verschiedenen Arten von Artikeln und bei unterschiedlichen Maßnahmen abschneidet, wurden die Artikel der oben aufgeführten Warengruppen zusätzlich in die folgenden Cluster eingeteilt.

Die Artikel-Cluster:

- Langsamdreher mit Preisänderung und Aktion (LPA)
- Langsamdreher mit Preisänderung ohne Aktion (LP)
- Schnelldreher mit Preisänderung mit Aktion (SPA)
- Schnelldreher mit Preisänderung ohne Aktion (SP)

Die Einteilung von Artikeln in Langsam- bzw. Schnelldrehern ist von Händler zu Händler und in der Literatur sehr unterschiedlich definiert. Kleinere Verkaufsflächen und niedrigere Kundenfrequenzen z. B. bei dm-Markt resultieren in einer ganz anderen Definition von Langsamdreher als z. B. bei den Globusmärkten. Manche definieren diese Kategorien auch artikelindividuell, z. B. über die Mindestbestellmenge. Ein Artikel ist dann ein Langsamdreher, wenn weniger als die Mindestbestellmenge pro Woche bzw. pro Bestellintervall verkauft werden.

Auf Grund der sehr unterschiedlichen Kauffrequenzen von Food- bzw. Drogerieartikeln ist es sinnvoll, jeweils andere Definitionen für diese Kategorien festzulegen. Anderenfalls würden entweder keine Schnelldreher in die Kategorie Drogerie oder fast alle Food-Artikel in die Gruppe Schnelldreher fallen. Da es sich hier um Globusdaten mit durchschnittlich sehr hohen Kauffrequenzen handelt, definieren wir einen Langsamdreher im Bereich Food als einen Artikel, der sich weniger als 150-mal pro Woche verkauft und im Bereich Drogerie als einen Artikel, der sich weniger als 50-mal die Woche verkauft. Alle anderen Artikel, die jeweils nicht unter diese Definitionen fallen, sind Schnelldreher.

Die folgende Tabelle enthält die für die Evaluation verwendete Anzahl an Artikeln pro Cluster im Bereich Food, die für die Evaluation zur Verfügung standen. Dabei wurde versucht möglichst korrekte Artikel, d. h., mit konsistenter und vollständiger Datenbasis, zu verwenden. Fehlerhafte und unvollständige Artikel wurden soweit möglich herausgefiltert. Es kann dennoch nicht ausgeschlossen werden, dass in der Testmenge fehlerhafte Artikel enthalten sind, da viele Fehler, z. B. fehlende Promotionsinformationen, nicht festgestellt werden können.

LPA	LP	SPA	SP
3501	3501	127	127

Tabelle 5: Die Anzahl der Artikel pro Cluster im Bereich Food

Die folgende Tabelle enthält die für die Evaluation verwendete Anzahl an Szenarien pro Cluster im Bereich Food:

LPA	LP	SPA	SP
212	181	83	145

Tabelle 6: Die Anzahl der Szenarien pro Cluster im Bereich Food

Die folgende Tabelle enthält die für die Evaluation verwendete Anzahl an Artikeln pro Cluster im Bereich Drogerie:

LPA	LP	SPA	SP
982	982	79	79

Tabelle 7: Die Anzahl der Artikel pro Cluster im Bereich Drogerie

Die folgende Tabelle enthält die für die Evaluation verwendete Anzahl an Szenarien pro Cluster im Bereich Drogerie:

LPA	LP	SPA	SP
46	585	43	184

Tabelle 8: Die Anzahl der Szenarien pro Cluster im Bereich Drogerie

Um die Simulation hinsichtlich der Länge des Lernintervalls und in verschiedenen Zeitabschnitten untersuchen zu können, wurde in einer weiteren Dimension die Konfiguration der Lern- und Suchintervalle variiert. In der folgenden Tabelle sind die verwendeten Lern- und Simulationsintervalle aufgelistet:

Konfiguration	Lernintervall	Suchintervall
K1	01.01.2003 – 28.02.2005	01.03.2005 – 17.09.2005
K2	01.01.2004 – 28.02.2005	01.03.2005 – 17.09.2005
K3	01.01.2003 – 29.02.2004	01.03.2004 – 31.12.2004

Das Lernintervall dient den Verhaltensnetzen jeweils vollständig als Trainingsmenge, innerhalb des Suchintervalls wird abhängig vom Artikel-Cluster nach Änderungen

(Preisänderungen oder Promotionen) an den Artikeln gesucht und diese maximal 28 Tage lang simuliert. Die Idee dabei ist, nicht einfach beliebige Intervalle, bei denen keine Veränderungen bei den Artikeln stattgefunden haben, zu simulieren, da dies nicht der späteren Verwendung des Systems durch einen Sortimentsmanager entspricht, sondern nur Intervalle zu verwenden bei denen eine Veränderung stattgefunden hat. Ein Sortimentsverantwortlicher wird in der Regel eine oder mehrere Änderungen an seinen Artikeln festlegen und diese vorab simulieren wollen. Jedes gefundene Szenario innerhalb des Suchintervalls entspricht deshalb einer Preisänderung oder Promotion, die je nach Länge maximal 28 Tage (4 Wochen) lang simuliert wird.

In der ersten Konfiguration K1 wurde der maximal zur Verfügung stehende Zeitraum, der in der Datenbank vorliegt, ausgenutzt. Bei K2 wurde das Lernintervall künstlich von 26 auf 14 Monate verkürzt. In K3 wird das Suchintervall vollständig auf das Jahr 2004 verlegt, um Simulationsergebnisse von zwei unterschiedlichen Jahren zu erhalten.

Die Topologie der verwendeten Netze

Die Topologie eines Verhaltensnetzes ist nicht manuell vorgegeben, sondern sie wird durch das verwendete Lernverfahren ermittelt. Abbildung 40 zeigt ein Beispiel für ein extrahiertes Artikelverhaltensnetz.

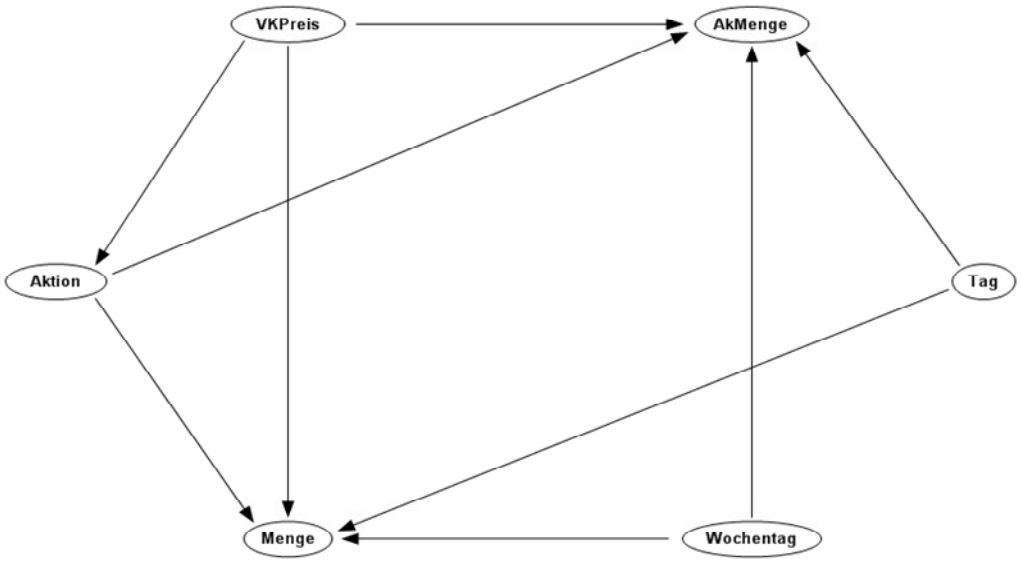


Abbildung 40: Beispielgraph eines Verhaltensnetz

Die zugehörige Diagrammansicht des obigen Netzes zeigt die Zustände der Knoten, die Diskretisierung dieser in Intervalle, deren Mittelwerte, die Wahrscheinlichkeiten und die Erwartungswerte. Hier handelt es sich wiederum nur um ein Beispiel, da sowohl die Anzahl

und Art der Intervalle, als auch alle dargestellten Werte durch das Lernverfahren bestimmt werden und somit für jeden Artikel individuell ermittelt werden.

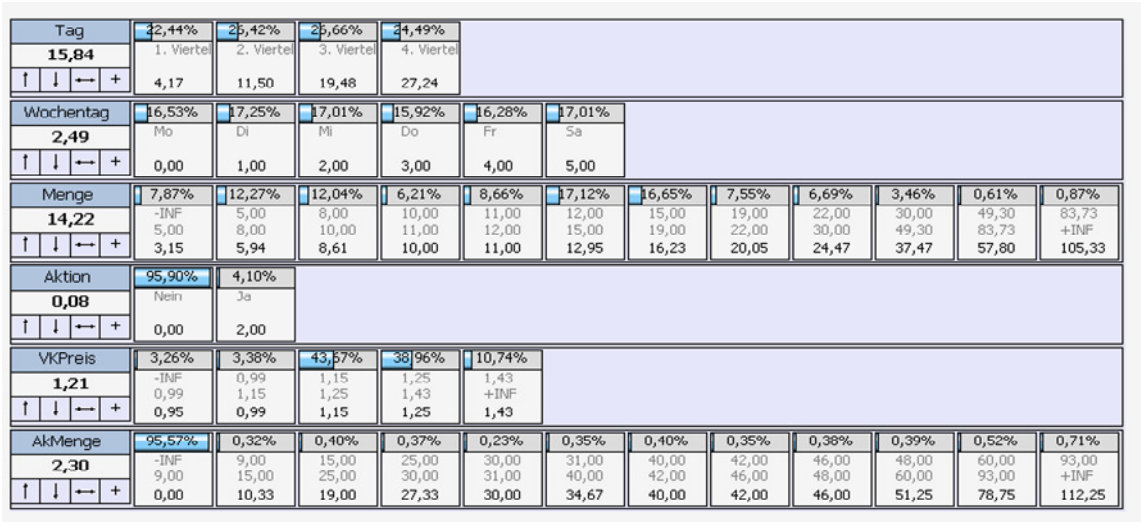


Abbildung 41: Diagrammansicht eines Evaluationsnetzes

Alle Simulationen wurden im Folgenden wochenweise mit den angegebenen Dimensionen durchgeführt. Für alle Artikelcluster SP, SPA, LP und LPA wurden Simulationen mit den Konfigurationen K1, K2 und K3 des Verhaltensnetzes (VN) durchgeführt und mit dem klassischen Prognoseverfahren exponentielle Glättung 2. Ordnung verglichen, das in vielen Systemen des Einzelhandels zum Einsatz kommt. Es wird sowohl der mittlere absolute prozentuale Fehler (MAPE), als auch die Prognosegüte in Prozentkategorien ermittelt.

Ergebnisse im Bereich Food

Cluster	Prognose	Konfiguration		
		K1	K2	K3
SP	VN	14,56	16,56	15,56
	exp. Gl.	20,34	22,89	20,77
SPA	VN	21,43	21,33	26,12
	exp. Gl.	43,15	43,42	61,23
LP	VN	18,65	16,77	16,97
	exp. Gl.	18,80	18,02	19,17
LPA	VN	18,57	24,43	36,55
	exp. Gl.	62,27	59,41	61,36

Tabelle 9: Die durchschnittlichen Prognosegüten (MAPE)

Tabelle 9 zeigt den mittleren absoluten Prognosefehler der evaluierten Food-Artikel. Man sieht, dass das die Simulation mit Verhaltensnetzen für alle Cluster und Konfigurationen besser als die Prognose mit exponentieller Ordnung 2. Ordnung ist. Besonders deutlich ist der Unterschied bei Szenarien mit Aktionen, wo die Verhaltensnetze doppelt so gute Ergebnisse liefern. Erwartungsgemäß sind die schnelldrehenden Artikel besser zu simulieren als die langsamdrehenden. Je größer die Änderung eines Artikels ausfällt, desto schwieriger wird die Simulation. Beworbene Artikel mit Preisänderung sind schlechter zu simulieren als Artikel mit reiner Preisänderung. Bei den Konfigurationen gibt es nur leichte Unterschiede. Tendenziell fallen die Ergebnisse bei Szenarien mit kürzerem Lernintervall (K2, K3) – mit Ausnahme von Cluster LP – schlechter aus. Szenarien aus Konfiguration K3 scheinen zudem etwas schlechter simulierbar zu sein als K2. Das könnte damit zusammenhängen, dass die vom Händler gelieferte Datenqualität im Laufe des Projektes zugenommen hat. Zu Beginn des Projektes kamen Dateninkonsistenzen besonders gehäuft vor.

Cluster	Prognose	Konfiguration								
		K1			K2			K3		
		≤20%	>20% ≤50%	>50%	≤20%	>20% ≤50%	>50%	≤20%	>20% ≤50%	>50%
SP	VN	81,82	14,55	3,64	75,56	20,00	4,44	64,44	35,56	0,00
	exp. Gl.	63,64	29,09	7,27	53,33	33,33	13,33	60,00	28,89	11,11
SPA	VN	54,29	42,86	2,86	57,14	40,00	2,86	53,85	23,08	23,08
	exp. Gl.	14,29	40,00	45,71	14,29	37,14	48,57	0,00	23,08	76,92
LP	VN	53,25	42,86	3,9	61,76	35,29	2,94	66,67	27,78	5,56
	exp. Gl.	59,74	35,06	5,19	66,18	29,41	4,41	63,89	27,78	8,33
LPA	VN	63,64	36,36	0,00	50,70	36,62	12,68	20,41	57,14	22,45
	exp. Gl.	0,00	9,09	90,91	8,45	19,01	72,54	2,04	24,49	73,47

Tabelle 10: Die durchschnittlichen Prognosegüten in Prozentkategorien

Tabelle 10 spiegelt im Wesentlichen die Ergebnisse von Tabelle 9 wieder. Hier ist der Unterschied zwischen K2 und K3 noch deutlicher zu erkennen. K1 liefert bessere Ergebnisse als K2 und K2 bessere als K3. Auch der Unterschied zwischen Aktionsszenarien bei der Simulation mit Verhaltensnetzen und der Prognose mit exponentieller Glättung wird in dieser Tabelle noch klarer. Bei LPA-K1 und SPA-K3 mit exp. Glättung liegt kein einziges Szenario innerhalb von 20% Abweichung. Bei den VN liegen die Ergebnisse bis auf eine Ausnahme immer über 50% in dieser Kategorie und selten mehr als 5% in der schlechtesten Kategorie. Das beste Ergebnis liefert VN-SP-K1, bei dem über 80% der Szenarien ein besseres Ergebnis als 20% Abweichung haben. Als problematisch hat sich VN-LPA-K3 erwiesen, was daran

liegen mag, dass in diesem Fall sehr wenig statistische Samples vorliegen, die zudem noch Dateninkonsistenzen haben.

Das folgende Balkendiagramm zeigt noch einmal auf übersichtliche Weise die Ergebnisse der beiden obigen Tabellen. Zu sehen sind die Prognosegüteverteilungen der Simulationen aufgeteilt in Kategorien für alle Cluster. Die Werte wurden über die Konfigurationen K1 – K3 gemittelt.

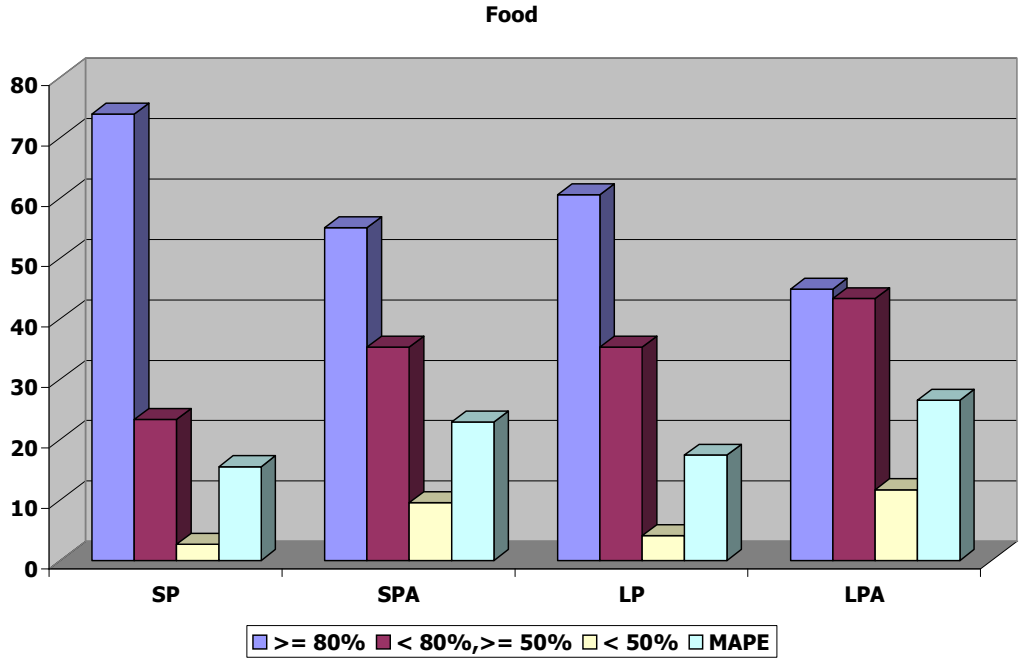


Abbildung 42: Durchschnittliche Güteverteilung im Bereich Food

Das folgende Kuchendiagramm zeigt das Ergebnis der Kategorie Food VN-SP-K1:

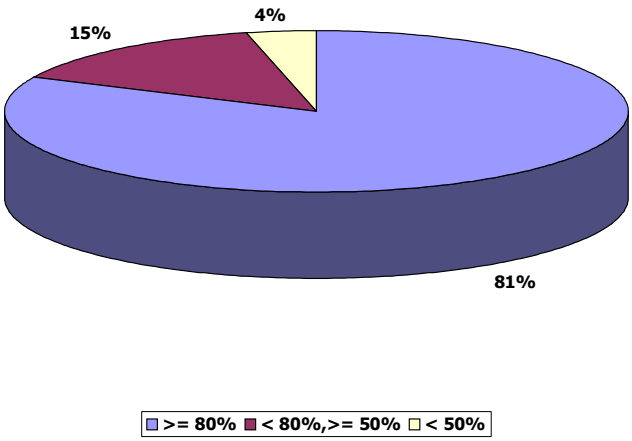


Abbildung 43: Güteverteilung des besten Clusters (Food)

Ergebnisse Bereich Drogerie

Cluster	Prognose	Konfiguration		
		K1	K2	K3
SP	VN	19,89	17,88	20,34
	exp. Gl.	23,88	27,45	25,49
SPA	VN	28,37	31,95	22,51
	exp. Gl.	72,28	74,18	26,61
LP	VN	22,92	21,70	25,22
	exp. Gl.	22,19	24,01	30,05
LPA	VN	27,90	32,01	38,95
	exp. Gl.	68,66	66,96	51,91

Tabelle 11: Die durchschnittlichen Prognosegüten (MAPE)

Für den Bereich Drogerie gelten ähnliche Aussagen wie im Bereich Food. Insgesamt fallen die Ergebnisse erwartungsgemäß schlechter aus als bei Food-Artikeln, da die Anzahl der Samples erheblich geringer ist. Dies gilt vor allem für die Schnelldreher, bei denen eine andere Definition für die Kauffrequenzkategorie angewandt wurde, als bei den Food-Artikeln. Sämtliche Schnelldreher im Bereich Drogerie würden bei den Food-Artikeln noch zur Gruppe der Langsamdreher gehören.

Cluster	Prognose	Konfiguration								
		K1			K2			K3		
		≤20%	>20% ≤50%	>50%	≤20%	>20% ≤50%	>50%	≤20%	>20% ≤50%	>50%
SP	VN	60,29	29,41	10,29	65,38	28,85	5,77	60,32	30,16	9,52
	exp. Gl.	58,82	27,94	13,24	44,23	38,46	17,31	57,14	28,57	14,29
SPA	VN	35,00	60,00	5,00	26,32	63,16	10,53	25,00	75,00	0,00
	exp. Gl.	0,00	15,00	85,00	0,00	0,00	100,0	75,00	0,00	25,00
LP	VN	51,02	39,18	9,80	53,47	38,78	7,76	53,68	32,63	13,68
	exp. Gl.	50,61	42,86	6,53	48,16	43,27	8,57	46,32	31,58	22,11
LPA	VN	50,00	25,00	25,00	33,33	51,85	14,81	26,67	53,33	20,00
	exp. Gl.	0,00	0,00	100,0	0,00	7,41	92,59	13,33	33,33	53,33

Tabelle 12: Die durchschnittlichen Prognosegüten in Prozentkategorien

Das folgende Balkendiagramm zeigt die Prognosegüteverteilungen der Simulationen aufgeteilt in Kategorien für alle Cluster. Die Werte wurden über die Konfigurationen K1 – K3 gemittelt.

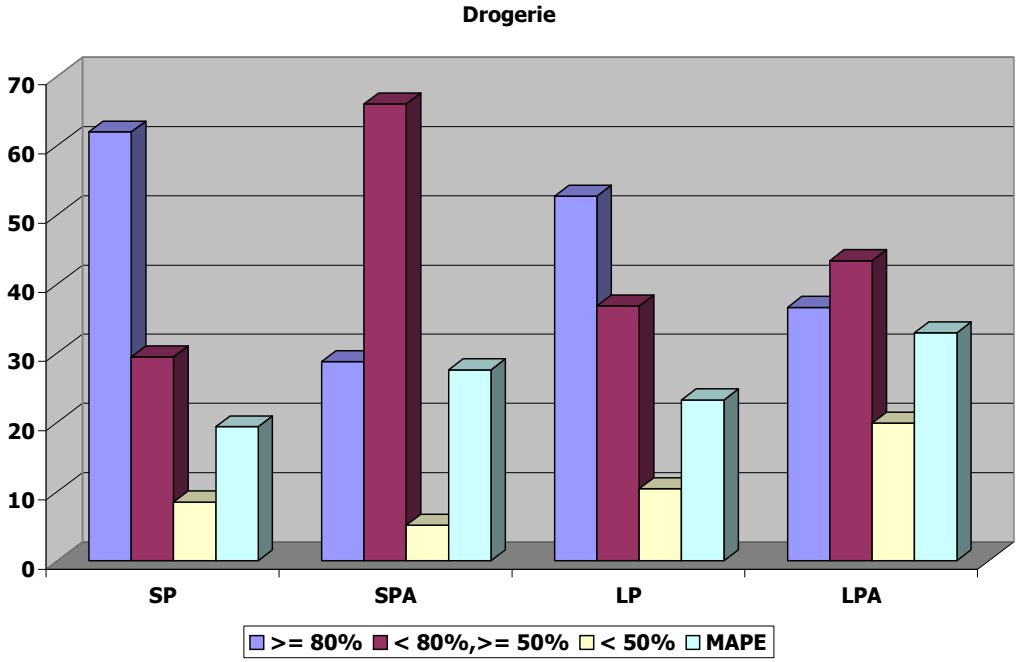


Abbildung 44: Durchschnittliche Güteverteilung im Bereich Drogerie

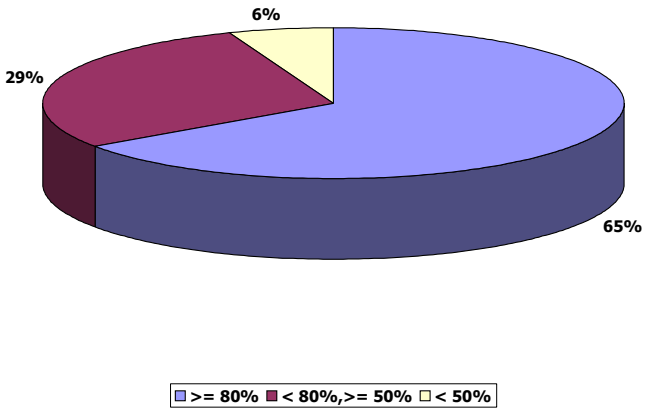


Abbildung 45: Güteverteilung des besten Clusters (Drogerie)

Analyse der Prognosegüte abhängig von der Kauffrequenz

Auf Grund der unterschiedlichen Ergebnisse der Bereiche Food und Drogerie habe ich, wie in Abbildung 46 zu sehen, die Prognosegüte in Abhängigkeit von der Kauffrequenz systematisch untersucht.

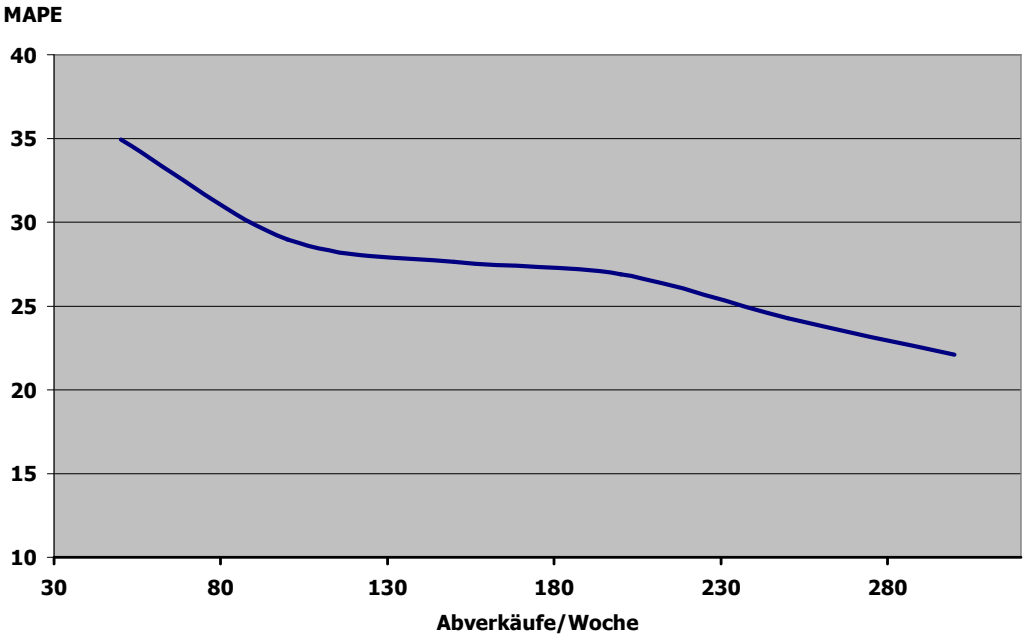


Abbildung 46: Einfluss der Kauffrequenz auf die Prognosegüte (MAPE)

Das Ergebnis bestätigt die Hypothese, dass die Prognosegüte erheblich von der Kauffrequenz eines Artikels abhängt. Deshalb sollte man, wann immer es geht, Artikel, z. B. Verbundartikel und VKPreis-Gruppen, vor einer Simulation zu einem virtuellen Artikel verschmelzen, um die Güte der Simulationen zu erhöhen. Außerdem ist es sinnvoll gewisse Fragestellungen nicht auf Filialebene, sondern auf Konzernebene zu untersuchen, indem die Daten aller Artikel über alle Filialen aggregiert werden.

Skalierung des Simulationssystems

Abbildung 47 zeigt, dass das Simulationssystems linear mit der Anzahl an Agenten skaliert. Durch Verteilung der Agenten auf ein Netzwerk von Rechnern lässt sich somit die Leistung des Gesamtsystems beliebig erhöhen.

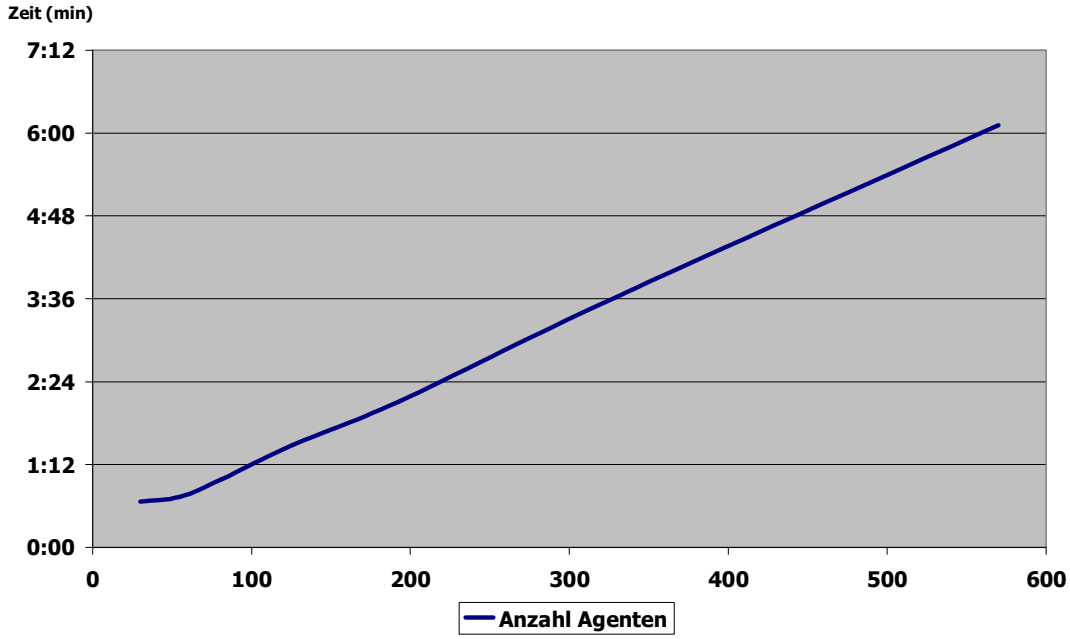


Abbildung 47: Die Skalierbarkeit des Simulationssystems

5.3.2 Simulation alternativer Szenarien

Dieser Abschnitt soll zeigen, wie ein Sortimentsmanager mit dem Simulationssystem arbeitet. Nach einer eingehenden Analyse der Situation mit rein analytischen Werkzeugen innerhalb einer Warengruppe wird er einige Artikel identifizieren, die nicht die gewünschten Ziele erreicht haben oder bei denen zumindest noch Spielraum vorhanden ist. Für jeden dieser Artikel wird er auf Basis seines Wissens und seiner Erfahrung Hypothesen über die Gründe

der jetzigen Situation und über die Auswirkungen von möglichen Maßnahmen treffen. Die möglichen Veränderungen an den Artikeln sind bereits eine sehr gute Einschränkung des großen Handlungsraumes. Um aus dieser Menge dann eine konkrete Handlungsoption auszuwählen kann nur aus Erfahrung erfolgen, daher ist die Wahrscheinlichkeit recht hoch, dass der Sortimentsverantwortliche auf Grund der Komplexität des Problems falsch liegt. Diese Lücke schließt das SimMarket-Simulationssystem durch die Fähigkeit Ex-Ante-Simulationen der vorliegenden Handlungsalternativen durchführen zu können. Abbildung 48 zeigt einen Screenshot des Systems mit einem angelegten Projekt, das fünf alternative Szenarien enthält. Im rechten unteren Teil sind die Szenarien mit ihren Preisänderungen und Promotionen zusehen, darüber wird das Ergebnis der Simulation dargestellt.

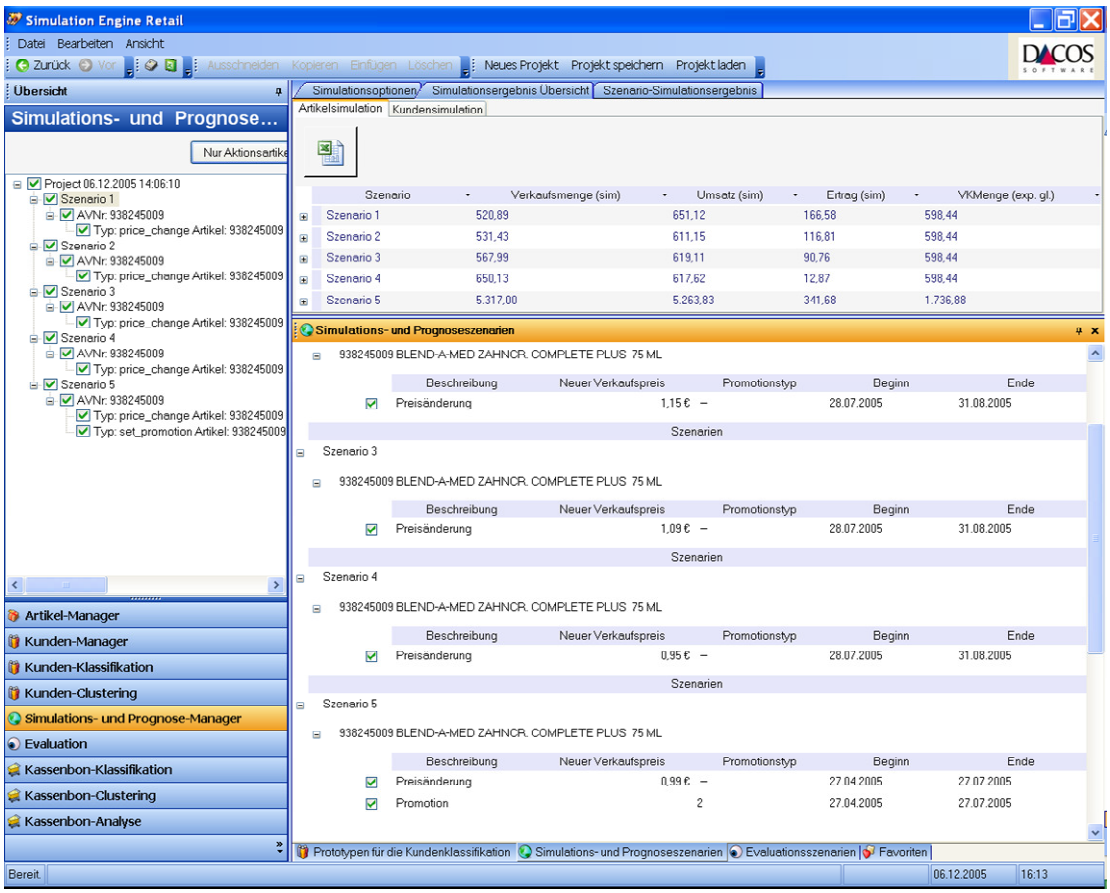


Abbildung 48: Simulation alternativer Szenarien

Im Folgenden soll nun gezeigt werden, wie man eine Menge von Handlungsalternativen ex ante simulieren und die Ergebnisse für die Entscheidungsfindung nutzen kann. Wie in Tabelle 13 zu sehen ist, wurde ein Artikel vom 28.07.2005-31.08.2005 mit verschieden

Preisen und in einem Fall mit einer Aktion simuliert. In den Szenarien 1-4 wird jeweils der Preis des Artikels gesenkt, in Szenario 5 wird der Artikel zusätzlich beworben.

ArtikelNr	AbDatum	BisDatum
938245009	28.07.2005	31.08.2005

Szenario	VK Preis (sim)	Aktion	Verkaufs-menge (sim)	Umsatz (sim)	Ertrag (sim)
Szenario 1	1,25	nein	520,89	651,12	166,58
Szenario 2	1,15	nein	531,43	611,15	116,81
Szenario 3	1,09	nein	567,99	619,11	90,76
Szenario 4	0,95	nein	650,13	617,62	12,87
Szenario 5	0,99	ja	5317,00	5263,83	341,68

Tabelle 13: Ergebnisse einer Simulation mit alternativen Szenarien

In Tabelle 13 sieht man sehr schön, dass die Senkung des Preises jeweils zu einem höheren Absatz des Artikels führt. Aber schon der Umsatz fällt bei Szenario 2 kleiner aus als bei Szenario 1. Für den Ertrag gilt diese Aussage im verstärkten Maße. Mit jeder Preissenkung sinkt der Ertrag des Artikels, obwohl der Absatz kontinuierlich steigt. In Szenario 5 dagegen wird dieser Artikel durch ein Faltblatt beworben, wobei der Preis sogar leicht höher ausfällt als in Szenario 4. Die Bewerbung hat einen erheblichen Anstieg des Abverkaufs und des Umsatzes zur Folge. Im Verhältnis zum Absatz fällt die Erhöhung des Ertrags allerdings gering aus. Dennoch kann mit dieser Variante der Ertrag auf ein Maximum gehoben werden. In dieser Rechnung fehlen allerdings noch die Kosten für die durchzuführende Promotion, die man in der Praxis mit einrechnen müsste. Diese würden den Ertrag in Szenario 5 weiter verringern. Man sieht sehr schön, dass in Nicht-Promotionsszenarien der Händler mit dem höchsten Preis am Besten fährt, da der Ertrag in Szenario 1 beim Preis von 1,25 Euro am höchsten ausfällt. Ein in dieser Simulation noch nicht berücksichtigter Einflussfaktor, der insbesondere bei Szenario 5 eine Rolle spielen könnte, ist die Wechselwirkung des beworbenen Artikels mit anderen Artikeln der Warengruppe. Es ist nicht auszuschließen, dass durch die Bewerbung des Artikel Kannibalisierungseffekte bei anderen Artikeln auftreten, die den Gesamtertrag der Warengruppe unverändert lassen oder sogar schmälern. Dies ist Thema des nächsten Abschnittes.

Der folgende Graph zeigt die Simulationsergebnisse in grafischer Form:

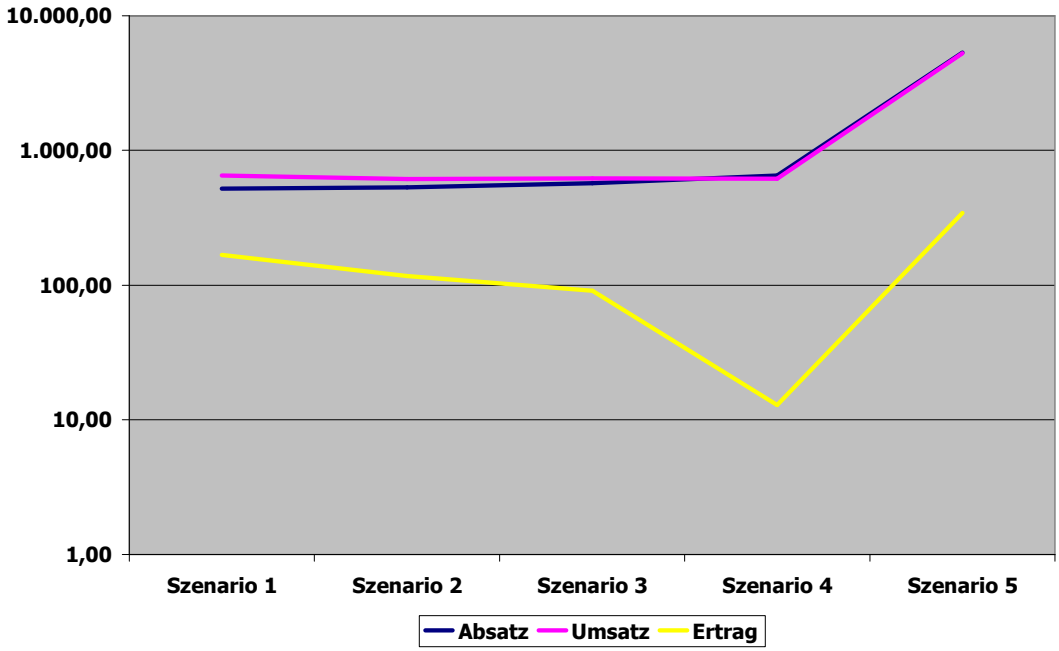


Abbildung 49: Simulation eines Artikels mit unterschiedlichen Szenarien

In Abbildung 49 sieht man, dass der Ertrag in Szenario 5 und 1 am höchsten ist, obwohl der Absatz von Szenario 1 bis 5 kontinuierlich steigt. Bei Szenario 4, dass den niedrigsten Preis simuliert, bricht der Ertrag trotz Absatzanstieg besonders stark ein.

5.3.3 Abhängigkeitsanalyse durch die Simulation mit Holonen

Im Folgenden wird der Nutzen der Holonenbildung für die Simulation von alternativen Szenarien und für Abhängigkeitsanalysen z. B. für die Optimierung von Werbefaltblättern gezeigt. Dazu wurde eine Holonsimulation mit zwei sich gegenseitig beeinflussenden Artikeln mit fünf verschiedenen Szenarien durchgeführt. Die generelle Vorgehensweise bei der Durchführung einer Abhängigkeitsanalyse oder einer Holonsimulation beginnt zunächst mit einer Vorauswahl der zu substituierenden Artikel. Der zu verändernde Artikel wird mit den Artikeln der Warengruppe und anderer potenzieller Substitutions- oder Cross-Selling-Artikel in einer Korrelationsanalyse verglichen. Die Idee ist, dass abhängige Artikel, egal, ob durch Push-, Pull- oder Kannibalisierungseffekte, zumindest zeitweise positive oder negative Korrelationen mit dem Fokusartikel haben müssen. Genau dies ermittelt der

Standardkorrelationskoeffizient nach Pearson, der im ersten Schritt auf die Artikel angewandt wird. Das Ergebnis wird nach den absoluten Korrelationswerten sortiert.

Artikel	Korrelationswert
6125004	1
6126001	0,509
141656005	0,32
141653004	0,308
357743001	0,3
...	...

Tabelle 14: Ergebnisse der Korrelationsanalyse

Tabelle 14 zeigt das sortierte Ergebnis einer solchen Korrelationsanalyse. Der erste Artikel auf der Liste ist der Fokusartikel, der mit sich selbst mit dem Wert 1 korreliert. Der Zweite auf der Liste ist ein anderer Artikel, bei dem es sich aber, wie man schon an der Artikelnummer erkennen kann, um die gleiche Marke oder das gleiche Produkt, nur mit einer anderen Geschmacksrichtung, handelt. Zwischen diesen Artikel existiert selbstverständlich eine große Ähnlichkeit, die aber für die Simulation nicht von Bedeutung ist, da man diese Produkte wie ein einzelnes Produkt auffasst. Der dritte Artikel auf der Liste ist ein Produkt einer anderen Marke und soll nun für eine Holonsimulation verwendet werden. Das Ziel einer Holonsimulation ist nicht nur herauszufinden wie sich der Fokusartikel unter bestimmten Bedingungen verkauft, sondern auch, wie die anderen Artikel des Holons auf die Veränderungen des Fokusartikels reagieren. Gibt es keine Abhängigkeiten zwischen diesen Artikeln oder lassen sich Push-, Pull- oder Kannibalisierungseffekte beobachten? Quantifizierte Aussagen über diese Effekte können dann für eine exakte Ertragsrechnung für die festgelegten Szenarien verwendet werden. Damit lässt sich das eigentliche Ziel einer jeden Maßnahme, die Ertragsoptimierung über die gesamte Warengruppe und darüber hinaus, durch ex ante Simulationen erreichen.

Im vorliegenden Fall wurden zwei stark korrelierende Artikel einer Warengruppe über den Zeitraum 12.07.2005 – 14.09.2005 in fünf verschiedenen Szenarien mit einem Holon simuliert.

ArtikelNr	AbDatum	BisDatum
6125004	12.07.2005	14.09.2005
141656005	12.07.2005	14.09.2005

Szenario	Artikel	VKPreis	EKPreis	Aktion	Verkaufs-menge	Umsatz	Ertrag	VKMenge (exp. Gl.)
Szenario 1	gesamt				1.609,13	4.319,57	2.062,14	1.308,73
	6125004	2,79	1,00	nein	395,54	1.103,56	708,02	517,54
	141656005	2,65	1,51	nein	1.213,59	3.216,00	1.354,12	791,20
Szenario 2	gesamt				1.563,13	3.969,84	1.802,00	1.308,73
	6125004	2,25	1,00	nein	431,16	970,12	538,95	517,54
	141656005	2,65	1,51	nein	1.131,97	2.999,72	1.263,05	791,20
Szenario 3	gesamt				3.334,50	7.387,07	3.444,38	1.308,73
	6125004	1,99	1,00	ja	2.196,00	4.370,04	2.174,04	517,54
	141656005	2,65	1,51	nein	1.138,00	3.017,03	1.270,34	791,20
Szenario 4	gesamt				2.995,65	6.088,61	2.641,75	1.308,73
	6125004	1,79	1,00	ja	2.151,00	3.850,29	1.699,29	517,54
	141656005	2,65	1,51	nein	844,65	2.238,32	942,46	791,20
Szenario 5	gesamt				2.266,51	3.097,17	1.928,68	1.308,73
	6125004	1,79	1,00	ja	1.168,50	2.091,62	923,12	517,54
	141656005	2,45	1,51	nein	1.098,00	2.690,12	1.005,56	791,20

Tabelle 15: Ergebnis der Holonsimulation

Tabelle 15 zeigt das Ergebnis der Holonsimulation. In den Szenarien 1-4 wurde jeweils der Preis des ersten Artikels schrittweise gesenkt, während der Preis des zweiten Artikels konstant bleibt. In Szenario 5 würde zusätzlich der Preis von Artikel 2 gesenkt. Außerdem wurde bei den Szenarien 3-5 der Artikel 1 beworben.

Zwischen Szenario 1 und 2 erkennt man sehr schön, wie auf Grund der Preissenkung die Abverkaufsmenge von Artikel 1 steigt. Leider tritt bei Artikel 2 ein Kannibalisierungseffekt auf, so dass dieser an Absatzmenge verliert. Außerdem kann die erhöhte Absatzmenge von Artikel 1 nicht den Rohertragsverlust durch die Preissenkung ausgleichen, so dass nicht nur das Gesamtergebnis von Szenario 2, sondern auch die Einzelergebnisse der Artikel schlechter ausfallen.

In Szenario 3 wird der Preis von Artikel 1 weiter gesenkt und zudem mit einer Aktion verknüpft. Diese ist von durchschlagendem Erfolg. Der Absatz steigt extrem und auch der

Ertrag kann den Sortimentsverantwortlichen zufrieden stellen. Artikel 2 verliert trotz der Aktion und der Preissenkung nicht weiter an Absatz, so dass das Gesamtergebnis für Szenario 3 sehr positiv ausfällt.

In Szenario 4 wird der Preis von Artikel 1 noch einmal gesenkt und die Bewerbung beibehalten. Leider führt das nicht zu einer verbesserten Abverkaufsmenge, sondern schmälert nur den Ertrag von Artikel 1. Die weitere Preissenkung von Artikel 1 hat allerdings noch einen weiteren negativen Effekt, der Absatz von Artikel 2 fällt ebenfalls. Der Grund hierfür könnte sein, dass durch die starke Preissenkung von Artikel 1 das Preisgefüge der gesamten Warengruppe durcheinander gekommen ist. Artikel 2 ist zwar nicht im Preis erhöht worden, allerdings ist der Preisunterschied und somit der relative Preis von Artikel 2 weiter gestiegen. Vermutlich sind die Kunden deshalb auf andere Artikel ausgewichen oder noch schlimmer, sie haben den Kauf ganz unterlassen. Insgesamt fällt der Ertrag von Szenario 4 damit schlechter aus als bei Szenario 3.

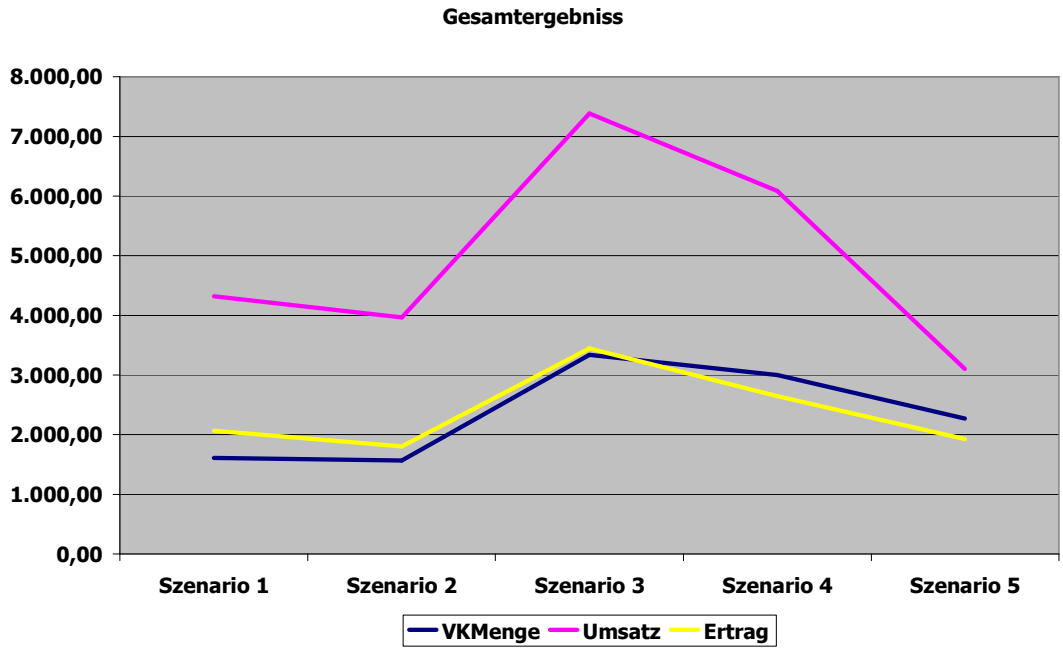


Abbildung 50: Das Gesamtergebnis einer Holonsimulation

In Szenario 5 wird nun zusätzlich der Preis von Artikel 2 gesenkt, Artikel 1 bleibt im Vergleich zu Szenario 4 unverändert. Diese Veränderung verringert den relativen Preis von Artikel 2, so dass wieder mehr Kunden diesen Artikel kaufen. Der vorherige Verlust kann aber nicht ganz ausgeglichen werden. Zusätzlich hat diese Preissenkung erhebliche Auswirkung auf Artikel 1. Anscheinend gelingt es nicht die verloren gegangenen Kunden des

Artikels 2 (siehe Szenario 4) zurückzugewinnen. Artikel 2 scheint sein Plus an Absatz im Vergleich zu Szenario 4 ausschließlich durch ehemalige Käufer des Artikels 1 zu erzielen. Darüber hinaus verliert Artikel 1 aber noch weitere Kunden. Vermutlich da nun der relative Preis wieder gestiegen ist und die Aktion durch die Preissenkung von Artikel 2 nicht mehr die gewünschte Wirkung zeigt. Insgesamt verliert Szenario 5 durch die Verluste bei Artikel 1 erheblich an Ertrag im Vergleich zu Szenario 4, obwohl der Absatz und auch der Ertrag von Artikel 2 steigt.

Abbildung 50 zeigt noch einmal grafisch die Gesamtergebnisse aller Szenarien mit Menge, Umsatz und Ertrag. Man sieht sehr deutlich, dass Szenario 3 insgesamt das beste Ergebnis liefert, obwohl in den Szenarien 4 und 5 noch weitere Preissenkungen durchgeführt werden.

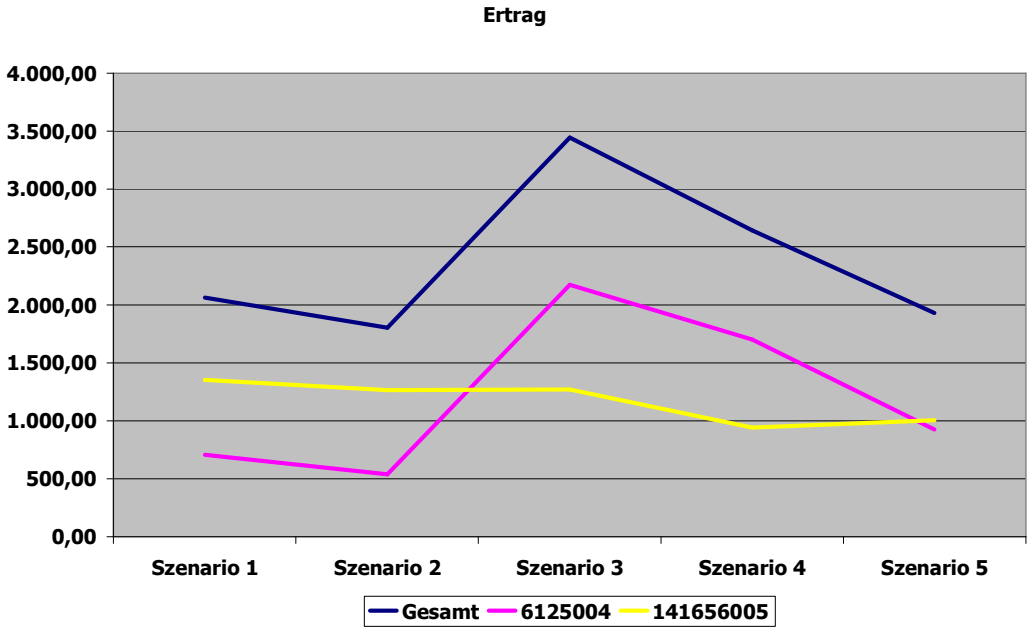


Abbildung 51: Der Ertrag der Szenarien für die verschiedenen Artikel des Holons

Abbildung 51 zeigt die Erträge der einzelnen Artikel und den Gesamtertrag. Man sieht, dass der Ertrag von Artikel 2 durch die Settings von Artikel 1 beeinflusst wird und dass das Gesamtergebnis von Szenario 3 das Beste ist, obwohl Artikel 2 in diesem Szenario nicht seinen höchsten Ertrag erzielt. In Szenario 5 fällt der Ertrag von Artikel 1 trotz Aktion und massiver Preissenkung sogar unter das Niveau von Artikel 2.

6 Fazit und Ausblick

6.1 Fazit

In dieser Arbeit wurde gezeigt, wie man mit probabilistischen Agenten das Abverkaufsverhalten von Artikeln im Einzelhandel ex ante simulieren kann. Dazu wurde ein probabilistisches holonisches Multiagentensystem (PH-MAS) entwickelt, bei dem die Wissensbasen der Agenten mittels Data-Mining-Verfahren aus den Realdaten der Händler extrahiert werden. Die gewonnenen Abverkaufsmuster werden in Verhaltensnetze kodiert, die für eine Simulation der repräsentierten Artikel verwendet werden. Es wurde gezeigt, wie der Kern der Verhaltensnetze durch Bayes'sche Netze realisiert werden kann, die für die Verwendung in der Retaildomäne in vielen Details erweitert werden mussten. So wurden die neuen Evidenzarten Soft- und Extrapolationsevidenzen eingeführt und konkretisiert. Für die Modellierung und Simulation von globalen Abhängigkeiten zwischen Artikelagenten wurde ein Verschmelzungsalgorithmus vorgestellt, der die probabilistischen Verhaltensnetze der Agenten in holonische Metanetze fusioniert. Es wurde gezeigt, wie mit den Verhaltensnetzen eine Trendanalyse und infolgedessen eine Trendkorrektur der prognostizierten Daten durchgeführt werden kann. Des Weiteren wurde eine mehrdimensionale Simulationssprache (MSL) für beliebige Verhaltensnetze und andere mehrdimensionale Wissensrepräsentationsformen vorgestellt. Schließlich wurde eine selbst optimierende Simulationsroutine präsentiert, die Szenarien in Abfolgen von Netzkonfigurationen konvertiert, um effiziente Simulationen mit den Verhaltensnetzen zu ermöglichen. Diese entwickelten Technologien wurden im Rahmen dieser Arbeit vollständig in die Agenten des PH-MAS integriert. Mit Hilfe des neu entwickelten verteilten Agentenframeworks MASg wurden die PH-Agenten auf der Basis der .NET-Technologie realisiert und zu einem umfangreichen Simulationssystem ausgebaut. Das gesamte SimMarket-Simulationssystem umfasst die folgenden Funktionalitäten:

- Analyse, Simulation und Prognose der Auswirkungen von Maßnahmen des Sortimentsmanagements, wie z. B. Preis-, Sortiments- und Platzierungsänderungen sowie Promotionsaktionen auf Artikelabverkäufe („Was-wäre-wenn-Szenarien"), auf der Basis von probabilistischen Verhaltensnetzen,
- selbst lernende Modellgenerierung und Adaption vorhandener Netze,
- Erkennen von Abhängigkeiten – z. B. Korrelationen – zwischen Artikeln,
- Modellierung und Simulation der Auswirkung von nicht linearen, externen Einflussfaktoren wie beispielsweise Saisonalitäten, Wetter, Aktionen der Konkurrenz, etc.

- Analyse, Simulation & Prognose durch Mikrosimulationsansatz auf allen Hierarchieebenen (Artikel-, Miniwarengruppen-, Warengruppen-, Filial-Ebene),
- quantitative Berücksichtigung von nicht linearen Abhängigkeiten zwischen Artikeln (Kannibalisierungs-, Push- und Pulleffekte) mittels Holonisierung von probabilistischen Netzen während der Simulation,
- Trendanalysekomponente, die zur Korrektur von trendbehafteten Entitäten während einer Simulation verwendet werden kann,
- umfassende Evaluationskomponente, zur Evaluierung des Systems mit Realdaten.

Abschließend lässt sich sagen, dass die Ergebnisse dieser Arbeit zeigen, dass bereits sehr viele Artikel zuverlässig simuliert werden können. Dennoch existieren noch einige Probleme bei der Simulation von bestimmten Kategorien von Artikel und Szenarien. So zeigen die Ergebnisse, dass insbesondere langsamdrehende Artikel und Produkte mit unzureichender Datengrundlage schlecht simuliert werden können. Dies ließe sich aber durch geschickte Aggregation von ähnlichen bzw. gleichen Artikeln lösen und ist damit im Wesentlichen ein Datenbankproblem.

6.2 Ausblick

6.2.1 Erweiterung der *Simulation Engine*

6.2.1.1 Analogien

Eine noch zu lösende Problematik ist die Simulation von Artikeln für die nur eine unzureichende oder gar keine Datengrundlage existiert. In solchen Fällen kann das System nur mit analogen Artikeln arbeiten, die ein sehr ähnliches Verhalten haben. Die Herausforderungen bestehen dabei 1. in der Definition von „ähnlichen Artikeln“, die sich sehr wahrscheinlich von Anwendungsfall zu Anwendungsfall unterscheidet, und 2. ein Verfahren zu finden, das zu einem gegebenen Artikel analoge Artikel findet. Ein solches Verfahren sollte nicht über die Daten sondern über Meta-Attribute automatisch die ähnlichsten Artikel findet. Diese (relativen) Attribute wie z. B. Größe, Geschmack, Farbe, Kategorie, Zielgruppe, Preis müssen daher in einer Datenbank vorliegen, was bisher bei fast allen Handelsunternehmen nicht der Fall ist. Sollten solche Beschreibungen aber vorliegen, dann könnte ein Sortimentsmanager diese dazu verwenden, einen neuen Artikel manuell durch Auswahl der zutreffenden Attribute zu beschreiben und anschließend mit automatischen Verfahren mit den anderen Artikeln in der Datenbank zu vergleichen. Als

vergleichende Verfahren eignen sich bei diesem Anwendungsfall prinzipiell alle vektorbasierten Clustering- und Klassifikationsverfahren. Sind zu einem neuen Artikel Ähnliche gefunden worden, können die Daten dieser für (relative) Simulationen verwendet werden. Damit könnte man schließen, dass die Erhöhung des Preises bei Artikel A zu einer Verringerung des Abverkaufs um 5% führt, folglich wird auch der Abverkauf von dem analogen Artikel B um 5% fallen, wenn dessen Preis um den gleichen Prozentsatz erhöht wird. Um zu solchen Aussagen zu kommen, müssen die Netze entsprechend mit relativen Knoten wie in Kapitel 4.5.2 beschreiben modelliert werden (*relative Netze*).

Liegen dagegen für einen Artikel Daten vor, die für eine Simulation unzureichend sind, so können die Daten von analogen Artikeln mit den Daten des zu simulierenden Artikels aggregiert werden, um den Umfang der Datenbasis auf ein brauchbares Niveau zu bringen. Sind die Artikel ausreichend ähnlich im Abverkauf, kann dieser virtuelle neue Artikel statt der einzelnen Artikel bei entsprechender Gewichtung der Ergebnisse für Simulationen verwendet werden.

6.2.1.2 Erweiterung der Verhaltensnetze

Im Rahmen dieser Arbeit wurden zahlreiche Techniken aus dem Forschungsgebiet der Bayes'schen Netze vorgestellt, implementiert, evaluiert und anschließend verbessert. Dennoch gibt es einige Erweiterungen dieser Technologie, die in dieser Arbeit nicht empirisch untersucht werden konnten, aber dennoch interessant genug klingen, um in weiteren Forschungsprojekten näher untersucht zu werden. Folgende Erweiterungen der Verhaltensnetze sollten noch implementiert und evaluiert werden:

- weitere Lernverfahren
- weitere Inferenzverfahren
- Adaptionsverfahren
- Kontinuierliche Knoten
- Entscheidungsnetze
- Dynamische Bayes'sche Netze
- Relative Netze
- Andere Verfahren zum Setzen von mehreren Softevidenzen
- Performanzoptimierungen der Netze und der Simulation

6.2.1.3 Funktionale Erweiterung der *Simulation Engine*

Nicht nur die Verhaltensnetze sollten in Zukunft erweitert werden, sondern auch die *Simulation Engine* an sich. So werden zu den bereits integrierten Verfahren, weitere

Clustering- und Klassifikations-Algorithmen, sowohl auf Vektor-, als auch auf Netzbasis integriert werden, um Segmentierungen auf der Menge alle Artikel durchführen zu können und Analogien in den Sortimenten zu finden. Außerdem werden weitere Korrelationsverfahren, insbesondere multivariate, in zukünftigen Versionen benötigt werden. Außerdem sollte untersucht werden, wie und unter welchen Bedingungen andere Prognoseverfahren mit den in dieser Arbeit entwickelten Techniken kombiniert werden können. Dies wurde ja bereits erfolgreich bei der Trendkorrektur realisiert.

Ein bisher in SimMarket noch nicht bearbeiteter Bereich ist die klassische *Verbundanalyse*[28], die das Verhältnis von Artikeln zueinander beim Abverkauf analysiert, z. B. welche Artikel werden bevorzugt miteinander gekauft. Diese Verfahren sind sehr einfach gehalten und lassen sich ohne weiteres in die *Simulation Engine* integrieren. Einige der von den Anwendern der Handelsunternehmen gewünschten Anwendungsfälle (siehe Anhang E) setzen eine derartige Komponente voraus.

6.2.2 Weitere Anwendungsmöglichkeiten der *Simulation Engine*

6.2.2.1 Konsumgüterindustrie

Eine nahe liegende Anwendung der SimMarket *Simulation Engine* ist der Bereich Konsumgüterindustrie, der sehr eng mit dem Einzelhandel verwandt ist. Problematisch ist allerdings häufig die unzureichende oder nicht vorhandene Datengrundlage, da der Hersteller häufig über keinerlei Abverkaufsdaten sondern nur über Lieferzahlen verfügt. Viele Effekte und Einflüsse sind allerdings in diesen Daten nicht zu finden z. B. kurzfristige Wettereinflüsse oder Konkurrenzaktionen, so dass eine Mikrosimulation im eigentlichen Sinne nicht möglich ist. Der Hersteller benötigt somit in vielen Fällen die Daten der Händler, die z. B. in gemeinsamen CPFR-Projekten oder durch andere Abkommen ausgetauscht werden müssen.

6.2.2.2 Finanzbereich

Eine weitere Anwendungsmöglichkeit besteht im Finanzbereich, bei Banken, Versicherungen und im Bereich Geldanlage z. B. bei Fond-/Depotmanagern. Bei all diesen potenziellen Anwendern liegen sehr umfangreiche und vollständige Daten über geschäftliche

[28] Ein sehr einfacher Spezialfall der Abhängigkeitsanalyse.

Transaktionen vor, die von der *Simulation Engine* genutzt werden könnten, um Simulationen bezüglich Wertentwicklung von Geldanlagen, Indices etc. durchführen zu können.

6.2.2.3 Makroökonomische Simulationen

Noch interessanter, aber auch schwieriger ist die Anwendung der *Simulation Engine* auf makroökonomische Problemstellungen, schwieriger deshalb, weil nicht immer die erforderlichen Daten und Lernfälle vorliegen. Man stelle sich vor, man könnte z.B. die Auswirkungen von Wahlen auf das Konsumverhalten und die Risikobereitschaft bzgl. des Konsums der Bevölkerung besser vorhersagen. Dies wäre für alle Parteien, für Volkswirtschaftswissenschaftler und Politiker, aber auch wiederum für Banken, Versicherungen, die Industrie, Konsumgüterhersteller und somit auch wieder für die Händler von großem Interesse, denn wenn ich bestimmte volkswirtschaftliche Tendenzen vorhersagen kann, können die Unternehmen entsprechend vorab reagieren und die Produktionslage darauf anpassen.

A Syntax der Simulationssprache MSL

Die Syntax von MSL wird im Folgenden in erweiterter *BNF (Backus-Naur-Form)* definiert:

```
<MSLquery> ::=  simulate <resultDimensions>
                with <BNetID> [ where <slicerDimension> ]


<resultDimensions> ::= <set> on columns [ “,” <set> on rows [
“,” <set> on pages [ “,” <set> on chapters [ “,” <set> on
sections ] ] ] ]

<BNetID> ::= “[“ BNetzreferenz “]”

<slicerDimension> ::= <tupel>

<set> ::= “{“ <tupel> { “,” <tupel> } “}”
          | “{“ <set> <setop> <set> “}”
          | “{“ <tupel> <tupelop> <tupel> “}”
          | filter “(“ <set> “,” <searchCondition> “)”

<tupel> ::= <node>
          | “(“ <node> “,” <node> { “,” <node> } “)”
          | <node> “.” <nodeop>
          | <state>
          | <state> “.” <stateop>
          | <member>

<setop> ::= “*” | “#” // CrossJoin() bzw. NonEmptyCrossJoin()

<tupelop> ::= “:”    // Range

<node> ::= <level> “.” “[“ nodeID “]”
          | “[“ nodeID “]”
          | <level> “.” <levelop>

<member> ::= <level> “.” “[“ memberID “]”
          | “[“ memberID “]”
```

<level> ::= “[“ *levelname* “]” {“.” “[“ *levelname* “]” }

<levelop> ::= **members**

<nodeop> ::= **exp**
| “[“ **exp** “]” “:=” “’” *expvalue* “’”
| “[“ **distr** “]” “:=” “’” <distrtupel> “’”

<state> ::= <node> “.” “[“ **states** “]” “.” “[“ **indices** “]” “.”
“[“ *index* “]”
| <node> “.” “[“ **states** “]” “.” “[“ **realavg** “]” “.”
“[“ *value* “]”

<stateop> ::= **prop** | **avg** | **realavg** | **lborder** |**rborder**

<distrtupel> ::= “(“ *prob* “ { “,” *prob* } “)”

<searchCondition> ::= Ein boolescher Ausdruck mit <tupel> als Operanden.

B Data-Mining-Methoden

Assoziationsanalyse

Die *Assoziationsanalyse* – auch *Verbundanalyse* genannt – findet Abhängigkeiten zwischen Variablen in großen Datenmengen (Transaktionen) in Form von Regeln. Das klassische Anwendungsgebiet der Assoziationsanalyse ist die *Warenkorbanalyse*, bei der Assoziationsregeln der Form „Wenn Artikel A verkauft wird, dann wird mit einer Wahrscheinlichkeit X auch Artikel B verkauft“ extrahiert werden. Die Wahrscheinlichkeit wird *Konfidenz* (*Confidence)* genannt und steht für die Stärke der Regel. Zusätzlich wird der *Support* ermittelt, der die Häufigkeit der Regel beschreibt. Die Assoziationsanalyse gibt wertvolle Hinweise zur Sortimentsoptimierung und deckt Cross-Selling-Potenziale auf. Oftmals liefert die Assoziationsanalyse Hypothesen, die mittels weiterer Data-Mining-Verfahren, wie Klassifikation oder Clusteranalyse, geprüft und weiter untersucht werden können.

Häufig ist es sinnvoll die Assoziationsanalyse nicht auf Artikelebene durchzuführen sondern auf einer höheren Ebene der Warengruppenhierarchie. So ist eine Regel der Form „Wenn Kunden Wein kaufen, dann kaufen sie in x% aller Fälle auch Käse“ sinnvoller als „Wenn Kunden Wein der Marke A des Jahrganges B kaufen, dann kaufen sie in x% aller Fälle auch Käse der Marke C in Produktgröße D“. Diese Regeln weisen im Allgemeinen eine sehr geringe Konfidenz und einen sehr geringen Support auf.

Ein Spezialfall der Assoziationsanalyse ist die *Sequenzanalyse*, die auch *Zeitreihenanalyse* genannt wird. Bei der Sequenzanalyse wird zusätzlich die zeitliche Reihenfolge der untersuchten Transaktionen berücksichtigt. Eine Sequenz ist eine zeitlich geordnete Menge [a, b, c] an Transaktionen. Der Support von [a, b, c] ergibt sich aus der Häufigkeit mit der genau diese Sequenz in der gesamten Transaktionsmenge vorkommt. Versicherungen nutzen die Sequenzanalyse, um die typische Reihenfolge von Versicherungsabschlüssen ihrer Kunden zu ermitteln. Das Ergebnis lautet dann beispielsweise, dass mit einem Support von x% als Erstes eine Kfz-Versicherung, dann eine Haftpflichtversicherung und anschließend eine Lebensversicherung abgeschlossen wird. Zudem wird die Sequenzanalyse für die Untersuchung von Webserver Log-Files genutzt, um das Surfverhalten von Käufern in Onlineshops zu analysieren.

Die Qualität der Ergebnisse einer Assoziationsanalyse sind sehr unterschiedlich. Es können Informationen gewonnen werden, die direkt in gewinnmaximierende Aktionen umgesetzt werden können, aber auch weniger brauchbare Regeln können das Ergebnis sein. Viele Regeln sind entweder trivial, weil sie schon vorher allgemein bekannt waren, oder unbrauchbar, weil sie nicht für sinnvolle Handlungen genutzt werden können. Ein Vorteil der Assoziationsanalyse ist, dass die extrahierten Regeln sehr gut von Menschen gelesen werden

können und somit gut interpretierbar sind. Dies ist bei anderen Data-Mining-Verfahren oft nicht der Fall.

Regeln können sowohl vorgegeben und während der Analyse nur geprüft werden oder es können regeln automatisch gefunden werden.

Ein Nachteil der Assoziationsanalyse ist, dass der Aufwand exponentiell mit der Anzahl der zu untersuchenden Artikel und der Komplexität der Regeln wächst. Für einen guten Überblick über die wichtigsten Algorithmen zur Generierung von Assoziationsregeln siehe [HanKam01].

Entscheidungsbäume

Beim Entscheidungsbaum-Verfahren wird eine gegebene Trainingsmenge sukzessiv anhand des jeweils trennschärfsten Merkmals in kleinere Teilmengen unterteilt. Durch das schrittweise Vorgehen ergibt sich nach n Teilungen ein Baum mit einer maximalen Pfadtiefe von n-1. Zielsetzung ist die Generierung möglichst homogener Gruppen auf der untersten Baumebene. Um diese Homogenität zu erreichen und eine Überanpassung des Modells an die Trainingsdaten (engl. overfitting) zu reduzieren, wird ein Baum häufig nach der Erzeugung zurechtgestutzt (engl. pruning) bzw. generalisiert. Dadurch wird unerwünschtes Datenrauschen und Ausreißer (engl. outliers) aus dem Baum entfernt und führt zu einer erhöhten Trefferquote. Der resultierende Baum kann anschließend zur Klassifikation neuer Objekte genutzt werden. Die Funktionsweise ist die Folgende: Jeder interne Knoten des Baumes repräsentiert einen Test (Entscheidungsknoten) und jeder ausgehende Ast einen möglichen Ausgang des Tests.

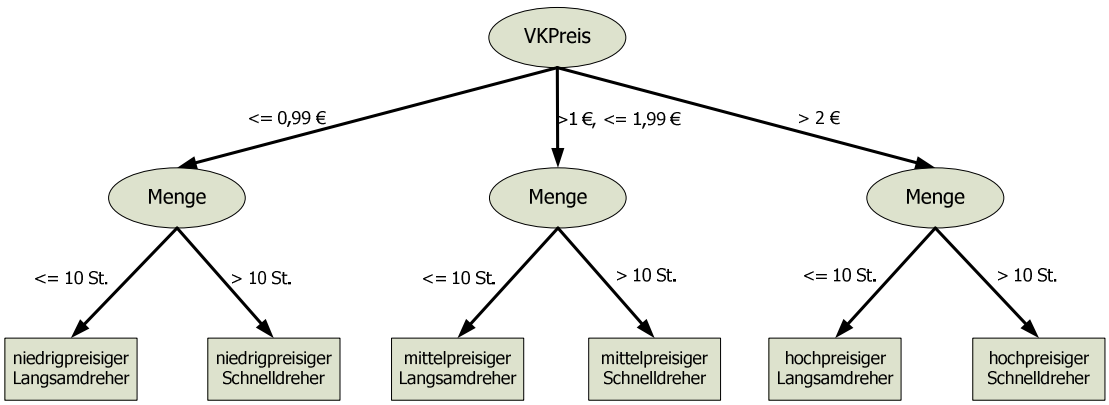


Abbildung 52: Beispiel für einen Entscheidungsbaum

Ein zu klassifizierendes Objekt wird an die Wurzel des Baumes übergeben, der Ausgang des Tests in der Wurzel und den anderen Knoten des Baums entscheidet über den Verlauf des

Pfad des Objekts durch den Baum bis runter zu einem Blatt. Das Blatt in dem das Objekt landet steht für die Gruppe, die diesem Objekt am ähnlichsten ist.

Entscheidungsbäume werden meistens als Flussdiagramme grafisch dargestellt und können deshalb leicht nachvollzogen und interpretiert werden.

Der bekannteste und zugleich einfachste Algorithmus um Entscheidungsbäume zu erzeugen ist der ID3-Algorithmus [HanKam01]. Die Klassifikationsgüte eines Entscheidungsbaumverfahrens wird im Wesentlichen durch das verwendete Attributselektionsmaß und das Pruning-Verfahren bestimmt, die im Nachfolgealgorithmus C4.5 deutlich verbessert worden sind. Des Weiteren wurde das Behandeln von kontinuierlichen und fehlenden Werten verbessert und eine höhere Ausführungseffizienz erreicht.

Diskriminanzanalyse

Unter dem Begriff Diskriminanzanalyse werden in der klassischen Statistik Methoden zur Klassifizierung von Objekten zusammengefasst. Die Diskriminanzanalyse geht davon aus, das eine Menge G von Objekten aus $S_i >= 2$ Teilmengen besteht und $\{S_1 \cap \ldots \cap S_n\} = \varnothing$ ist. Die Objekte lassen sich nur durch die gleichzeitige Betrachtung von $p > 1$ messbaren Merkmalen unterscheiden. Für jedes Objekt muss also für die Diskriminierung ein p-dimensionaler Merkmalsvektor $x \in \Re^p$ vorliegen. Liegt außerdem für ausreichend viele Objekte die Klassenzugehörigkeit k vor, so lassen sich eine oder mehrere Trennfunktionen (Diskriminanzfunktionen) bestimmen mit deren Hilfe sich Objekte ohne Klassenzugehörigkeit in eine der vorliegenden Gruppen zuordnen lassen. Im Laufe der Zeit haben sich drei wesentliche Vorgehensweisen der Diskriminanzanalyse herauskristallisiert:

- Lineare Diskriminanzanalyse nach Fisher,
- Nächste-Nachbarn-Diskriminanzanalyse,
- Maximum-Likelihood-Diskriminanzanalyse.

Der *Ansatz von Fisher* (1936) basiert auf der Verwendung einer linearen Transformation $y = \mathbf{b}x$ der p-dimensionalen Größen x in eine eindimensionale Größe y durch einen reellen Vektor $\mathbf{b} = (b_1,\ldots,b_p)$. Die Komponenten von **b** müssen derart bestimmt werden, dass die Zerlegung der x-Werte der Trainingsmenge möglichst gut wiedergegeben wird. Dies ist der Fall, wenn die Mittelwerte $MW(y_i)$ der Gruppen (Gruppenzentroide) möglichst weit auseinander liegen. Durch Maximierung der Abstände zwischen den Zentroiden erhält man eine Menge von Diskriminanzfunktionen, die für die Klassifizierung eines Objektes ohne Klassenzugehörigkeit genutzt werden können. Ein Objekt mit dem Merkmalsvektor x wird der Klasse i zugeordnet zu dessen Zentroid $MW(y_i)$ der Abstand $D_i(x) = |y - MW(y_i)| = |\mathbf{b}x -$

$MW(y_i)|$ am geringsten ist. Die Maximierung der Abstände erreicht man, wenn man die Summe der mit den Gruppenumfängen gewichteten Abstände der $MW(y_i)$ vom Gesamtmittel MW(y) relativ zur Summe der quadratischen Fehler maximiert. Diese Optimierung ist auf ein Eigenwert-Eigenvektorproblem zurückführbar und bedient sich der entsprechenden Methoden der linearen Algebra.

Die *Nächste-Nachbarn-Diskriminanzanalyse* basiert auf der Definition eines Abstandsmaß $d(x, x_{ij})$ zwischen dem Merkmalsvektor x eines nicht zugeordneten Objektes und den Merkmalsvektoren x_{ij} der Trainingsmenge. Dabei steht i = 1,..,g für die Gruppenzugehörigkeit und $j = 1,\ldots,n_i$ für die unterschiedlichen Elemente einer Gruppe in der Trainingsmenge. In der Literatur existieren zahlreiche Abstandsmaße wie der euklidische Abstand oder der Mahalanobis-Abstand, die bei dieser Analyse verwendet werden können. Ein Objekt mit dem Merkmalsvektor x wird der Gruppe z zugeordnet, die den Vektor der Trainingsmenge mit dem geringsten Abstand zu x enthält $d(x, x_{zj}) = \min\{d(x, x_{ij}), i=1,\ldots,g; j=1,\ldots,n_i\}$. Eine Erweiterung der Nächste-Nachbar-Regel besteht darin, dass man statt einem k nächste Nachbarn von x aus der Trainingsmenge berücksichtigt (*k-Nächste-Nachbarn-Diskriminanzanalyse*). Infolgedessen erhält man k nächste Nachbarn vom Merkmalsvektor x, wobei die k Nachbarn aus unterschiedlichen Gruppen i stammen können. Ein zu klassifizierendes Objekt wird dann der Gruppe i^* (i = 1,…,g) mit der maximalen Anzahl an Nachbarn aus der Menge der k-nächsten Nachbarn zugeordnet $k_i^* = \max\{k_1,\ldots,k_g\}$. Leider ist diese Entscheidungsregel nicht immer eindeutig.

Bei unterschiedlichen Umfängen der Trainingsmengengruppen n_i kann es sinnvoll sein bei der Klassifikation die Anzahl der k-nächsten Nachbarn mit der jeweiligen Trainingsmengengröße zu gewichten. Außerdem ist es in der Regel sinnvoll die Anzahl der zu bestimmenden nächsten Nachbarn pro Gruppe festzulegen. Dazu nimmt man die Summe der Abstände zum zu klassifizierenden Objekt, um die ermittelten Nachbarn damit zu gewichten. Zunächst bestimmt man aus jeder Gruppe die k_i nächsten Nachbarn x_{ij} mit $j=1,\ldots,k_i$ und $k = \sum_{i=1}^{g} k_i \geq g$. Die Anzahl der k_i sollte dabei proportional zur Gruppengröße n_i in der Trainingsmenge sein. Die Entscheidungsregel lautet dann:

$$\frac{k_{i^*}}{\sum_{j=1}^{k_{i^*}} d(x, x_{i^*j})} = \max\left(\frac{k_1}{\sum_{j=1}^{k_1} d(x, x_{ij})}, \ldots, \frac{k_g}{\sum_{j=1}^{k_g} d(x, x_{ij})} \right).$$

Weitere Entscheidungsregeln z.B. mit a-priori Wahrscheinlichkeiten sind möglich.

Bei der *Maximum-Likelihood-Diskriminanzanalyse* handelt es sich um ein statistisches Schätzverfahren bei dem man die Merkmalsvektoren $x_i = (x_1,\ldots,x_n)$ der Objekte als Realisierung einer Zufallsgröße $\underline{x}_i$ auffasst. Bei gegebener Trainingsmenge und Annahmen über die Verteilung der relevanten Variablen wird geprüft bei welchen Parametern in der Grundgesamtheit die Daten der Trainingsmenge am wahrscheinlichsten sind. Es muss das Maximum einer Funktion L gefunden werden, die diese Wahrscheinlichkeiten ausrechnet. Diese Funktion wird Likelihood-Funktion genannt und gibt an, welche geschätzten Parameter bei gegebenen Daten die größte Wahrscheinlichkeit aufweißt, den realen Parametern in der Grundgesamtheit zu entsprechen oder anders ausgedrückt bei gegebenen Merkmalsvektor x gibt die Funktion die Wahrscheinlichkeit des Auftretens dieser Kombination an. Daher der Name Maximum-Likelihood-Diskriminanzanalyse. Eine Likelihood-Funktion kann je nach Verteilung mehrere lokale Maxima haben, was zur Folge hat, dass häufig das globale Maximum nicht gefunden wird. Es kann sogar sein, dass die Likelihood-Funktion kein Maximum hat und die Schätzung nicht konvergiert. Für die n Zufallsgrößen ist die Likelihood-Funktion definiert als das Produkt der Einzelwahrscheinlichkeiten der Zufallsgrößen $\underline{x}_i$. Daher ist der Wert der Funktion in der Regel schlecht handhabbar. Deshalb maximiert man häufig den Logarithmus von der Funktion (Log-Likelihood).

Bei der Verwendung der Likelihood-Funktion in der Diskriminanzanalyse wird für jede gegebene Klasse eine solche Funktion ermittelt. Ein zu klassifizierendes Objekt mit einem Merkmalsvektor x wird der Gruppe zugeordnet bei der die zugehörige Likelihood-Funktion maximal ist. Die Likelihood-Funktion dient hier also als Diskriminanzfunktion. Das Maximum-Likelihood-Verfahren (*ML-Verfahren*) ist eine allgemeine Methode, die in mehreren spezielleren Verfahren zur Anwendung kommt. So nutzt beispielsweise die Logistische Regression, die die Abhängigkeiten zwischen mehreren binären Variablen analysiert, das ML-Verfahren zur Schätzung der Regressionsparameter. Sollen Variablen mit mehr als zwei Ausprägungen analysiert werden, wendet man die erweiterten Verfahren *Multinominales Logit* bei nominal skalierten Merkmalen oder *Ordinales Logit* bei ordinalskalierten Merkmalen an. Beide Verfahren nutzen ebenfalls das ML-Verfahren zur Schätzung der Regressionsparameter.

Weitere Klassifizierungsmethoden

Eine weitere sehr aktive Forschungsrichtung ist das *Klassifizieren mit Hilfe von Assoziationsregeln* (siehe auch 0). Diese Ansätze sind stark verwandt mit Entscheidungsbaum-Verfahren. Im ersten Schritt werden Regeln der Form $Bedingung_1 \wedge \ldots \wedge Bedingung_n \Rightarrow Klasse_i$ aus einer Trainingsmenge extrahiert. Im zweiten Schritt werden für ein zu klassifizierendes Objekt ebenfalls Assoziationsregeln aus seiner Datenbasis (Zeitreihe) generiert. Im letzten Schritt werden die gelernten Objektregeln mit den Klassenregeln

verglichen. Das Objekt wird dann der Klasse mit den größten Gemeinsamkeiten in den Assoziationsregeln zugeordnet. Die existierenden Verfahren unterscheiden sich hauptsächlich in der Methode, wie Objektregeln und Klassenregeln miteinander verglichen werden und in der Generierung der Klassenregeln. Diese Methoden sind häufig stark mit den Methoden zur Erzeugung von Entscheidungsbäumen verwandt. Die Struktur der Klassenregeln muss allerdings bei einer grafischen Darstellung keinen Baum ergeben.

Ein Nachteil von regelbasierten Systemen ist, dass ursprünglich kontinuierliche Werte in den Regeln ‚scharf' diskretisiert werden müssen. Man betrachte die folgende Regel für einen Artikel:

IF (Abverkaufsmenge > 100) $\wedge$ (Rohertrag > 10%) THEN Artikel ist wertvoll.

Diese Regel besagt, dass ein Artikel für eine Firma wertvoll ist, wenn der Artikel einen Rohertrag von über 10% erwirtschaftet und über 100-mal pro Tag im Durchschnitt verkauft wird. Häufig ist eine solche scharfe Regel mit festen Schwellenregeln sehr schwer zu definieren, da ein Artikel mit 99 Abverkäufen pro Tag sicherlich nicht viel weniger wertvoll ist als ein Artikel der sich 101-mal verkauft und auf den diese Regeln zutreffen würde. In solchen Fällen wurden *Fuzzy-Sets* eingesetzt, die keine scharfen Schwellenwerte haben, sondern einen Wahrheitswerte zwischen 0 und 1, der den Grad der Zugehörigkeit zu einer Kategorie angibt. Infolgedessen hätte die Abverkaufsmenge von 99 Stück aus obigen Beispiel einen hohen Zugehörigkeitswert zu einer Klasse, die das Abverkaufsverhalten eines Artikels ausdrückt, z.B. „Schnelldreher". Dennoch wäre der Zugehörigkeitswert eines Artikels der sich 101-mal pro Tag verkauft zur Klasse „Schnelldreher" höher. Die generelle Anwendung von Fuzzy-Sets verläuft im Allgemeinen so, dass im ersten Schritt kontinuierliche Werte in diskrete Klassen abgebildet werden und eine Fuzzy-Funktion definiert wird, die die Zugehörigkeit eines Wertes zu diesen Klassen berechnet. Da auf ein zu klassifizierendes Objekt durchaus mehr als eine Fuzzy-Regel passen kann, werden die Wahrheitswerte für eine Klasse kombiniert. Dadurch trägt jede Fuzzy-Regel Wahrheitswerte für die Zugehörigkeit zu einer oder mehrerer Klassen bei. In der Regel summiert man diese Werte auf und erhält so Summen von Wahrheitswerten, die für die Gewichtung der Klassen genutzt werden können. Fuzzy-Sets können auf alle Arten von Regelsystemen insbesondere für Klassifikationsaufgaben angewendet werden.

C Marktanalyse Bayes'sche-Netz-Tools

Zu Beginn der Analyse wurde zunächst ein Anforderungskatalog aufgestellt, der ein für uns ideales BN-Tool beschreibt. Anhand dieses Kataloges wurden dann die 38 wichtigsten Programme ausgewertet, die wir bei unserer Recherche gefunden haben.

Anforderungskatalog:
Ein Bayes'sches Netz-Tool sollte folgende Funktionalitäten unterstützen:

- Lernalgorithmen, um die Struktur eines BN aus einer Datenmenge zu lernen,
- Lernalgorithmen, um die Parametrisierung des BN aus einer Datenmenge zu lernen,
- einen Diskretisierungswizard, der automatisch Vorschläge machen kann, der aber auch manuell bedienbar ist,
- den Lernverfahren sollte man beliebiges Domänenwissen vorgeben können, wie z. B. Kanten/Relationen, Ordnungen, Wurzelknoten, Blattknoten,
- die Lernverfahren müssen hinreichend gute Ergebnisse liefern,
- einen Inferenzalgorithmus, um Anfragen an das Netz stellen zu können,
- das Setzen von Likelihoods (Wahrscheinlichkeitsverteilungen),
- die „Adaption" von gelernten Netzen,
- Funktionen, um die Prognose- bzw. Diagnosegüte eines Netzes ermitteln zu können (z. B. durch Splitten der Inputdaten in eine Lern- und Testmenge),
- evtl. andere Knotenarten zu unterstützen, wie z. B. Utility-Knoten und Entscheidungsknoten (Influence Diagrams / Decision Networks),
- möglichst viele Inputformate sollten unterstützt werden: Datenbanken, Textdateien, CSV-Dateien, andere BN-Tool-Formate, ebenso sollten entsprechende Outputformate generierbar sein,
- eine einfach zu bedienende, übersichtliche und funktionale Benutzeroberfläche,
- das Programm sollte stabil laufen und bugfrei sein,
- guter Support vonseiten des Herstellers,
- eine gute Dokumentation,
- das Tool sollte auf .NET als Managed Code laufen und eine programmierbare API mitbringen und
- idealerweise frei verfügbar (kostenlos) bzw. bezahlbar sein.

Es wurden 38 BN-Tools analysiert, wobei davon letztlich die folgenden fünf vielversprechendsten Programme im Detail anhand von Fallbeispielen getestet und bewertet wurden.

WinMine	Belief Network PowerConstructor	Bayesware Discoverer 1.0
- kein Inferenzalgorithmus	- kein Inferenzalgorithmus	+ Inferenzalgorithmus
o Lernalgorithmus, allerdings mit schlechten Ergebnissen	+ Lernverfahren	+ Lernalgorithmus
+ Prognosegüte messbar	- Prognose nicht möglich	+ Prognosegüte messbar
o Diskretisierung, aber nur automatisch	+ Diskretisierungswizard	+ Diskretisierungswizard
- Domänenwissen lässt sich nicht vorgeben	+ Domänenwissen kann vorgeben werden	- Domänenwissen lässt sich nicht vorgeben
- inkompatibel zu anderen BN-Tools	o kann Netica- und Hugin-Dateien exportieren, kein Import	- inkompatibel zu anderen BN-Tools
- keine Influence Diagrams	- keine Influence Diagrams	- keine Influence Diagrams
- Likelihoods können nicht gesetzt werden - schlechte Dokumentation	- Likelihoods können nicht gesetzt werden o Dokumentation sehr knapp	- Likelihoods können nicht gesetzt werden o Dokumentation sehr knapp
- schlechte GUI	- kein GUI	- schlechte GUI
+ kostenlos	+ kostenlos	- kommerziell
- API nicht verfügbar	o API verfügbar	- API nicht verfügbar

Netica 2.17	Hugin 6.1
+ Inferenzalgorithmus	+ Inferenzalgorithmus
o Lernverfahren, kann allerdings keine Strukturen lernen, nur Parameter	o Lernalgorithmus, allerdings nicht auf dem aktuellen Stand der Forschung
+ Prognosegüte messbar	- Prognosegüte nicht messbar

- kein Diskretisierungswizard	+ Diskretisierungswizard
- Domänenwissen lässt sich nicht vorgeben	+ Domänenwissen kann vorgeben werden
o Import von Hugin- und anderen Dateiformaten möglich, der Export nicht	- inkompatibel zu anderen BN-Tools
- keine Influence Diagrams	+ Influence Diagrams mit Utilityknoten und Entscheidungsknoten
+ Likelihoods können gesetzt werden	+ Likelihoods können gesetzt werden, Adaption der Netze
- keine Dokumentation bzw. nicht bewertbar	+ gute Dokumentation
+ gute GUI	+ gute GUI
- kommerziell, Evaluierungsversion stark beschränkt	- kommerziell & teuer (1666,- € Forschungsversion bzw. 6300,- € kommerzielle Version), guter Support, Testversion beschränkt
+ API verfügbar	+ API verfügbar

Die Analyse der Tools hat ergeben, dass keines der verfügbaren Programme alle unsere Anforderungen erfüllen kann. Insbesondere lief kein Tool auf .NET. Von allen getesteten Programmen war Hugin 6.1 mit Abstand das Tool mit dem größten Funktionsumfang. Leider erfüllt Hugin nicht alle Anforderungen. Insbesondere der hohe Preis sprach gegen den Einsatz in unserem Projekt. Deshalb und auf Grund des zu erwartenden Erfahrungsgewinns haben wir uns entschieden, unser eigenes Bayes'sches Netz-Tool zu implementieren.

D SimMarket-Datenmodell

TRANSAKTIONEN

Die Tabelle enthält alle Transaktionen der Kassen.

Name	Datentyp	Größe	Null	Beschreibung
Zeit	Date	8		Datum + Uhrzeit
BonNr	Number	18		eindeutige Bonnummer über alle Bons
ArtikelNr	Number	18		Die interne Artikelnummer im System, Fremdschlüssel für Artikeltabelle
WGR	Varchar	8		Warengruppenkennzeichen
Menge	Number	12,4		verkaufte Menge des Artikels
VK_Brutto	Number	12,4		Verkaufspreis
KundenID	Number	10	J	Kundennummer, Fremdschlüssel für Kundenstammtabelle
Rabatt	Number	1	J	Rabattkenner
RabattWert	Number	5,2		Rabattwert in Prozent für diese Position; Default: 0
Preisrang	Number	5,4		Preisrang

Tabelle 16: SimMarket-Datenmodell – Transaktionen

Die Transaktionen enthalten nur die Verkaufspositionen eines Bons. Es werden keine Positionen mit Summen, Zahlungen etc. gespeichert. Die interne Artikelnummer ist die Nummer, unter der ein Artikel im Kundensystem geführt wird. Sie kann über die ArtikelNr-Referenz in eine externe Artikelnummer (EAN, PLU …) übersetzt werden. Der Preisrang gibt das Preisniveau eines Artikels im Vergleich zu anderen Artikeln der gleichen Miniwarengruppe an.

ARTIKEL
Bewegliche Artikeldaten

Name	Datentyp	Größe	Null	Beschreibung
ArtikelNr	Number	18		Fremdschlüssel für Artikelstamm
AbDatum	Date	8		Intervall, für das genau diese Eigenschaften des Artikeldatensatzes gelten
BisDatum	Date	8		
AktionsNr	Number	6	J	Aktionsnummer, Fremdschlüssel für Aktionstabelle
Platzierung	Number	8	J	Platzierungsinformation, Fremdschlüssel für Platzierungstabelle
VK_Brutto	Number	12,4		Verkaufspreis
EK_Brutto	Number	12,4	J	Einkaufspreis
Dispo	Number	1		Kennzeichen, ob der Artikel im Moment verfügbar ist. 0: nicht verfügbar 1: verfügbar
gelistet	Number	1		Kennzeichen, ob Artikel gelistet ist 0: nicht gelistet 1. gelistet

Tabelle 17: SimMarket-Datenmodell – beweglicher Artikelstamm

Eindeutiger Schlüssel: ArtikelNr + AbDatum + BisDatum

Die Artikeldaten sind beweglich und mit einem entsprechenden Zeitintervall verknüpft. Wenn sich für den Artikel mindestens eine der Eigenschaften AktionsNr, VK_Brutto, EK_Brutto, Platzierung, Dispo oder gelistet ändert, dann wird ein neues Intervall erzeugt. Ein neues Intervall erhält als BisDatum „unendlich“. Intervalle dürfen sich in keinem Falle überschneiden sich!
Beispiel: Bei der ersten Einlistung eines Artikels wird das erste Intervall erzeugt. Das AbDatum enthält das Datum der Einlistung, in dem Feld BisDatum wird „unendlich“ eingetragen.
Wird zu einem Tag X z. B. der VK_Brutto geändert, dann erhält das neue Intervall Tag X als AbDatum und als BisDatum „unendlich“. Alle anderen Inhalte (bis auf den geänderten VK_Brutto) entsprechen denen des 1. Intervalls. Beim 1. Intervall wird gleichzeitig das BisDatum von „unendlich“ auf Tag X-1 geändert.

Zu beachten ist, dass eine Aktion zwei neue Intervalle erzeugt. Das erste Intervall wird erzeugt, wenn die Aktion beginnt, da sich die AktionNr von NULL auf „Aktionsnummer“ ändert. Das zweite Intervall wird erzeugt, wenn die Aktion endet, da die AktionsNr wieder auf NULL gesetzt wird.

ARTIKELSTAMM

Name	Datentyp	Größe	Null	Beschreibung
ArtikelNr	Number	18		eindeutige interne Artikelnummer
BasisWGR	Varchar2	8		Basiswarengruppe (unterste Ebene der Warengruppenhierarchie), Fremdschlüssel für Warengruppentabelle (Größe wurde von 4 auf 8 gesetzt, damit Globus-Nummern direkt übernommen werden können)
Bezeichnung	Varchar2	40		Bezeichnung des Artikels
Gruppe	Number	11	J	Fremdschlüssel für Gruppentabellen
Artikeltyp	Number	6	J	Fremdschlüssel für Artikeltypentabelle

Tabelle 18: SimMarket-Datenmodell – fester Artikelstamm

Eindeutiger Schlüssel: ArtikelNr

Der Artikelstamm enthält die Artikeldaten, die sich nicht ändern, bzw. Daten, die im Fall einer Änderung überschrieben werden.

GRUPPEN

Stammdatentabelle für Gruppen (Gruppierung von Artikel)

Name	Datentyp	Größe	Null	Beschreibung
Gruppe	Number	11		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Bezeichnung/Name der Gruppe

Tabelle 19: SimMarket-Datenmodell – Gruppen

Eindeutiger Schlüssel: Gruppe

Die Tabelle Gruppen ist für eine zusätzliche Gruppierung des Artikels vorgesehen, wie z. B. Verkaufspreisgruppen.

ARTIKELTYPEN
Stammdatentabelle für Artikeltypen

Name	Datentyp	Größe	Null	Beschreibung
Artikeltyp	Number	6		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Bezeichnung/Name des Artikeltyps

Tabelle 20: SimMarket-Datenmodell – Artikeltypen

Eindeutiger Schlüssel: Artikeltyp
ArtikelTyp kann zur zusätzlichen Klassifizierung des Artikels genutzt werden. Bei Bedarf können mehrere disjunkte Artikeltypen vorliegen.

PLATZIERUNGEN
Stammdatentabelle für Platzierungsdaten

Name	Datentyp	Größe	Null	Beschreibung
Platzierung	Number	8		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Bezeichnung/Name der Platzierung

Tabelle 21: SimMarket-Datenmodell – Platzierungen

Eindeutiger Schlüssel: Platzierung

Enthält z. B. Regalnummer/Regalbezeichnung.
ARTIKELNR_REFERENZ

Name	Datentyp	Größe	Null	Beschreibung
ArtikelNr	Number	18		Artikelnummer
EAN_PLU	Number	14		EAN oder PLU

Tabelle 22: SimMarket-Datenmodell – ArtikelNr-Referenz

Zu einer Artikelnummer kann es mehrere EAN/PLU geben.

WARENGRUPPEN

Name	Datentyp	Größe	Null	Beschreibung

WgrNr	Varchar2	8		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Bezeichnung/Name der Warengruppe
WgrNrVater	Varchar2	8	J	übergeordnete Warengruppe (Zeiger auf WgrNr); für Warengruppen, die an der Spitze der Hierarchie stehen, ist dieses Feld leer

Tabelle 23: SimMarket-Datenmodell – Warengruppen

Eindeutiger Schlüssel: WgrNr

Zu jeder Warengruppe wird nur die unmittelbar darüber liegende Vaterwarengruppe gespeichert. So ist eine beliebige Hierarchie abbildbar. Im Artikelstamm wird nur die Warengruppe der untersten Ebene in der Hierarchie gespeichert (Basiswarengruppe). Von der Warengruppentabelle wird kein historischer Verlauf gespeichert. Bei den Warengruppen der obersten Hierarchiestufe ist das Feld WgrNrVater leer.

AKTIONEN
Tabelle enthält alle Aktionen

Name	**Datentyp**	**Größe**	**Null**	**Beschreibung**
AktionsNr	Number	6		eindeutige Aktionsnummer
Aktionstyp	Number	6		Klassifizierung der Aktion, Fremdschlüssel für AktionsTypen
Bezeichnung	Varchar2	40		Bezeichnung/Name der Aktion
AbDatum	Date	8		Datum, ab der die Aktion gültig ist
BisDatum	Date	8		Datum, bis zu der die Aktion gültig ist
Erscheindatum	Date	8	J	Erscheinungsdatum

Tabelle 24: SimMarket-Datenmodell – Aktionen

Eindeutiger Schlüssel: AktionsNr

AKTIONSTYPEN
Stammdaten zu Aktionen

Name	**Datentyp**	**Größe**	**Null**	**Beschreibung**
AktionsTyp	Number	6		eindeutige Nummer für Aktionstyp
Bezeichnung	Varchar2	40		Bezeichnung/Name des Aktionstyps

Tabelle 25: SimMarket-Datenmodell – Aktionstypen

Eindeutiger Schlüssel: AktionsTyp

Kundenstamm

Name	Datentyp	Größe	Null	Beschreibung
KundenID	Number	10		eindeutige KundenID
Anrede	Number	1	J	1: Frau 2: Herr 3: Eheleute 4: Firma 5: Verein
Geburtstag	Date	8	J	
PLZ	Varchar2	8	J	Postleitzahl
Ort	Varchar2	50	J	Ortsbezeichnung
Land	Varchar2	3	J	Länderkennzeichen

Tabelle 26: SimMarket-Datenmodell – Kundenstamm

Eindeutiger Schlüssel: KundenID

KUNDENKARTEN

KundenID + beliebige Anzahl von EC-/Kreditkartennummern.

Name	Datentyp	Größe	Null	Beschreibung
KundenID	Number	10		KundenID
KartenNr	Number	20		Kundenkartennummer, EC-Kartennummer oder Kreditkartennummer
Kartentyp	Number	1		Kennzeichen für Kartentyp 1: Kundenkartennummer, 2: EC-Kartennummer 3: Kreditkartennummer

Tabelle 27: SimMarket-Datenmodell – Kundenkarten

Die Kundenkartentabelle ist eine Detailtabelle zu der Kundentabelle. Für jeden Kunden aus dem Kundenstamm können mehrere Karten hinterlegt werden.

LIEFERANTEN
Lieferanten bzw. Herstellerstammdaten

Name	Datentyp	Größe	Null	Beschreibung
LieferantenNr	Number	8		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Name des Lieferanten
ILN	Number	16	J	Hersteller – International Location Number

Tabelle 28: SimMarket-Datenmodell – Lieferanten

Eindeutiger Schlüssel: LieferantenNr

ARTIKEL_LIEFERANTEN
Kreuztabelle Artikellieferanten

Name	Datentyp	Größe	Null	Beschreibung
ArtikelNr	Number	18		Fremdschlüssel für Artikelstammtabelle
LieferantenNr	Number	8		Fremdschlüssel für Lieferantentabelle

Tabelle 29: SimMarket-Datenmodell – Kreuztabelle Artikellieferanten

KONKURRENZ
Wettbewerberstammdaten

Name	Datentyp	Größe	Null	Beschreibung
KonkurrenzNr	Number	8		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Name des Wettbewerbers
Entfernung	Number	2	J	Entfernung zum Gemeindemittelpunkt

Tabelle 30: SimMarket-Datenmodell – Wettbewerber

Eindeutiger Schlüssel: KonkurrenzNr

KONKURRENZ_ARTIKEL
Kreuztabelle Artikelwettbewerber

Name	Datentyp	Größe	Null	Beschreibung
ArtikelNr	Number	18		Fremdschlüssel für Artikelstammtabelle
KonkurrenzNr	Number	8		Fremdschlüssel für Konkurrenztabelle

VK_Brutto	Number	12,4		Verkaufspreise des Artikels beim Mitbewerber
ErhebDatum	Date	8		Datum der Erhebung
Aktion	Number	1	J	Flag, ob der Artikel bei dem Mitbewerber zum Zeitpunkt der Erhebung in einer Promotion war.

Tabelle 31: SimMarket-Datenmodell – Kreuztabelle Artikelwettbewerber

GRUPPENMITGLIEDER

Zuordnung von Kunden zu einer Gruppe

Name	Datentyp	Größe	Null	Beschreibung
GruppenID	Number	4		Fremdschlüssel auf Kundengruppen
KundenID	Number	10		Fremdschlüssel auf Kundenstamm

Tabelle 32: SimMarket-Datenmodell – Gruppenmitglieder

Tabelle wird nicht importiert, sondern von SimMarket erzeugt.

KUNDENGRUPPEN

Stammdatentabelle der Gruppenmitglieder

Name	Datentyp	Größe	Null	Beschreibung
GruppenID	Number	4		eindeutiger Schlüssel
Bezeichnung	Varchar2	40		Name/Bezeichnung der Kundengruppe

Tabelle 33: SimMarket-Datenmodell – Kundengruppen

Eindeutiger Schlüssel: GruppenID

Data-Mining-Tabelle

Inputtabelle für die Verhaltensnetze der Agenten

Name	Datentyp	Größe	Null	Beschreibung
Datum	DateTime	8		Startdatum des beobachteten Intervalls
InVariable 1..N	Double	12,4		Messbare Einflussfaktoren, die das Einkaufsverhalten beeinflussen. Jede Variable bekommt eine eigene Spalte.

OutVariable 1..M	Double	12,4		Gemessene Zielvariablen, die prognostiziert werden sollen. Jede Variable bekommt eine eigene Spalte.

Tabelle 34: SimMarket-Datenmodell – Data-Mining-Tabelle

Eindeutiger Schlüssel: Datum

Diese Tabelle muss nicht gespeichert werden, sondern kann auch on demand von der Datenbank erzeugt werden. Eine Speicherung ist meist nicht sinnvoll, da das abgerufene Zeitintervall stark differieren kann.

Data-Mining-Tabellen könne für unterschiedliche Zeitbasen erstellt werden. Jede Zeile dieser Tabelle entspricht einem Lernfall und kann sowohl einen Tag als auch einen größeren Zeitraum wie z. B. eine Woche repräsentieren. Das Datum bezeichnet immer den Beginn des Zeitraumes eines Lernfalles. Das oben angegebene Format entspricht dem Format, das von den Verhaltensnetzen erwartet wird und nicht unbedingt dem Gespeicherten in der Datenbank.

Allgemeine Anmerkungen:
Datum „unendlich“ : 31.12.9999

E Anwendungsfälle der SimMarket-Simulation-Engine

Kategorien von Anwendungsfällen

- Preisänderung / -optimierung (Simulation)
- Substitutionsanalyse
- Promotionsanalyse /-optimierung (Simulation)

Die benötigten Technologien sind:

- ⇨ manuelle Simulation
- ⇨ automatische Simulation
- ⇨ Substitutionsanalyse mit Korrelationsanalyse
- ⇨ performantes Data Warehouse
- ⇨ flexible GUI

für zukünftige Versionen:

- ⇨ Klassifikation
- ⇨ Clustering
- ⇨ Assoziationsanalyse

Anwenderrollen:

- ⇨ Sortimentsverantwortlicher (Ein- / Auslistungen, Bepreisung, Category Management, Regalmanagement)
- ⇨ Promotionsplaner (Promotionen, wie z. B. Faltblattwerbung, Radiowerbung etc.)
- ⇨ Kundenmanager (CRM – Mailings, kundengruppenspezifische Maßnahmen)
- ⇨ Filialleiter (Performance Measurement)
- ⇨ Einkäufer (Demand Management)

Konkrete Fragestellungen

Die **fett** markierten Anwendungsfälle wurden im Rahmen dieser Arbeit in den Forschungsprototypen integriert.

Analyse (der Vergangenheit und des Ist-Zustandes)

Fokus-Dimension Artikel / WGR:

Artikelanalyse:

- **Basiskennzahlen: absolute und relative (Dimensionen Zeit, WGR, Artikel, Filiale)**
- Was hat eine Kassenschüttenplatzierung gebracht (Menge/Ertrag)? Flacht der Abverkauf danach wieder ab?
- Verbundanalyse – Was wird mit meiner Marke/Artikel zusammengekauft?

Kundeninformationen /-analyse:

- Wie hat sich die Kundentypologie verändert (eines Artikels / der WGR)?
- Was haben die Kunden einer Neulistung vorher gekauft?
- Wie viele Erstkäufer/Wiederkäufer hat meine Neulistung/neue Marke im Zeitverlauf (Wochen)? Und im Vergleich zu einer bereits eingeführten Marke/Artikel (Benchmark)?
- Wie viele Erstkäufer einer Neulistung wechseln nach dem ersten Kauf wieder zur alten Marke / Artikel?
- Was haben die Wiederkäufer meiner Neulistung/neue Marke beim Wiederkauf noch gekauft? => spezielle Warenkorbanalyse
- Wie hoch ist der Durchschnittsbon der Wiederkäufer einer Neulistung? Wie hoch war der D-Bon bei diesen Käufern?
- Wie repräsentativ sind die Kundenboninformationen (pro Sortiment)?
- Exklusivanalyse – Wie viele Kunden kaufen nur die Marke x aus einer WGR? Wie viele Kunden erreiche ich mit einer Marke? Wie viele kaufen was anderes?
- Loyalitätsanalyse – Wie viel % der Kunden sind zu einer Marke loyal?
- Ausgaben pro Jahr, z. B. Wie viel gibt ein Shampoo-Kunde aus? (im Vergleich zu Haarkur-Kunden?)
- Wie viele Neukäufer haben wir in der WGR durch Neulistung gewonnen? (Anzahl der Käufer, die vorher noch nie in der WGR gekauft haben)

Fokus-Dimension Maßnahmen:

- Coupon/Gewinnspiel – Wie hoch war die Rücklaufquote? Käuferprofil der Rückläufer?
- Mailings – Wie viele Erstkäufer? Was haben diese vorher gekauft? Wie viele Einmalkäufer?
- Gibt es Bevorratungseffekte (z. B. bei 5fach-Bepunktung o. Ä.)?

- Wie viele Kunden haben nach dem Mailing eine Filiale besucht?
- Konnte der Marktanteil der Marke/Artikel in der WGR durch ein Mailing erhöht werden?
- Werbeblattoptimierung/-simulation

Fokus-Dimension Kunden / Kundengruppe:

- Wie hat sich die Kundentypologie verändert (der Filiale)?
- Kennzeichnung eines Kunden, dessen Warenkorb einen großen Anteil an Eigenmarken / Bio aufweist? Eigenschaften, Verhaltensweisen, Attribute.
- Welche kundengruppenspezifischen Marketingmaßnahmen erscheinen sinnvoll & Erfolg versprechend?
- Wie definiert sich der typische
    - Smartshopper
    - Markenkäufer
    - „A-Artikel"-Käufer
    - Vielkäufer, hinsichtlich Warengruppen, Aktionsverhalten, Flugblättern, demografischen Kriterien, Einkaufshäufigkeit, Einkaufsstättentreue?
- Welche speziellen Kundengruppen kaufen in ländlichen Märkten?
- Welche Unterscheidungen finden sich generell von der Kundengruppe Stadt- zur Landbevölkerung?
- Welche nach der Zielgruppentopologie „Sinus-Milieu" definierten Kundentypen finden sich vorwiegend in Filiale X?

Simulation

Bemerkung:
Die Komplexität der Szenarien bestimmt, ob eine Fragestellung mit einer *manuellen* oder einer vergleichbar einfachen verdeckten automatischen Simulation möglich ist oder, ob eine komplexe *automatische Simulation* nötig ist. Der entscheidende Faktor ist die Zeit. Soll eine Aussage über einen längeren Zeitraum mit mehreren Zustandsänderungen gefällt werden, ist eine automatische Simulation nötig.

Prognostische Simulation

Fokus-Dimension Artikel / WGR:

Artikelsimulation:

- **Wie verändern sich die Abverkaufsmenge und der Ertrag des Produkts (Menge/Ertrag) bei Preisänderungen?**

  - **Gibt es eine signifikante Abverkaufssteigerung für Artikel x bei einer Preissenkung von y Euro? (Effizienz der Maßnahme „Preisänderung“)**
  - **Wie verhält sich die Preiselastizität bei Artikel X bei Preissenkung um Y Euro?**
- **Wie verändern sich die Abverkaufsmenge und der Ertrag des Produkts (Menge/Ertrag) bei Promotionen?**
  - Präevaluierung: Welchen maximalen Abverkauf könnten wir mit unserer „2+1-Aktion“ mit „Mars“, „Kitkat“ oder „Bounty“ haben?
  - Welchen Erfolg hat eine saisonale Aktion mit „Ferrero Überraschungseiern“ bei gleichzeitiger Aktion des Mitbewerbers?
  - Welche Nachkaufsrate lässt sich für Artikel X, Artikel Y oder Artikel Z nach einer Couponaktion in Höhe von 0,50 Euro prognostizieren?
- **Welche Artikel der Warengruppe X zeigen eine hohe saisonale Abhängigkeit?**
- Upgrading: Mit welchem Erfolg lässt sich bei einem Upgradingversuch im Rahmen einer Flugblattaktion von der Marke „Ballantine’s“ hin zur Marke „Johnnie Walker“ rechnen?
- Neulistung:
  - Welche Absatzentwicklungsszenarien ergeben sich, wenn die Produktneueinführung „Senf mit Johanniskraut“ im Vorfeld bzw. parallel zum Erscheinungstermin im Flugblatt promotet wird?

Artikelsimulation mit Substitutionsinformation:

- **Preisänderung / Promotion: Wie wird/hat sich der AV der umliegenden Produkte (Menge/Ertrag) bzw. der WGR verändern/verändert (Substitutionseffekte)?**
  - **Kannibalisiert bei einer Flugblattaktion Marke X die nicht in Aktion befindliche Marke Y?**
  - **Kann eine Steigerung des Absatzes des Komplementärproduktes „Holzkohle“ durch eine Verbundplatzierung von „Kugelgrills & Holzkohle“ bewirkt werden?**
- **Zweitplatzierung: Wie verändern sich die Abverkaufsmenge und der Ertrag des Produkts (Menge/Ertrag) bei einer Zweitplatzierung?**
  - **Wie wird sich eine Kassenschüttenplatzierung (Zweitplatzierung) auswirken (Menge/Ertrag)? Flacht der Abverkauf danach wieder ab?**
  - **Welche Auswirkungen hat eine Aktion mit einem Doppelpack/Überfüllgröße (Zweitplatzierung) auf den Normalartikel?**
- **Welche Artikel sind <u>wie</u> von einander abhängig?**
- **Auslistungen:**

  - **Weichen die Kunden bei einer Auslistung auf andere Produkte aus (anhand von Abverkaufszahlen, nicht Kundeninfo)?**
  - **Welche Artikel werden den auszulistenden Artikel X wahrscheinlich substituieren?**
  - Welche Auswirkungen gibt es durch eine Auslistung durch Verbundkäufe auf andere WGR (Menge/Ertrag)?
- Neulistungen:
  - Wächst die WGR durch die Einführung der Neulistung oder werden ausschließlich Wettbewerber verdrängt?
  - Zeigt eine Zusatzlistung der Marke X um die Sorte XY (z. B. Kümmerling + Kümmerling Orange; Mars traditionell + Mars Mandel) eine Abverkaufssteigerung oder kommt es zu einer Kannibalisierung der Sorten bei Marke X?
  - Inwiefern kannibalisiert der neu eingelistete Artikel X einen anderen Artikel aus der entsprechenden Warengruppe?

Kundenanalyse:

- Preisänderung:
  - Wie wird sich die Kundentypologie ändern?
  - Welche Artikel zeigen eine ähnliche Käuferschicht auf?
- Neulistung:
  - Welche Marken/Artikel verlieren nach der Einführung die meisten Käufer?
  - Wie wird sich die Kundentypologie ändern?
- Auslistung:
  - Weichen die Kunden bei einer Auslistung auf andere Produkte aus? Welche Auswirkungen gibt es durch eine Auslistung durch Verbundkäufe auf andere WGR (Menge/Ertrag)?
  - Wie wird sich die Kundentypologie ändern?

Fokus-Dimension Maßnahmen:

- Welchen Einfluss könnten externe Faktoren (insbesondere Wettbewerb, Wetter) auf unsere „xy-Promotion“ hinsichtlich Absatzverhalten haben?
- Wäre eine Faltblattaktion mit oder ohne Kombination von Preisänderungen einiger Artikel sinnvoll?

Fokus-Dimension Kunden / Kundengruppen:

- Kundensimulation => im Prinzip gleiche Fragestellungen wie bei der Artikelsimulation

- Wie verhält sich der Prototyp des „Biokunden“ bei einer Preiserhöhung der Bio-Artikel bei Obst und Gemüse?
- Inwieweit ließe sich die Rücklaufquote einer geschlechtsspezifischen zu unspezifischen Mailingaktion für singlehaushaltrelevante Artikel prognostizieren?
- Welche Rücklaufquote lässt sich bei einer zielgruppenspezifischen Mailing-Coupon-Aktion im Segment „Baby“ erwarten?

Optimierende Simulation:
Der Unterschied zur prognostizierenden Simulation besteht darin, dass bei der optimierenden keine konkreten Szenarien vorgegeben werden, sondern nur die gewünschten Ziele einer Maßnahme. Die *Simulation Engine* muss aus den definierten Zielen sinnvolle Szenarien generieren, diese simulieren und anhand der zu optimierenden Kennzahlen sortiert auflisten (Ranking).

Fokus-Dimension Artikel / WGR:

- Wie wäre der optimale Standardpreis für Artikel X, um dessen Basisabverkaufsmenge zu steigern?

Fokus-Dimension Kunden / Kundengruppen:

- Wo liegt die maximale Toleranzschwelle für „Bio-Vielkäufer“, einen Aufpreis für Bio contra konventionell auszugeben?
- Bis zu welchem maximalen Betrag sind Weinkäufer bereit, Premiumqualität zu bezahlen?

F Literaturverzeichnis

[AbeMam97] Abe, N., Mamitsuka, H., *Predicting Protein Secondary Structure Using Stochastic Tree Grammars*, Machine Learning, 29(2/3): 275--301, 1997.

[Agosta04] Agosta, L., Data Warehousing lessons learned: Data Mining is dead – long live Predictive Analytics, in DMReview, Januar, 2004.

[Almond95] Almond, R. G., *Graphical Belief Modeling,* Chapman&Hall, London, 1995.

[ArnCorPro87] Arnborg, S., Corncil, D. G., Proskurowski, A., *Complexity of finding embeddings in a k-tree*, Journal of SIAM, Algebraic Discrete Methods 8, Seiten 177-184, 1987.

[BacHocMal93] Bachem, A., Hochstättler, W., Malich, M., *The Simulated Trading Heuristic for Solving Vehicle Routing Problems*. Technical Reports 93.139, Mathematisches Institut der Universität Köln, 1993.

[BecGei96] Becker, A., Geiger, D., *A sufficiently fast algorithm for finding close to optimal junction trees*, In Proceedings of the Twelfth Conference of Uncertainty in Artificial Intelligence, Portland, Morgan Kaufman, 1996.

[Bohn04] Bohnenberger, T., *Decision-Theoretic Planning for User-Adaptive Systems: Dealing With Multiple Goals and Resource Limitations*, Dissertation, Universität des Saarlandes, 2004.

[Bolstorff03] Bolstorff, *Supply Chain Excellence – A Handbook for Dramatic Improvement Using the SCOR Model*. American Management Association, 2003.

[BonGas98] Bond, A. H., Gasser, L., *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann Publishers Inc., 1998.

[BrUlAnPr00] Brannon, E. L., Ulrich, P. V., Anderson, L. J., Presley, A. B., *Agent-Based Simulation of the Consumer's Apparel Purchase Decision*, National Textile Center Annual Report I98-A09, Auburn University, November 2000.

[Brooks86] Brooks, R. A., *A robust layered control system for a mobile robot. In IEEE Journal of Robotics and Automation*, Ausgabe RA-2 (1), Seiten 14-23, 1986.

[BucMaz01] Buchta, C., Mazanec, J., *SIMSEG/ACM – A Simulation Environment for Artificial Consumer Markets.* Working Paper Nr. 79 März 2001.

[BürFisVie98] Bürkert, H.-J., Fischer, K., Vierke, G., *Transportation Scheduling with Holonic MAS – The TeleTruck Approach*, In PAAM, 1998.

[CACM05] Günther, O., Spiekermann, S., *RFID and Perceived Control – The Consumer's View*, Humbolt-Universität zu Berlin, forthcoming in CACM, ACM, September 2005.

[ChiGeiHec94] Chickering, D. M., Geiger, D., Heckerman, D., *Learning Bayesian networks is NP-hard*, Technical Report MSR-TR-94-l7, Microsoft Research, Microsoft Corporation, 1994.

[CHHKMPS02] Cheng, J., Hatzis, C., Hayashi, H., Krogel, M., Morishita, S., Page, D. and Sese, J., *KDD Cup 2001 Report. ACM SIGKDD Explorations* Volume3, Issue 2, January 2002.

[Chickering96] Chickering, D., *Learning Bayesian networks is NP-Complete,* In Learning from Data, 121-130, Springer-Verlag, 1996.

[CooHolSu01] Cook, D. J., Holder, L. B., Su, S., Maglothin, R., Jonyer, I., *Structural Mining of Molecular Biology Data,* IEEE Engineering in Medicine and Biology, special issue on Advances in Genomics, 2001.

[CooHer91] Cooper, G. F., Herskovits, E., *A Bayesian method for constructing Bayesian belief networks from databases*, In UAI91 – Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence, Seiten 86-94, 1991.

[Cooper84] Cooper, G. F., NESTOR: *A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge*, Ph.D. Thesis, Department of Computer Science, Stanford University, 1984.

[CowDawSpi99] Cowell, R. G., Dawid, A. P., Spiegelhalter, D. J., *Probabilistic Networks and Expert Systems*, Springer Verlag, 1999.

[CriSha00] Christianini, N., Shawe-Taylor, J., *An Introduction to Support Vector Maschines and other kernel-based learning methods*, Cambridge University Press, 2000.

[CrispDM00] Chapman (NCR), P., Clintion (SPSS), J., Kerber (NCR), R., Khabaza (SPSS), T., Reinartz (DaimlerChrysler), T., Shearer (SPSS), C., Wirth (DaimlerChrysler), R., CRISP-DM 1.0 – Step-by-step data mining quide (CRoss-Industry Standard Process for Data Mining), 2000.

[Csi75] Csiszár, I., *I-divergence geometry of probability distributions and minimization problems*, Ann. Prob., 3(1), Seiten 146-158, 1975.

[Dempster66] Dempster, A. P., *New Methods for Reasoning Towards Posterior Distributions Based on Sample Data*, Annals of Mathematical Statistics 37, Seiten 355-374, 1966.

[DemLaiRub77] Dempster, A., Laird, N., Rubin, D., *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royals Statistical Society, B 39, Seiten 1- 38, 1977.

[FisSchSie03] Fischer, K., Schillo, M., and Siekmann, J., *Holonic Multiagent Systems: The Foundation for the Organization of Multiagent Systems.* Proceedings of the First International Conference on Applications of Holonic and Multiagent Systems (HoloMAS'03), 2003.

[Flügel03] Flügel, G., Data-Mining im Gesundheitswesen: *Ein Praxisleitfaden für Mediziner, Pharmazeuten und Krankenkassen*, Businessvillage, Göttingen, 2003.

[Friedman97] Friedman, N., *Learning Belief Networks in the Presence of Missing Values and Hidden Variables*, In Proceedings of 14th International Conference on Machine Learning, Seiten 125-133, 1997.

[Friedman98] Friedman, N., *The Bayesian Structural EM Algorithm*, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998.

[FrGeKoPf99] Friedman, N., Getoor, L., Koller, D., Pfeffer, A., *Learning Probabilistic Relational Models*, Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI), Stockholm, Schweden, 1999.

[FrieGol02] Friedman, N., and Goldszmidt, M., *Learning Bayesian Networks with Local Structure*, In: Learning in Graphical Models, Ed. Jordan, M. I., January 16, 2002.

[FriLinNac00] Friedman, N., Linial, M., Nachman, I., *Using Bayesian Networks to Analyse Expression Data*, Journal of Computational Biology, 2000.

[GemGem84] Geman, S., Geman, D., *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*, IEEE Transactions on Pattern Analysis and Maschine Intelligence 6, Seiten 721-742, 1984.

[Gerber04] Gerber, A., *Flexible Kooperation zwischen autonomen Agenten in dynamischen Umgebungen*, Dissertation, Universität des Saarlandes, Lehrstuhl für Künstliche Intelligenz Prof. Siekmann, 2004.

[GerSieVie99] Gerber, C., Siekmann, J., Vierke, G., *Holonic Multi-Agent Systems*. Research Report RR-99-03, DFKI, 1999.

[Golumbic80] Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[Hab74] Haberman, S., *The Analysis of Frequency Data*, The University of Chicago Press, 1974.

[HanKam01] Han, J., Kamber, M., *Data Mining – Concepts and Techniques*, Morgan Kaufmann Publishers, Academic Press, 2001.

[HecGeiChi95] Heckerman, D., Geiger, D., Chickering, D. M., *Learning Bayesian networks: The combination of knowledge and statistical data*, Machine Learning, 20(3):197-243, 1995.

[Heckerman96] Heckerman, D., *A Tutorial on Learning with Bayesian Networks*, Technical Report MSR-TR-95-06, Microsoft Research, Redmond, 1996.

[Hertel99] Hertel, J., *Warenwirtschaftssysteme – Grundlagen und Konzepte*, 3. überarbeitete und erweiterte Auflage, Physica-Verlag, 1999.

[Hindle01] Hindle, T., *Die 100 wichtigsten Management-Konzepte*, Econ, http://www.ephorie.de/hindle_top_100.htm , 2001.

[Inmon96] Inmon, B., *Building the Data Warehouse*, John Wiley & Sons, 2. Auflage, 1996.

[Inmon99] Inmon, B., *Building the operational Data Store*, John Wiley & Sohn, 1999.

[Jager00] Jager, W., *Modelling Consumer Behaviour*. Rijksuniversiteit Groningen, Dissertation Juni 2000.

[Jameson03] Jameson, A., Adaptive Interfaces and Agents, In J. A. Jacko & A. Sears (Eds.), *Human-Computer Interaction Handbook*, Seiten 305-330, Mahwah, NJ: Erlbaum, 2003.

[JenJen94] Jensen, F. V., Jensen, F., *Optimal Junction Trees*, In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, Washington, 1994.

[Jensen01] Jensen, F. V., *Bayesian Networks and Decision Graphs*. Springer Verlag 2001.

[JenWoo98] Jennings, N. R., Wooldridge, M. J., *Agent Technology – Foundations, Applications and Markets*, Springer-Verlag, 1997.

[Jir95] Jiroušek, R., *Solution of the marginal problem and decomposable distributions*, Kybernetika, 27(5), Seiten 403-412, 1991.

[JungFisch01] Jung, C., Fischer, K., *Theory and Practice of Hybrid Agents*, Research

Report RR-01-01, DFKI GmbH, 2001.

[Kemper03] Kemper, H.-G., Vortrag: *Innovative Ansätze zur Unterstützung der betriebswirtschaftlichen Entscheidungsfindung*, 2002.

[KepHanGre00] Kephart, J., Hanson, J., Greenwald, A., *Dynamic pricing by software agents*, Computer Networks, Vol. 32, No. 6, Seiten 731-752, 2000.

[KinGeo97] Kinny, D. N., Georgeff, M. P., *Modeling and design of multi-agent systems*, In J. P. Müller, M. J. Wooldridge und N. R. Jennings, Herausgeber, Intelligent Agents III, Ausgabe 1193 of LNAI, Seiten 1-20, Springer-Verlag, Heidelberg, 1997.

[Kirchner02] Kirchner, K., *Anwendung von Spatial Data Mining Algorithmen bei Energieversorgungsunternehmen*, Workshop Arbeitskreis Knowledge Discovery, 2002.

[Klügel01] Klügel, F., *Multiagentensimulation*, Addison-Wesley-Verlag, 2001.

[Kolmogorov33] Kolmogorov, A. N., *Grundbegriffe der Wahrscheinlichkeitsrechnung*, Springer Verlag, Berlin, 1933.

[KorNic04] Korb, K. B., Nicholson, A. N., *Bayesian Artificial Intelligence*, Chapman & Hall/CRC, 2004.

[Köster02] Köster, F., *KDD & Data Mining als Analyseinstrument im Kontext agentenbasierter Assistenzsysteme zur Bildung/Festigung virtueller Lerngruppen*, Workshop Arbeitskreis Knowledge Discovery, 2002.

[Kuk05] Kuklin, A., *Implementation of Inference Algorithms in Belief Networks*, Master Thesis, University of Applied Science HTW Saarbrücken, 2005.

[LamBac93] Lam, W., Bacchus, F., *Learning Bayesian belief networks: An approach based on the MDL principle*. Computational Intelligence 10, Seiten 269-293, 1993.

[LauSpi88] Lauritzen, S. L., Spiegelhalter, D. J., *Local computations with probabilities on graphical structures and their application to expert*

systems, Journal of the Royal Statistical Society B, 50(2):157-224. 1988.

[LawKel00] Law, A. M., Kelton, W. D., *Simulation, Modeling and Analysis*, McGraw Hill Higher Education Verlag, 2000.

[Lehner03] Lehner, W., *Datenbanktechnologien für Data-Warehouse-Systeme – Konzepte und Methoden*, dpunkt.Verlag, 2003.

[Mertens94] Mertens, P., *Prognoserechnung*, 5. Auflage, Physica-Verlag, 1994.

[MerRäss05] Mertens, P., Rässler, S., *Prognoserechnung*, 6. Auflage, Physica-Verlag, 2005.

[Müller96] Müller, J. P., *The Design of Intelligent Agents – A Layered Approach*, Ausgabe 1177 von LNAI, Springer-Verlag, Heidelberg, 1996.

[Neapolitan89] Neapolitan, R. E., *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. A Wiley-Interscience Publication, John Wiley & Sons, Inc. New York, 1989.

[OleLauJen92] Olesen, K. G., Lauritzen, S. L., Jensen, F. V., *aHugin: A system creating adaptive causal probabilistic networks*, In Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference, Seiten 223-229, Morgan Kaufmann, 1992.

[PanJen01] Panzarasa, P., Jennings, N. R., *The organisation of sociality: a manifesto for a new science of multi-agent systems*, Proc 10th European Workshop on Multi-Agent Systems (MAAMAW-01), Annecy, Frankreich, 2001.

[Pearl88] Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers Inc., San Mateo, California, 1988.

[PeoSha91] Peot, M. A., Shachter, R. D., *Fusion and propagation with multiple observations in belief networks*, Artificial Intelligence Vol. 48, No. 3, Seiten 299-318, 1991.

[PohYaoJas98] Poh, H.-L., Yao, J., Jasic, T., *Neural Networks for the Analysis and Forecasting of Advertising and Promotion Impact*, Intelligent Systems in Accounting, Finance and Management, Vol. 7, No. 4, 1998.

[RaoGeo91] Rao, A. S., Georgeff, M. P., *Modeling rational agents within a BDI-architecture*, Technical Report 14, Australian AI Institute, Carlton, Australia, 1991.

[Rissanen78] Rissanen, J., *Modeling by shortest data description*, Automatica 14, Seiten 465-471, 1978.

[RouSan99] Roure, J., Sangüesa, R., *Incremental methods for Bayesian network learning*, Tech.Report LSI-99-42-R, Software Department at the Technical University of Catalonia, 1999.

[RusSchSta04] Russ, C., Schwaiger, A., Stahmer, B. P., *Category-Management und Customer-Relationship-Management: SimMarket - Grundkonzeption und Anwendungserfahrungen*, in Performance Leadership im Handel, Zukunft im Handel Band XIX, Deutscher Fachverlag, 2004.

[RusNor03] Russel, S. J., Norvig, P., *Artificial Intelligence – A Modern Approach*, Prentice-Hall International, Inc., Englewood Cliffs, New Jersey, Second Edition, 2003.

[RusNor95] Russel, S. J., Norvig, P., *Artificial Intelligence – A Modern Approach*, Prentice-Hall International, Inc., Englewood Cliffs, New Jersey, 1995.

[SaiBouDro02] Said, L. B., Bouron, T., Drogoul, A., *Agent-based Interaction Analysis of Consumer Behavior*, In proceedings of AAMAS Bologna 2002.

[Sandholm93] Sandholm, T. W., *An Implementation of the Contract Net Protocol based on Marginal Cost Calculations*. In Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Seiten 295-308, 1993.

[Schäfer98] Schäfer, R., *Benutzermodellierung mit dynamischen Bayes'schen Netzen als Grundlage adaptiver Dialogsysteme*, Dissertation, Lehrstuhl Prof. Wahlster, Universität des Saarlandes, 1998.

[Schillo04] Schillo, M., *Multiagent Robustness: Autonomy vs. Organistion*, Dissertation, Universität des Saarlandes, 2004.

[Schlicker06] Schlicker, S., *SimAgent – Eine verteilte und mobile Multiagentenplattform für den Einsatz in einem Supermarkt-Simulations-System*, Diplomarbeit, Universität des Saarlandes, Lehrstuhl Prof. Dr. Jörg Siekmann, 2006.

[Schmidt05] Schmidt, P., *Design und Implementierung einer generischen Datenbankarchitektur und geeigneter Schnittstellen zur Bereitstellung der Datengrundlage für adaptive probabilistische Agenten im Rahmen des Projektes SimMarket.XT*, Diplomarbeit, Lehrstuhl für Künstliche Intelligenz Prof. Siekmann, Universität des Saarlandes, 2005.

[Schneider04] Schneider, W., *Marketing und Käuferverhalten*, Oldenbourg Verlag, 2004.

[SchSmo02] Schölkopf, B., Smola, A., *Learning with Kernels: Support Vector Maschines, Regularization, Optimization and Beyond*, The MIT Press, Cambridge, 2002.

[SchSta03] Schwaiger, A., Stahmer, B., *SimMarket: Multiagent-Based Customer Simulation and Decision Support for Category Management*, Multiagent System Technologies, MATES Conference 2003, Seiten 74-84, Springer-Verlag, 2003.

[SchSta04] Schwaiger, A., Stahmer, B., *SimMarket – Agentenbasierte Simulation menschlichen Kaufverhaltens*, Research Report, DFKI GmbH, 2004.

[SchSta05] Schwaiger, A., Stahmer, B., *SimMarket – Simulation von Kundenverhalten mit probabilistischen Agenten*, In Marketing- und Management-Transfer, Institut für Handel & Internationales Marketing, Oktober, 2005.

[SchSta05b] Schwaiger, A., Stahmer, B., *Probabilistic Holons for Efficient Agent-Based Data Mining and Simulation*, In Lecture Notes in Computer Science, Springer-Verlag, Proceedings of Holonic and Multi-Agent

Systems for Manufacturing: Second International Conference on Industrial Applications of Holonic and Multi-Agent Systems, HoloMAS 2005, Copenhagen, Denmark, August 22-24, 2005.

[SchStaRus05] Schwaiger, A., Stahmer, B., Russ, C., Erfinder von: *Verfahren und Vorrichtung zur Simulation von nicht linearen Abhängigkeiten zwischen physikalischen Entitäten und über Sensoren gemessene Einflussfaktoren auf der Basis eines Mikro-Simulationsansatzes mittels in Objekten eingebetteten probabilistischen Netzen*, Patentanmeldenr.: 05108352.5, Europäisches Patentamt, September, 2005.

[Schwaiger01] Schwaiger, A., *The RescueBots Simulator System*, Diplomarbeit, Lehrstuhl Prof. Siekmann, Universität des Saarlandes, 2001.

[Schwaiger06] Schwaiger, A., *Modellierung, Simulation und Vergleich individuellen Konsumentenverhaltens mit probabilistischen Holonen*, Dissertation, Lehrstuhl Prof. Siekmann, Universität des Saarlandes, 2006.

[Schwarz78] Schwarz, G., *Estimating the dimension of a model*, Annals of Statistics 6, Seiten 461-464, 1978.

[ShoGei97] Shoikhet, K., Geiger, D., *Finding optimal triangulations via minimal vertex separators*, a preliminary version appeared in Proceedings of AAAI, 1997.

[SmiDav81] Smith, R. G., Davis, R., *Frameworks for cooperation in distributed problem solving*. IEEE Transactions on Systems, Man and Cybernetics, 11(1):61-70, 1981.

[SpiLau90] Spiegelhalter, D. J., Lauritzen, S. L., *Sequential updating of conditional probabilities on directed graphical structures*, Networks, 20, Seiten 579-605, 1990.

[Spiegelhalt86] Spiegelhalter, D. J., *Probabilistic reasoning in predictive expert systems*, In Uncertainty in Artificial Intelligence, Amsterdam, Seiten 47-68, 1986.

[SprGlySch93] Sprites, P., Glymour, C., Scheines, R., *Causation, Prediction and Search*, No. 81 in Lecture Notes in Statistics, Springer-Verlag, Heidelberg, 1993.

[Stahmer01] Stahmer, B. P., *Hybrid Agents in the RoboCup Rescue Domain*, Diplomarbeit, Lehrstuhl Prof. Siekmann, Universität des Saarlandes, 2001.

[StaSch04] Stahmer, B. P., Schwaiger, A., *Holonic Probabilistic Agent Merging Algorithm*, Proceedings of Intelligent Agent Technology, IEEE/WIC/ACM IAT Conference Peking, 2004.

[Stephenson00] Stephenson, T. A., *An Introduction to Bayesian Network Theory and Usage*, Research Report, IDIAP - Dalle Molle Institute for Perceptional Artificial Intelligence, 2000.

[Suzuki96] Suzuki, J., *Learning Bayesian belief networks based on the minimum description length principle*, In Proceedings of the Thirteenth International Conference on Maschine Learning, Seiten 462-470, 1996.

[Thiesing98] Thiesing, F. M., *Analyse und Prognose von Zeitreihen mit Neuronalen Netzen*, Shaker Verlag, 1998.

[ValKimVom01] Valtora, M., Kim, Y.-G., Vomlel, J., *Soft Evidential update for Probabilistic Multiagent Systems*, Department of Computer Science and Engineering, University of South Caroline und University of Aalborg, 2001.

[VerPea90] Verma, T. S., Pearl, J., *Equivalence and synthesis of causal models*, In Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence, San Mateo, CA, Morgan Kaufmann, Seiten 220-227, 1990.

[Vom99] Vomlel, J., *Methods of Probabilistic Knowledge Integration*, PhD thesis, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University, 1999.

[Wahlster03] Wahlster, W., SmartKom: *Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell*, In: Krahl, R., Günther, D. (eds): Proceedings of the Human Computer Interaction Status Conference, Seiten 47-62, Berlin, 2003.

[WalBou68] Wallace, C. S., Boulton, D. M., *An information measure for classification*, The Computer Journal 11, Seiten 185-194, 1968.

[Weiss00] Weiss, G., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge, 2000.

[WieBuc03] Wiedmann, K.-P., Buckler, F., *Neuronale Netze im Marketing-Management – Praxisorientierte Einführung in modernes Data-Mining*, 2. Auflage, Gabler Verlag, Wiesbaden, 2003.

[Wittig03] Wittig, F., *Maschinelles Lernen Bayes'scher Netze für benutzeradaptive Systeme*, Akademische Verlagsgesellschaft Aka Berlin, Dissertation, Universität des Saarlandes, FB Prof. Wahlster, 2003.

[ZePrKi00] Zeigler, B. P., Praehofer, H., Kim, T. G., *Theory of Modeling and Simulation*, 2. Ausgabe, Academic Press, San Diego, 2000.

[Zentes00] Zentes, J., *One To One Marketing – Sinnvoll Und Umsetzbar?*, BestFoods Forum, TrendForum Verlag, 2000.

[ZenBieKle04] Zentes, J., Biesiada, H., Schramm-Klein, H., *Performance Leadership im Handel*, Zukunft im Handel Band XIX, Deutscher Fachverlag, 2004.

[ZenJanMor99] Zentes, J. , Janz, M., Morschett, D., *New Dimensions in Retail Marketing*, Institute for Commerce and International Marketing & SAP AG, 1999.

[Zweig98] Zweig, G. G., *Speech Recognition with Dynamic Bayesian Networks*, Ph.D. Thesis, University of California, Berkeley, 1998.