Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Tim Dethlefs

Ein verbraucherorientiertes Energiesystem für Smart Grids
-
Entwicklung eines Multi-Agenten-Systems zur dezentralen Optimierung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Tim Dethlefs

Ein verbraucherorientiertes Energiesystem für Smart Grids
-
Entwicklung eines Multi-Agenten-Systems zur dezentralen Optimierung

Masterthesis eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Informations- und Kommunikationstechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Wolfgang Renz
Zweitgutachter : Prof. Dr. Ing. Holger Kapels

Abgegeben am 22. Mai 2014

Tim Dethlefs

Thema der Masterthesis

Ein verbraucherorientiertes Energiesystem für Smart Grids - Entwicklung eines Multi-Agenten-Systems zur dezentralen Optimierung

Stichworte

Dezentrale Optimierung, Smart-Grids, Demand-Side-Management, Energielogistik

Kurzzusammenfassung

Im Smart Grid der Zukunft wird das Demand-Side-Management (DSM) den Rang einer Schlüsseltechnologie einnehmen, mit dem ein neuer Freiheitsgrad zur Kontrolle geschaffen wird, um die Verluste und Schwankungen durch volatile Distributed Energy Resources zu vermindern. In dieser Arbeit soll ein Architekturansatz für ein verbraucherorientiertes Energiesytem spezifiziert und implementiert werden. Dieses soll den Kunden ermöglichen, anhand eines Marktmodells, spontane verteilte Optimierungen des Energiebedarfs durchzuführen. Dadurch sollen die Transparenz, der Automatisierungsgrad und die Akzeptanz von DSM-Maßnahmen erhöht werden.

Tim Dethlefs

Title of the Masterthesis

A demand-side-oriented Energy-System - Development of a Multi-Agent-System for decentralized Optimization

Keywords

decentralized optimization, smart-grids, demand-side-management, energy logistics

Abstract

Demand-Side-Management (DSM) is one of the key applications in the future smart grid, creating a new degree of control in order to reduce losses and fluctuations caused by volatile distributed energy resources. In this thesis an architectural approach on a consumer-orientated demand-side-application will be proposed. Utilizing methods of multi-agent-systems (MAS) the architecture enables consumers to form spontaneous networks in order to optimize their demand.

Danksagung

”It was the best of times, it was the worst of times,...”

Charles Dickens, A Tale Of Two Cities

Die Erstellung dieser Arbeit wäre ohne die Hilfe und Unterstützung einer Vielzahl von Personen unmöglich gewesen. Ich möchte an dieser Stelle allen voran bei meinem betreuenden Professor Dr. Wolfgang Renz für die unermüdliche Unterstützung und Anleitung bedanken. Die persönliche und fachliche Begleitung in dieser Zeit durch meine geschätzten Kollegen Gregor Balthasar, Dirk Beewen und Thomas Preisler haben diese Arbeit stets bereichert, genauso wie auch meinen Alltag. Ebenso danke ich der Arbeitsgruppe C4DSI um Hans Schäfers, meinem Zweitgutachter Dr. Holger Kapels und den Entwicklern von Jadex Active Components Dr. Lars Braubach und Dr. Alexander Pokahr.

Meiner Familie gebührt besonderer Dank, da sie in allen Höhen und Tiefen in dieser Zeit bei mir standen, ebenso wie Anne Gärtner, die mir ein ums andere mal die nötige Kraft gab.

Abschließend danke ich auch meinen Kommilitonen des Informations- und Kommunikationstechnik-Master Jahrgang 2012. Es war eine tolle Zeit mit euch!

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

Die Energieversorgung befindet sich weltweit vor einem grundlegenden Paradigmenwechsel. Der Wandel von der intensiven Nutzung fossiler Brennstoffe, sowie nuklearen Energieträgern, hin zu einer regenerativen und nachhaltigeren Energieerzeugung basiert auf der fundamentalen Feststellung, dass die globale Erwärmung, ausgehend von der weltweit steigenden Industrialisierung, als auch den wachsenden Emissionen konventioneller Kraftwerke, die Menschen in vielen Teilen der Erde zunehmend mit klimatischen Veränderungen und extremen Wetterphänomenen konfrontiert (vgl. Potsdam Institute for Climate Impact Research and Climate Analytics (2012)). Auch sind die traditionellen Quellen für Energieträger, wie Öl, Gas, Kohle und radioaktive Isotope, begrenzt. Zwar sind bei radioaktiven Energieträgern und Kohle mittelfristig keine Engpässe abzusehen (Studie von Cramer u. a. (2009)), doch beim Erdöl wird der kritische Zeitpunkt der maximalen globalen Förderung (Peak-Oil) je nach Prognose zwischen 2020 bei Odell (2000) und 2035 (vgl. Cramer u. a. (2009)) taxiert. Der Wandel der Energiegewinnung kann demnach bereits jetzt als eine der zentralen globalen Herausforderungen des beginnenden 21. Jahrhunderts angesehen werden.

Fossile Quellen, wie zum Beispiel Erdgas und Öl und insbesondere die Kohle, sind seit der Industrialisierung die primären Energieträger. 2005 wurden immer noch 81 % des weltweiten Energiebedarfs mit diesen fossilen Brennstoffen gedeckt (siehe IEA (2007)), während der Weltklimarat feststellte, dass deren Emissionen (Kohlendioxid, Ruß, Stickoxide) durch die intensive Nutzung erheblich zur globalen Erwärmung beitragen (siehe IPCC (2013)). In den 1950er Jahren wurde in die Kernspaltung als billige und saubere Energieform der Zukunft große Hoffnungen gesetzt. Dieser Glaube wurde jedoch durch die Kraftwerksunglücke von Tschernobyl 1986 und Fukushima 2011 weltweit stark erschüttert. Infolgedessen setzte in vielen Nationen ein Umdenken ein, so zum Beispiel durch die in Deutschland 2011 beschlossene Energiewende[1], die den vollständigen Ausstieg aus der Kernenergie bis 2022 vorsieht. Bereits im Jahr 2000 wurde das Erneuerbare-Energien-Gesetz (EEG)[2] durch die damalige Bundesregierung in Kraft gesetzt, wodurch der Bau und Betrieb von regenerative Quellen, wie Photovoltaik- oder Windkraftanlagen, gefördert werden. Ziel war eine deutliche Steigerung des Anteils regenerativer Energien an der Gesamterzeugung auf bis zu 80 % Anteil an Erneuerbaren bis 2050 [3].

[1] §7 des Dreizehnten Gesetzes zur Änderung des Atomgesetzes, Bundesgesetzblatt Nr. 43 (2011)
[2] Offizieller Titel: "Gesetz für den Vorrang Erneuerbarer Energien", Inkrafttreten: 1. April 2000
[3] §1 Abs. 2 EEG

1.1. Motivation

Das Einspeiseziel der Bundesrepublik Deutschland und anderer Staaten für einen erhöhten Anteil erneuerbarer Energien am Strommix, sowie die technische Entwicklung kleiner, preisgünstiger und rentabler Erzeuger, die mittels hochverfügbarer (Gas, Biomasse) oder gar praktisch unlimitierter Ressourcen (Sonne, Wind, Wasserkraft) betrieben werden können, führt zu einer zunehmenden Dezentralisierung und Diversifizierung der Erzeugerlage der elektrischer Energie. Rechtlich flankiert werden diese Maßnahmen durch eine Bevorzugung regenerativer Energieträger bei der Einspeisung[4] und Subventionen für den Betrieb solcher Anlagen.

Die Verfügbarkeit und volle Erzeugungskapazität dieser Erzeuger hängt jedoch zum Teil von nicht-beeinflussbaren und vorhersagebasierten Faktoren wie der Wetterlage ab, wodurch Energieversorgungsunternehmen (EVUs) in ein Spannungsfeld zwischen der vorangingen Einspeisung regenerativer Energien durch das EEG, der Gewährleistung der Versorgungssicherheit[5] und dem wirtschaftlichen Betrieb der eigenen Kraftwerke geraten. Um kurzfristige Defizite von erneuerbaren Energien kompensieren zu können, benötigt es beispielsweise regelfähige konventionelle Kraftwerke und entsprechende Energiespeicher.

Die Nutzung des Potentials der Verbraucherseite als zusätzlichen Freiheitsgrad um die Situation abzumildern, wurde bereits in den 1980er Jahren, unter anderem von Gellings (1985), als *Demand-Side Management* (DSM) vorgestellt. Neben den Ausprägungen des DSM, bei denen Stromeinspareffekte durch effizientere Verbraucher oder bewusstere Nutzung erzielt werden, ist auch die aktive Lastbeeinflussung auf der Verbraucherseite als kurzfristige DSM-Maßnahme zunehmend von Interesse. Da bei der elektrischen Energie keine Quality of Service (QoS) flächendeckend gehandelt werden kann, konzentrieren sich die Geschäftsmodelle dabei auf das Planungspotential und die Flexibilität, d.h. der zeitlichen Verschiebung von Lasten. Früher begrenzende Faktoren für dieses DSM, wie den Zeit- und Einarbeitseinsatz und der Reaktionsgeschwindigkeit wurden seitdem laut Palensky und Dietrich (2011) durch den stark steigenden Automatisierungsgrad und durch die breite Verfügbarkeit von vernetzen *embedded systems*, sowohl auf Betreiber- als auch auf Kundenseite, weitgehend vermindert. Es wurden daher viele Ansätze entwickelt, die Verbraucherseite aktiv in Smart Grid Konzepte einzubinden. Abgesehen von einigen Forschungsvorhaben mit Industriepartnern und Haushalten, wie den eEnergy-Projekten, wurde jedoch bislang keine flächendeckende Integration erreicht, obwohl die Smart Meter flexible Tarife[6] unterstützen.

Die möglichen Anwendungsfälle des Demand-Side Managements sind vielseitig. In der Forschung finden sich zumeist Systeme, die zum *Load Balancing* eingesetzt werden, um den

[4] §8 Abs. 1 S. 1 EEG
[5] §13 Abs. 1 EnWG
[6] § 40 EnWG

Verbrauch mit der Erzeugung abzustimmen und Abweichungen zwischen diesen auf unterschiedlichen Zeitskalen zu minimieren. Dabei wurden Systeme von der Haushaltsebene (zum Beispiel Matallanas u. a. (2012)) bis hin zur Liegenschafts- (siehe Kok u. a. (2005) oder Kamper (2010)) und Verteilnetzebene (bspw. bei Gabaldon u. a. (2003)) entwickelt.

In der Literatur lassen sich zwei Typen von Lastbeeinflussungen identifizieren. Der erste Ansatz nutzt Steuersignale, um alle angeschlossenen Verbraucher direkt zu kontrollieren (etwa bei Lünsdorf und Sonnenschein (2009)). Diese werden in der Regel bei hochreaktiven (Echtzeit-) Systemen eingesetzt, die verlässliche Antwortsignale liefern sollen, um beispielsweise Primärregelleistung und Sekundärregelleistung anzubieten.

Der zweite Ansatz sind Marktbasierte Systeme. Dabei können drei Arten von Systemen unterschieden werden: Auktionsbasierte, tarifbasierte und börsenbasierte Marktmodelle. Das *PowerMatcher*-System von Kok u. a. (2005) ist ein Beispiel für eine Auktionsplattform. Dabei werden Auktionen zwischen einzelnen oder aggregierten Verbrauchern und den Erzeugern durchgeführt. Tarifbasierte Systeme sind am weitesten verbreitet, auch da diese per EnWG durch die Smart Meter unterstützt werden müssen. Dem Smart Meter in den Zielhaushalten wird dabei, statt eines fixen Tarifs, eine Preiskurve übermittelt, wodurch für verschiedene Tageszeitpunkte unterschiedliche Strompreise gelten. Angewendet wird dieser Ansatz unter anderem von Ulbig und Andersson (2010) und Ramchurn u. a. (2011). Börsenorientierte Modelle bieten für jeden Handelszeitpunkt Preise mit dazugehörigen Mengen an. Diese Modelle werden in der Praxis im Energiehandel, zum Beispiel der EEX oder der EPEX-Spot, eingesetzt. In der Forschung wurden Börsenmodelle für kleine Märkte bislang jedoch kaum beachtet.

Viele der hier vorgestellten Ansätze sind entweder sehr spezifisch für eine Anwendungsdomäne respektive einen Verbrauchertyp definiert und daher nicht unbedingt übertragbar beziehungsweise generalisierbar oder enthalten Definitionsprobleme die den Praxiseinsatz durch komplizierte Steuermechanismen erschwerten. Als Beispiel sei der von Kamper (2010) beschriebene *Lawineneffekt* genannt, der in Abschnitt 3.2 näher erläutert wird. Es wird demnach eine generische Architektur benötigt, die viele Verbrauchertypen abbilden kann und deren Stromverbrauch ohne besondere externe Steuerungsmaßnahmen, aber dennoch unter Berücksichtigung der lokalen Nebenbedingungen der Verbraucher, wie beispielsweise *Must-Run*-Kriterien (siehe Abschnitt 3.2), optimieren kann.

1.2. Zielsetzung

Im Rahmen dieser Arbeit soll eine Architektur für ein Energiesystem, welches eine leichtgewichtige, lose-gekoppelte und verteilte Optimierung von Lasten auf der Verbraucherseite bereitstellen soll, entwickelt werden. Dazu werden die notwendigen Komponenten, Kommunikationsprotokolle und Schnittstellen beschrieben. Angewandt werden Methoden aus dem Bereich der Multi-Agenten Systeme, um den Verbrauchern die Möglichkeit zu geben, spontane Netzwerke zu formen, um ihren Verbrauch zu optimieren. Grundlage dafür ist ein zu entwickelndes generisches Verbrauchermodell, welches möglichst viele Verbrauchertypen abbildet. Es soll gezeigt werden, dass das entwickelte Verbrauchermodell in der Lage ist, die beiden für das DSM interessanten Verbraucherklassen, entsprechend der Definition von Lünsdorf und Sonnenschein (2009), in einem Modell zusammenzufassen.

Das zugrundeliegende Marktmodell soll eine transparente und verständliche Preisbildung ermöglichen. Die Skalierbarkeit, sowie die Adaptivität des Modells in Hinblick auf die flexible Anwendung auf verschiedene Geschäftsmodelle und Betriebsmodi, stehen dabei ebenfalls im Fokus.

Das Ziel ist die Nutzung der Flexibilität auf der Verbraucherseite, um verschiedene Anwendungsszenarien zu realisieren, wie zum Beispiel die effizientere Nutzung vorhandener Ressourcen, die Planung von Lasten und die Bereitstellung einer Plattform für zukünftige Geschäftsmodelle auf der Verbraucherseite. Dabei sollen auch die folgenden Forschungsfragen untersucht werden. Allen voran steht die Frage, ob der verteilte Ansatz mit den gewählten Metaheuristiken in der Lage ist, die angestrebte Bandbreite an praxisrelevanten Energieproblemen zu lösen und ob ein generisches Verbrauchermodell entwickelt werden kann, welches über definierte und vorher festgelegte Schnittstellen gegenüber den Optimierungsalgorithmen möglichst viele Verbraucher abbildet.

Im zweiten Schritt soll die entwickelte Architektur in ein Simulationssystem umgesetzt und untersucht werden. Dabei sollen insbesondere Haushaltsgeräte als Verbraucher dienen, da die Nutzung dieser Gerätetypen gut untersucht und dokumentiert ist und statistische Daten vorliegen. Damit soll das System anhand von Anwendungsfällen aus der Praxis und wissenschaftlichen Grundlagenliteratur untersucht werden.

1.3. Aufbau der Arbeit

Die Arbeit ist in sieben Kapitel unterteilt. Nach der Einleitung werden im zweiten Kapitel die wichtigsten Grundlagen und technischen Entwicklungen skizziert, die für das Verständnis der Arbeit bedeutsam sind. Dabei wird zuerst auf das Spannungsfeld zwischen rechtlichen, wirtschaftlichen und technischen Entwicklungen im Stromnetz eingegangen und der Bezug zum *Smart Grid* und dem Demand-Side Management hergestellt. Ferner werden die theoretischen Grundlagen der verwendeten Optimierungsalgorithmen behandelt und Multi-Agenten-Systeme (MAS) als Teil der verteilten Anwendungen behandelt.

Im dritten Kapitel werden die Anforderungen an das zu entwickelnde Energiesystem beschrieben. Diese sind in wirtschaftliche Anforderungen und Eigenschaften der zu entwickelnden Architektur, sowie Anforderungen an das zu implementierende Referenzsystem, geteilt. In der technischen Analyse wird das ausgewählte Multi-Agenten-System näher spezifiziert, das Optimierungsproblem mathematisch beschrieben und Testcases definiert.

Im vierten Kapitel wird die Architektur des Systems spezifiziert. Es werden sowohl die einzelnen Komponenten definiert, als auch die verteilten Optimierungsalgorithmen für die Anwendungsdomäne angepasst.

Das fünfte Kapitel widmet sich der Referenzimplementierung der entwickelten Architektur in Jadex Active Components als Simulationssystem. Der Fokus liegt hierbei auf der Anpassung des Systems an die Anforderungen eines Energiesimulationssystems und des genutzten Multi-Agenten-Systems.

Die Validierung und Simulation von Test- und Usecases ist das Thema des sechsten Kapitels. Neben der Validierung der entwickelten Simulation werden ausgewählte Use-Cases aus Literatur und Praxis simuliert, um die Funktionsweise des Systems zu demonstrieren. Abschließend wird die Arbeit evaluiert und ein Ausblick auf weiterführende Forschungsthemen gegeben.

2. Grundlagen und Stand der Technik

In diesem Kapitel sollen die wichtigsten Grundlagen zum Verständnis der Arbeit dargestellt und erläutert werden. Dabei werden sowohl die Energie- und Stromversorgung, als auch Grundlagen zur Optimierung und Multi-Agenten-Systemen behandelt.

2.1. Energie und Energieversorgung

Die Energieversorgung erfährt derzeit einem tiefgreifenden Wandel, insbesondere im Bereich der Elektrizitätsversorgung. Die Elektrizität ist durch ihre Verfügbarkeit und vielseitigen Einsatzmöglichkeiten zu einem der wichtigsten und bedeutendsten Energieträger geworden. Stetig neue Anwendungen und Nutzungspotentiale, aber auch Gesetze, Normen und Richtlinien, sowie gesellschaftliche Entwicklungen, generieren stetig neue Anforderungen an die Erzeugung, Netzinfrastruktur und zunehmend auch an den Verbraucher. In diesem Abschnitt sollen sowohl die rechtlichen, marktwirtschaftlichen, als auch die technischen Aspekte der Stromversorgung in den Kontext der aktuellen Entwicklungen, wie beispielsweise der zunehmenden Einbindung der Verbraucher, gesetzt werden.

2.1.1. Rechtliche Rahmenbedingungen

Die rechtlichen Entwicklungen in der Energieplanung und Energieeffizienz sind historisch stark verankert mit den Ölkrisen der 1970er Jahre sowie den großen Atomkraftwerkshavarien in Tschernobyl 1986 und Fukushima 2011. In den USA wurde die Entwicklung der Energieverteilung maßgeblich durch die Verteuerung der Öls im Zuge der Ölkrise angestoßen. Auch heute noch maßgebliches Gesetz ist der *National Energy Conservation Policy Act* (NECPA) von 1978 mit dem Ordnungen und Normen zur Stabilisierung des Energieverbrauchs der Vereinigten Staaten geschaffen wurden. In den Folgejahren stieg der Stromverbrauch laut U.S. Energy Information Administration (2013) jedoch weiterhin stark an. Im Gegensatz dazu steht Kalifornien, wo bereits 1972 mit einer eigenen gesetzlichen Verordnung zur Steigerung der Geräteeffizienz, neuen Bauvorschriften und Effizienzprogrammen der Energieversorger die Nachfrage stabilisiert werden konnte und dies 2006 mit einem neuen Gesetz manifestierten (siehe Mitchell u. a. (2009)). 2007 wurde der *Energy Independence and Security Act*

(EISA) als Erweiterung des NECPA geschaffen, um insbesondere die Kraftstoff- und Lichteffizienz zu steigern. Neben diesen bislang vorwiegend passiven Maßnahmen werden von staatlichen Stellen wie der Federal Energy Regulatory Comission (FERC) zunehmend Studien zur Steigerung des Anteils erneuerbarer Energieressourcen und zur aktiven Steuerung des Verbrauchs in Auftrag gegeben (als Beispiel die Studie der Federal Energy Regulatory Commission (2009)).

In Deutschland wurde im Jahr 2000 das *Gesetz für den Vorrang Erneuerbarer Energien* (EEG) als Bundesgesetz inkraft gesetzt, welches die bevorzugte Einspeisung von Strom aus erneuerbaren Energieerzeugern regelt und diesen eine feste Einspeisevergütung für 20 Jahre garantiert. Bis 2020 sollen mindestens 35% des Stroms von erneuerbaren Energieerzeugern generiert werden. Dieser Anteil soll bis 2050 auf mindestens 80% steigen. 2013 betrug, laut Statistisches Bundesamt (2013), der Anteil der Erneuerbaren an der Bruttostromerzeugung bereits 23,4%. Aufgrund der Volatilität vieler erneuerbarer Energieerzeuger steht die vorrangige Einspeisung im Konflikt mit klassischen Betriebskonzepten und fordert eine hohe Dynamik von regelfähigen Kraftwerken zum Ausgleich von Energiedefiziten, beziehungsweise die Schaffung von entsprechenden Speicherkapazitäten. Das *Energiewirtschaftsgesetz* (EnWG) fordert von Übertragungsnetzbetreibern, dass die Sicherheit und Zuverlässigkeit des Versorgungssystems in der jeweiligen Regelzone bei Störungen oder Gefährdungen durch netz- oder marktbezogene Maßnahmen sicherzustellen ist[1]. Das EnWG regelt jedoch nicht nur die Pflichten der Erzeugerseite, sondern ermöglicht auch eine genauere Messdatenerfassung[2] und Maßnahmen zur Lastbeeinflussung auf Seiten der Verbraucher durch last- und tageszeitabhängige Tarife[3]. Der verpflichtende Einbau von intelligenten Zählern, den *Smart Metern*, bei Neubauten und umfangreichen Renovierungen soll dabei zur flächendeckenden Verbreitung der beschriebenen verbraucherorientierten Maßnahmen führen[4].

[1] § 13 EnWG und § 14 EnWG
[2] § 21b EnWG
[3] § 40 EnWG
[4] § 21c EnWG

2.1.2. Rollen- und Architekturmodelle im Smart Grid - Umfeld

Um diese Situation handhabbar zu machen, werden in Zukunft nahezu alle Bereiche des Netzes, vom Erzeuger bis zum Endverbraucher, automatisiert werden. Durch die geringen Investitionskosten der DERs und den damit verbundenen neuen Betreibergruppen, wie bei Solaranlagen, Bürgerwindparks und Micro-BHKWs diffundiert auch die traditionelle Rollenverteilung von Erzeugern und Konsumenten. Klassische Energieunternehmen werden zunehmend zu Energiedienstleistern, die neben der Netzinfrastruktur auch die Kommunikationsinfrastruktur anbieten. Dieses neue Rollen- oder Domänenmodell wurde insbesondere in der NIST[5]-Framework-Roadmap (NIST (2010)) definiert. Dieser NIST-Standard bildet für weitere Publikationen in diesem Bereich, auch über die Grenzen der USA hinaus, eine Grundlage. Das NIST-Modell definiert sieben Domänentypen (Abbildung 2.1), welche hier kurz beschrieben werden sollen.

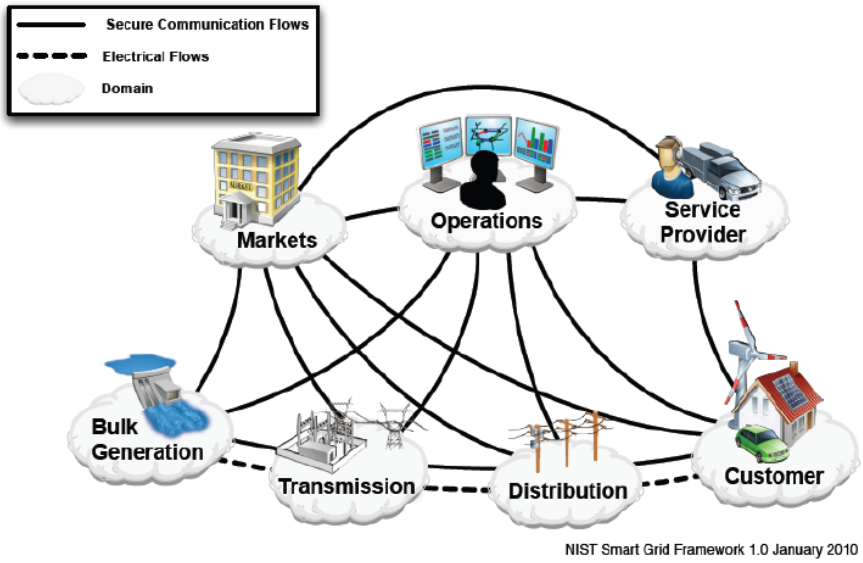

Abbildung 2.1.: NIST-Domänenmodell des Smart Grids (aus NIST (2010))

- **Customer:** Die Verbraucherdomäne stellt den Endverbraucher dar. Dieser kann entweder ein Haushalt, ein Gewerbe oder eine Industrieanlage sein. Die Steuerungsschnittstelle zu den anderen Domänen ist das Energy Service Interface (ESI) im Smart-Meter des Verbrauchers, welches den Energiebedarf und unter Umständen vorhandene DERs verwaltet.
- **Markets:** Auf den Märkten wird nicht nur Energie gehandelt, sondern auch dazugehörige Services . Auf den Markt wird im folgenden Abschnitt näher eingegangen.

[5]National Institute of Standards and Technology - Standardisierungsbehörde des US-Wirtschaftministeriums

- **Service Provider:** Die Service Provider Domäne ist recht breit gefasst. Diese Domäne umfasst sowohl netzspezifische Services, wie zum Beispiel die Verbrauchsabrechnung oder die Unterhaltung des Netzes, als auch die Entwicklung neuer Marktkonzepte und Produkte.
- **Operations:** Diese Domäne ist im derzeitigen Netz eine der wichtigsten, da sie die Stabilität des Netzes sicherstellt. Laut NIST-Modell sollen jedoch in Zukunft mehr und mehr Funktionen durch die Service Provider Domäne wahrgenommen werden. Allerdings wird die Operations-Domäne immer eine wichtige Rolle im Smart-Grid einnehmen, da die Netzstabilität und -funktionalität existentielle Bedeutung hat.
- **Bulk Generation:** In dieser Domäne sind Großkraftwerke zusammengefasst.
- **Transmission:** Hierbei handelt es sich um die Übertragungsnetzbetreiber, die die Energie der Großkraftwerke mit dem Verteilnetz (Distribution) koppeln.
- **Distribution:** Das Verteilnetz verbindet DERs, Übertragungsnetze und Energiespeicher mit den Verbrauchern.

Europäische Organisationen und Forschungseinrichtungen haben das NIST-Modell aufgegriffen und an die lokalen Anforderungen angepasst und weiterentwickelt. Insbesondere wurde die DER-Domäne in der European Smart Grid Reference Architecture (verfasst von Bruinenberg u. a. (2012)) hinzugefügt, welche vorher im NIST-Modell nur implizit in der Distribution- und Customer-Domain vorhanden war. Ebenfalls wurde bei Bruinenberg u. a. (2012) die Layer-Architektur ”SGAM” (Smart Grid Architecture Model, Abbildung 2.2) des Smart-Grids eingeführt. Dabei handelt es sich um ein informationstechnisches Modell, welches die Kommunikationsebenen im Smart-Grid repräsentieren soll. Dieses wurde auch im VDE-Positionspapier ”Energieinformationsnetze und -systeme” von Hübner (2012) aufgegriffen.

Neben den Rollen- und Domänenmodellen wurden im Einklang dazu Smart Grid - spezifische Kommunikationsprotokolle und -modelle entwickelt, beziehungsweise befinden sich in der Entwicklung. Allen voran stehen das Common Information Model (CIM, beschrieben in Uslar u. a. (2012)) und das Smart-Grid Kommunikationsprotokoll IEC 61850 als nachrichtenorientierte Kommunikationsmodelle für Smart Grid Anwendungen.

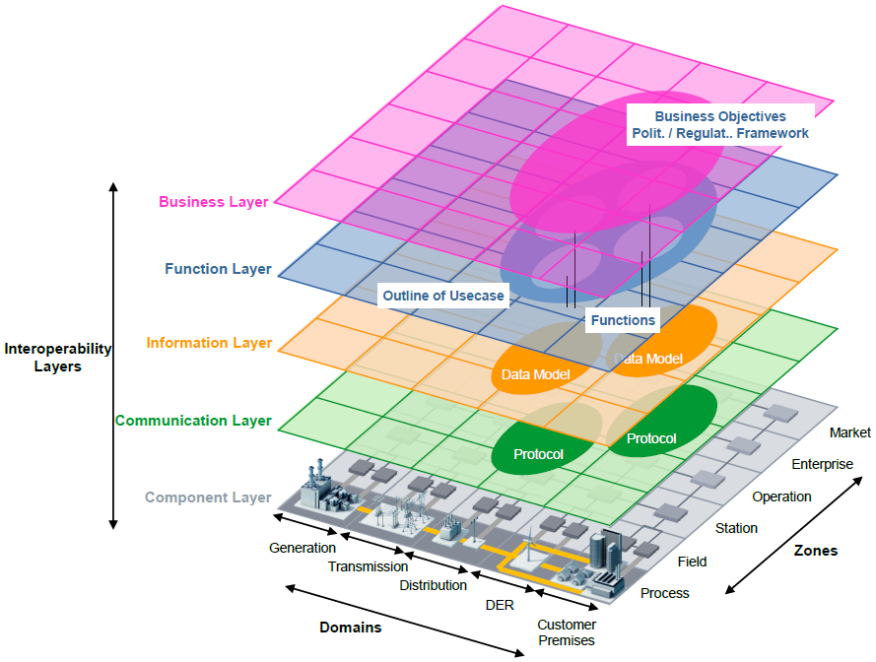

Abbildung 2.2.: Smart Grid Architecture Model (SGAM), aus Bruinenberg u. a. (2012)

2.1.3. Strommärkte

Der heutige Strommarkt in Europa ist, aufgrund des Bestrebens der Wahrung der Netzsicherheit, stark reguliert und durch die großen Handelsvolumina überwiegend nur für Großverbraucher und -Erzeuger zugänglich. Strom wird entweder durch Verträge zwischen den Anbietern und Abnehmern oder über eine Börse gehandelt. Bilaterale Verträge werden hauptsächlich bei langfristigen Handelsbeziehungen geschlossen.

Für Deutschland und Kontinentaleuropa ist die führende Energiebörse die *European Energy Exchange* (EEX)[6] in Leipzig, beziehungsweise ihr auf Stromprodukte spezialisierter Spotmarketableger, die *European Power Exchange* (EPEX SPOT)[7]. Die EEX ist auf langfristigen Energiehandel spezialisiert, während die EPEX Spot den kurzfristigen Stromgroßhandel abdeckt ist und neben Day-Ahead Auktionen auch einen Intraday-Markt anbietet.

Der Auktionsmarkt dient zur Day-Ahead Preisermittlung für Stromlieferverträge. Dieser Preisindex wird Physical Electricity Index (PEHELIX) genannt und aus einem zweiseitigen Sealed-Bid-Auktionsverfahren ermittelt, bei dem der Schnittpunkt von Angebots- und Nachfragekurven für jede Stunde oder Block des Folgetages berechnet wird (zum Verfahren siehe

[6] www.eex.com
[7] www.epexspot.com

Varian (2010)). Das kleinste Handelsvolumen ist 0,1 MW für eine Stunde oder Block. Blöcke sind vordefinierte Handelszeiträume, die mehrere Stunden umfassen.

Der Intraday-Markt ist ein Markt für kurzfristige Handelsverträge bis 45 Minuten vor der Auslieferung. Es kann ab 15:00 Uhr des laufenden Tages für den folgenden Tag gehandelt werden. Die Zeitgrößen sind 15 Minuten oder eine Stunde und die minimale Handelsgröße ist 0,1 MW.

Die Ausschreibungsbedingungen für Regelenergie wurden 2011 von der Bundesnetzagentur (BNetzA) reformiert, sodass der Marktzugang zur Sekundärregelleistung und Minutenreserve, unter anderem für abschaltbare Lasten, erleichtert wurde. So wurden die Mindestgrößen der Angebotsabgabe reduziert, die Aggregation erlaubt und die Möglichkeit geschaffen, Leistungen durch Dritte absichern zu lassen.

Aufgrund der großem Handelsmengen divergieren die dezentrale Erzeugungssituation und die Handelsmöglichkeiten. Durch geringe Investitionsrisiken und kurze Amortisationszeiträume haben laut Kok u. a. (2005) kleine und mittelgroße Energieerzeuger in den letzten Jahren zunehmend an Attraktivität gewinnen können. Diese verteilte Erzeugersituation führt jedoch einerseits dazu, dass die Koordination derselben fortwährend an Komplexität gewinnt, aber andererseits DERs aufgrund ihrer geringen Produktion nicht unmittelbar an den oben vorgestellten Strommärkten partizipieren können. Eine Möglichkeit zur Lösung dieses Problems ist die Gruppierung von DERs zu größeren Einheiten, den sogenannten Virtual Power Plants (VPP), wie sie beispielsweise von Bergmann u. a. (2010) oder Kok u. a. (2010) beschrieben werden. Dadurch werden größere Marktteilnehmer generiert, die an der Börse teilnehmen können. Um ein Virtual Power Plant betreiben zu können, ist die Aggregation von Erzeugern alleine oft nicht ausreichend, sodass auch automatisiert steuerbare und flexible Verbraucher zur Bereitstellung von negativer Regelenergie eingebunden werden können.

2.1.4. Technische Rahmenbedingungen

In Europa wird das Wechselspannungsnetz gemäß IEC 60038 (2009) mit einer sinusförmigen Frequenz von 50 Hz betrieben. Erzeugung und Verbrauch müssen immer im Gleichgewicht sein, um die Netzfrequenz stabil zu halten. Kommt es zu Abweichungen wird die sogenannte Regelenergie eingesetzt (siehe ENTSOE (2004)). Die Regelenergiearten werden anhand ihrer Zeithorizonte und Leistungsgrößen unterschieden, die in Abbildung 2.3 dargestellt werden. Die höchsten Anforderungen werden an die Primärregelleistung gestellt. Sie muss praktisch sofort abrufbar sein und spätestens nach 30 Sekunden die volle Leistung erreicht haben. Die zweite Regelleistungsart ist die Sekundärregelleistung. Die Sekundärregelleistung kann mit einer Reaktionszeit von 30 Sekunden angefordert werden und muss nach fünf Minuten ihre volle Leistung erreicht haben, um spätestens nach 15 Minuten die Primärregelleistung abzulösen, damit diese wieder verfügbar ist. Die Sekundärregelleistung

kann maximal vier Stunden abgerufen werden. Die dritte Regelleistungsart ist die Minutenreserve. Sie muss spätestens 15 Minuten nach der Anforderung ihre volle Leistung erreicht haben und ist, wie die Sekundärregelleistung, bis zu vier Stunden verfügbar.

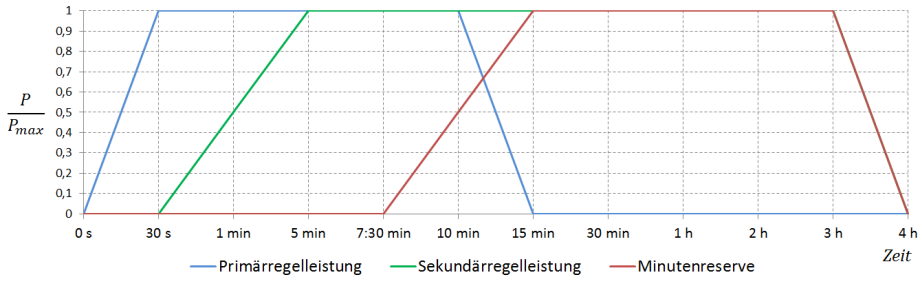

Abbildung 2.3.: Zeithorizonte des Abrufs von Regelleistungen im Stromnetz

Aufgrund der bereits dargestellten Volatilität der DERs kann es interessant sein, sich die Verbraucherseite als zusätzlichen Freiheitsgrad zunutze zu machen, um entweder die Energie durch Planung effizienter zu nutzen, oder negative Regelenergie zur Verfügung zu stellen. Dazu sind sowohl die IKT-Infrastruktur, als auch Heimautomatisierungssysteme und Energy Service Interfaces (siehe NIST (2010)) erforderlich. Diverse Projekte aus dem Energieumfeld nutzen aufgrund der wohldefinierten Standards und der weiten Verbreitung das Internet und das TCP/IP-Protokoll als bestehende Kommunikationsinfrastruktur für ihre Energieanwendungen (siehe Palensky und Dietrich (2011)).

Einen interessanten Ansatz für ESI in der Haushaltsautomatisierung bietet das OGEMA-Projekt (siehe Nestle u. a. (2010)). Es soll in der Zukunft die Anbindung von Haushaltsgeräten über ein Gateway-Interface an die Leitstelle ermöglichen und die Verbraucher abhängig vom variablen Strompreis automatisiert steuern können. OGEMA soll dabei sowohl in der Lage sein, diverse Kommunikationsstandards für Haushaltsverbraucher zu unterstützen, als auch die Verbraucher aggregieren und nach außen über definierte Schnittstellen anzubieten. Daneben etabliert sich das durch Piette u. a. spezifizierte OpenADR als System für automatisierte Demand-Response. OpenADR ist jedoch auf relativ statische top-down DR-Geschäftsmodelle ausgelegt, die vom System Operator ausgehen.

Laut Palensky und Dietrich (2011) ist noch kein einheitlicher Kommunikationsstandard für DSM definiert. Der im *Virtual Power Plant* - Umfeld verbreitete Standard IEC 61850 bietet auch zunehmend Beschreibungsmöglichkeiten für elektrische Verbraucher an und ist daher in einer vielversprechenden Position.

Eine Registry für DERs und Verbraucher, mit der alle verfügbaren Energieservices abgerufen werden können, fehlt jedoch nach derzeitigem Stand noch (siehe Dethlefs und Renz (2013)). Somit gestaltet es sich derzeit als schwierig, abseits von Forschungsprojekten, wie den eEnergy-Projekten, flächendeckende DSM-Kapazitäten nutzen zu können.

2.1.5. Demand-Side-Management

Die Definition des Begriffes ”Demand-Side-Managements” (DSM) unterliegt einer fortwährenden Weiterentwicklung. Gellings (1985) definiert DSM als Planung und Umsetzung von Aktivitäten der elektrischen Versorger, um die Nutzung der Energie durch die Verbraucher so zu beeinflussen, dass die gewünschten Änderungen des Lastganges umgesetzt werden. Identifiziert wurden sechs Lastbeeinflussungsarten des DSM:

- **Peak Clipping:** Reduktion der Lastspitzen durch direkte Lastkontrolle.
- **Valley Filling:** Aufbau von Lasten neben den Lastspitzen, um eine gleichmäßigere Auslastung zu erzielen.
- **Load Shifting:** Die Lastverschiebung dient dazu, Teile von Lastspitzen in weniger kritische Bereiche zu verschieben.
- **Strategic Conservation:** Dies bezeichnet Maßnahmen zu Senkung des Verbrauchs, zum Beispiel durch Energieeinsparungen oder effizientere Geräte.
- **Strategic Load Growth:** Der Aufbau des Energieabsatzes über das Valley Filling hinaus, beispielsweise durch die Elektrifizierung von Fahrzeugen (eMobility).
- **Flexible Load Shape:** Der Verbrauch von Energie anhand einer vorgegeben Lastplanung.

Die Gründe für die Anwendung von DSM können vielfältig sein. Eine Studie der Federal Energy Regulatory Commission (2009) geht für die USA von einer stark steigenden Energienachfrage aus, deren Spitzenlast von 775 GW im Jahr 2009 auf über 900 GW im Jahr 2019 ansteigen könnte. Diese Entwicklung würde den Neubau von etwa 2000 Kraftwerken, einschließlich der entsprechenden Infrastruktur, im selben Zeitraum erfordern. Demand-Side-Management wird hier als Maßnahme gesehen, dieses Wachstum auf 800 GW zu begrenzen oder sogar zu stabilisieren, um den steigenden Kosten und den Auswirkungen auf die Umwelt zu begegnen (siehe Huber u. a. (2011)). Im deutschsprachigen Raum ist neben Energieeinsparungen auch die gezielte Nutzung volatiler Effekte der erneuerbaren Energien von Bedeutung (siehe Palensky und Dietrich (2011)).

Laut Lünsdorf und Sonnenschein (2009) lassen sich Geräte in Bezug auf DSM in drei Klassen teilen.

- **Nachfragegetrieber Betrieb:** Das Geräte wird aufgrund der direkter Nachfrage des Nutzers betrieben und kann nicht beeinflusst werden. Beispiele hierbei wären Licht, Fernseher, Heimcomputer etc.
- **Programmgetriebener oder halbautomatischer Betrieb:** Der Benutzer initialisiert den Verbraucher, aber das Gerät muss nicht sofort starten. Der Verbraucher kann dementsprechend in einem gewissen, vom Benutzer vorgegebenen, Zeitfenster seine Aufgabe erfüllen. Beispiele wären Waschmaschinen oder Geschirrspüler.
- **Vollautomatischer Betrieb:** Die Geräte werden innerhalb gewisser Parameter betrieben und verfügen über Sensorik und Aktoren, um diesen Zustand automatisiert zu halten. Thermische Verbraucher, wie zum Beispiel Heizungen oder Kühlschränke fallen in diese Kategorie.

Im Jahr 2011 betrug der Nettostromverbrauch in Deutschland 535,2 TWh, wobei 46,6% des Verbrauchs auf die Industrie und 25,5% auf Haushalte entfielen (Quelle: BDEW (2012)). Die Aufteilung nach Bereichen kann Abbildung 2.4 entnommen werden.

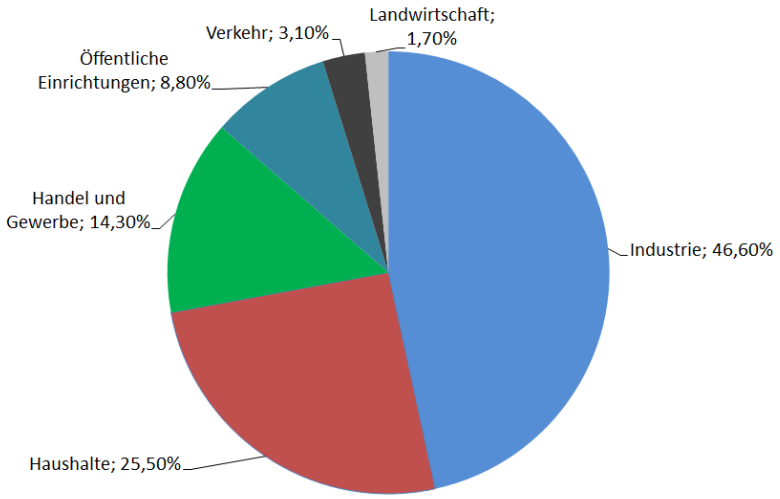

Abbildung 2.4.: Anteile am Bruttostromverbrauch in Deutschland 2012 nach den Daten vom BDEW (2012)

Die Betrachtung von Industrieanwendungen für das DSM gestaltet sich, aufgrund der branchen- und aufgabenspezifischen Anforderungen, ohne konkreten Anwendungsfall als schwierig. In zweitgrößten Verbrauchssektor, den Haushalten, zeigen Untersuchungen der Energieagentur NRW (2006), dass ca. 54 % aller Geräte halb- oder vollautomatisch betrieben werden können. Der Anteil von Stromheizungen, die als thermische Speicher von besonderem Interesse sind, nimmt laut Frey u. a. (2007) derzeit sogar zu.

Intelligente Stromzähler, auch als *Smart Meter* bezeichnet, könnten in Zukunft als Aggregationsplattform für Haushaltsgeräte eingesetzt werden und somit als Gateway in das Smart Grid fungieren. Grundlage dafür ist, dass Smart Meter in Zukunft Steuerungsstandards, wie das openADR-Protokoll oder den IEC 61850 - Standard beherrschen (siehe Palensky und Dietrich (2011)). Das Migrationsziel[8] der Europäischen Union, dass bis 2022 80% der Haushalte mit Smart Metern ausgestattet sein sollen, unterstreicht diese Bemühungen. Derzeit fungieren Smart Meter in Deutschland, abgesehen von einigen Modellregionen, jedoch beinahe ausschließlich als Transparenzplattform, die den Stromverbrauch nachprüfbar machen. Da gegenüber den hohen Einbaukosten keine Einspareffekte vorhanden sind, ist die Akzeptanz der Smart Meter laut Studie der Verbraucherzentrale Sachsen (2012) gegenwärtig noch sehr niedrig. Daher könnten als Alternative für die Hausautomatisierung auch Router infrage kommen, die mit einer Betriebssoftware und den entsprechenden Protokollen ausgestattet werden müssten (siehe Kamper (2010) und Rusitschka u. a. (2009)).

[8] Richtlinie 2009/72/EG vom 13. Juli 2009 über gemeinsame Vorschriften für den Elektrizitätsbinnenmarkt und zur Aufhebung der Richtlinie 2003/54/EG, Annex I (2)

2.2. Zu den Grundlagen der Optimierung

In diesem Abschnitt sollen zuerst die Grundlagen der Optimierung beschrieben werden und danach auf die Ant Colony Optimization und die Genetischen Algorithmen als Optimierungsheuristiken eingegangen werden.

2.2.1. Allgemeine Definition des Optimierungsproblems

Mit der *Optimierung* wird ein Teilgebiet der angewandten Mathematik bezeichnet. Folgend der Definition von Merz (2003) und Gerdts (2013) werden dabei Verfahren eingesetzt, um die Eingabewerte $\vec{x}$ eines Systems so zu bestimmen, dass der Wert der Zielfunktion f dieses System, unter den i.A. vorhandenen Nebenbedingungen, maximiert oder minimiert wird (siehe Formel 2.1).

$$f(\vec{x}) = min! \text{ unter der Nebenbedingung } \vec{x} \epsilon X \tag{2.1}$$

In vielen Fällen können die Probleme nicht analytisch beschrieben werden, sodass Approximationen oder numerische Verfahren eingesetzt werden müssen. Die Menge der möglichen Lösungen können, je nach Anwendungsgebiet, durch eine oder mehrere Nebenbedingungen eingeschränkt sein. Mehrdimensionale Optimierungen führen in der Regel auch zu Topologien der Lösungsräume in denen neben einem globalen Maximum bzw. Minimum mehrere lokale Minima oder Maxima auftreten können.

Optimierungsprobleme treten in vielen Disziplinen auf, so zum Beispiel in der Wirtschaft (Preis- oder Materialeinsatzoptimierung), der Logistik (Routenoptimierung), der Energietechnik (Kraftwerkseinsatzplanung), der Elektrotechnik (Antennendesign, Leiterplattenplanung) und der Physik (Thermodynamik).

2.2.2. Kombinatorische Optimierung

Kombinatorischen Probleme bilden eine besondere Klasse in den Optimierungsproblemen. Ein kombinatorisches Optimierungsproblem ist laut Merz (2003) folgendermaßen definiert:

Ein kombinatorisches Optimierungsproblem besteht aus einer Menge E von Instanzen, einer endlichen Menge $H(I)$ von (möglichen) Lösungen für jede Instanz $I \epsilon D$ und einer Funktion f, die jeder Lösung $\vec{x} \epsilon H(I)$ zu jeder Instanz $I \epsilon E$ einen reellwertigen Lösungswert $f(\vec{x}, I)$ zuordnet.

Viele Probleme der kombinatorischen Optimierung sind in der Problemklasse *NP-schwer*[9], wodurch eine effiziente Lösung häufig nicht möglich ist. Die Lösungsverfahren teilen sich hier in zwei Gruppen. Auf der einen Seite sind exakte Lösungsverfahren, die garantiert das globale Minimum berechnen, aber häufig sehr lange Rechenzeit benötigen, während auf der anderen Seite Metaheuristiken sehr schnell gute Lösungen generieren, bei denen jedoch nicht garantiert ist, dass sie die optimale Lösung finden.

Ein typisches Beispiel für ein kombinatorisches Optimierungsproblem ist das *Travelling Salesman Problem* (TSP). Beim TSP muss ein Handelsreisender mehrere Städte nacheinander besuchen und dabei eine möglichst kurze Route wählen. Er darf dabei jede Stadt nur einmal besuchen. Es zeigt sich, dass das Problem mit einer zunehmenden Anzahl von Städten über-exponentiell wächst und bisher noch kein optimaler Routen-Algorithmus existiert.

Das TSP ist definiert als Minimierungsproblem der Tourlänge L durch n Städte:

$$\min L(x) = \sum_{i=1}^{n} d_{ij} x_{ij}, \tag{2.2}$$

wobei d_{ij} als Wert aus der Distanzmatrix den Abstand zwischen zwei Städten beschreibt. Im Zusammenhang mit der obigen Definition des kombinatorischen Optimierungsproblems beschreibt eine Instanz die Eingabemenge, i.d.R. als Koordinaten der Städte oder als Distanzmatrix. Die Menge der möglichen Lösungen $H(I)$ ist beschränkt durch Nebenbedingungen, z.B. das jede Stadt nur einmal besucht werden darf und jede Stadt besucht werden muss.

[9] Nichtdeterministische Polynomialzeit

2.2.3. Ant Colony Optimization

Unter dem Begriff *Ant Colony Optimization* wird eine Reihe von Algorithmen verstanden, die das natürliche Verhalten von Ameisen nachbilden sollen. Ameisen sind in der Lage den schnellsten Weg zwischen Nest und Nahrungsquelle zu finden. Dazu greifen sie auf eine Pheromon-basierte Wegfindung zurück. In Abbildung 2.5 ist dieses Verhalten skizziert. In der ersten Phase (links) nehmen die Ameisen zufällige Pfade um das Hindernis herum, sodass auf allen Pfaden etwa gleich viele Ameisen unterwegs sind und emittieren dabei eine Pheromonspur (gestrichelte Linie). Nachdem sie die Nahrungsquelle erreicht haben, kehren sie zurück zum Nest. Unter der Voraussetzung, dass alle Ameisen in etwa gleich schnell sind, kehren die Ameisen über den kürzeren Pfad schneller zurück und der Pheromongehalt auf dem kürzeren Pfad steigt gegenüber dem längeren Weg, wie im Bild rechts dargestellt. Ameisen, die nun vom Nest kommen, werden sich anhand des höheren Pheromongehalts für die kürzere Strecke entscheiden und dort noch mehr Pheromone platzieren, während gleichzeitig immer weniger Ameisen den langen Pfad wählen.

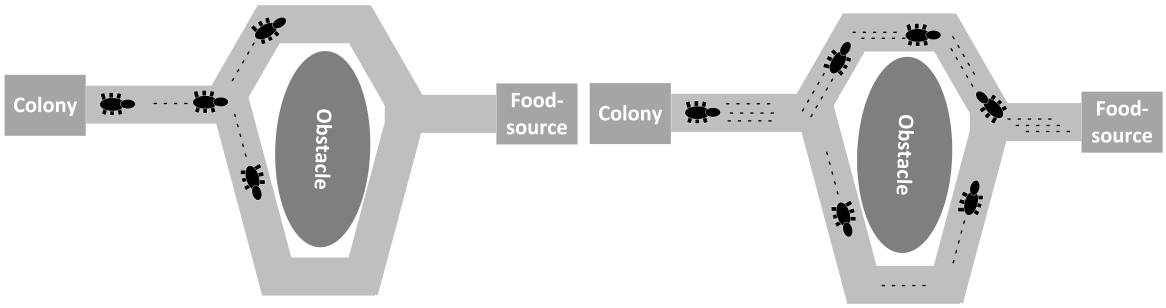

Abbildung 2.5.: Wegfindung mittels Pheromonen bei Ameisen

Dorigo u. a. (1996) haben dieses Prinzip für Optimierungsprobleme adaptiert. Der Fokus liegt dabei auf der Lösung des oben definierten NP-schweren ”Traveling Salesman Problem” (TSP). Die Ant Colony Optimization soll als Metaheuristik einen Lösungsansatz zur schnellen und effizienten Routengenerierung bereitstellen, welcher jedoch nicht zwangsläufig den besten Pfad findet oder Aussagen über die Lösungsgüte treffen kann.

Im darauffolgenden Abschnitt wird das Basissystem ”Ant System” (AS) vorgestellt. Danach folgt die Beschreibung des modernen, optimierten ”Ant Colony Systems” (ACS). Um die Anschaulichkeit zu wahren, wird die Beschreibung der Algorithmen im TSP-Kontext vorgenommen. Knoten des Wegfindungsgraphens sind als Städte zu verstehen und Kanten dementsprechend als Routen.

Der Abstand zwischen den Städten i und j ist als $d(i,j)$ definiert. Der Pheromongehalt dieser Kante als $\tau(i,j)$.

Ant System (AS)

Ant System ist der erste publizierte Optimierungsalgorithmus basierend auf der Ameisen-Metapher. Dieser und die ersten Weiterentwicklungen werden in dem Artikel von Dorigo u. a. (1996) zusammengefasst.

Jede Ameise erzeugt in diesem Algorithmus eine vollständige Tour durch alle Städte. Die Städte werden anhand einer Auswahlwahrscheinlichkeit, basierend auf Abstand und Pheromongehalt, ausgewählt. Dadurch werden kurze Routen mit hohem Pheromongehalt bevorzugt. Nachdem alle Ameisen-Agenten ihre Tour konstruiert haben, wird ein globales Update der Pheromonwerte durchgeführt. Dabei fügt jede Ameise einen bestimmten Anteil des Pheromons auf den Kanten seiner Tour hinzu. Die Menge der von den Ameisen emittierten Pheromonen richtet sich nach der jeweiligen Tourlänge. Kanten auf kürzeren Touren erhalten dabei einen deutlich höheren Anteil an Pheromonen als Kanten auf langen Touren. Zusätzlich sinkt der Pheromongehalt aller Kanten nach jeder Runde um einen gewissen Anteil. Dies sorgt dafür, dass seltener besuchte Kanten zunehmend weniger attraktiv werden.

Die Auswahlwahrscheinlichkeit $p(i,j)$ für eine Stadt j von einer Ameise, die sich in der Stadt i befindet, errechnet sich durch Formel (2.3).

$$p(i,j) = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{u=1}^{n} \tau_{iu}^{\alpha} \cdot \eta_{iu}^{\beta}} \tag{2.3}$$

τ_{ij} ist der Phermongehalt der Kante und $\eta = \frac{1}{d_{ij}}$ das Inverse der Distanz. α und β sind Wichtungsparameter der Distanz gegenüber dem Pheromongehalt. Die Summe ist der Normierungsfaktor über alle verbleibenden Städte.

Der Pheromongehalt einer Kante wird nach jeder Runde mit Formel (2.4) global angepasst:

$$\tau(i,j) \leftarrow (1-\rho) \cdot \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_k(i,j) \tag{2.4}$$

Dabei ist ρ der Pheromon-Verlustparameter. Dieser sorgt dafür, dass der Pheromongehalt der Kante mit der Zeit abnimmt.

$\Delta\tau_k(i,j)$ beschreibt die Menge an emittierten Pheromonen von einer Ameise k und m ist die Anzahl aller beteiligten Ameisen.

Der $\Delta\tau_k(i,j)$ - Term kann nun so gestaltet werden, dass der emittierte Pheromongehalt die Ameisen in gewünschter Weise beeinflusst. Dorigo u. a. (1996) schlagen für TSP-Probleme

die Formel (2.5) vor. L_k bezeichnet dabei die Länge der gesamten Tour einer Ameise k. Dadurch werden kurze Strecken besonders hoch gewertet.

$$\Delta\tau_k(i,j) = \begin{cases} \frac{1}{L_k} \text{ wenn ij } \epsilon \text{ Tour von Agent k} \\ 0, \text{ sonst} \end{cases} \tag{2.5}$$

Beim Ant System - Basisalgorithmus emittieren alle Ameisen Pheromone. Es wäre jedoch auch denkbar, dass nur ausgewählte Ameisen Pheromone verteilen, um eine noch feinere Gewichtung zu erhalten.

Ant Colony System (ACS)

Im Lauf der Zeit wurde der AS-Algorithmus, insbesondere für das TSP, konsequent weiterentwickelt und angepasst. Speziell die Entwicklung des Ant-Q Algorithmus durch Gambardella und Dorigo (1995) und die Betrachtung der Eigenschaften durch Dorigo und Gambardella (1996) stellten besondere Meilensteine dar. Die Besonderheit an Ant-Q ist, dass die Ameisen laufend neue Pfade traversieren. Dadurch wird ein frühes Konvergieren in lokalen Minima der Zielfunktion vermieden und größtenteils bessere Ergebnisse als mit AS erzielt. In der Arbeit von Dorigo und Gambardella (1997) wurde Ant-Q nochmals um eine lokale Update-Regel zum ”Ant Colony System” (ACS) erweitert.

Die ”State Transition Rule” s, mit der bestimmt wird, welche Stadt von der Ameise als nächstes ausgewählt wird, ist im ACS gegenüber dem AS deutlich modifiziert:

$$s = \begin{cases} \arg\max_{u \epsilon J_k(r)} \{\tau(r,u) \cdot \eta(r,u)^\beta\}, \text{ wenn } q \leq q_0 \text{ (exploitation)} \\ \text{S, sonst (biased exploration)} \end{cases} \tag{2.6}$$

mit q als Zufallszahl zwischen [0...1] und q_0 als Parameter zwischen $0 \leq q_0 \leq 1$, welcher die relative Bedeutung zwischen *Exploitation* und *biased Exploration* einstellt. Ist $q \leq q_0$, wählt die Ameise eine Route mit minimaler Länge und maximalem Pheromongehalt. Ansonsten wird die Verteilungsfunktion aus Formel (2.3) eingesetzt, um die nächste Stadt zu bestimmen.

In diesem Algorithmus emittiert ausschließlich die beste Ameise eines Optimierungslaufs Pheromone, um die Suche gerichteter zu machen. Mit der ”Global Update Rule” werden die Kantengewichte global nach einem Lauf angepasst.

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} \text{ wenn (i,j) zur besten Tour gehören,} \\ \tau_{ij}, \text{ sonst.} \end{cases} \tag{2.7}$$

mit ρ als globale Pheromon-Verlustvariable und $\Delta\tau(i,j)$ als Pheromonmenge, die pro Optimierungslauf emittiert wird und sich wie folgt bestimmt:

$$\Delta\tau_{ij} = \frac{1}{L_{best}} \tag{2.8}$$

wobei die Länge L_{best} entweder die beste bisher global gefundene Länge L_{gb} sein kann oder die Iterations-beste Länge L_{ib}. Dorigo und Gambardella (1997) haben festgestellt, dass sich beide Ansätze nur marginal im Ergebnis unterscheiden, wobei L_{gb} leicht bessere Ergebnisse erzielt.

Eine Besonderheit des ACS gegenüber dem Ant-Q Algorithmus ist die ”Local Update Rule”, welche jedes mal durchgeführt wird, wenn sich eine Ameise an einer Kante entlangbewegt.

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi\cdot\tau_0 \tag{2.9}$$

φ ist die lokale Pheromon-Verlustvariable und τ_0 der Pheromongrundwert. Ziel der Local Update Rule ist die Diversifizierung der Suche. Durch die konsequente Verringerung des Pheromongehalts von häufig traversierten Kanten sollen Ameisen zunehmend ermuntert werden, andere Kanten zu nutzen, um neue Lösungen zu generieren.

2.2.4. Genetische Algorithmen

Bei der Gruppe der Genetischen Algorithmen (GA) handelt es sich um Optimierungsverfahren, die von der Evolutionstheorie nach Darwin (1859) inspiriert sind. Ziel ist es, bestimmte vorteilhafte Eigenschaften durch kontinuierliche Anpassung der Population zu erhalten und zu vermehren. In der Biologie sind diese Eigenschaften in den DNA-Strängen der Chromosomen als Gene codiert. Gene können sich in der synthetischen Evolution auf zwei Arten verändern: Selektion und Mutation (siehe Nissen (1994)). Grundlage für die Selektion ist die sogenannte Fitness. Die Fitness bestimmt, in welchem Maße die Gene eines Individuum positive Eigenschaft enthalten. Individuen mit einer großen Fitness werden sich aufgrund ihrer guten Anpassung an die Umwelt besonders häufig vermehren und daher viele Nachkommen in der folgenden Population haben. Die Gene der Nachkommen setzen sich wiederum aus einer Mischung der Gene ihrer Eltern zusammen. Wenn (theoretisch) davon auszugehen ist, dass sich überwiegend Individuen mit hohen Fitnesswerten vermehren, sollte die Folgegeneration im Mittel eine höhere Fitness aufweisen. Individuen mit niedriger Fitness finden nur wenige Partner und ihre Genkombinationen sind geringer in der Folgepopulation vertreten.

Die Mutation ist zweite Operator. Die spontane Mutation verändert ein oder mehrere Gene eines Individuums und kann so zu neuen Eigenschaftskombinationen führen, die alleine durch Selektion nicht zustande gekommen wären.

Nach dieser kurzen Einführung der Begrifflichkeiten soll nun im folgenden die Beschreibung der Metapher der Genetischen Algorithmen im Vordergrund stehen.

Einführung des Chromosoms für die GA

Laut Nissen (1994) ist das Chromosom (oder auch String) in den meisten Fällen die genetische, logische Repräsentation der Zielfunktion.

Als binäre Repräsentation sind die Variablen der Zielfunktion logisch codiert und zu einem String zusammengefasst, wie in Abbildung 2.6 dargestellt. Dieses Verfahren bietet sich insbesondere bei der Lösung von numerischen Problemen an, beispielsweise wenn Startwerte für Neuronale Netze generiert werden sollen. Zur Übersetzung der binär codierten Gene in die Zielfunktion werden eine Decodierfunktion D und die Zielfunktion F benötigt.

Im TSP Kontext könnte ein Chromosom als direkte ganzzahlige Repräsentation des Programms, also der Tour, aufgefasst werden. Das erste Gen bezeichnet dementsprechend die Stadt, mit der gestartet wird, das zweite Gen die darauffolgende Stadt und so weiter.

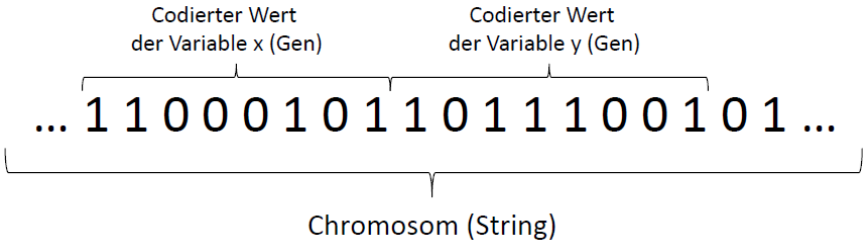

Abbildung 2.6.: Beispiel eines einfachen binären Chromosoms für Genetische Algorithmen

Der genetische Basisalgorithmus

Der Genetische Basisalgorithmus (abgeleitet von Nissen (1994)) ist eine grundlegende Referenzimplementierung der GA. Dabei werden iterativ aus der Startpopulation G_0 Folgegenerationen $G_t(t = 1, 2, \ldots)$ generiert. Eine Population besteht aus n Individuen (Chromosome) $\vec{a_i}$ $(i = 1, 2, \ldots, n)$. Es werden im Basisalgorithmus acht Schritte durchlaufen, die im folgenden näher beschrieben werden.

1. Erzeugen einer zufälligen Startpopulation: Es muss festgelegt werden, aus wie vielen Individuen die Startpopulation und ihre Nachfolger bestehen sollen. Dann werden zufällige Chromosome $\vec{a_i}$ generiert, bis die Startpopulation voll ist.

2. Quantifizierung der Lösungsgüte: Die Zielfunktion muss in die Fitnessfunktion übersetzt werden, da die Selektion im Basis-GA positive Fitnesswerte $\Phi\{\vec{a}\}$ voraussetzt. Zur Übersetzung wird neben der Decodierfunktion D und der Zielfunktion F zuweilen auch eine Skalierungsfunktion S eingesetzt.

$$\Phi\{\vec{a}\} = S(F(D(\vec{a}))) \tag{2.10}$$

3. Fitness-proportionale Selektion und Replikation: In diesem Schritt wird die Replikationswahrscheinlichkeit eines Individuums berechnet. Dazu wird die Auswahlwahrscheinlichkeit P_s aus dem Fitnesswert $\Phi\{\vec{a_i}\}$ des Individuums $\vec{a_i}$ relativ zur Summe aller Fitnesswerte entsprechend Formel (2.11) bestimmt.

$$P_s(\vec{a_i}) = \frac{\Phi\{\vec{a_i}\}}{\sum_{j=1}^{n} \Phi\{\vec{a_j}\}} \tag{2.11}$$

Interpretiert als Urnenversuch, wird nun zweimal aus dem Populationspool mit Zurücklegen gezogen, um eine Paarung zu bilden. Da das Ziehen der Individuen auf Basis der Auftrittswahrscheinlichkeit geschieht, werden Individuen mit hoher Fitness mit höherer Wahrscheinlichkeit gezogen als Individuen mit niedrigerer Fitness.

4. Crossover: Beim Crossover handelt es sich um den Hauptoperator des Basis-GA. Mit diesem sollen neue und bessere Lösungen gefunden werden. Zuerst wird mit einer Wahrscheinlichkeit von P_c ermittelt, ob ein Crossover durchgeführt werden soll. Danach wird an einem zufälligen Punkt der Chromosomen, die im dritten Schritt ausgewählt wurden, ein Bruch erzeugt und die Segmente vertauscht (Abbildung 2.7). Dadurch entstehen zwei Nachkommen für die Nachfolgepopulation.

5. Mutation: Die Mutation ist ein Hintergrundoperator und beugt dem endgültigem Verlust von Allelen (Bitwerten) vor. Dazu wird mit der Wahrscheinlichkeit P_m ein Chromosom invertiert oder ein Bit geändert.

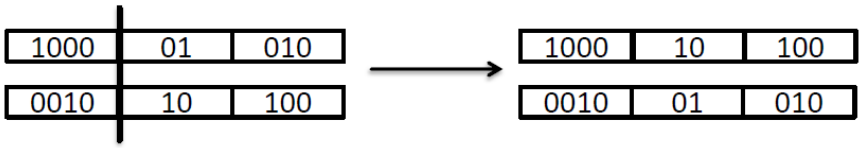

Abbildung 2.7.: Exemplarische Darstellung eines Crossovers

6. Nachkommen der Folgepopulation hinzufügen: Die beiden gezogenen Individuen werden nun der Folgepopulation hinzugefügt.

7. Nachfolgepopulation füllen: Die Schritte 3. - 6. werden wiederholt, bis die Nachfolgepopulation ebenfalls n Individuen hat.

8. Konvergenz prüfen: ansonsten weiter mit 2.

Verteilungskonzepte Genetischer Algorithmen

Bei der Verteilung Genetischer Algorithmen werden zwei Hauptrichtungen unterschieden. Zum einen wird für jeden Prozess oder Prozessor eine lokale Population angelegt (**Migrationsmodell**), oder eine Population ist global über alle Prozesse verteilt (**Diffusionsmodell**). Deutlich häufiger eingesetzt und besser untersucht wurde bisher laut Cantú-Paz (1998) das Migrationsmodell. Es wurden inzwischen auch Mischformen entwickelt, welche aber hier aufgrund des thematischen Umfanges nicht weiter betrachtet werden.

Das Migrationsmodell, entwickelt in den grundlegenden Arbeiten von Pettey u. a. (1987) und Tanese (1987), basiert auf der Idee, dass jeder Prozessor eine eigene Population hat und diese unabhängig von anderen Populationen entwickelt. Von Zeit zu Zeit werden jedoch Individuen zwischen den Populationen ausgetauscht (siehe Abbildung 2.8). Damit ist die Heterogenität der Lösungsansätze gewährleistet, da die Populationen größtenteils unabhängig sind. Die Gefahr, dass der Optimierungslauf in einem suboptimalen, lokalen Minimum endet, ist daher deutlich gemindert. Sollte sich dennoch eine Population durch ein lokales Minimum nicht mehr weiterentwickeln können, geben neue, von außen hinzukommende, Individuen von anderen Populationen neue Optimierungsimpulse. Tatsächlich stellte Cantú-Paz (1998) in seiner Untersuchung fest, dass das Migrationsmodell in der Regel schneller Lösungen findet und weniger zu suboptimalen Lösungen neigt als ein serieller GA. Da die Subpopulationen des Migrationsmodells jedoch meistens kleiner sind, werden Defizite bei der Feineinstellung der Lösung sichtbar.

Sowohl Nissen (1994) als auch Cantú-Paz (1998) betonen in ihren Übersichtsarbeiten ausdrücklich, dass es bislang keine allgemeingültigen Parametrisierungen des GAs gibt, son-

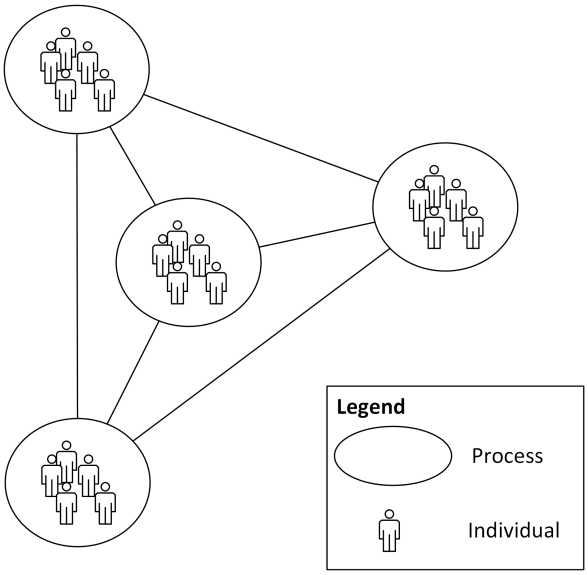

Abbildung 2.8.: Migrationsmodell der verteilten Genetischen Algorithmen

dern diese im hohen Maße von der Art des Problems abhängen und daher üblicherweise empirisch bestimmt werden müssen. Nissen (1994) weist darauf hin, dass darüber hinaus folgende Aspekten bei dem Migrationsmodell definiert werden müssen:

1. **Synchroner oder asynchroner Austausch:** Es ist zu definieren, ob Individuen gleichzeitig zwischen zwei Populationen ausgetauscht werden, oder sie einen unilateralen Transfer von Individuen durchführen.
2. **Häufigkeit des Austausches:** Es könnte zum Beispiel alle n Iterationen ein Austausch stattfinden, oder durch bestimmte Ereignisse ausgelöst werden.
3. **Globaler oder nachbarschaftlicher Austausch:** Es ist zu klären, ob Individuen nur zwischen Nachbarn ausgetauscht werden und wie diese Nachbarschaft definiert ist. Oder ob der Austausch zwischen zufälligen globalen Populationen stattfindet.
4. **Emigration oder Immigration:** Bei der Emigration gehen die transferierten Individuen ihrer alten Population verloren, während sie bei der Immigration erhalten bleiben.
5. **Anzahl der ausgetauschten Individuen und Art des Austausches:** Neben der reinen Anzahl der auszutauschenden Individuen muss ebenfalls festgelegt werden, welche Individuen ausgetauscht werden. Neben der zufälligen Auswahl könnten auch immer die besten n Individuen ausgetauscht werden.

Das zweite Modell, welches in Abbildung 2.9 skizziert ist, wird als Diffusionsmodell bezeichnet. Hier hält jeder Prozess nur ein Individuum (*Fine-Grained-GA*, siehe Abbildung 2.9) oder

ist für die Ausführung eines bestimmten Operation zuständig (*Master-Slave-GA*), während hier nur eine einzige globale Population vorliegt.

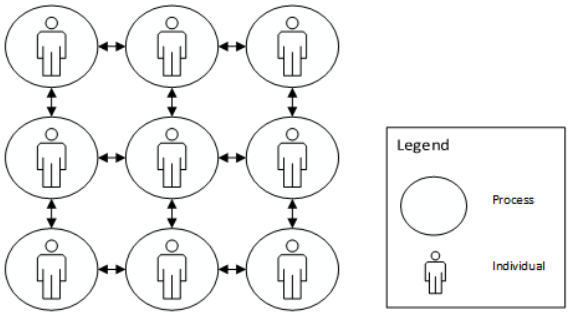

Abbildung 2.9.: Diffusionsmodell der verteilten Genetischen Algorithmen

Der Fine-Grained-Ansatz basiert auf einer räumlichen Ordnung der Individuen, die mit ihren Nachbarn Crossover durchführen. In der Literatur wurden sowohl zwei- als auch mehrdimensionale Nachbarschaften in Form von Hyper-Cubes untersucht. Ob diese Organisationsform jedoch Vorteile gegenüber anderen hat, konnte noch nicht abschließend geklärt werden. Es deutet jedoch vieles darauf hin, dass diese Struktur sehr gute Leistungsmerkmale bei massiv-parallelen Simulationen mit einer Vielzahl an verfügbaren Prozessoren hat. Begrenzender Faktor ist dabei jedoch die Kommunikationsbandbreite (vergleiche dazu Cantú-Paz (1998)).

In der Master-Slave-Organisation läuft ein Großteil des GA in einem einzigen Prozessor ab. Einzelne Funktionalitäten, wie die Ermittlung der Fitness werden auf Sub-Prozessoren verlagert. Vorteil ist hier, dass die Funktion des ursprünglichen Genetischen Algorithmus nicht verändert wird. Es ist gleichwohl abzuwägen zwischen dem erhöhten Kommunikationsaufwand und dem Vorteil paralleler Berechnungen. Daher sind auch nur aufwändige Berechnungen der Fitnessfunktionen für diese Form geeignet, während bei einfachen Operationen, wie Crossover oder Mutation, der Kommunikationsaufwand die Vorteile der parallelisierten Verarbeitung deutlich übersteigt.

2.3. Verteilte Systeme

Laut Tanenbaum und Van Steen (2002) ist ein verteiltes System eine Ansammlung unabhängiger Computer, die dem Benutzer als ein einziges System erscheinen. Die Einzelsysteme haben keinen gemeinsamen Speicher und kommunizieren über ein Netzwerk mittels Nachrichten miteinander. Dem Benutzer gegenüber erscheint das System durch die Software als eine Einheit, während die Hardware physisch getrennt ist.

Diese Art der Systemorganisation hat eine Vielzahl von Vorteilen. So ist mit ihnen echte Nebenläufigkeit realisiert und es ist durch das Hinzufügen von zusätzlichen Einzelsystemen in der Regel skalierbar. Auch kann durch Redundanz und Replikation eine erhöhte Ausfallsicherheit gewährleistet werden. Es gibt jedoch auch deutliche Nachteile. So ist die Detektion von Fehlerzuständen, wie Ausfällen, zuweilen schwierig. Ferner gibt es keine gemeinsame Zeitbasis und Prozesse können unterschiedlich schnell bearbeitet werden. Das System ist Nicht-Deterministisch und muss daher entsprechend ausgelegt werden. Die Kommunikation über Nachrichten kann lange dauern oder fehleranfällig sein. Jedoch überwiegen für viele Anwendung, wie das World Wide Web oder SETI@home[10], die Vorteile der Verteilung. Sei es, weil die Zuverlässigkeit und Ausfallsicherheit eines unitären Systems nicht ausreicht, oder das anfallende Datenvolumen es erforderlich macht.

Anwendungen, die auf verteilten Systemen laufen, werden *Verteilte Anwendungen* genannt. Zwischen der verteilten Anwendung und dem lokalen Betriebssystem liegt in den meisten Fällen eine weitere Softwareschicht, die *Middleware*. Sie dient als Framework für die Verteilung und sorgt maßgeblich für die einheitliche Erscheinung der Software für den Benutzer.

Ein typisches Beispiel für eine Anwendung, die sich von einem zentralen Service zu einer verteilten Anwendung wandelte, ist der Domain-Name-Service (DNS) mit dem die Adressauflösung von IP-Adressen in für Menschen leicht lesbare, semantische Adressen - dem Uniform Resource Locator (URL) - vorgenommen wird. Mit dem Übergang des ARPANET in das World Wide Web stellte sich heraus, dass das bisherige Konzept, eine zentral verteilte und lokal abgespeicherte Adresstabelle zum Mapping von Namen auf numerische Adressen einzusetzen, nicht mehr praktikabel war. Die Datei *hosts.txt* wurde manuell und zentral gepflegt und konnte nicht mehr mit der schnellen Entwicklung des WWW mithalten. Mockapetris (2003) erstellte daher mehrere Request for Comments (RFC) zum Aufbau eines verteilten Systems zur Namensauflösung. Die Struktur des Systems ist baumförmig hierachisch organisiert, wobei jedes Servercluster für eine eigene Zone, beziehungsweise *label*, zuständig ist. Dieses System lässt sich damit dezentral verwalten und ist skalierbar.

[10] http://setiathome.ssl.berkeley.edu

2.3.1. Multi-Agenten-Systeme

Laut Wooldridge (2004) ist ein Agent ein Softwaremodul, welches zu autonomen Verhalten fähig ist und gleichzeitig mit anderen Agenten interagiert. Das autonome Verhalten basiert auf eigenständigen Aktionen in Wechselwirkung und Interaktion mit der Umgebung (oder Umwelt) des Agenten und den Zielen die dieser verfolgt. Die Interaktion mit anderen Agenten beschränkt sich jedoch nicht auf das bloße Austauschen von Daten, sondern auch auf abstraktere Formen sozialer Interaktion, wie Koordination und Kooperation. Ist mehr als ein Agent in einem System vorhanden und interagieren diese auf die oben beschriebene Weise, wird in der Regel von einem Multi-Agenten-System gesprochen (Abbildung 2.10).

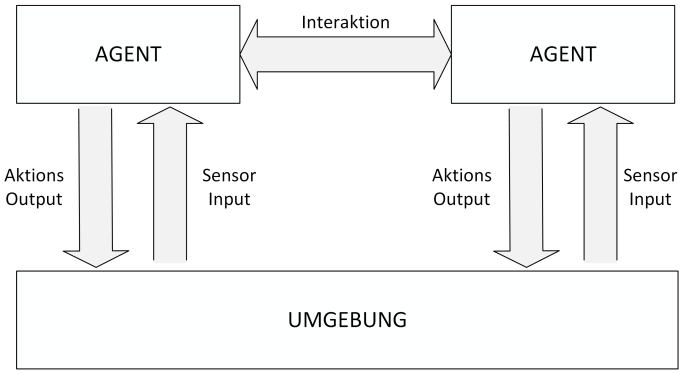

Abbildung 2.10.: Agenten mit Koordinations- und Interaktionspfaden

Agenten können laut Wooldridge (2004) im Allgemeinen in drei Klassen unterteilt werden, wobei es darüber hinaus eine Vielzahl von Unterklassen und Spezialfällen gibt.

- **Practical Reasoning Agents:** Practical Reasoning Agents nehmen ihre Umwelt wahr und versuchen sie proaktiv zu modifizieren. Dazu braucht der Agent Ziele, die dieser im *delibaration* genannten Schritt aufgrund seines Umgebungswissens auswählt. Ist ein Ziel ausgewählt, muss der Agent definieren, wie er das Ziel, und damit den gewünschten Umgebungszustand, erreichen kann. Dies wird im Allgemeinen als *Means-End Reasoning* bezeichnet, an dessen Ende ein Plan steht, wie das Ziel zu erreichen ist. Diese Pläne werden in der Regel durch den Programmierer des Agenten vorgegeben. In Abbildung 2.11 ist das Procedural Resoning System (PRS) skizziert. Das PRS ist eng verwandt mit dem *Belief-Desire-Intention*-Modell (BDI) zur Entwicklung von entscheidungsfähigen Agenten.
- **Reaktive Agenten:** Dieser Typ von Agenten basiert auf einer Entwicklung des Agentenparadigmas in den 1980er Jahren, weg von der symbolischen künstlichen Intelligenz durch explizite Repräsentation der Ziele, hin zu einem neuen Verständnis des Begriffes ”Umwelt”. Die Intelligenz in einem Softwaresystemen ist laut dieser Ansicht

mehr als ein Produkt der Interaktion zwischen Agent und Umwelt aufzufassen (Wooldridge (2004)). Die Agenten sind einfach aufgebaut, sodass sie schnell auf ihre Umwelt reagieren und interagieren können. Dadurch wird emergent (also ohne explizite Repräsentation der Ziele) ein viel komplexes intelligentes Verhalten, als in den Einzelkomponenten symbolisch programmiert, erreicht.

Reaktive Agenten haben meist nur ein Ziel, wodurch die Goal-Deliberation wegfällt. Daraus resultiert auch ihr Geschwindigkeitsvorteil gegenüber Practical Reasoning Agents. Neben der Geschwindigkeit und der Einfachheit ihres Aufbaus gibt es jedoch auch einige Nachteile. Das gravierendste Probleme ist, dass es keine allgemein anerkannte Methodik gibt, emergentes Verhalten ohne explizite Repräsentation zuverlässig zu schaffen. Dadurch wird der Designprozess teilweise erheblich erschwert. Andererseits sind diese Agenten nicht besonders flexibel, da sie nur einen kleinen Fundus an Verhaltensweisen haben.

- **Hybride Agenten:** Um die Vorteile beider Agententypen zu nutzen, können Hybriden als Mischform konstruiert werden. Dabei ist laut Wooldridge (2004) der naheliegende Ansatz das System einer Dekomposition zu unterziehen und in reaktive und proaktive Subsysteme aufzuteilen. Diese würden in interagierende Schichten unterteilt werden. Im Minimalfall wären dies zwei, eine proaktive und eine reaktive Schicht, aber es können theoretisch beliebige Grade an Aktivitätsschichten definiert werden.

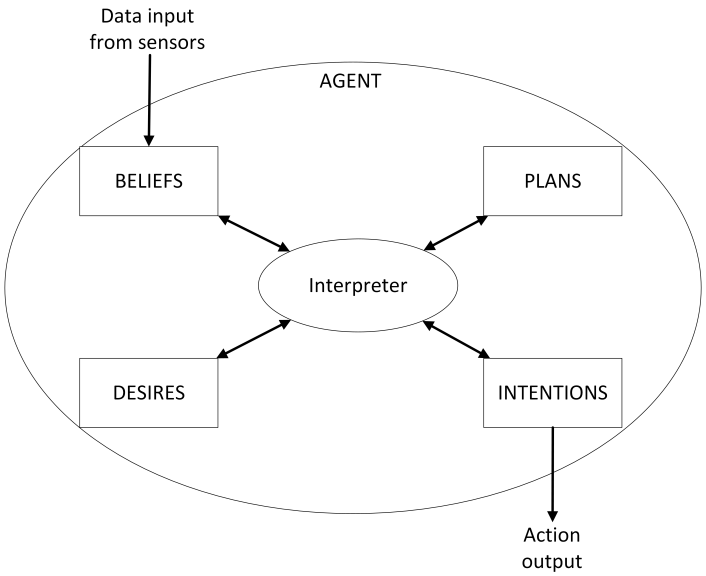

Abbildung 2.11.: Das Procedural Reasoning System (PRS) aus Wooldridge (2004)

2.3.2. Jadex Active Components

Jadex[11] (siehe Pokahr u. a. (2005)) ist eine an der Universität Hamburg entwickelte Multi-Agenten-Plattform, die aus dem verbreiteten *JADE*-MAS hervorging. Jadex erweiterte ursprünglich das FIPA[12]-konforme *JADE* um das BDI-Paradigma.

Aus dieser anfänglichen Erweiterung heraus ist Jadex inzwischen von einer Middleware zu einer Plattform für sogenannte Active Components geworden, die neben der Entwicklung von BDI-Agenten auch Micro-Agenten und andere Softwarekonzepte unterstützt. Micro-Agenten sind leichtgewichtige, serviceorientierte Agententypen, deren Programmablauf auf drei Phasen (Erzeugung, Laufzeit und Ende) beschränkt ist. Jadex beinhaltet darüber hinaus eine serviceorientierte Kommunikation, die auch in das BDI-Konzept eingebunden werden kann und somit eine Kommunikation über generische Interfaces zwischen den unterschiedlichen Agententypen ermöglicht. Ferner bietet Jadex umfangreiche Möglichkeiten und Unterstützungen zur asynchronen Programmierung mit sogenannten Futures und ResultListenern. Aufgrund dieser Eigenschaften ist Jadex im hohen Maße zur Umsetzung der oben definierten Systemziele und Anforderungen geeignet.

2.3.3. Agentenorientierte Modellierungsmethoden

Für die Anforderungsanalyse sowie das Design ist es unerlässlich, auf beschreibende Darstellungen und Ablaufdiagramme zurückgreifen zu können. In diesem Abschnitt sollen die in dieser Arbeit genutzten Methodiken vorgestellt werden. Teilweise sind für die eindeutige Beschreibung in der MAS-Domäne Erweiterungen erforderlich, welche ebenfalls hier eingeführt werden.

Tropos

Tropos[13] ist ein Agenten- und Goal-orientiertes Modellierungswerkzeug, erstmals vorgestellt durch Bresciani u. a. (2004), um die Softwareentwicklung von Multi-Agenten-Systemen zu unterstützen. Giorgini u. a. (2005) entwickelten Tropos weiter zu einer Methodik für die Goal-orientierte Anforderungsanalyse für BDI-Agenten.

Die Anforderungsanalyse mit Tropos ist in zwei Phasen geteilt: *Early Requirements* und *Late Requirements*. In der Early Requirements Phase werden Rollen und Aktoren definiert und deren Abhängigkeiten aufgezeigt. Es werden auch gemeinsame Ressourcen und Ziele

[11] www.activecomponents.org
[12] Foundation for Intelligent Physical Agents
[13] http://www.troposproject.org

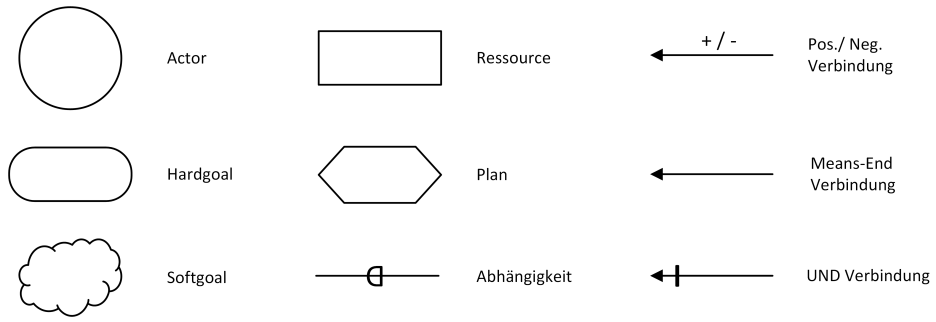

Abbildung 2.12.: Auswahl der wichtigsten Tropos-Designelemente

identifiziert. Ziele können sowohl konkrete Goals im Sinne des BDI-Paradigmas sein (siehe Abschnitt 2.3), aber auch sogenannte Soft-Goals. Dies sind Ziele, deren Erfüllung unter Umständen nicht genau modelliert, quantifiziert oder definiert werden kann. Ein Beispiel für ein Soft-Goal wäre ein *„Glücklicher Kunde“*. In Abbildung 2.12 sind die wichtigsten Designelemente von Tropos dargestellt. Neben den bereits eingeführten Aktoren, Soft- und Hardgoals sowie Ressourcen und Plänen existieren noch eine Reihe von Verbindungselementen. Abhängigkeiten werden durch Pfeile modelliert, wobei diese wie folgt zu interpretieren sind, wenn es sich bei s und t um Objekte (Ressourcen, Ziele oder Agenten) handelt: $s \rightarrow t \Leftrightarrow$ *„s hängt von t ab“*. Ziele können auch im Sinne der Verbindungstypisierung von Von Martial (1990) positiv (*verstärkend*) oder negativ (*abschwächend*) beeinflusst werden. Um ein Top-Level-Goal umzusetzen, also ein Hauptziel des Agenten, können mehrere Teilziele erforderlich sein. Diese werden mittels einer UND-Verknüpfung mit dem Top-Level-Goal verbunden. Pläne werden an Goals mit Means-End-Pfeilen angeschlossen, mit denen symbolisch ausgedrückt werden soll, dass der betreffende Plan eine mögliche Auswahl für das Goal ist.

In der Abbildung 2.13 wird exemplarisch die dritte Phase des Tropos-Designprozesses für ein selbst entwickeltes, einfaches Kunden/Erzeuger-System dargestellt. Dabei wird eine Dekomposition des Systems durchgeführt und auch technische Anforderungen und Designmerkmale miteinbezogen.

Bei Tropos ist nur die Interaktion zwischen unterschiedlichen Agententypen berücksichtigt. In dieser Arbeit hat jedoch die Kommunikation gleichrangiger Agenten einen hohen Stellenwert, die mit den Methodiken von Tropos nicht ausreichend abgebildet werden kann. Daher wird im Anhang B eine Methode vorgestellt, die von Tropos beeinflusst ist, sich jedoch zunehmend der graphischen Dokumentation der Agenten zuwendet.

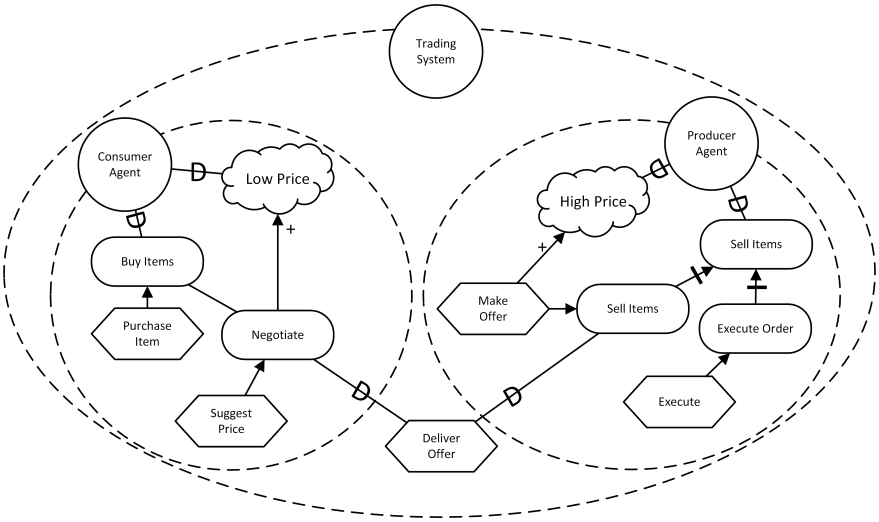

Abbildung 2.13.: Architecural Design eines Producer/Consumer-Systems

Temporallogische Modellierung von MAS

Eine temporallogische Beschreibungsmöglichkeit für MAS wird von Xing und Singh (2003) vorgestellt. Dabei wird, ausgehend von deklarativen Zustandsdiagrammen, eine *Computation Tree Logic*(CTL)-Struktur abgeleitet.

Die CTL ist eine temporale, formale & deklarative Semantik zur Spezifikation und Verifikation von Systemen und ist eine Untermenge der generalisierten CTL*, beziehungsweise eine Erweiterung der Aussagenlogik. Mit dieser Semantik werden Systemzustände und ihre Übergänge als Baumstruktur beschrieben. Dazu werden Boolsche Operatoren und Temporaloperatoren verwendet um aussagenlogische Formeln aufzustellen. Die detaillierte Beschreibung aller Operatoren würde jedoch den Rahmen dieser Arbeit übersteigen. Daher wird hier nur die Baumstruktur eingeführt, da diese im Design genutzt wird. Mit dieser können nicht-triviale Systemzustände und Übergänge übersichtlich dargestellt werden.

Der Methodik von Xing und Singh (2003) besteht aus drei einfachen Schritten: Zuerst werden mögliche Systemzustände identifiziert. Durch die Definition von Übergängen (State Transitions) werden die Zustände miteinander zu einem deklarativen Zustandsdiagrammen verbunden. Damit kann die Komposition des Systems verdeutlicht werden. Im dritten Schritt wird das Zustandsdiagramm in CTL umgesetzt.

Das Vorgehen kann an einem einfachen Beispiel demonstriert werden. Ein System soll eine einfache Serveranwendung bereitstellen, die auf einen externen Request eine Berechnung durchführt und das Ergebnis zurückliefert. Entsprechend des Vorgehensmodells von Xing und Singh (2003) werden Grundzustände und Übergänge identifiziert (Abbildung 2.14) und

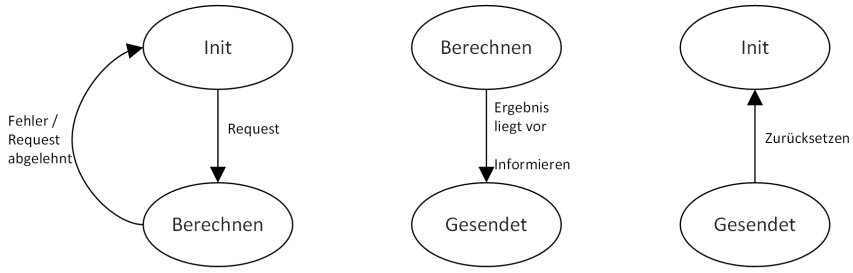

Abbildung 2.14.: Dekomposition der Systemzustände mit Übergängen

durch Pfeile dargestellt. Der Übergang vom Zustand s in den Zustand t wird dementsprechend $s \rightarrow t$ notiert. Diese Zustandsmuster werden darauf folgend zu einem Gesamtsystem zusammengefügt, sodass eine Zustandsdiagramm, welches die Komposition des Systems repräsentiert, entsteht (Abbildung 2.15).

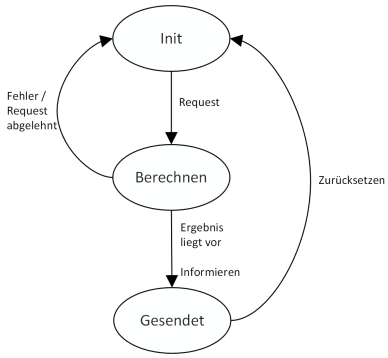

Abbildung 2.15.: Zustandsdiagramm nach der Komposition der einzelnen Zustände

Um das Zustandsdiagramm in eine CTL-Struktur zu transformieren, müssen die Zustandsübergänge bewertet werden. *Fehler, Ergebnis liegt vor, Ergebnis senden* und *Zurücksetzen* sind *internal events*. Diese Zustandsänderungen laufen dementsprechend ohne äußere Einflüsse ab. *Request* ist ein *external event*, der von einem externen System ausgelöst wird. Alle Zustandsübergänge werden ebenfalls in Zustandsellipsen überführt und, sofern es sich um *external events* handelt, mit einem Pfeil zur Selbstreferenz $s \rightarrow s$ versehen. Das Ergebnis ist in Abbildung 2.16 dargestellt.

In Multi-Agenten-Systemen mit Interaktionen sind *external events* ein häufig eingesetztes Mittel, sodass durch eine einfache Selbstreferenz nicht in jedem Falle deutlich ist, mit wem interagiert wird und welche Inhalte kommuniziert werden. Daher wird hier das Schema um zwei Elemente erweitert. Eine Figur stellt eine andere Domäne dar, zum Beispiel einen anderen Agenten oder einen Client, mit der kommuniziert wird. Kommunikationsverbindungen

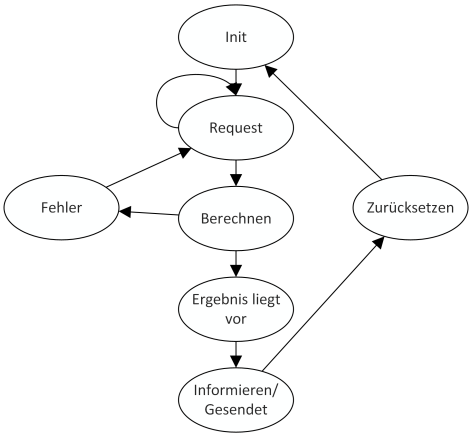

Abbildung 2.16.: CTL-Struktur der Serveranwendung

mit anderen Domänen werden durch gestrichelte Linien angezeigt. In Abbildung 2.17 wurde das Schema aus Abbildung 2.16 um beide Elemente erweitert. Sei x ein Agent und dieser hat einen Task t der Bezeichnung *ergebnisliegtvor(x,t)* erfolgreich abgeschlossen und dessen Ergebnisvektor $\pi(\vec{V})$ ist WAHR, dann informiert x einen Agenten y (andere Domäne) über $\pi(\vec{V})$. Die Selbstreferenzierung $t \rightarrow t$ wird dementsprechend durch eine Referenz

$$\textit{Informieren(x,y,}\pi\textit{)} := \forall\ \vec{v},\ t.\ AG[\textit{ergebnisliegtvor(x,t)} \rightarrow AF[\textit{informieren}(x, y, \pi(\vec{v}))]] \tag{2.12}$$

ersetzt. Die Abkürzungen und Symbole bedeutet im CTL-Kontext:

- AG: *auf allen Pfaden gilt in jedem Zustand* ϕ
- AF: *es wird immer ein Zustand erreicht, der* ϕ *erfüllt*

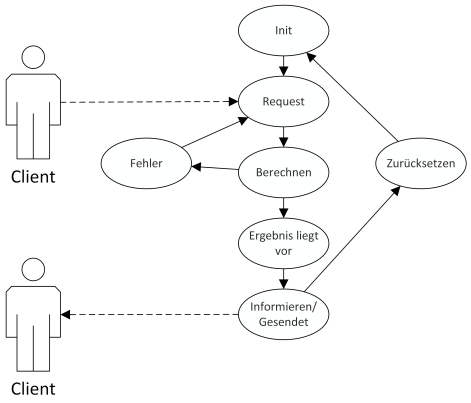

Abbildung 2.17.: Erweiterung des CTL-Beispiels um Kommunikationspfaden und Agentendomänen

2.3.4. MAS-Simulationen im Energiesektor

Das MAS-Softwareparadigma wird von einer Vielzahl von Arbeitsgruppen genutzt, um Simulationen im Smart Grid Umfeld durchzuführen. Multi-Agenten-Systeme können aufgrund ihrer Verteilungs- und Autonomieeigenschaften relativ analog zu den tatsächlichen Gegebenheiten im Netz entwickelt werden. Beinahe allen Ansätzen ist gemeinsam, dass ein Agent die Repräsentation eines Erzeugers, eines Verbrauchers oder einer Netzentität ist (beispielsweise Leitstelle oder Transformatorstation). Ziel ist zumeist, die Last an die Erzeugung mehr oder weniger unmittelbar anzugleichen. Die Zeithorizonte der Regelung erstrecken sich dabei von Planungen des nächsten Tages mit dem *PowerMatcher* von Kok u. a. (2005) bis nahezu Echtzeitanwendungen bei Linnenberg u. a. (2011).

Die Simulationssysteme unterscheiden sich hauptsächlich in der Art des Demand-Side-Managements, bei denen laut Sonnenschein u. a. (2010) indirekte und direkte Kontrollmöglichkeiten differenziert werden können. Während bei der direkten Kontrolle in erster Linie die verwendeten Protokolle interessant sind, können bei der indirekten Steuerung noch weitere Typen unterschieden werden, sodass sich folgendes Schema ergibt:

- Direkte Steuerung
- Indirekte Steuerung

  Preissignale

  Auktionsbasierte Ansätze

  Optimierung der Verbraucherseite

Ein Beispiel für die direkte Steuerung von Lasten ist die Arbeit von Lünsdorf und Sonnenschein (2009), bei der die Verbundsteuerung von Lasten untersucht wurde. Vorteilhaft bei diesen Systemtypen ist die verlässliche Systemantwort bei den definierten Signalen, da die angeschlossenen Verbraucher sich nach der jeweiligen Spezifikation verhalten müssen. Die Kontrolle über die Steuerung liegt alleine bei dem betreffenden *Service Provider* oder *Operator*. Diese Betriebsweise impliziert jedoch auch einige harte Anforderungen, so zum Beispiel die Modellierung von Anforderungen der Verbraucherseite, etwa unter welchen Bedingungen die Lasten zentral geschaltet werden dürfen. Die Leitstelle benötigt zusätzlich genaue Informationen darüber, welche Lasten in welchem Zeithorizont kontrolliert werden können, um abschätzen zu können, welche Effekte durch die Steuersignale hervorgerufen werden. Aufgrund der Komplexität und Heterogenität von Haushaltsgeräten, erscheint dieser Aufwand eher für industrielle Großverbraucher interessant.

Im Bereich der indirekten Steuerung von Lasten sind deutlich divergierende Ansätze untersucht worden. Nahe an der direkten Steuerung ist die Kontrolle mit Preissignalen, beispielsweise bei Ramchurn u. a. (2011) und wie sie im OpenADR-System oder in den Smart-Metern spezifiziert sind. Diese MAS sind stark an der energietechnischen Praxis orientiert. Durch die einfache und für den Kunden nachvollziehbare Flexibilisierung der Tarife über den Tag und die gewohnte top-down Kommunikation von Preissignalen durch die zentrale Leitwarte ist dieser Ansatz auch eine konsequente Weiterentwicklung des bestehenden Tarifmodells. Allerdings können bei automatisierten Verbrauchern durch simple Preisminimierungen in der Summe Lastspitzen auftreten, die durch eine entsprechende Echtzeitüberwachung und andere Maßnahmen durch die Leitwarte vermieden werden müssten. Daher sollten in Zukunft Verfahren eingesetzt werden, die Netzrestriktionen und Nebenbedingungen implizit abbilden.

Verbreitet sind auch Auktionsansätze, bei denen entweder die Verbraucher direkt mit den DERs oder über Aggregationsagenten verhandeln. Diesen Ansatz verfolgen beispielsweise Kok u. a. (2005), Linnenberg u. a. (2011) und Beer u. a. (2011). Dabei werden Erzeugung und Last in einem bilateralen Verfahren aufeinander abgestimmt und Preise ausgehandelt. Eine Schwierigkeit hierbei ist die Festlegung von Preisschwellen und Akzeptanzniveaus a priori für jede Kleinanlage durch die Erzeuger und die Kunden.

In der letzten Klasse wird eine direkte Optimierung der Verbraucherseite durchgeführt, um einen, je nach zugrundeliegendem Geschäfts- oder Betriebsmodell, optimalen Nutzungsgrad zu generieren. Kamper (2010) und Logenthiran u. a. (2012) beschreiben dabei exemplarische Anwendungsfälle, welche recht spezifisch in den Betriebsmodellen sind. Kamper (2010) beschreibt zusätzlich für eine dezentrale Anwendung ein generische Verbrauchermodell.

3. Anforderungen und Analyse

In diesem Kapitel soll die Funktion des Systems beschrieben und die daraus resultierenden Anforderungen analysiert werden. Dazu wird zuerst das System funktional beschrieben und die Forschungsfragen definiert. Daraufhin sollen energiewirtschaftliche und technische Anforderungen an die Architektur des Systems identifiziert werden. Die zu entwickelnde Architektur soll ferner als Simulationssystem implementiert werden, um Anwendungsfälle zu demonstrieren und die Forschungsfragen untersuchen zu können.

Zuletzt werden in der technischen Analyse Aspekte wie die Plattform und die zu implementierenden Optimierungsalgorithmen nach eingehender Prüfung ausgewählt.

3.1. Systembeschreibung

Das System soll die Flexibilität von Energieverbrauchern nutzen, um diese zu optimieren und Fahrpläne für die Verbraucher zu erzeugen, um unter anderem dem Wandel, von der Erzeugung basierend auf der Nachfrage hin zur Erzeugung nach Verfügbarkeit, gerecht zu werden. Dazu soll anhand der Verschiebungspotentiale eine verteilte Optimierung mittels einer lose gekoppelten und leichtgewichtigen Architektur durchgeführt werden. Dabei seht die Frage im Fokus, ob das System dazu genutzt werden kann, typische Anwendungsfälle auf der Verbraucherseite abzubilden und ob durch die Anwendung weicher Bedingungen, wie zum Beispiel von Preissignalen und Marktdaten, ein deterministisches Verhalten generiert werden kann. Dabei soll bestenfalls auf die bestehende, oder sich entwickelnde Infrastruktur zurückgegriffen werden und die zunehmende Verbreitung von vernetzten Geräten im *Internet der Dinge* und im *Internet der Energie* ausgenutzt werden, um dynamische Optimierungsnetze zu erstellen. Diese sollen flexibel, skalierbar und spezifisch auf Anforderungen und Geschäftsmodelle des Smart Grids reagieren können. Dazu soll eine generische High-Level-Architektur entwickelt werden, die ohne spezifisches Domänenwissen Optimierungen durchführen kann.

3.2. Energiewirtschaftliche Anforderungen

Das System sich sollte möglichst flexibel und generisch auf eine Reihe von zu definierenden Anwendungsfälle adaptieren können. Die Zeitbasis soll je nach Anwendung definierbar sein, solange sich diese auf eine diskrete Zeitbasis abbilden lässt. Genauso sollen die Leistungsniveaus beliebige Größen annehmen können. Das zugrundeliegende Energiewirtschafts- und Handelsmodell soll in der Lage sein, eine Vielzahl von Geschäftsmodellen und Anwendungsfällen adäquat abzubilden und diese ohne zusätzliche externe Anforderungen oder Regeln zu planbaren Ergebnissen führen.

Bei den möglichen Anwendungsfällen sollten folgende, teilweise zusammenhängende, oder aufeinander abbildbare Szenarien unterstützt werden:

- **Verbesserte Nutzung von systemimmanenten Energieressourcen unter Berücksichtigung externer Ressourcen.** Dieser Anwendungsfall beschreibt eine Liegenschaft, die interne Energieressourcen besitzt und diese mit Demand-Side-Management durch eigene Verbraucher möglichst gut ausnutzen möchte. Können die internen Energieressourcen den Bedarf nicht decken, stehen teure externe Energieversorger zur Verfügung. Dieser Anwendungsfall wird beispielsweise von Logenthiran und Srinivasan (2011) betrachtet.
- **Dynamische Mehrtarifmodelle, basierend auf Marktdaten (Flexible Hoch- und Nieder-/Nachtstromtarife).** Wie in der Arbeit von Ulbig und Andersson (2010) beschrieben, ist zuweilen eine Vereinfachung und Abstraktion des eigentlichen Marktes durch Tarifmodelle vonnöten, um den Kunden einen Zugang ohne tiefgreifenden Zeit- und Einarbeitungseinsatz zu ermöglichen, um so die Akzeptanz zu erhöhen.
- **Mini- und Mikrobörsen für Kleinverbraucher und -erzeuger.** Der steigenden Durchdringung der Erzeugerseite von volatilen und dezentralen Kraftwerken sind klassische Märkte laut Kok u. a. (2010) noch nicht angepasst und daher würde mittelfristig ein Bedarf an kleinen, automatisierten und dezentralen Energiemärkten entstehen. Auf diesen Märkten könnten Aktoren anhand von Prognosen ihre Erzeugung, respektive ihren Verbrauch, abschätzen und automatisiert ihren Bedarf decken.
- **Teilnahme von (industriellen) Verbraucherverbänden an größeren Märkten.** Als Beispiel dient hier das e-harbours Projekt (siehe Verbeeck u. a. (2014)), in dem unter anderen Industrieanlagen in Häfen mit intelligenten Steuerungen zur optimierten Planung ihrer Bedarfe ausgerüstet wurden. In diesem Anwendungsfall könnten auch Modelle aus dem ersten Punkt berücksichtigt werden.

Um all die oben beschrieben Anwendungsdomänen abbilden zu können, benötigt es ein adäquates Marktmodell. Ausgehend von der Erzeugerseite liegen als grundlegende Daten sowohl das Einspeisepotential, als auch die (kalkulatorischen) Kosten für den Betrieb vor.

Auf der Verbraucherseite sind dementsprechend Preisrahmen und (heuristische) Prognosen über den Verbrauch, sowie teilweise Verbrauchsplanungen vorhanden. Wie bereits im Grundlagenkapitel erläutert wurde, ist der rechtliche Stand, sowie die derzeitige technische Grundlage vieler Arbeiten und Projekte, ein top-down orientiertes, flexibles Tarifmodell. Im eEnergy-Projekt *MeRegio* (siehe Frey (2013)) wurden Preisstufen eines flexiblen Tarifs über ein im Haus installiertes Interface farblich dargestellt. Nach drei Monaten orientierten jedoch nur noch 2% bis 12% der Kunden ihren Verbrauch anhand dieses Signals. Beim Abschluss des eEnergy-Projektes adressierte Breuer (2013) von RWE daher den Bedarf an regionalen Energiemärkten für die Endkunden, um diese besser und automatisiert einzubinden.

Das implementierte Marktmodell sollte daher Börsencharakter haben. Basierend auf dem Energieangebot können sich Verbraucher anhand ihrer Parameter entsprechende Leistungen, oder Optionen darauf (Termingeschäfte wie z.B. Regelenergie), erkaufen.

In der Literatur wird zuweilen eine vereinfachte Variante eines Börsenmodells angewendet, bei dem keine Mengenberücksichtigung stattfindet, sondern den Verbrauchern einzig ein flexibler Tarif angeboten wird. Bei Ramchurn u. a. (2011) wird allen Agenten die gleiche Preiskurve offeriert. Eine für Agenten in einem solchen System vorteilhafte Strategie wäre es, eine Minimumsuche durchzuführen. Da alle Agenten zumeist unabhängig voneinander agieren können, kann dies zu starken Lastspitzen im System führen, wenn viele Agenten das gleiche Minimum finden. Von Kamper (2010) wird dieses Verhalten als *Lawineneffekt* bezeichnet. Ramchurn u. a. (2011) fordern daher, dass sich nicht alle Agenten gleichzeitig optimieren dürfen, damit der Stromanbieter rechtzeitig durch neue Preiskurven gegensteuern kann. Dies ist jedoch eine harte Anforderung, die zur korrekten Funktionalität des Systems eingehalten werden muss, in der Realität aber nicht immer garantiert werden kann.

Da der vorletzte Punkt durch Kok u. a. (2005) mit dem *Powermatcher*-MAS bereits umfangreich untersucht wurde und der letzte Punkt meist durch die speziellen Gegebenheiten und Anforderungen der Industrie häufig nicht zu übertragbaren Ergebnissen führt, werden in dieser Arbeit insbesondere die ersten beiden Szenarien als Anwendungsfälle weiter untersucht.

Für das Demand-Side-Management sind die Prognose und Planung des Energieverbrauchs entscheidend. Basierend auf den oben definierten Anwendungsdomänen und deren Planungshorizonte ergeben sich demnach Anforderungen an die Zeitbasis (siehe auch Kapitel 2.1.4). Diese sollte je nach Anwendungsdomäne frei wählbar sein. Typische diskrete Zeitbasen sind die Viertelstundenbasis mit 96 Zeitschritten pro Tag (für Day-Ahead-Anwendungen) und die Minutenbasis mit 1440 Zeitschritten pro Tag (Minutenreserve). Für Anwendungen im Primär- und Sekundärregelleistungsbereich wäre eine Zeitbasis auf Sekundenschritten erforderlich.

Ein wichtiger Aspekt für die Anwendung ist die Berücksichtigung von Must-Run-Kriterien. Mit Must-Run werden Zustände beschrieben, in denen die Verbraucher unabhängig von der

derzeitigen Marktsituation oder dem gewählten Fahrplan, in der Lage sind, ihr Laufzeitverhalten selbst zu bestimmen, wenn die Situation es erfordert. Ein Nutzer muss beispielsweise bei einem Haushaltsgerät immer die Möglichkeit haben, es sofort zu starten. Genauso muss eine automatisierte Heizung in der Lage sein selbstständig Heizimpulse anzuregen, wenn Temperaturgrenzen verletzt wurden, auch wenn der Fahrplan noch keine Heizimpulse vorsieht.

In Tabelle 3.1 sind die wichtigsten Energiewirtschaftlichen Anforderungen noch einmal zusammenfasst dargestellt. Dabei wurde eine Abschätzung der Wichtigkeit der Anforderungen vorgenommen.

Tabelle 3.1.: Energiewirtschaftliche Anforderungen

Bezeichnung	**Priorität**
Entwicklung eines generischen Wirtschaftsmodells	hoch
Berücksichtigung verschiedener Zeithorizonte	hoch
Berücksichtigung von Must-Run-Kriterien	mittel

3.3. Eigenschaften der Architektur des Energiesystems

Im Bereich der Eigenschaften und Anforderungen des zu entwickelnden Energiesystems sind zwei große Bereiche zu berücksichtigen. Zum einen die architektonischen Eigenschaften an das Energiesystem als solches und zum anderen die damit unmittelbar zusammenhängenden Anforderungen an die Optimierung und die Optimierungsalgorithmen.

Die Architektur des Energiesystems sollte aus mehreren kombinierbaren Komponenten bestehen, die die jeweiligen Systemfunktionalitäten abbilden.

- **Verbraucherkomponente:** Die Verbraucherkomponente repräsentiert die flexiblen elektrischen Verbraucher. Sie muss neben der Sensorik und Aktoren zur Überwachung und Steuerung des zugeordneten elektrischen Verbrauchers auch ein Simulationsmodell zur Vorhersage des Energiebedarf und Informationen über die eigenen Lastprofile haben.
- **Infrastukturkomponente:** Die Infrastrukturkomponente ist die Schnittstelle zwischen Steuerungskomponenten und Verbraucherkomponenten und bieten Services zum Suchen, Auffinden und Buchen von Komponenten nach definierbaren Kriterien.
- **Steuerungskomponente:** Die Steuerungskomponente stellt die Energieservices für das Smart Grid zur Verfügung indem Verbraucher zu Virtuellen Load Plants aggregiert werden.

- **Optimierungskomponente:** Die Optimierungskomponenten sorgen dafür, dass die Fahrpläne der Verbraucher dahingehend angepasst werden, dass die von der Steuerungskomponente vorgegebenen Geschäftsmodelle umgesetzt werden können.
- **Service Bus:** Zur Bereitstellung der Kommunikation zwischen den Komponenten wird ein Service Bus benötigt.

Alle Entitäten im System sollen in der Lage sein, die Rollen der Verbraucherkomponente, der Optimierungskomponente und der Steuerungskomponente übernehmen zu können, um spontane Netzwerke für Energieservices bilden zu können. Dadurch können zum einen eine große Bandbreite von Geschäftsmodellen abgebildet werden, andererseits wird die Abhängigkeit von externen Dienstleistern gesenkt, da die Rollen systemimmanent wahrgenommen werden können. Diese Dynamik erlaubt es, auf kurzfristige Entwicklungen flexibel zu reagieren.

Die Skalierbarkeit des Systems muss in zwei Dimensionen ermöglicht werden. In der ersten Dimension soll eine große Anzahl an Verbraucher-Agenten an dem System teilnehmen können. Dazu ist es erforderlich, obgleich die Anzahl möglicher Lösungen für das Optimierungsproblem exponentiell oder darüber hinaus wachsen könnte, dass der Zeit- und Speicherbedarf eines jeden Optimierungsschrittes idealerweise nur linear oder nahezu linear steigen sollte. Aus dieser Skalierbarkeitsanforderung ergibt sich dementsprechend auch eine physikalische Grenze für die Größe des Systems, abgeleitet aus dem für die Nutzer und der Anwendung akzeptablen Speicher- und Zeitbedarf.

Die zweite Dimension der Skalierbarkeit bezieht sich auf die Abbildung der Verbraucher. Im Sinne der Consumer-Domain des NIST Framework soll das Energy Management System auch dazu dienen, die darunterliegende physikalische Ebene zu maskieren und zu abstrahieren. Der gewählte Ansatz zur Implementierung der Verbraucher sollte dementsprechend in der Lage sein, sowohl die Geräteebene als auch aggregierte Lasten abzubilden.

Im engen Zusammenhang mit der Skalierbarkeit, aber auch in Hinblick auf die Nutzung von *embedded systems*, steht die Anforderung, dass die oben beschriebenen Systemkomponenten bezüglich der verwendeten Hardware möglichst adaptiv sind. Dazu gehört insbesondere, dass die Optimierung in einfache Teilschritte aufteilbar ist und die Anzahl der durchzuführenden Optimierungsschritte von der Plattform je nach Systemressourcen lokal angepasst werden kann, ohne das Gesamtsystem zu beeinflussen. Diese Anforderungen an die Systemeigenschaften, wie Skalierung und Robustheit, können praktisch nur von einem verteilten Optimierungssystem erfüllt werden.

In Tabelle 3.2 sind die wichtigsten technischen Anforderungen noch einmal zusammenfasst dargestellt. Dabei wurde eine Abschätzung der Wichtigkeit der Anforderungen vorgenommen.

Tabelle 3.2.: Anforderungen und Eigenschaften der Architektur des Energiesystems

Bezeichnung	Priorität
Entwicklung einer Service-basierten Architektur	hoch
Entwicklung eines generischen und skalierbaren Verbrauchermodells	hoch
Definition der Infrastrukturkomponente	mittel
Definition der Steuerkomponente	hoch
Identifikation von geeigneten Schnittstellen für die Kommunikation	hoch

3.4. Anforderungen an das Simulationssystem

Die entwickelte Architektur des Energiesystems soll als Simulationssystem umgesetzt werden, um die Funktionalität des Systems zu demonstrieren und die diesbezüglichen Forschungsfragen anhand von Anwendungsfällen in der Energiedomäne näher untersuchen zu können. Dazu wird eine Plattform benötigt, mit der ein skalierbares und verteiltes Multi-Agenten-System entwickelt werden kann und dass auch eine Service-basierte Kommunikation der Agenten untereinander ermöglicht.

Das Multi-Agenten-System soll Verbraucher aus dem Haushaltsdomäne repräsentieren, da Gewerbe- und Industrieanwendungen und -verbraucher meist sehr spezifisch sind. Ferner liegen mit der Smart-A Studie von Stamminger u. a. (2008) detaillierte statistische Daten über die Flexibilitätspotentiale und Verbreitung von Haushaltsverbrauchern vor. Besonders interessant, auch im Vergleich mit der Literatur, sind hierbei Day-Ahead- und Intraday-Simulationen auf Basis von 96 Zeitabschnitten an einem Tag.

Dabei soll zum einen Referenzimplementierung von programmgetriebenen Verbrauchern vorgenommen werden. Diese Geräte sollen anhand der Smart-A Studie statistisch parametrisiert werden, sodass ihre Startzeitpunkte und Flexibilitätspotentiale zufällig gewählt werden, um eine möglichst realitätsnahe Simulation gewährleisten.

Als zweiter Verbrauchertyp soll die exemplarische Anwendung eines vollautomatisierten Verbrauchers implementiert werden. Dazu bietet sich unter anderem Nachtspeicherheizungen an, deren thermische Simulation auf Basis realer Außentemperaturdaten durchgeführt werden kann.

Das zu implementierende System soll ferner über eine Möglichkeit zur Subsimulation verfügen, also die Verbraucher anhand der erstellten Fahrpläne parametrisieren und deren Betrieb simulieren. Dabei sollen verschiedene Ereignistypen auftreten können, die teilweise

neue Optimierungen nötig machen. Der erste Ergeignistyp betrifft die thermischen Verbraucher und soll einen Temperaturabfall unterhalb der Grenzwerte simulieren, beispielsweise durch eine offene Tür. Der Heizungsverbraucher ist so gezwungen, die Temperaturplanung wieder in den Temperaturkorridor zu bringen. Aufgrund der veränderten Fahrplansituation sollen sich alle Agenten danach neu optimieren. Der zweite Ereignistyp soll eine plötzliche Veränderung des Marktmodells simulieren. Dies könnte auftreten, wenn die Prognose über die verfügbare Energie unzutreffend war, weil beispielsweise aufgrund des Wetters weniger Windkraft oder Solarenergie verfügbar ist.

Zur Klärung der Forschungsfragen und um die Anwendbarkeit des Systems in Bezug auf Energiesimulationen zu demonstrieren, sollen die folgenden Szenarien als exemplarische Anwendungsfälle näher untersucht werden.

- **Day-Ahead-Optimierung:** In diesem Anwendungsfall soll die Fähigkeit des Systems untersucht werden, sich an gegebene Marktszenarien im Vorwege anzupassen.
- **Gezielte Lastverlagerung durch Verbundoptimierung:** Dieser Fall ist eine Erweiterung des ersten Anwendungsfalls. Es soll untersucht werden, ob das System mittels der weichen Preisgrenzen unter der Berücksichtigung von Must-Run-Kriterien in der Lage ist, auf neue Preisimpulse zu reagieren. Grundlage dabei soll ein Vergleich mit den Ergebnissen von Lünsdorf und Sonnenschein (2009) sein.
- **Untersuchung des Grenzverhaltens:** Aus wirtschaftlicher Sicht ist die Betrachtung von Grenzbereichen der Optimierung besonders interessant, wenn der Bedarf geplant werden muss. Es soll untersucht werden, wie sich die Optimierungsalgorithmen unter zunehmender Verknappung von Ressourcen verhalten.

Abschließend nach der Definition der zu untersuchenden Anwendungsfälle sind die Anforderungen an das Simulationssystem in Tabelle 3.3 zusammengefasst und in der Priorität geschätzt worden.

Tabelle 3.3.: Simulationsanforderungen

Bezeichnung	**Priorität**
Auswahl einer geeigneten Simulationsplattform	hoch
Implementierung von halbautomatischen Verbrauchern	hoch
Implementierung von vollautomatischen Verbrauchern	hoch
Implementierung von verteilten Optimierungsalgorithmen	hoch
Implementierung einer einfachen Subsimulation	mittel
Parametrisierung des Modells mit statistischen Daten	mittel

3.5. Anforderungen an die Optimierung

In diesem Abschnitt werden Optimierungsalgorithmen auf ihre Eignung bezüglichen der Problemstellung untersucht. Ferner werden Testcases zur Sicherstellung der korrekten logischen Funktionalität der Simulation definiert.

3.5.1. Mathematische Beschreibung des Optimierungsproblems

Das Optimierungsproblem der Minimierung des Gesamtpreises I wird formal beschrieben durch

$$I = min \sum_{t=0}^{T} \sum_{j=1}^{A} I_j(t) P_j(t) \quad \text{wobei} \quad \sum_{j=1}^{A} P_j(t) \stackrel{!}{=} L_N\{\tau_n\}(t) \tag{3.1}$$

T bezeichnet die maximale Anzahl der zu berücksichtigenden diskreten Zeitschritte und A für jeden Zeitpunkt t die Anzahl der Angebote auf dem Markt. Angebote sind definiert durch einen Angebotspreis $I_J(t)$ und einer Menge $m_J(t)$. $P_J(t)$ beschreibt die Nachfrage (benötigte Leistung) unter Berücksichtigung der Bedingung, dass $P_J(t) \leq m_J(t)$. Die nachgefragte Leistungssumme zum Zeitpunkt t liefert

$$L_N\{\tau_n\}(t) := L_N\{\tau_1, \tau_2, \ldots, \tau_N\}(t) = \sum_{m=0}^{M} L_m\{\tau_1, \tau_2, \ldots, \tau_{n_m}\}(t) \tag{3.2}$$

$$L_m\{\tau_1, \tau_2, \ldots, \tau_{n_m}\} = \sum_{i=0}^{n_m} \mathcal{L}_m(t - \tau_i), \tag{3.3}$$

wobei M die Anzahl der im System vorhandenen Verbraucher ist und $L_m(t)$ die Leistungsfunktionen der Verbraucher als Summe der benötigten Leistung $\mathcal{L}(t-\tau_i)$ über n_m beschreibt und damit die i.a. variable Anzahl der Startzeitpunkte N beschrieben wird durch

$$N = \sum_{m=0}^{M} n_m \tag{3.4}$$

wobei n_m die i.a. variable – abhängig von der Lage der Startzeitpunkte – Anzahl der Startzeitpunkte eines Devices zählt.

3.5.2. Auswahl der Optimierungsverfahren

An die zu verwendenden Optimierungsalgorithmen müssen spezifische Anforderungen gestellt werden, die im folgenden näher identifiziert wurden:

1. **Verteilbarkeit:** Die Algorithmen müssen sich, zur Effizienzsteigerung, durch einfache Service-basierte Ansätze auf mehrere Instanzen verteilen lassen, zum Beispiel durch den Austausch von Zwischenlösungen.
2. **Skalierbarkeit:** Innerhalb von physikalischen und logischen Grenzen sollte die Anzahl der beteiligten Optimierungsinstanzen unerheblich sein, ebenso wie die Größe des Optimierungsproblems.
3. **Adaptivität:** Die Algorithmen müssen sich in kleine Teilschritte unterteilen lassen, sodass verschiedene Systemtypen (*embedded devices* bis Workstations) in der Lage sind, sich im Rahmen ihrer Leistungskapazitäten an der Lösung zu beteiligen, ohne das Verfahren zu beeinträchtigen.
4. **Verlässlichkeit:** Die verwendeten Algorithmen müssen gültige Lösungen in akzeptabler Zeit generieren können, um die Funktionalität des Systems sicherzustellen, auch wenn noch nicht das Optimum gefunden wurde.

Zur Lösung des kombinatorischen Optimierungsproblems[1] kommen zwei Lösungsansätze infrage. Zum einen exakte Verfahren, wie die *vollständige Enumeration*, *branch and cut* oder *branch and bound*, die die optimale Lösung liefern können, als auch Metaheuristiken, die darauf spezialisiert sind, schnell gute Lösungen zu finden, aber nicht unbedingt die beste Lösung.

Laut Merz (2003) kommt bereits für kleine Problemstellungen die vollständige Enumeration nicht infrage, da viele kombinatorische Optimierungsprobleme in der Komplexitätsklasse *NP-schwer* liegen und damit die Zahl der möglichen Lösungen exponentiell oder überexponentiell ansteigen. Ebenso seien exakte Verfahren wie *branch and cut* und *branch and bound* zwar in der Lage, die optimale Lösung zu finden und sind auch verteilbar (siehe auch Androulakis und Floudas (1999)), benötigen jedoch eine genaue Problemmodellierung und haben einen hohen Zeitbedarf zur Lösung des Problems.

Im Gegensatz zu den vorherigen Lösungsansätzen stehen Metaheuristiken, deren Ziel es ist, in der zur Verfügung stehenden Zeit eine möglichst gute Lösung zu finden (siehe Merz

[1] Ansätze zur statistischen Modellierung des Optimierungsproblems, wie beispielsweise bei Ruthe u. a. (2013), erlauben eine Approximation des erwarteten Lastverlaufs, welche jedoch direkt von der Güte der Modellierung abhängt. Diese Lösungsmethode ist aufgrund ihrer statistischen Natur besonders bei der Planung von größeren Verbänden interessant. Da in der Literatur bislang nur wenige Arbeiten dazu veröffentlicht wurden, sei diese Methode nur der Vollständigkeit halber erwähnt.

(2003)) und damit auch die vierte Anforderung erfüllen können. Sie können jedoch im Gegensatz zu den exakten Verfahren größtenteils keine Aussage über die Güte der Lösung treffen. Da in dem Anwendungsfeld die optimale Lösung nur eine untergeordnete Rolle spielt und es entscheidender ist, gute Lösungen in kurzer Zeit zu generieren, kann diese Einschränkung akzeptiert werden.

Bei der Auswahl der für diese Arbeit geeigneten Metaheuristik kann auf eine Vielzahl von etablierten Verfahren zurückgegriffen werden. Aufgrund der gut dokumentierten Eigenschaften und dem hohen Verbreitungsgrad, auch im DSM-Umfeld (etwa bei Logenthiran u. a. (2012)), bieten sich Genetische Algorithmen als etablierter Ansatz an. Obwohl bereits deutlich fortschrittlichere Optimierungsansätze für DSM in der Forschung behandelt wurden, wie zum Beispiel *Support Vector Machines* bei Bremer und Sonnenschein (2013), sollten Genetische Algorithmen hier als verständliche und etablierte Referenzimplementierung für andere Optimierungsansätze dienen können.

Um zu demonstrieren, dass die kombinatorische Betrachtung des Problems und die Beschreibung der Startzeitpunkte als Vektoren als generische Eingabeform geeignet sind, um auch andere Optimierungsalgorithmen zu implementieren, soll die *Ant Colony Optimization* (ACO) als Beispielimplementierung für eine Metaheuristik aus dem *reinforced learning*-Umfeld eingesetzt werden. *Ant Colony Optimization* ist eine im Optimierungsumfeld bereits weitläufig anerkannte Metaheuristik (siehe Merz (2003) und de Castro (2006)), die jedoch bislang noch nicht im DSM-Umfeld eingesetzt wurde. In dieser Arbeit soll beantwortet werden, ob die ACO geeignet ist, Problemstellungen aus dem Demand-Side-Management zu lösen.

3.5.3. Test-Cases

Neben rein technischen Tests der Jadex-Plattform durch Prototypen sind Test-Cases für die Validierung der Anwendungslogik besonders wichtig. Ebenso müssen die Optimierungsalgorithmen parametrisiert werden. Dorigo u. a. (2006) und de Castro (2006) für die ACO und Cantú-Paz (1998) sowie Nissen (1994) für die Genetischen Algorithmen betonen, dass keine Aussagen über allgemeingültige Parameter für diese Optimierungsalgorithmen getroffen werden können und diese häufig domänenspezifisch experimentell bestimmt werden müssen.

Aufseiten der funktionalen Logiktests für die einzelnen Komponenten sollen folgende Szenarien untersucht werden:

- **Validierung des statistischen Modells der halbautomatischen Verbraucher:** Die Agenten sollten dahingehend überprüft werden, dass die Startzeitpunkte der Geräte so gewählt werden, dass sich das Startzeithistogramm der Verbraucher an den Erwartungswert der Startzeitwahrscheinlichkeitsfunktion des jeweiligen Verbrauchertyps annähert, um eine möglichst genaue und realistische Simulation zu ermöglichen.
- **Validierung des Verhaltens der automatisierten Verbraucher:** Es sollte die korrekte Einhaltung der Temperaturgrenzen unter Optimierung der Betriebszeitpunkte gezeigt werden.
- **Verifikation der Subsimulation:** Es ist zu zeigen, dass ein vollautomatischer Verbraucher in der Lage ist, einen plötzlichen Temperaturabfall unterhalb der Temperaturgrenzen zu kompensieren und die darauf folgenden Betriebszeitpunkte wieder zu optimieren. Auch soll gezeigt werden, dass das System in der Lage ist, bei wechselnden Marktdaten mitten am Tag eine Neuoptimierung durchzuführen.
- **Bestimmung von Parametern der Optimierungsalgorithmen:** Es sollen die in der einschlägigen Literatur behandelten Startparameter für die gewählten Optimierungsalgorithmen untersucht und unter Umständen domänenspezifisch angepasst werden.

4. Architektur und Design des Energiesystems

In diesem Kapitel soll die Architektur eines generischen Energiesystems für die Verbraucheroptimierung anhand der vorhergehenden Analyse und den daraus folgenden Anforderungen beschrieben werden. Zentrale Elemente sind dabei die Beschreibung der beteiligten Agenten und deren Kommunikationsinfrastruktur. Wie in Abbildung 4.1 funktional dargestellt, soll das System anhand von Marktdaten einer Energiebörse eine Optimierung vornehmen. Dabei schließen sich die Agenten zu einer interagierenden Gruppe zusammen und schätzen ihren jeweiligen Verbrauch und die zur Verfügung stehende Flexibilität anhand eines Simulationsmodells ab. Kooperativ wird eine Lösung des Optimierungsproblems erzeugt. Als Ergebnis wird ein Fahrplan generiert, mit dem die Agenten parametrisiert werden.

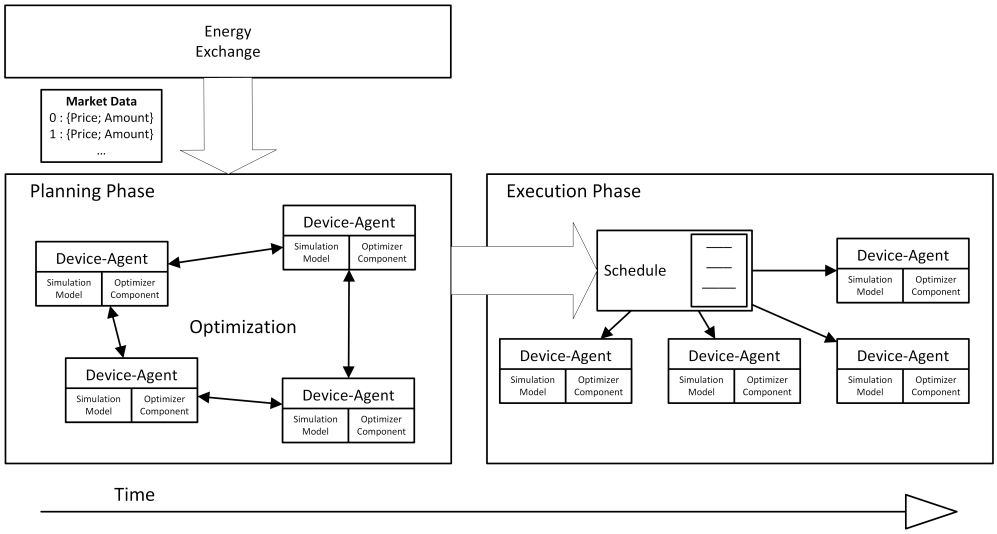

Abbildung 4.1.: Funktionale Beschreibung des Systems

Aus dieser Beschreibung heraus kann eine Architektur (siehe Abb. 4.2) entworfen werden, die die Problemstellung abstrahiert und damit auch für andere Domänen anwendbar macht. Die Verbraucherkomponenten sind der domänenspezifischen Konfiguration zugeordnet und

werden der Umgebung (Environment) zugerechnet. Die Steuerungskomponente (oder Control Component) abstrahiert das domänenspezifische Problem und sucht über die Infrastructure Component geeignete Optimierungskomponenten, die das abstrahierte Problem lösen und die Lösung an die Steuerungskomponente zurückliefern. Diese wendet die Lösung auf das Environment an. In diesem Fall handelt es sich bei der Lösung um den Fahrplan für die Verbraucherkomponenten.

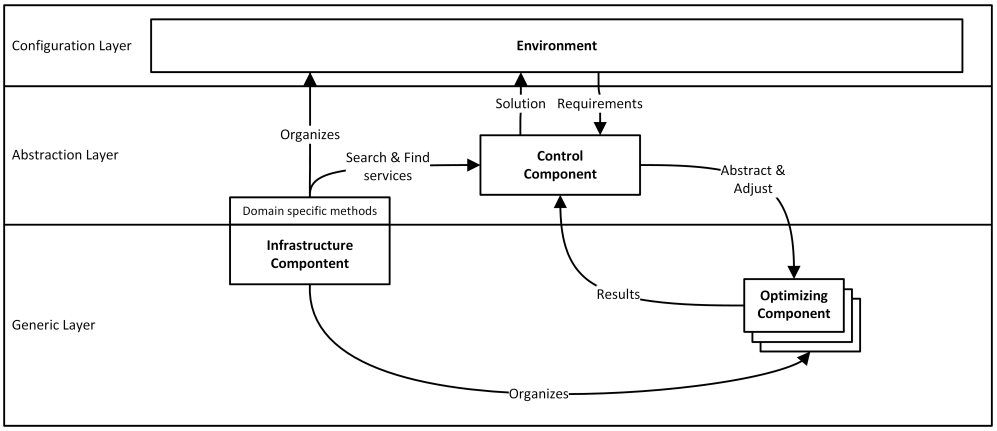

Abbildung 4.2.: Abstrakte Architektur des Optimierungssystems

Wie bereits im vorhergehenden Kapitel herausgestellt wurde, konzentriert sich der architektonische Ansatz auf die Optimierung der Startzeitpunkte der Verbraucher. Dabei ist eine vektorielle Repräsentation der Flexibilität zur Durchführung ganzzahliger Optimierungen erstrebenswert, um erprobte Methoden der evolutionären Algorithmen einsetzen zu können.

Im Folgenden sollen die Architektur des Systems und seiner Subsysteme, sowie die getroffenen Annahmen und Designentscheidungen vorgestellt und erläutert werden.

4.1. Beschreibung der anwendungsspezifischen Architektur

Das vorgeschlagene System soll die Lasten einer Gruppe von Agenten innerhalb bestimmter Rahmenbedingungen anhand eines vorgegebenen Geschäftsmodells auf Basis ihrer temporalen Flexibilität optimieren. Die Agenten sind dabei abstrahiert und transparent, dementsprechend ist es für die Optimierung unerheblich, ob die Agenten die physikalische Ebene repräsentieren oder aggregierte Virtual Devices. Dies wird im folgenden Abschnitt, in dem die Device-Agenten beschrieben werden, näher erläutert. Entscheidend ist jedoch, dass die

Optimierung innerhalb bestimmter Rahmenbedingungen stattfindet. Wie bereits oben beschrieben, können Lasten in zwei Kategorien eingeteilt werden: nicht-verschiebbare Lasten und verschiebbare Lasten, die auch zu späteren Zeitpunkten laufen können. Für das Energiesystem werden nur die verschiebbaren Lasten betrachtet. Die verschiebbaren Lasten sind zumeist an gewisse Nebenbedingungen gebunden, die verhindern, dass die Last beliebig verschoben werden kann. So darf das Optimierungsergebnis in Haushalten gewisse Komfort-Ansprüche nicht beeinflussen, um die Nutzerakzeptanz nicht zu minimieren. Die Agenten stellen daher ihr Optimierungspotential innerhalb dieser Rahmen und Nebenbedingungen zur Verfügung.

Das Geschäftsmodell beschreibt eine Marktsituation für die Gruppe der Agenten. Für jeden Systemzeitpunkt t, wobei gilt $t \in T$ und üblicherweise $T = \{1, \ldots, 96\}$, da in den meisten aktuellen Arbeiten und vergleichbaren Börsenmodellen der Tag in Viertelstunden geteilt ist, können eine beliebige Anzahl von Angeboten O platziert werden. Ein Angebot ist ein (Preis, Mengen) - Tupel, welches eine definierte Leistung m zu einem bestimmten Wert I auf dem Markt anbietet. Somit gilt: O = (I, m). Wenn das System keine Ausfälle von Lasten erlaubt, muss für den Fall, dass zu irgendeinem Zeitpunkt t mehr Energie angefragt als angeboten wird, ein Höchstpreis existieren für den gilt $O_{max} = (I_{max}, \infty)$, sodass jede Anfrage im Sinne der Energieverfügbarkeit bedient werden kann. Energiewirtschaftlich könnte dieses Angebot als Preis für den Bilanzkreisausgleich begründet werden, mit der ein Leistungsdefizit im Netz durch andere Anbieter ausgeglichen wird, um die Netzstabilität zu gewährleisten.

Alle Agenten haben gleichrangigen Zugriff auf den definierten Markt. Es wären damit zwei mögliche Verhaltensszenarien denkbar. Zum einen ein kompetitives Szenario, in dem die Agenten egoistisch um die günstigsten Angebote konkurrieren und im Wettbewerb stehen. Je früher eine Last angemeldet wird, desto wahrscheinlicher ist es, dass der Agent einen günstigen Preis erzielt. Allerdings können hier suboptimale Zustände auftreten, die verhindern, dass der Gesamtpreis für die Gemeinschaft an sich minimiert wird. Beispielsweise kann ein Agent ein Angebot annehmen, welches für ihn optimal ist, jedoch von anderen Agenten viel effizienter genutzt werden könnte.

Um den Einfluss dieser suboptimalen Ergebnisse zu vermindern, können kooperative Strategien angewendet werden. Im Forschungsgebiet der Spieltheorie wurde durch Axelrod (1984) untersucht, unter welchen Umständen Kooperation, also die gezielte Zusammenarbeit von egoistischen Individuen auftritt. In der Arbeit wurde festgestellt, dass Kooperation immer dann wünschenswert und für alle Teilnehmer profitabel ist, wenn Individuen immer wieder die Wahl haben, eine gemeinsame Ressource entweder bis zum maximalen Eigennutz auszunutzen, oder mit anderen zu kooperieren, um den gemeinschaftlichen Nutzen zu maximieren. In Anbetracht dieser Situation wird als kooperative Strategie zum Demand-Side-Management eine gemeinsame Optimierung vorgeschlagen. Die Agenten nehmen dabei in Kauf, dass sie unter Umständen nicht den jeweils für sich optimalen Preis wählen können.

Dafür werden durch die kooperative Optimierung bessere lokale Minima und unter Umständen globale Maxima erreicht, die die Gesamtkosten für alle Teilnehmer senken.

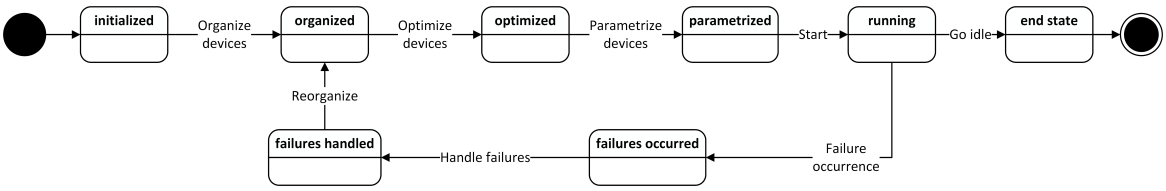

Abbildung 4.3.: Zustandsautomat des Gesamtsystems

In Abbildung 4.3 sind die globalen Zustände und Zustandsübergänge des Gesamtsystems dargestellt. In dem ersten Zustand ist das System definiert. Das bedeutet, die Menge der teilnehmenden Agenten und die zugrundeliegende Marktsituation sind bekannt, ebenso der Zeitkorridor, der optimiert werden soll. Darauf folgend wird eine Teilmenge aus den Agenten gebildet, die in dem definierten Zeitrahmen ihre Last optimieren können. Es findet eine Laufzeitoptimierung der Agenten statt und darauf folgend werden die Agenten werden mit ihren Startzeiten parametrisiert. Wenn die Geräte mit ihren Parametern laufen, können entweder Fehler auftreten, die eine neuerliche Optimierung erforderlich machen, oder das System läuft fehlerfrei und geht über in den Endzustand.

Kernanforderungen an den oben beschriebenen Zustandsautomaten sind Skalierbarkeit, Adaptivität und Flexibilität. Um dies zu gewährleisten, muss definiert werden, wie die Komponenten logisch und räumlich aufgeteilt sind. Kernbestandteil ist der Device-Agent, der die virtuellen oder physischen Geräte abbildet und ihr Flexibilitätspotential nach außen propagiert. Um das System skalierbar und robust zu machen, ist eine Nutzung der Geräte, auf denen die Device-Agenten laufen, für die Optimierung wünschenswert. Dieses Optimierungsmodul, welches aus dem passiven Device-Agenten, der bislang nur seine Flexibilitätspotentiale zur Verfügung stellte, zu einem aktiven Agenten macht, muss dementsprechend einen verteilten Optimierungsalgorithmus implementiert haben. Dieser Algorithmus muss leichtgewichtig sein, damit er die auf den Zielplattformen enthaltenen Ressourcen optimal nutzen kann. Die Skala dürfte sich hier im Bereich von Single-Core-Embedded-Systems für eine $1:1$ Device-Abbildung bis zur Multi-Core-Workstation, die speziell für autarke Optimierungen bereitgestellt ist, bewegen. Da jedes Gerät nur einen kleinen Teil des Problems lösen muss und der Algorithmus damit auf nahezu jedem Gerätetypen lauffähig ist, braucht es keine externe Ressource zur Optimierung. In statischen Netze, die sich kaum in ihrer Struktur verändern, könnten aber auch Multi-Core-Workstations genutzt werden, die auf jedem Kern ein Optimierungsmodul laufen lassen. Dies wäre zum Beispiel im industriellen Umfeld interessant, in dem eine klare Trennung zwischen Optimierung und Verbraucherebene vorgenommen werden soll. Das System wäre weiterhin skalierbar, indem einfach weitere CPUs hinzugefügt werden.

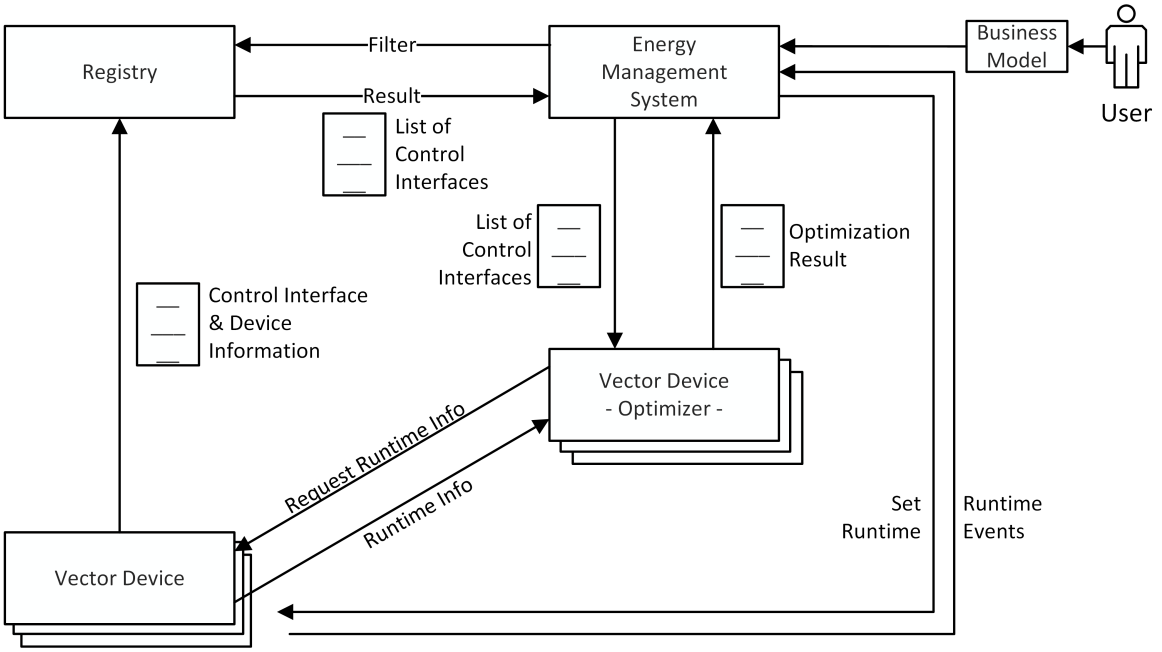

Abbildung 4.4.: Systementwurf mit den wichtigsten Kommunikationspfaden

Die zweite Komponente implementiert das Auffinden von Devices. Dieser Service wird im Folgenden als Registry bezeichnet und ermöglicht die Bildung der oben beschriebenen Netze, indem sich Agenten mit ihren Daten registrieren und dadurch aufgefunden und angefragt werden können.

Die letzte Kernkomponente des Systems ist das Energy-Management-System (EMS), welches für die Organisation verantwortlich ist. Ein Benutzer gibt über das EMS das gewünschte Geschäftsmodell ein, dieses sucht entsprechende Devices in der Registry und initialisiert danach die Optimierung. Der Registry ist, zusammen mit dem Energy-Management-System, im Zustandsautomatendiagramm (Abb. 4.3) der Zustandsübergang *Organize devices* zugeordnet. Zusätzlich übernimmt das Energy-Management-System auch die Behandlung von Planabweichungen zur Laufzeit und entscheidet anhand des vom Benutzer definierten Rahmens, ob eine spontane Neuoptimierung nötig ist.

In Abbildung 4.4 sind als Übersicht alle Komponenten mit den wichtigsten Kommunikationspfaden abgebildet. Die Agenten werden in den folgenden Abschnitten näher beschrieben.

4.1.1. Der Device-Agent

Der Device-Agent ist der zentrale Agententyp des Systems. Es handelt sich dabei um einen Software-Agenten, der in erster Linie die nachgelagerte physikalische Schicht gegenüber den Optimierungsinstanzen abstrahiert. Instanzen, die auf diese Ressource zugreifen, können also nicht feststellen, ob es sich um ein einzelnes physikalisches Gerät handelt, oder um ein transparentes, aggregiertes & virtuelles Device. Durch diese Generalisierung der Device-Schnittstelle wird eine tiefgreifende Flexibilität des Systemansatzes gewährleistet. Es ist möglich das System sowohl auf einem sehr niedrigen Abstraktionslevel zu betreiben, bei dem ein Gerät einen Agenten entspricht (Abbildungsverhältnis Agent/Devices = 1 : 1), als auch auf einem sehr hohen Grad, bei dem der Device-Agent eine Vielzahl von aggregierten Geräten transparent abbildet (1 : n).

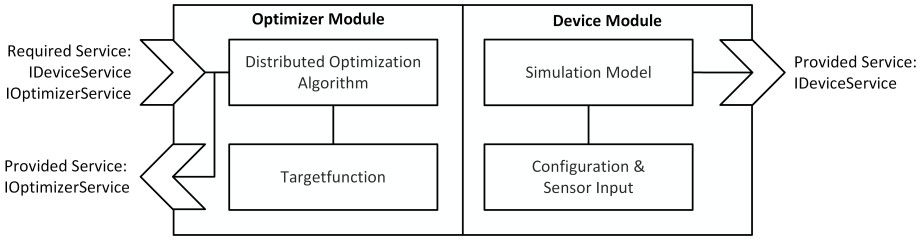

Abbildung 4.5.: Modulares Konzept des Device-Agenten

In Abbildung 4.5, in dem Übersicht über die mögliche modulare Aufteilung des Device Agenten dargestellt ist, wird die Komponente, die für die Berechnung der möglichen Laufzeitpunkte vorgesehen ist, als Simulationsmodell bezeichnet. Das Simulationsmodell berechnet mithilfe des Umgebungswissens und der Agentenkonfiguration die Flexibilitätspotentiale und stellt den Optimierungsinstanzen diese für die Optimierung über die getVector(...)-Methode des Device-Services zur Verfügung, die im Schnittstellendiagramm des Device-Agenten in Abbildung 4.6 definiert wird. Die getVector(...)-Methode liefert einen Vektor von möglichen diskreten Startzeitpunkten zurück, aus denen die Optimierungsinstanzen einen Zeitpunkt wählen kann. Bei Geräten, die in dem Optimierungszeitraum nur einmal laufen müssen, ist dieser einzelne Zugriff ausreichend. Muss ein Gerät oder Anlage jedoch mehrfach am Tag laufen, so wird dies iterativ geplant, um auch Laufzeitabhängigkeiten abbilden zu können. Dieses Vorgehen wird in Abbildung 4.7 visualisiert. Agent Y ist das angefragte und zu optimierende Device und Agent X die Optimierungsinstanz. Agent X fragt über die Service-Schnittstelle mögliche Startzeitpunkte von Agent Y an. Agent Y liefert daraufhin einen Vektor mit diskreten Startzeitpunkten zurück. X wählt aus dem Vektor einen möglichen Zeitpunkt aus und fügt ihn einem temporären Laufzeit-Vektor hinzu. X fragt dann erneut Laufzeitpunkte von Y an, diesmal mit dem bereits gewählten Laufzeitpunkt. Y simuliert mit dem übertragenen Laufzeit-Vektor und liefert entweder einen leeren Vektor zurück, wenn Y in dem Optimierungszeitraum nicht nochmals laufen muss, oder weitere Zeitpunkte, aus denen X

dann wiederum einen auswählt, diesen dem Laufzeit-Vektor hinzufügt und wiederum anfragt, solange bis Y einen leeren Vektor zurückliefert.

Mit diesem Verfahren können iterativ Laufzeitvektoren, beispielsweise für Heizungen, erzeugt werden, ohne dass die Optimierungsinstanz spezielles Domänenwissen benötigt. Die Logik zur Generierung der Laufzeiten verbleibt vielmehr bei den einzelnen Device-Agenten. Dies unterstützt sowohl Generalisierungsaspekte, da die Optimierungsinstanzen nicht an neue Gerätetypen angepasst werden müssen, sofern sich deren Logik im oben beschriebenen Paradigma ausdrücken lässt, als auch Aspekte des Datenschutzes und der Privatsphäre, da es für die Optimierung nicht vonnöten ist, genauere Spezifikationen des Gerätes zu wissen.

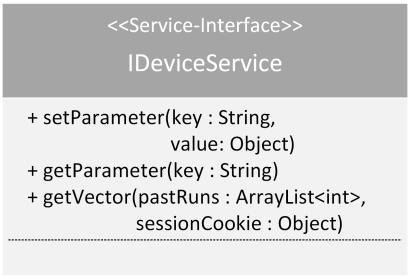

Abbildung 4.6.: Service-Interface des Device-Agenten

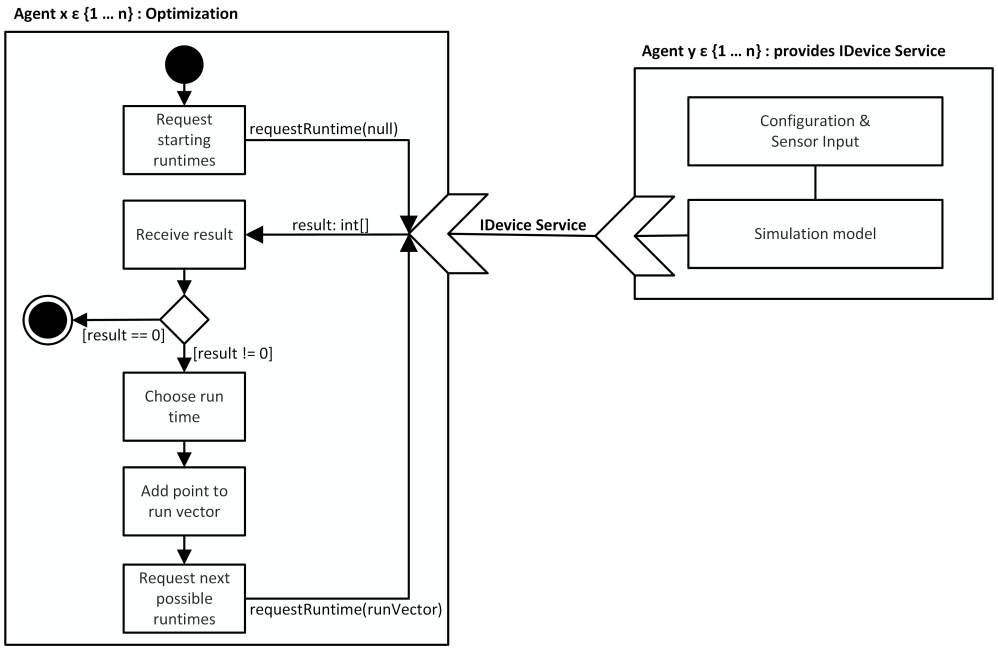

Abbildung 4.7.: Iterative Erzeugung von Laufzeitvektoren

Wie bereits mehrfach erwähnt wurde, kann ein Device-Agent bei systemimmanenten Optimierungen auch ein Optimierungsmodul enthalten, welches die Schnittstellen und Funktionalitäten für die Optimierung bereitstellt. Das Optimierungsmodul wird in Abschnitt 4.2 näher beschrieben.

4.1.2. Die Registry

Um die Device-Agenten such- und auffindbar zu machen, ist eine Registry erforderlich, die als Schnittstelle und Service-Plattform zwischen dem EMS und den Device-Agenten dient. In Abbildung 4.4 ist die Registrierung von Device-Agenten links zwischen Registry und Agent dargestellt. Dabei wird die *register(...)*-Methode des Registry-Services (siehe Abbildung 4.8) genutzt.

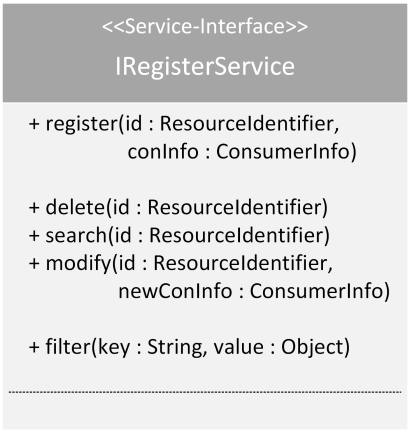

Abbildung 4.8.: Interface der Registry

Device-Agenten registrieren sich hierbei sowohl mit einem *Resource Identifier*, mit dem sie eindeutig identifizierbar sind, als auch mit ihren relevanten Daten, die in dem als *ConsumerInfo* bezeichneten Objekt gespeichert sind. Die Auswahl der zu übertragenden Daten sollte dem Device-Agenten überlassen bleiben, auch um Nutzern die Möglichkeit zu geben, von vornherein die Teilnahme an bestimmten Geschäftsmodellen auszuschließen. Dementsprechend generisch und flexibel sollte die interne Datenstruktur des ConsumerInfo-Objekte angelegt sein.

Zwingende Informationen, die in dem Objekt jedoch vorgegeben sein sollten, sind Informationen über den Stromverbrauch, der abgerufen werden kann und einem Zeithorizont, in dem das Device genutzt werden kann, sowie eine Zeitbasis, in der das Device reagieren kann, um an Szenarien wie Day-Ahead-Handel, Minutenreserve oder Sekundärregelleistung teilnehmen zu können. Diese Parameter enthalten alle notwendigen Informationen für die Klas-

sifizierung und Optimierung. Optionale Parameter könnten beispielsweise Beschreibungen enthalten, oder weitergehende Steuerungen zulassen.

Die Devices werden mit diesen Informationen in die Datenhaltung der Registry aufgenommen. Ein Energiemanagement-System kann nun nach einzelnen Devices suchen, oder nach Gruppen filtern. Ein Filter beschreibt eine Eingrenzung der Suche in der Datenbasis der Registry nach bestimmten Attributen. So könnte nach Devices gesucht werden, die eine gewisse Mindestleistung aufweisen, oder eine bestimmte Flexibilität zulassen. Über die *filter(...)*-Methode des Registry-Services können diese Attribute übergeben werden, woraufhin eine Liste der zur Suchanfrage passende Devices zurückgegeben wird.

Die in Abbildung 4.8 dargestellte Schnittstellendefinition ist keineswegs vollständig. So wird hier beispielsweise kein Rechtemanagement definiert, mit dem die Service-Aufrufe (insbesondere Modifikations- und Löschzugriffe) verifiziert werden.

4.1.3. Das Energiemanagement-System

Das Energiemanagement-System (EMS) ist die Schnittstelle zwischen Benutzer, Betriebsmodellen und Device-Ebene und für die Organisation der zugeordneten Device-Agenten zuständig.

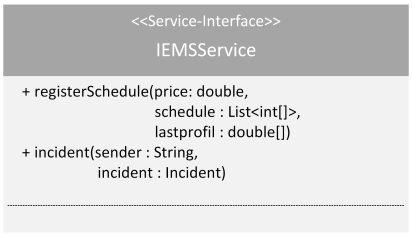

Abbildung 4.9.: Interface des Energiemanagement-Systems

In Abbildung 4.10 ist der generelle Organisationsablauf dargestellt. Das EMS nimmt im ersten Schritt von einem als Client bezeichneten Dritten das umzusetzende Optimierungsziel als Beschreibung entgegen. Aus der Beschreibungssemantik setzt es die Eingabe in Ziele und Anforderungen an die Struktur und Eigenschaften des Optimierungssystems, unter Berücksichtigung von Nebenbedingungen, um. Um das eingegebene Ziel zu verfolgen, benötigt das EMS in der Regel bestimmte Verbrauchertypen oder Flexibilitäten in bestimmten Zeiträumen. Diese Anforderungen werden in Filter-Parameter übersetzt, mit denen das EMS die *filter(...)*-Methode des Registry-Services aufruft. Die Registry liefert daraufhin eine Liste von adäquaten Devices zurück (Zustände 3 und 4 in Abb. 4.10), aus denen das EMS Verbraucher auswählt, die als Verbund optimiert werden sollen. Diese werden parametrisiert, indem ihnen Informationen zur Optimierung mitgeteilt werden, beispielsweise, welche Agenten in

dem Verbund enthalten sind, welche Algorithmik genutzt werden soll etc. Die optimierenden Agenten melden nach erfolgter Optimierung über die Methode *registerSchedule(...)* (siehe Abb. 4.9) das beste Ergebnis zurück. Das EMS sendet daraufhin die Startparameter an alle Device-Agenten des Verbundes und wartet auf die erfolgreiche Durchführung des Plans.

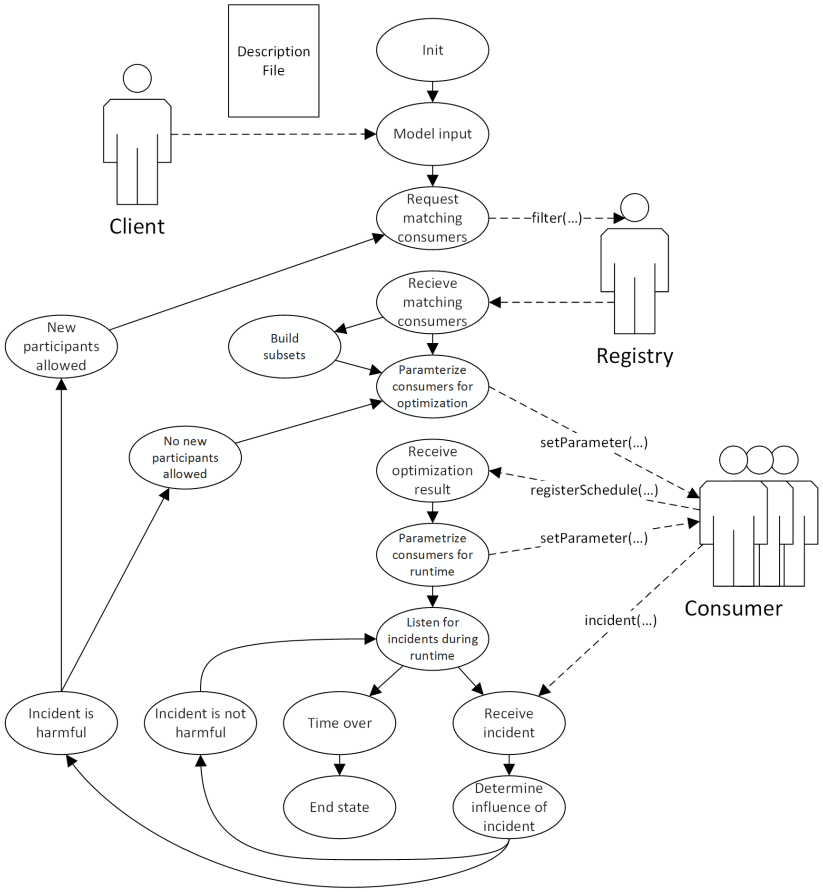

Abbildung 4.10.: An die CTL angelehnte Darstellung des Energiemanagement-Systems

Falls es zu Abweichungen im Fahrplan kommt, senden die Agenten Fehlerberichte über die Service-Methode *incident(...)* an das EMS, welches beurteilt, ob eine Neuoptimierung oder anderweitige Handlungsoption nötig ist, oder das Problem innerhalb von definierten Toleranzgrenzen auftritt.

4.2. Anwendungsspezifisches Design der Metaheuristiken zur Optimierung

In Kapitel 3 wurde hergeleitet, dass Genetische Algorithmen sowie die Ant Colony Optimization geeignet sind, um das definierte Optimierungsproblem zu lösen. In diesem Abschnitt sollen sowohl das Optimierungsmodul der Device-Agenten näher beschrieben werden, als auch die Anpassungen der in der Literatur beschriebenen Algorithmen auf die Problemstellung.

Im vorhergehenden Abschnitt wurde auf die (passive) Rolle der Device-Agenten und der Funktion des EMS in Bezug auf die Optimierung eingegangen. Das Optimierungsmodul fügt dem reaktiven Device-Agenten eine aktive Komponente hinzu und realisiert einen Teil des verteilten Optimierungsalgorithmus sowie die dazu notwendigen Kommunikationsschnittstellen.

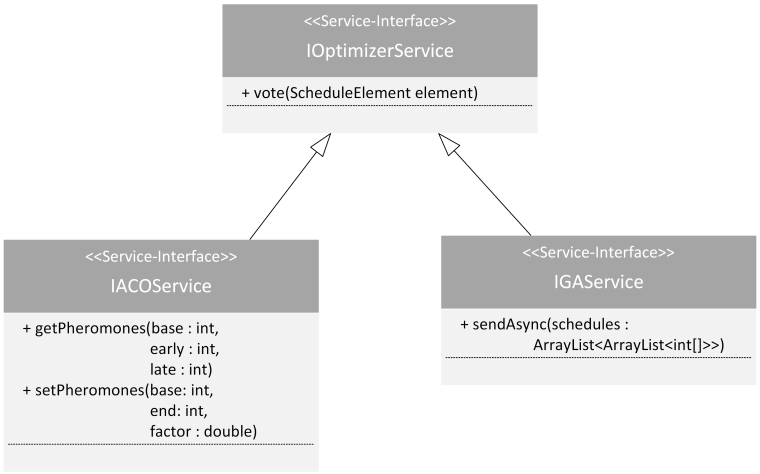

Abbildung 4.11.: Service-Interface-Definition des Optimierungsmoduls

Sendet das EMS ein Signal *sendPulse(...)* (siehe Service-Definition in Abb. 4.11) mit den entsprechenden Parametern (Verbunddaten, Optimierungsverfahren und -parameter etc.) an den Agenten, so kann dieser, unter Berücksichtigung der lokalen Gegebenheiten wie Prozessor-, Speicher- und Kommunikationskapazitäten, das Problem lokal bearbeiten und sich mit anderen Optimierungsinstanzen austauschen. Die Algorithmen müssen dementsprechend skalierbar sein, sodass ein Device-Agenten den Optimierungsumfang lokal anpassen kann, ohne dass dies Einfluss auf andere Optimierungsinstanzen haben darf.

Neben der *sendPulse(...)*-Methode ist für den Optimierungsservice eine weitere Methode erforderlich, mit der die Agenten sich über das beste Optimierungsergebnis abstimmen können, die in der Schnittstellendefinition als *vote(...)* bezeichnet wird.

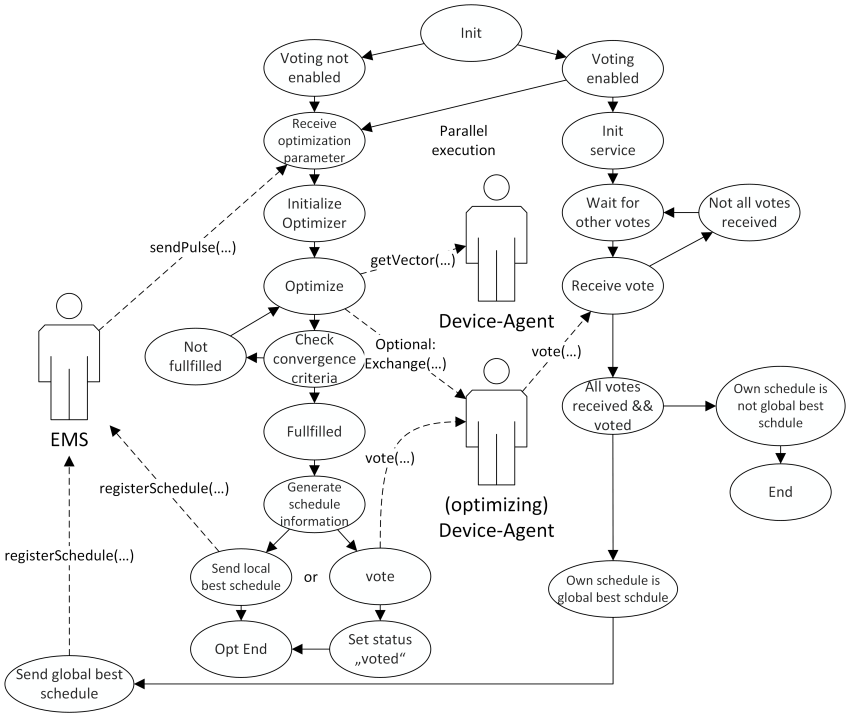

Abbildung 4.12.: An die CTL angelehnte Darstellung des Optimierungsmoduls

In Abbildung 4.12 ist der generelle Ablauf der Optimierung dargestellt. Ausgehend davon, ob abschließend eine Abstimmung über die beste Lösung durchgeführt wird oder nicht, teilt sich der Ablauf in ein oder zwei parallele Pfade. Der rechte Pfad in der Abbildung repräsentiert den Service, der in Bereitschaft ist, Votes entgegen zu nehmen. Auf dem linken Pfad ist die eigentliche Optimierung dargestellt. Der optimierende Agent erhält die Parameter sowie eine Liste von beteiligten Optimierungsinstanzen und den zu optimierenden Device-Agenten von dem EMS. Die Optimierung wird solange iterativ durchgeführt, bis das Konvergenzkriterium erfüllt ist. Dabei werden die *getVector(...)*-Methoden der Device-Agenten sequentiell aufgerufen und Startzeitpunkte nach der gewählten Optimierungsvorschrift gewählt. Optional können auch noch Daten oder Lösungen zwischen den optimierenden Agenten während der Optimierung ausgetauscht werden, um bessere Ergebnisse zu erzielen. Findet keine abschließende Abstimmung statt, sendet jeder optimierende Agent seinen besten Fahrplan an das EMS, welches die Ergebnisse auswertet. Wird eine Abstimmung vorgenommen, sendet jeder optimierende Agent seine beste Lösungen per *broadcast* an die *vote(...)*-Methoden der beteiligten Optimierungsservices. Der beste Agent teilt dann seine Lösung dem EMS direkt mit.

4.2.1. Komplexität und Komplexitätsreduktion

Aufgrund der spezifischen Beschreibung der Flexibilitätspotentiale durch Vektoren ergeben sich besondere Anforderungen an die Komplexität des Optimierungsproblems. Um dies zu verdeutlichen kann das Problem als gerichteter, nicht-zyklischer Graph definiert werden, indem die Zeitpunkte Knoten und die möglichen Verknüpfungen als Kanten aufgefasst werden. Um das Problem eingangs zu vereinfachen, repräsentieren Agenten Knoten 1. Ordnung und die möglichen Startzeitpunkte Knoten 2. Ordnung. Zur Planung muss ein Pfad durch den Graphen gesucht werden, bei dem jeder Knoten 1. Ordnung nur ein mal aufgesucht werden darf (Traversierung) und für jeden Knoten 1. Ordnung genau ein zugeordneter Knoten 2. Ordnung gewählt wird.

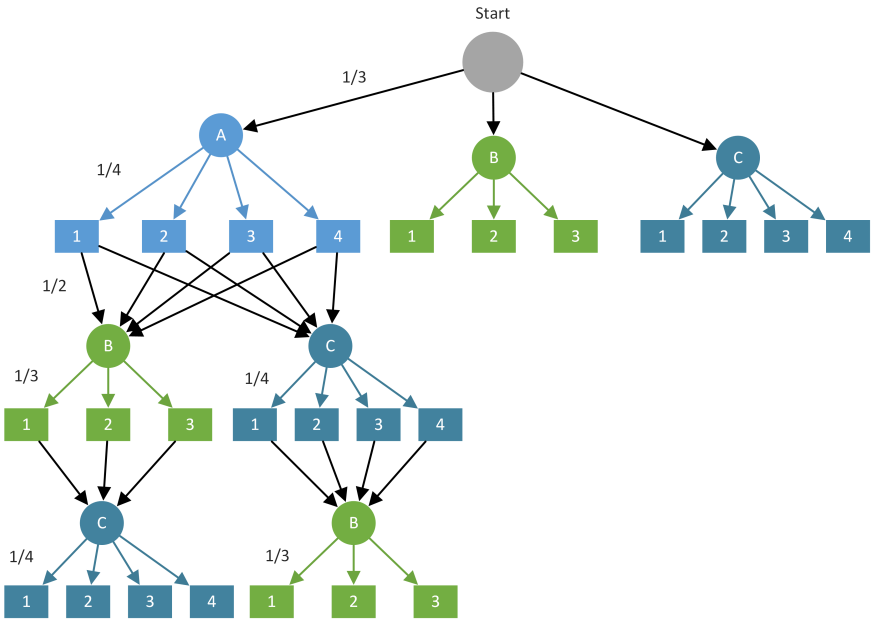

Abbildung 4.13.: Beschreibung des Graphenproblems anhand eines einfachen Beispiels.

Die obige Beschreibung soll anhand eines einfachen Beispiels demonstriert werden. Es sollen drei Devices A, B und C optimiert werden (Knoten 1. Ordnung). Die Knoten 2. Ordnung sind $A = \{1, 2, 3, 4\}, B = \{1, 2, 3\}$ & $C = \{1, 2, 3, 4\}$. Im Sinne der Problemstellung dieser Arbeit würde dies bedeuten, dass A vier mögliche Startzeitpunkte hat, B drei und C wiederum vier. In Abbildung 4.13 ist der Entscheidungsgraph mit den möglichen Pfaden und Auswahlwahrscheinlichkeiten dargestellt. Aus Gründen der Übersichtlichkeit ist nur der linke Teil des Baumes unter dem Knoten A komplett illustriert. Vom Start ausgehend, können drei Knoten 1. Ordnung gewählt werden. Ohne Berücksichtigung der Metriken der Kanten ist die Zugwahrscheinlichkeit bei $p_1 = \frac{1}{3}$. Wurde beispielsweise A gewählt, wird danach ein Knoten 2. Ordnung von A ebenfalls zufällig ausgewählt. Nun wird ein neuer Knoten 1. Ordnung

aus den verbleibenden B & C gezogen und wieder ein Knoten 2. Ordnung gewählt, solange, bis ein Pfad vollständig ist. Können die Knoten 1. Ordnung wie in der eingangs definierten Vorschrift frei gewählt werden, so ergeben sich hier bereits 288 mögliche Pfade durch den Graphen und das Problem ist unter Umständen äquivalent zu anderen über-exponentiellen Problemen wie dem Travelling Salesman Problem. Bei genauerer Betrachtung ist die Zugwahrscheinlichkeit (ohne Berücksichtigung von Metriken) jedes Pfades gleich, was nahelegt, dass alle Permutationen der Knoten gleichrangig sind. Wird nun festgelegt, dass die Knoten 1. Ordnung in einer festen Rangfolge traversiert werden, so reduziert sich die Anzahl der möglichen Pfade deutlich, wie in Abbildung 4.14 dargestellt wird. Für das Beispiel verbleiben 48 relevante Pfade.

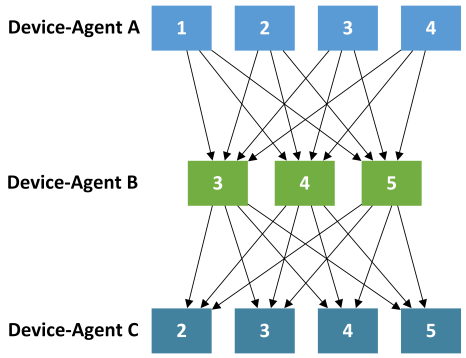

Abbildung 4.14.: Vereinfachung des Graphenproblems

Die Einschränkung, dass jedes Device nur einen Knoten 1. Ordnung repräsentiert, kann in Anbetracht von Devices, die mehrfach in einem Zeitraum betrieben werden müssen, nicht aufrecht erhalten werden. Jeder weitere Laufzeitvektor, der während der iterativen Planung von einem Device übergeben wird, erweitert praktisch den Graphen um einen weiteren Knoten 1. Ordnung. Da es schwierig ist, a priori abzuschätzen, wie viele Planungsiterationen für jedes Device auftreten, und wie viele Zeitpunkte in jedem Schritt möglich sind, ist auch eine vorhergehende Abschätzung der Pfadanzahl sehr komplex. Die Berechnungsvorschrift für die Anzahl der Pfade im allgemeinen lautet:

$$N_{\text{pfade}} = \prod_{m=1}^{M} n_m, \tag{4.1}$$

wobei M die Anzahl der der im System enthaltenen Verbraucher ist und n_m die variable Anzahl der Startzeitpunkte eines Device-Agenten beschreibt (analog zu Abschnitt 3.5.1). Es ist zu erkennen, dass die Anzahl der Pfade mit jedem neuen Knoten als Produkt wächst und dementsprechend eine vollständige Traversierung des Graphen zur Auffindung des besten Pfades bereits bei kleinen M nicht mehr zielführend ist.

4.2.2. Design des verteilten Genetischen Algorithmus

Der hier vorgestellte Genetische Algorithmus (siehe 4.15) orientiert sich an die in der Literatur vorgeschlagenen Basisalgorithmen (vgl. Nissen (1994) und de Castro (2006)), welche in Abschnitt 2.2.3 und Abschnitt 2.2.4 näher beschrieben wurden.

Bei dem Entwurf der Chromosomen wurde eine direkte Repräsentation der Devices gewählt. Jedes Device repräsentiert ein Genom, wobei in dem Genom die Laufzeitpunkte des Devices mit Integerwerten codiert sind (Codierung eines Tages von {0, ..., 96} für Viertelstunden- oder {0, ..., 1440} für Minutenwerte). Sollte ein Device mehrere Laufzeitpunkte haben, so sind diese als Vektor im Genom abgelegt. Um die Interoperabilität und Austauschbarkeit von Chromosomen zu gewährleisten und um den Speicheraufwand zu reduzieren legt das EMS eine bestimmte Reihenfolge der Devices im Chromosom a priori fest. Somit ist zu jedem Zeitpunkt ein Genom an einer bestimmten Stelle im Chromosom immer demselben Device zugeordnet.

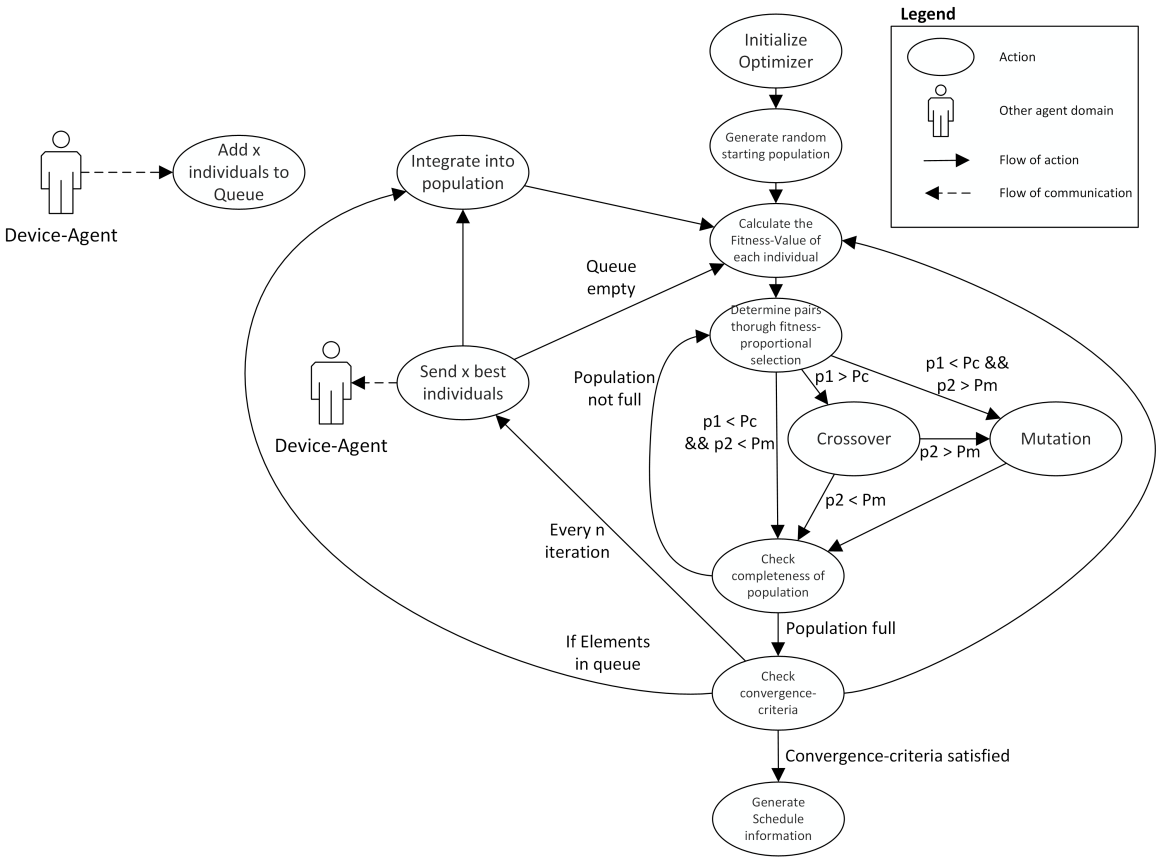

Abbildung 4.15.: An die CTL angelehnte Darstellung des Genetischen Algorithmus

Bei der Erstellung der Startpopulation (2. Zustand in Abbildung 4.15) wird, wie in Abbildung 4.7 bereits vorgestellt wurde, iterativ vorgegangen. Der optimierende Agent wählt den ersten Device-Agenten aus der definierten Reihenfolge aus und fordert einen Startzeitpunktvektor an. Aus diesem Vektor wählt der optimierende Agent einen Zeitpunkt zufällig aus und fügt ihn dem Genom zu. Mit diesem Genom ruft der optimierende Agent die Methode erneut auf und erhält entweder keine weiteren Zeitpunkte und das Genom wird abgeschlossen, oder einen weiteren Startzeitvektor, aus dem wiederum ein Zeitpunkt ausgewählt wird, der dem Genom hinzugefügt wird. Dies wird fortgesetzt, bis der zurückgegebene Startzeitvektor keine Elemente mehr enthält.

Schritt	Chromosom	Aktuelles Device	Aktuelles Genom	Rückgabe-wert	Zufällige Auswahl	Neues Genom
1	{ [], [], [] }	A	{ }	{ 3, 4, 5, 6 }	4	{ 4 }
2	{ [4], [], [] }	A	{ 4 }	{ 6, 7, 8, 9 }	8	{ 4, 8 }
3	{ [4, 8], [], [] }	A	{ 4, 8 }	{ 9, 10, 11 }	11	{ 4, 8, 11 }
4	{ [4, 8, 11], [], [] }	A	{4, 8, 11}	{ }	-	-
5	{ [4, 8, 11], [], [] }	B	{ }	{ 7, 8, 9, 10 }	9	{ 9 }
6	{ [4, 8, 11], [9], [] }	B	{ 9 }	{ }	-	-
7	{ [4, 8, 11], [9], [] }	C	{ }	{ 2, 3, 4 }	2	{ 2 }
8	{ [4, 8, 11], [9], [2] }	C	{ 2 }	{ 10, 11, 12 }	12	{ 2, 12 }
9	{ [4, 8, 11], [9], [2, 12] }	C	{ 2, 12 }	{ }	-	-

Abbildung 4.16.: Beispielhafte Erzeugung eines Chromosoms mit drei Devices A,B & C

In Abbildung 4.16 wird dieses Verfahren anhand eines Beispiels illustriert. Es sei ein Chromosom aus drei Devices A, B und C zu erstellen. Das Chromosom besteht demnach aus drei Genomen (eckige Klammern), wobei das erste Element A repräsentiert, das zweite B und das dritte C. Der optimierende Agent beginnt nun mit A und ruft die Service-Methode *getVector(...)* mit dem aktuellen (leeren) Genom $g_A = \{\}$ auf. Device A gibt daraufhin einen Vektor zurück, der die folgenden möglichen Startzeitpunkte enthält: $r_A = \{3, 4, 5, 6\}$. Der Optimierer wählt einen dieser Zeitpunkte zufällig aus und wiederholt das Verfahren solange, bis eine leere Menge zurückgeliefert wird. Dann plant der Optimierer den nächsten Verbraucher. In dem Beispiel startet das Device A zu den Zeitpunkten 4, 8 und 11, läuft im Optimierungszeitraum dementsprechend drei Mal. Das Device B startet zum Zeitpunkt 9 und muss nur einmal in diesem Zeitraum laufen. C wiederum startet zu den Zeitpunkten 2 und 12. Die Planung des Chromosoms erforderte in diesem Beispiel neun Nachrichten und Rückgabewerte.

Nachdem die Startpopulation generiert wurde, ist der nächste Schritt die Berechnung der Fitness (siehe Abb. 4.15). Hier sind mehrere Ansätze möglich, die Fitness eines Individuums zu bestimmen. Die vielleicht naheliegende Variante ist es, die Energiekosten jedes Chromosoms anhand der zugrundeliegenden Marktdaten zu berechnen. Da sich höhere Fitnesswer-

te stärker vermehren, jedoch niedrigere Preise bevorzugt werden sollen, kann der Kehrwert des Preises als Fitnesswert verwendet werden.

Nach der Fitnessberechnung wird die Nachfolgepopulation erstellt. Es werden zwei Individuen zufällig gezogen, wobei die Zugwahrscheinlichkeit von dem Fitnesskoeffizienten abhängt. Mit der Wahrscheinlichkeit P_m wird ein Crossover durchgeführt. Dabei wird ein zufälliger Genomübergangspunkt bestimmt und die nachfolgenden Genome zwischen den zwei Individuen getauscht. Innerhalb von Genomen findet kein Crossover statt, um keine ungültigen Lösungen zu erzeugen.

Der Nebenoperator des genetischen Basisalgorithmus ist die Mutation. Mit der Wahrscheinlichkeit P_m wird eine Mutation durchgeführt. Der Versuch wird zweimal, einmal für jedes der ausgewählten Individuen, durchgeführt. Bei der Mutation wird zuerst ein Genom zufällig ausgewählt. Danach ein Wert in dem Genom. Ab diesem Zeitpunkt bis zum Ende des Genoms wird iterativ neu geplant. Dadurch ist sichergestellt, dass regelmäßig neue Genomwerte auftreten.

Die fitness-proportionale Selektion sowie die Operatoren werden solange durchgeführt, bis die vorgegebene Populationsgröße erreicht ist. Ist dann das Konvergenzkriterium nicht erreicht, so startet ein neuer Optimierungslauf auf Basis der letzten erstellten Population. Die Optimierung erfolgt in parallelen Populationen, das heißt jeder Optimierer hat eine eigene Subpopulation. Dürfen die Optimierer Individuen austauschen, so werden alle n Iterationen x Individuen emigriert (Linker Pfad in Abb. 4.15). Basierend auf Cantú-Paz (1998) gehen dabei die Individuen für die eigene Population verloren und werden an einen zufälligen Nachbarn geschickt. Um die Agenten nicht synchronisieren zu müssen, werden sie in der Population-Queue des Nachbarn abgelegt. Dieser kann folglich asynchron beim Start einer neuen Optimierungsiteration prüfen, ob Individuen in der Population-Queue vorliegen und diese in seine Population integrieren, wobei die Populationsgröße um mindestens x kurzzeitig ansteigt. Die Populationsgröße reduziert sich aber durch die Selektion und die fixe Größe der Nachfolgegeneration nach einem vollen Optimierungslauf automatisch wieder auf die voreingestellte Größe.

4.2.3. Verteilter Entwurf der Ant Colony Optimization

Die algorithmische Umsetzung der Ant Colony Optimization erfolgt weitgehend nach den Arbeiten von Dorigo u. a. (2006), während insbesondere das Service-basierte Verteilungskonzept speziell für diese Architektur entworfen wurde.

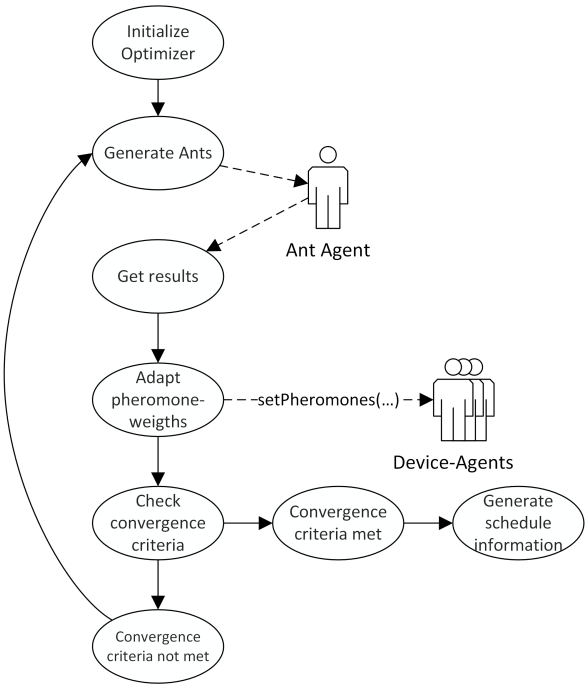

Abbildung 4.17.: Übergeordnetes Verhalten eines ACO-Optimierers

In Abbildung 4.17 ist das globale Verhalten des Algorithmus dargestellt. Nach der Initialisierungsphase generiert jede Optimierungsinstanz eine Anzahl an Ant-Agenten, seine lokale Kolonie, die den Graphen selbstständig traversieren. Die Ant-Agenten können laut Bullnheimer u. a. (1997), je nach Architektur und Möglichkeiten der verwendeten Implementierung, seriell oder parallel ausgeführt werden (siehe Abb. 4.18).

Um eine möglichst optimale Ausnutzung der verteilten Ressourcen zu erreichen, wird die in Abschnitt 4.2.1 Graphenbeschreibung des Optimierungsproblems zugrunde gelegt. Die *getVector(...)*-Methode liefert in dieser Methapher die nächsten besuchbaren Knoten zurück. Auch die Pheromone der Kanten werden lokal gespeichert. Übergibt ein Optimierer seinen Ausgangsknoten, welcher in Abb. 4.11 im IACOService als *base* der Methode *getPheromones(...)* bezeichnet wird, liefert der Device-Agent die dazugehörigen Pfad-Pheromonwerte zurück. Die anderen zwei Parameter können dazu genutzt werden, um a priori die Spannbreite der zurückzugebenden Pheromonwerte zeitlich zu begrenzen.

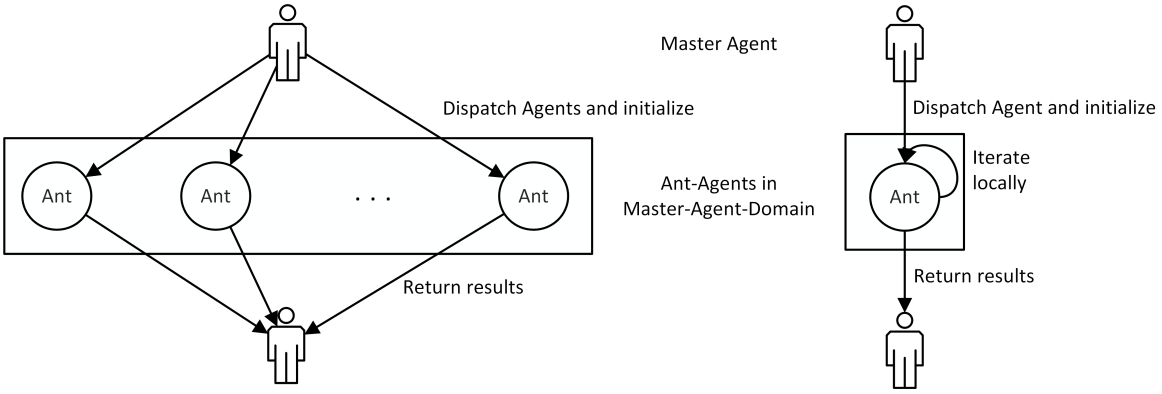

Abbildung 4.18.: Gegenüberstellung paralleler und serieller Ausführung von Ant-Agenten

Nach dem Lauf werden die Pheromongewichte angepasst. Dabei wird nach dem Elitist-Konzept vorgegangen (siehe de Castro (2006) und Dorigo u. a. (2006)), bei dem nur die besten Ant-Agenten Pheromone verteilen dürfen, um die Pfadsuche gerichteter zu machen. Dorigo u. a. (2006) schlägt dabei vor, nach jeder Iteration entweder auf dem Pfad des *iteration-best-* oder des *best-so-far-*Agenten Pheromone zu verteilen. Dorigo u. a. (2006) geben dabei aber keine eindeutige Präferenz ab. Laut de Castro (2006) ist insbesondere die Begrenzung der Pheromonemissionen auf einige wenige ”elitäre” Ant-Agenten der ausschlaggebende Faktor zur Verbesserung des Optimierungsergebnisses. In Anbetracht dieser Erkenntnisse und unter Berücksichtigung der Service-basierten Ant-Colony-Optimization, welche in dieser Arbeit spezifiziert wurde, wird die Gruppe der Elitist-Ants wie folgt definiert: Jeder Optimizer hat eine lokale Kolonie von Ant-Agenten. Nach jeder lokalen Iteration, d.h. nachdem alle Ant-Agenten der lokalen Kolonie einmal den Graphen traversiert haben, werden auf dem *best-so-far*-Pfad der lokalen Kolonie Pheromone platziert. Dabei werden die Device-Agenten noch einmal durchlaufen und die Pheromongewichte durch Service-Calls mit *setPheromones(...)* angepasst. Da die optimierenden Agenten durch verschiedene Parameter und Kapazitäten unterschiedlich schnell sein können, werden die Pheromone asynchron angepasst. Dieser Ansatz geht jedoch mit dem Risiko einher, dass schnelle Optimierungsinstanzen vergleichsweise viele Pheromone auf suboptimalen Routen platzieren können und diese damit verstärken. Gleichzeitig ist aber auch kein weiterer Koordinations- und Synchronisationsaufwand nötig. Durch die Nutzung des ACS-Algorithmus von Dorigo und Gambardella (1997) und der damit verbundenen intensivierten Suche neben etablierten Routen, sollte der verstärkende Faktor der suboptimalen Routen explizit vermindert werden können.

Das lokale Verhalten eines Ant-Agenten, welches in Abbildung 4.19 dargestellt wird, folgt größtenteils den ACS-Paradigmen, sowie der iterativen Planungsvorschrift zur Erzeugung von Vektoren (Abbildung 4.7). Der Master-Agent übergibt dem Ant-Agenten die Liste mit den beteiligten Device-Agenten. Der Ant-Agent fragt den ersten Vektor von dem ersten Device-Agenten in der Liste an und wählt zufällig einen Startknoten aus. Dieser Startknoten wird ei-

nem temporären Laufzeitvektor hinzugefügt. Danach wird der nächste Vektor mit Pheromonvektor ausgehend von dem zuvor gewählten Knoten angefordert. Ist der zurückgegebene Wert *null*, so ist die Planung dieses Device-Agenten abgeschlossen und es kann der nächste Device-Agent geplant werden, bis die Liste der Device-Agenten vollständig durchlaufen wurde. Ist der Wert nicht *null* wird, unter Berücksichtigung der jeweiligen ACS-Vorschrift, aus dem empfangenen Vektor ein nächster Knoten ausgewählt. Die Kantengewichte werden anhand der Marktpreise des Leistungsbedarfs der Device-Agenten berechnet.

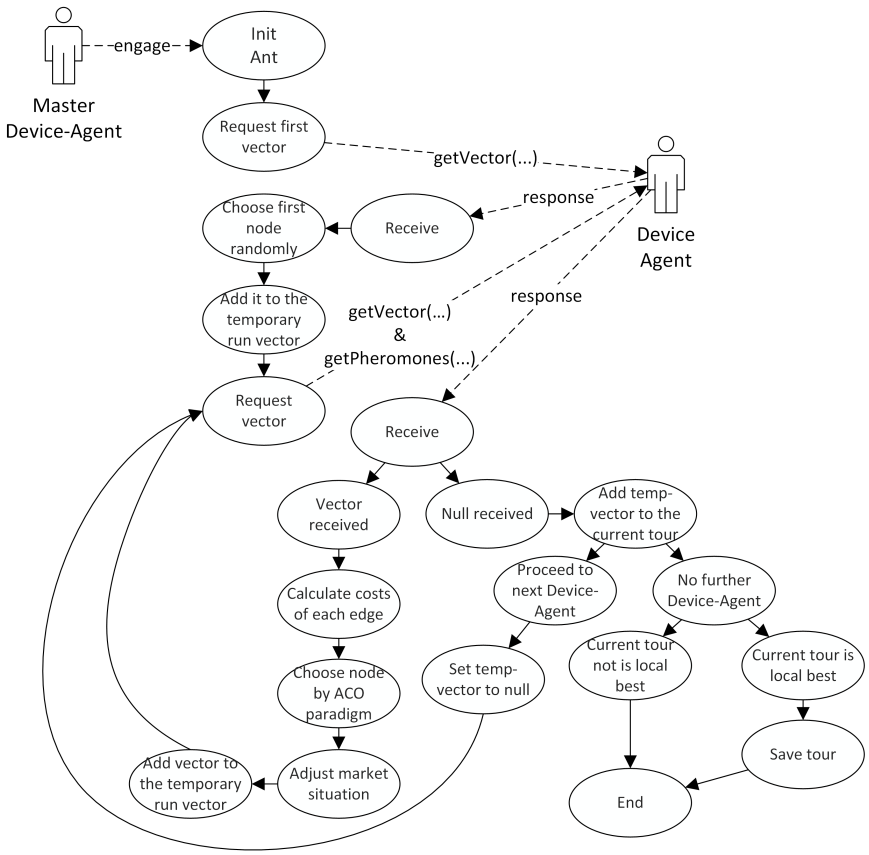

Abbildung 4.19.: An die CTL angelehnte Darstellung des Verhaltens eines Ant-Agenten

In dieser Arbeit wurde der ACS-Algorithmus von Dorigo und Gambardella (1997) als Referenz implementiert. Aufgrund der lokalen Update-Regel zur Pheromonverminderung und der globalen Update-Regel, bei der nur die Elitist-Ants Pheromone platzieren, kann dieser ohne weitere Änderungen in das bestehende Konzept implementiert werden.

5. Implementierung des Energiesystems in Jadex Active Components

Das System soll die oben definierte Spezifikation und die Architektur auf eine Simulationsumgebung abbilden, um die Forschungsfragen untersuchen zu können. Dazu wurden mehrere Device-Agenten, eine angepasste Registry und ein EMS implementiert. Das System soll in erster Linie als reines Simulationssystem dienen. Es ist sowohl erforderlich, dass eine kooperative Optimierung durchgeführt werden kann, als auch die Agenten mit den Werten aus der Optimierungslösung parametrisieren zu können und zu simulieren. Jeder Simulationsdurchlauf hat mindestens vier Phasen (siehe Abb. 5.1): Zuerst werden die verfügbaren Device-Agenten organisiert und dann optimiert. Danach wird eine (eventbasierte) Sub-Simulation durchgeführt. Treten dort Abweichungen oder Fehler auf, nimmt das System ab dem Zeitpunkt des frühsten Fehlers eine neue Optimierung vor, wonach die Sub-Simulation mit den aktualisierten Parametern neu gestartet wird. Dieser Vorgang wird wiederholt, bis keine Fehler mehr auftreten und das System geht in den Endzustand über.

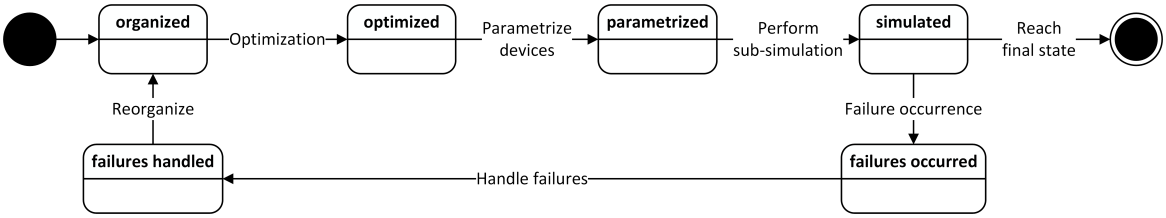

Abbildung 5.1.: Ablauf einer Simulation

Zu Projektbeginn kamen zwei mögliche Jadex-Entwicklungszweige in Betracht. Zuerst Jadex 2.0, dessen Entwicklung abgeschlossen wurde. Bei dieser Jadex-Version werden die BDI-Agenten in einer XML-Datei beschrieben und ihre Pläne in Java-Klassen geschrieben. Es handelt sich bei Jadex 2.0 (Unterversion 2.2.1) um ein relativ erprobtes Konzept, welches jedoch nicht weiterentwickelt wird. Das Risiko, dass ein Fehler in der Plattform auftritt, der den Entwicklungsfortschritt nachhaltig negativ beeinflusst, ist demnach als hoch einzuschätzen. Zusätzlich ist das Debugging durch die Trennung der BDI-Agenten in Java-Klassen und XML-Dateien sehr schwierig, wodurch auch hier mit einem erhöhten Entwicklungsaufwand gerechnet werden muss. Kurz vor Beginn der Systementwicklung wurden erste Ent-

wicklungsversionen von Jadex 2.3 veröffentlicht. Mit dieser Version wurden die neu konzeptionierten BDIv3-Agenten vorgestellt, welche sich mehr an den Progammierparadigmen der Micro-Agenten orientierten. Die BDIv3-Agenten können demnach vollständig in Java geschrieben werden, wodurch die gesamte Bandbreite an IDE-Tools für die Entwicklung und das Debugging genutzt werden kann. Zusätzlich ist der Speicherbedarf der Agenten stark reduziert worden, was massive Simulationen mit deutlich mehr Agenten ermöglicht. Obwohl sich in ersten Tests zeigte, dass diese Version, bedingt durch das frühe Entwicklungsstadium, noch erhebliche Fehler enthielt, überwiegen die Vorteile einer integrierten Entwicklung mit Zugriff auf IDE-Werkzeugen. Vorteilhaft war auch die Kommunikationsmöglichkeit mit den Jadex-Entwicklern und die fortwährende Weiterentwicklung der Plattform über Nightly-Builds.

Die Referenzimplementierung der oben beschriebenen Architektur wurde mit JADEX Active Components[1] durchgeführt. Dabei wurde die Entwicklung mit der JADEX-Version 2.3 Nightly-Build vom 03.11.2013 begonnen. Da sich jedoch zeigte, dass sich die neue, Java-integrierte, 3. Version der BDI-Agenten noch in einem frühen Entwicklungsstadium befand, wurde das Projekt fortlaufend auf neue JADEX-Versionen migriert, sodass der vollständige Entwicklungszyklus der Version 2.4 durchlaufen wurde und abschließend auf den Nightly-Build 2.5 vom 09.01.2014 angepasst wurde.

Für die Implementierung wurden BDIv3-Agenten und Micro-Agenten gleichermaßen verwendet, je nachdem, ob der jeweilige Agent eine aktive oder passiv/reaktive Rolle im System wahrnimmt. Für die Kommunikation der Agenten untereinander wurde die JADEX Service-Metapher genutzt. Die Parametrisierung des Simulationsmodells und der Agenten erfolgt durch XML-Dateien, die mit dem JAXB-Framework[2] eingebunden werden.

[1] www.activecomponents.org
[2] https://jaxb.java.net/

5.1. Implementierte Agententypen

Zur Demonstration der Funktionalität wurden zwei Typen von Device-Agenten implementiert. Es handelt sich dabei zum einen um die Implementierung eines programmgetriebenen Gerätes, welches im Entwicklungskontext als Semiautomatic-Device bezeichnet wird und zum anderen um einen vollautomatischen Verbraucher, welcher hier, aufgrund des Startzeitenvektors, auch *Vector-Device* genannt wird.

Daneben wurden auch die für die Organisation verantwortlichen Agenten, die Registry und das EMS, umgesetzt, sowie ein Agent, der das Management der Simulationsumgebung und die Parametrisierung der Agenten zu Systemstart vornimmt, implementiert.

5.1.1. Management-Agent

Um ein Simulationsmanagement und insbesondere einen geregelten Systemstart zu ermöglichen wurde ein Micro-Agent implementiert, der Simulationen gemäß der Nutzerspezifikation startet und durchführt. Die Simulationsparameter erhält der Agent zum Systemstart von dem in Abschnitt 5.3.1 beschriebenen Nutzerinterface. Dieses übergibt die Simulationskonfiguration über die *initializeApplication(...)*-Methode des IManagementServices (siehe Abb. 5.2). In mehreren IComponentSteps werden nacheinander ein EMS-Agent, eine Registry und die definierte Anzahl von Semiautomatic-Agenten und Vector-Agenten gestartet und initialisiert:

1. Starten des EMS
2. Starten der Registry
3. Initialisierung der Registry
4. Starten der Semiautomatic-Devices
5. Starten der Vector-Devices
6. Semiautomatic- und Vector-Devices initialisieren

IComponentSteps sind in Jadex eine Möglichkeit, asynchrone Prozesse strukturiert zu realisieren. Diese sind nötig, da der Start der Agenten asynchron stattfindet und der Callback dementsprechend abgefangen werden muss. Im Listing 5.1 ist der IComponentStep zum Starten des EMS dargestellt. Es ist erkennbar, dass die Klasse eine Methode *execute(...)* implementiert, die ausgeführt wird, wenn der IComponentStep aufgerufen wird. In der *execute(...)*-Methode wird zuerst asynchron der IComponentManagementService geholt. Der ResultListener ruft die Methode *resultAvailable(IComponentManagementService result)* auf, wenn das Ergebnis vorliegt. Mit dem IComponentManagementService können nun Agenten

gestartet werden. Dies wird mit der Methode *createComponent(...)* durchgeführt. Ein weiterer Result-Listener wartet auf das Ergebnis der Agentenerzeugung. Ist dieser gestartet und damit definitiv verfügbar, wird der nächste IComponentStep zur Erzeugung des Registry-Agenten mit *agent.scheduleStep(new StepStartRegister())* angesetzt.

Abbildung 5.2.: Service-Definition des Managment-Agenten

```
class StepStartEMS implements IComponentStep<Void>
{
 @Override
 public IFuture<Void> execute(IInternalAccess ia)
 {
  SServiceProvider.getServiceUpwards(agent.getExternalAccess().getServiceProvider(), IComponentManagementService.class).
        addResultListener(new  DefaultResultListener<IComponentManagementService>()
  {
          @SuppressWarnings("deprecation")
          public void resultAvailable(IComponentManagementService result)
          {
                  final IComponentManagementService cms = result;

                  cms.createComponent(null, "ems/EMSBDI.class", null, null).addResultListener(
                  new DefaultResultListener<IComponentIdentifier>()
                  {
                          public void resultAvailable(IComponentIdentifier cid)
                          {
                             agentList.add(cid);
                             agent.scheduleStep(new StepStartRegister());
                          }
                  });
          }
  });
  return IFuture.DONE;
 }
}
```

Listing 5.1: Code des IComponentSteps zum Start des EMS in Jadex

Durch diese Verschachtelung ist es möglich, dass jeder Agent erst gestartet wird, wenn sein Vorgänger vollständig erzeugt wurde. Somit kann ein definierter und geplanter Systemstart sichergestellt werden.

Der Management-Agent ist ebenfalls dafür zuständig, dass das System nach erfolgreicher Simulation neu gestartet wird, sofern in der Simulationskonfiguration festgelegt wurde, mehrere Simulationen zu durchlaufen. Dabei werden alle Agenten beendet und neu gestartet, nachdem das EMS über die *restartApplication()*-Methode des IManagementService dem Management-Agenten mitgeteilt hat, dass die Simulation erfolgreich beendet wurde. Der Management-Agent startet daraufhin den *stepRestart*-IComponentStep, in dem geprüft wird, ob eine neue Simulationsrunde gestartet werden muss, oder das System vollständig durchlaufen wurde.

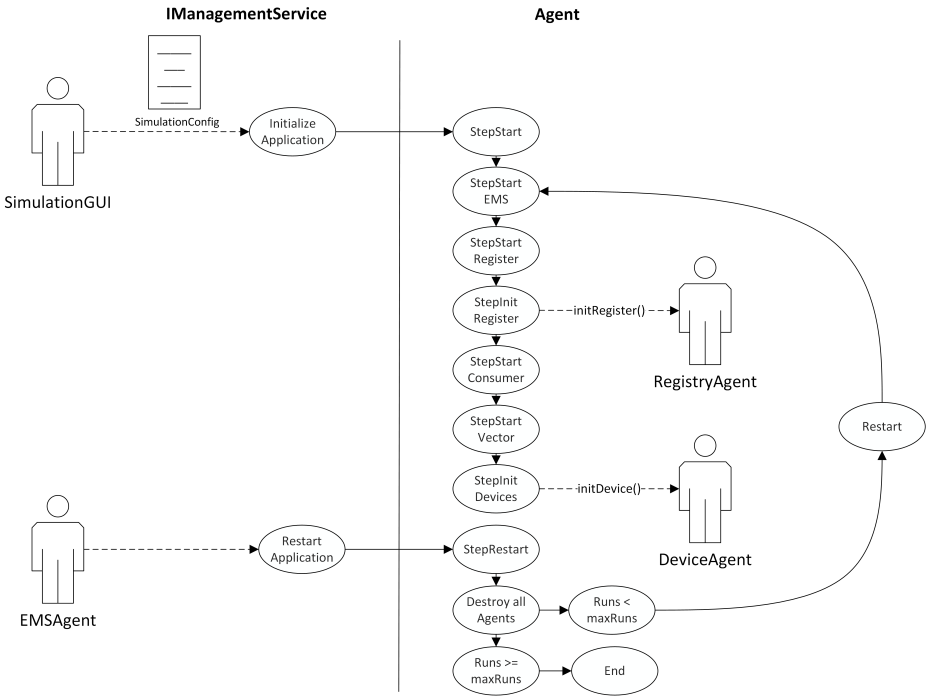

Abbildung 5.3.: An die CTL angelehnte Darstellung des Management-Agenten

5.1.2. Base-Agent

Der Base-Agent ist die BDI-Basisklasse für alle Agenten, welche Verbraucher repräsentieren und stellt eine weitgehende Implementierung der Device-Agent-Architektur dar. In Abbildung 5.4 ist die BDI-Struktur in einem modifizierten Tropos-Diagramm dargestellt. Neben dem IDeviceService wurden drei Main-Goals, das *RegisterGoal*, das *GetFahrplanGoal* und das *SimulateGoal*, implementiert. Das *RegisterGoal* ist Bestandteil der Initialisierung des Agenten und wird durch den Aufruf der *initDevice(...)*-Methode aufgerufen. Mit diesem registriert sich der Agent bei der Registry. Danach geht der Agent in einen reaktiven Zustand und wartet darauf, durch ein EMS parametrisiert zu werden.

Die Buchung eines Device-Agenten wird über die *sendSyncPulse(...)*-Methode vorgenommen, durch die das EMS dem Agenten Informationen und Parameter für die Optimierung übermittelt. Es handelt sich dabei um eine Aggregation der *setParameter(...)*-Methode für eine übersichtlichere Codestruktur. Dieser Synchronisationspuls wird, entsprechend der Service-Metapher in Jadex, als Broadcast an alle gebuchten Device-Agenten gesendet und dient als Signal für die beginnende Optimierung.

In dem realisierten System erhalten jedoch nicht alle Agenten den gleichen Startimpuls. Es wurden zwei polymorphe Methoden implementiert, da der Threading-Overhead in dem

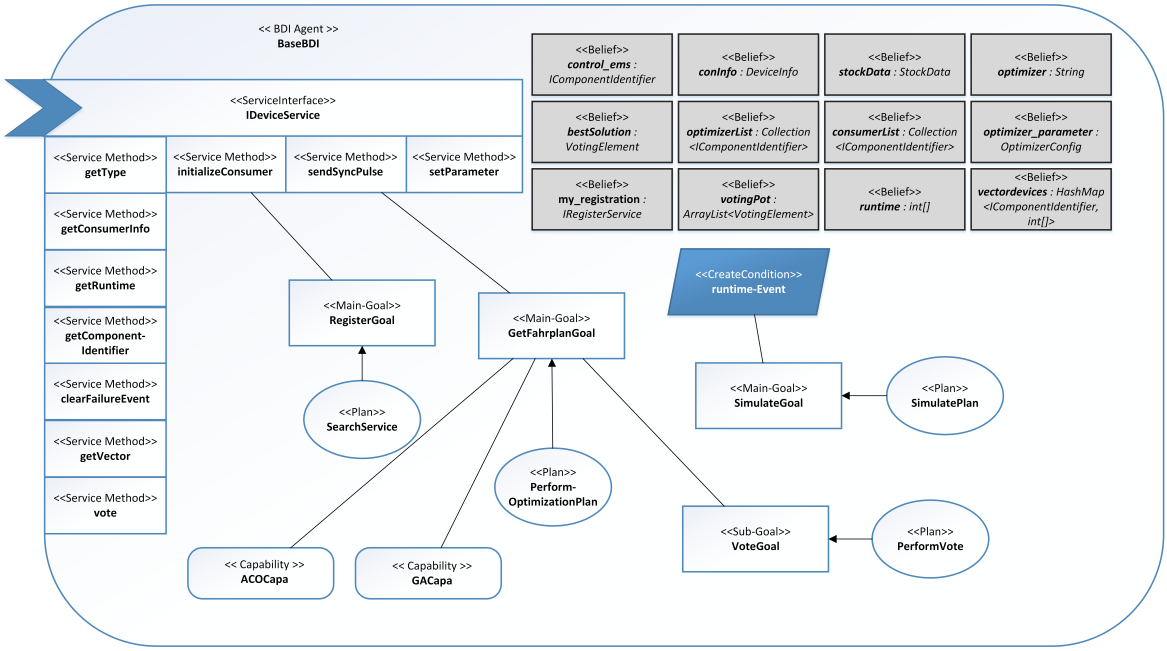

Abbildung 5.4.: Goal-Plan-Diagramm des Base-Agenten

massiv-parallelen Programm zu großen Verzögerungen führt, wenn alle Agenten gleichzeitig optimieren. Durch den EMS-Agenten wird daher beim Systemstart die Anzahl der vorhandenen Prozessorkerne überprüft und entsprechend viele Agenten zu optimierenden Agenten ernannt, während der Rest als reaktive Agenten auf die Anfragen der optimierenden Agenten reagiert. Abhängig vom empfangenen Puls verbleibt der Agent entweder in einem reaktiven Zustand, bei dem nur auf Service-Requests von optimierenden Device-Agenten geantwortet wird oder geht in die aktive Optimierung über, indem das *GetFahrplanGoal* ausgeführt wird.

Der Base-Agent implementiert als Optimierungsalgorithmen sowohl eine ACO-, als auch eine GA-Capability. Diese werden aus dem *PerformOptimizationGoal* heraus genutzt. Der Plan erzeugt dabei, abhängig von dem ausgewählten Optimierungsalgorithmus und den Einstellungen in der Konfiguration, zuerst ein Initialisierungs-Subgoal für den jeweiligen Algorithmus. Der Agent führt dann iterativ entweder einen GA-Optimierungschritt (*GAPerformGoal*), optional abwechselnd mit einem Individuenaustausch (*GASendAsynchronouslyGoal*), oder einen ACO-Schritt (*ACOGoal*), abwechselnd mit dem *ACOadaptWeigths*-Goal, zur Anpassung der Pheromongewichte im verteilten Graphen, aus.

```
/** Top Level Goal */
@Goal(unique=true)
public class SimulateGoal
{
	protected BaseBDI agent;

	@GoalCreationCondition(beliefs="runtime")
	public SimulateGoal(BaseBDI agent)
	{
		this.agent = agent;
	}

	@GoalDropCondition
	boolean checkDrop(BaseBDI agent)
	{
		return (agent.runtime == null);
	}
}
```

Listing 5.2: SimulateGoal mit Ausführungsbedingungen

Nach der Optimierung wird eine Abstimmung zwischen allen optimierenden Agenten über den besten Fahrplan durchgeführt. Hat das EMS die beste Lösung erhalten, parametrisiert es jeden Agenten mit seiner Startzeit. Wird die *runtime*-Variable verändert, löst Jadex einen Event aus, wodurch im *SimulateGoal* automatisch die Goal-Bedingungen geprüft werden (siehe Listing 5.2).

```
@Plan(trigger=@Trigger(goals=SimulateGoal.class))
protected class SimulatePlan
{
	@PlanBody
	protected IFuture<Void> body(final IPlan plan)
	{
		final Future<Void> ret = new Future<Void>();

		IFuture<IEMSService> fut = agent.getServiceContainer().getService(IEMSService.class, control_ems);
		fut.addResultListener(new DefaultResultListener<IEMSService>()
		{
			@Override
			public void resultAvailable(IEMSService result)
			{
				SimulationResult simulationResult = new SimulationResult();

				if(conInfo.getSimulationIncident() != null)
					simulationResult = new SimulationResult(agent.getComponentIdentifier(), false, runtime, conInfo
						.getSimulationIncident());
				else
					simulationResult = new SimulationResult(agent.getComponentIdentifier(), true, runtime, null);

				result.simulationResult(agent.getAgentName(), simulationResult).get();
				ret.setResult(null);
			}
		});

		return ret;
	}
}
```

Listing 5.3: SimulatePlan des Base-Agenten

In dem *SimulatePlan* wird geprüft, ob in der Agentenkonfiguration ein Simulationsfehler (*conInfo.getSimulationIncident()*) definiert wurde. Ist dies nicht der Fall, so wird davon ausgegangen, dass der Agent den Fahrplan einhalten kann. Ansonsten wird die Fehlermeldung an das EMS gesendet und dort behandelt. Wurde der Fehler im EMS behandelt, wird dieser mit dem Aufruf der *clearFailureEvent()*-Methode des IDeviceServices (Abb. 5.5) gelöscht.

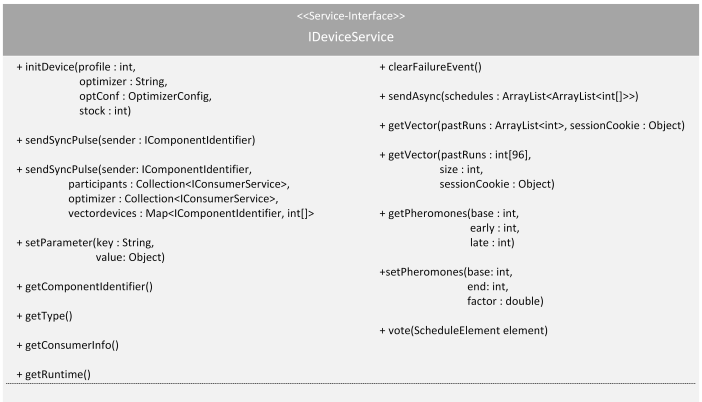

Abbildung 5.5.: Service-Interface des Base-Agenten

Die Serviceschnittstelle (siehe Abb. 5.5) wurde, im Gegensatz zum Architekturentwurf, stark an die Anforderungen der Simulation angepasst, um die Simulation zu beschleunigen und den Code übersichtlicher zu gestalten. Ihre Implementation ist das Hauptunterscheidungsmerkmal zwischen den einzelnen Device-Agententypen, die im folgenden näher beschrieben werden. Da in dem System nur der Genetische Algorithmus mit der *sendAsync(...)*-Methode und der ACO-Algorithmus mit *setPheromones(...)* und *getPheromones(...)* implementiert wurden, sind diese zur Vereinfachung in den IDeviceService integriert.

5.1.3. Semiautomatic-Device

Der Semiautomatic-Device-Agent ist die Umsetzung einmal am Tag laufender, halbautomatischer Geräte, wie zum Beispiel Waschmaschinen oder Geschirrspüler. Halbautomatisch bedeutet in diesem Zusammenhang, dass der Nutzer das Gerät starten muss, aber der tatsächliche Startzeitpunkt innerhalb gewisser Parameter von dem Gerät eigenständig gewählt werden kann. Die einzige Anpassung gegenüber dem Base-Agenten ist eine spezialisierte Implementierung des IDeviceService zur Generierung des charakteristischen Verhaltens.

In Anhang A.1 ist die XML-Struktur des Semiautomatic-Devices dargestellt. Im Vektor *PowerProfile* wird das Lastprofil des Verbrauchers in kW und 15-Minutenschritten angegeben. Um zu bestimmen, in welchem Zeitraum sich für den Agenten gültige Laufzeiträume befinden, benötigt dieser zwei Parameter. Zum einen den als *runtimeVector* bezeichneten Vektor. In diesem bis zu 96 Elemente langen Vektor wird ausgehend vom Zeitpunkt 0 die kumulierte Startwahrscheinlichkeit für jeden Zeitpunkt des Tages angegeben. Wird der Agent mit der *initDevice(...)*-Methode initialisiert, kann mithilfe eines Zufallsversuches dessen frühster möglicher Startzeitpunkt bestimmt werden. Um den spätest möglichen Startzeitpunkt zu

bestimmen, benötigt der Agent den *flexibilityVector*. Dieser gibt die kumulierte Wahrscheinlichkeit für n Zeitslots Flexibilität an. Mittels eines weiteren Zufallsversuches wird dadurch die Anzahl an erlaubten Zeitslots bestimmt.

Wird nunmehr die *getVector(...)*-Methode des Agenten aufgerufen, gibt der Agent den oben bestimmten Flexibilitätsvektor zurück. Die Optimierungsalgorithmen können aus diesem Vektor einen beliebigen Zeitpunkt als Startzeitpunkt wählen.

5.1.4. Vector-Device

Das Vector-Device ist eine Erweiterung des Base-Agenten und die Implementierung einer Stromheizung als Beispiel für ein vollautomatisches Gerätes, welches iterativ mit der im Architekturteil entwickelten Vorschrift geplant wird. Das Vector-Device enthält eine einfache, in Java geschriebene, thermische Simulation eines Hauses mit Elektroheizung.

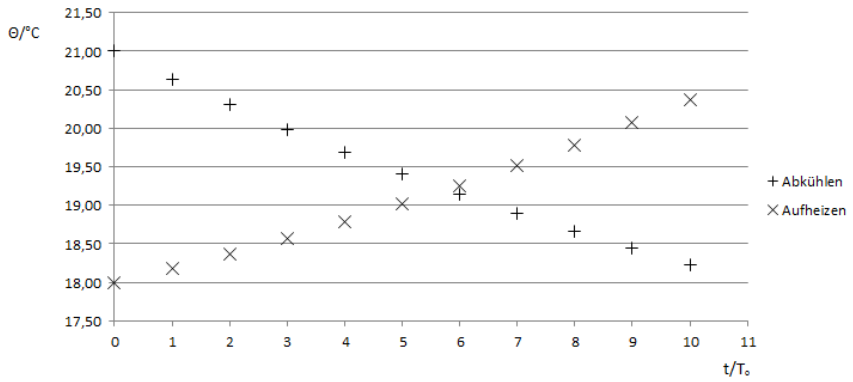

Abbildung 5.6.: Aufheiz- und Abkühlverhalten mit verschiedenen Parametern

Die Aufheiz- bzw. Abkühlfunktion wurde von Paerschke (2005) übernommen. Die Berechnung der Raumtemperatur Θ_{t+1} für die nächste Minute erfolgt mit Formel 5.1. Dabei ist Θ_t die momentane Innentemperatur, Θ_{aussen} die dazugehörige Außentemperatur und $P_{Heizung}(t)$ die Heizleistung in kW. Der Wärmedurchgangskoeffizient der Wände h wird multipliziert mit der Wandfläche A. C ist die beheizte Luftmenge in kg multipliziert mit der Dichte der Luft ($1,225 kg/m^3$) und der spezifischen Wärmekapazität der Luft ($1 kJ/kg/K$).

$$\Theta_{t+1} = (P_{Heizung}(t) + (\Theta_{aussen} - \Theta_t)hA) \cdot \frac{1}{C} + \Theta_t \tag{5.1}$$

Das Vektor-Device wird mittels einer deskriptiven XML parametrisiert, deren Struktur in Anhang A.2 dargestellt ist. Der *day*-Parameter bezeichnet den für die Außentemperatur zugrundeliegenden Tag des Jahres. Dieser Wert kann zwischen 1 und 365 liegen. Im *PowerProfile* kann ein fixer Stromverbrauchs-Vektor auf 15 Minutenbasis definiert werden, sofern der Stromverbrauch nicht im Agenten durch ein Modell berechnet wird. Um die Simulation innerhalb gewisser Temperaturgrenzen ablaufen zu lassen, sind obere und untere Schwelle der Raumtemperatur angegeben. Diese sollten so gewählt werden, dass eine realistische Berücksichtigung der Komfortbedürfnisse etwaiger Bewohner gewährleistet ist. Der *raumtemperatur*-Parameter gibt die initiale Raumtemperatur zum Start der Simulation an.

5.1.5. Registry-Agent

Der Registry-Agent wurde als reaktiver Micro-Agent ausgelegt, da dieser passiv auf Anfragen an den Registry-Service wartet. Der Service wurde um eine Methode zur Initialisierung *initializeRegister(int exp_number_participants)* erweitert. Dabei wird die Anzahl der erwarteten Registrierungen übergeben, um einen definierten Systemstart zu ermöglichen. Nach dem Erreichen der erwarteten Zahl sendet die Registry ein Signal an die vorhandenen Energy-Management-Systeme, dass sich alle geplanten Device-Agenten angemeldet haben und damit das Simulationssystem bereit ist.

Die Methoden *delete(...)*, *modify(...)* und *search(...)* wurden nicht umgesetzt, da sie in dem begrenzten Umfang der Simulation nicht angewendet werden. Die *filter(int fromTimeslot)*-Methode erlaubt derzeit eine Filterung der Device-Agenten nach der frühsten erlaubten Startzeit.

5.1.6. EMS-Agent

Der EMS-Agent wurde als BDI-Agent in Jadex implementiert und stellt die zentrale Schnittstelle zwischen den Device-Agenten und den Betriebsmodellen dar. In Abbildung 5.7 ist das Goal-Plan-Diagramm des Agenten dargestellt. Nachdem der Agent durch den Management-Agenten gestartet wurde, befindet sich dieser in einem inaktiven Zustand, um auf den abgeschlossenen Start des Gesamtsystems zu warten. Haben sich beim Registry-Agenten alle Device-Agenten registriert, ruft der Registry-Agent die *systemReady*-Methode des IEMS-Service auf (Abb. 5.7), wodurch das *systemRunning*-Belief auf *true* gesetzt wird und der Agent in den aktiven Status wechselt. Das Setzen von *systemRunning = true* erfüllt die Erzeugungsbedingung des *AchieveDayAheadGoals*.

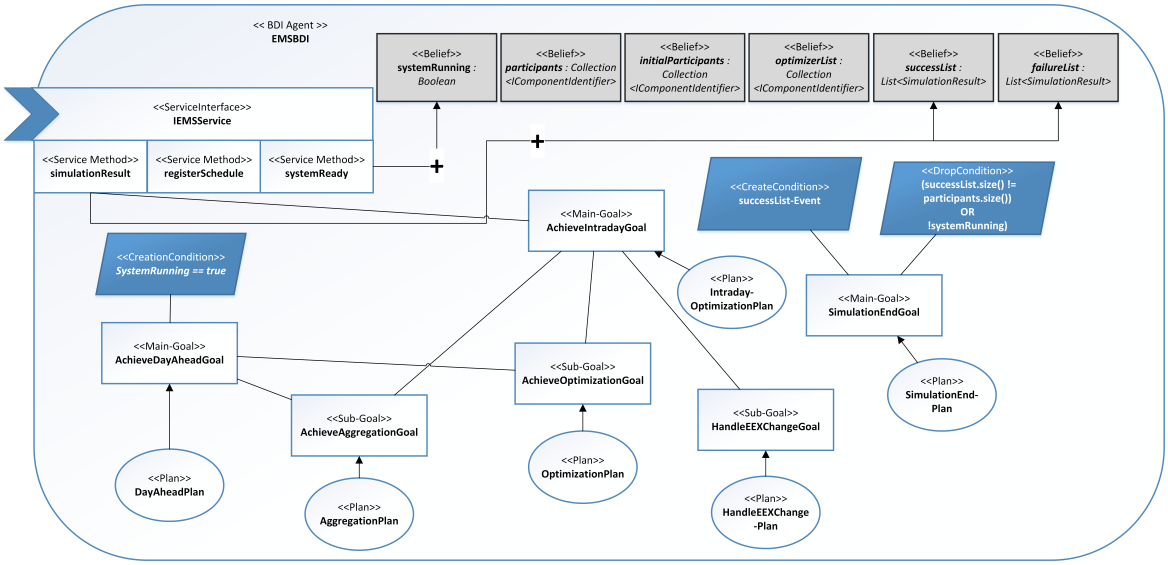

Abbildung 5.7.: Goal-Plan-Diagramm des EMS-Agenten

In dem *DayAheadPlan* wird zuerst das *AchieveAggregationGoal* ausgeführt. Mit diesem Goal wird die Registry nach passenden Device-Agenten gefragt, woraufhin diese eine *Collection* von Device-Agenten zurückgibt. Das EMS speichert alle zu buchenden Agenten in der *participants*-Collection. Es wählt zusätzlich aus der Collection maximal so viele Agenten aus, wie Prozessoren zur Verfügung stehen und legt sie in der *optimizerList* ab. Diese sind die aktiven Optimierungsagenten. Diese Einschräkung wurde vorgenommen, um den begrenzten Ressourcen eines Simulationssystems gerecht zu werden.

Mit dem *AchieveOptimizationGoal* und dem zugeordneten Plan werden die Device-Agenten parametrisiert und die Optimierung startet. Der EMS-Agent wartet nun, bis über die *registerSchedule*-Methode das Optimierungsergebnis vorliegt und parametrisiert dann die Agen-

ten der *participants*-Collection mit ihren jeweiligen Startzeitvektoren, woraufhin die Device-Agenten mit ihrer Subsimulation starten.

Die Agenten melden das Ergebnis der Subsimulation an das EMS über die *simulationResult*-Methode. Sind alle Ergebnisse vorhanden und kein Fehler aufgetreten, wird automatisch das *SimulationEndGoal* aktiviert, womit dem Manager-Agenten mitgeteilt wird, dass die Simulation abgeschlossen ist.

Ursprünglich war, entsprechend des BDI-Paradigmas, geplant das *AchieveIntradayGoal* auszuführen, wenn sich das Umweltwissen (Beliefs) des Agenten ändert. Dazu sollte das Goal auf Events der beiden Listen *successList* und *failureList* reagieren. Da in Jadex während der Entwicklung Probleme bestanden, Änderungen in ArrayLists als Events zu detektieren, wird das *AchieveIntradayGoal* derzeit aus dem Service direkt gestartet, wenn alle Device-Agenten ihr Simulationsergebnis abgeliefert haben und der Inhalt der *failureList* nicht null ist. Ist ein Simulationsfehler aufgetreten, der eine neue Optimierung erforderlich macht, wird das *AchieveIntradayGoal* ausgeführt. Es wird dabei der frühste Fehler zuerst behandelt. Das EMS erstellt mithilfe der Registry eine neue *participants*-Collection aus allen Semiautomatic-Agenten, die bis zu dem Zeitpunkt des frühsten Fehlers noch nicht gestartet sind und allen Vectordevices. Die Selektion aller Vectordevices erfolgt unter der Annahme, dass diese für den ganzen Tag geplant werden und auch zu jedem Tageszeitpunkt laufen können. Die Vectordevices (Key) mit ihren Laufzeitvektoren (Value) werden in einer Hashmap gespeichert, ebenso der Zeitpunkt, ab dem neu optimiert werden muss, unter dem Key *null*. Die Hashmap dient im Simulationssystem der Nachverfolgung von Zwischenergebnissen. Mittels dem *AchieveOptimizationGoals* wird eine neue Optimierung gestartet. Dieses Verfahren wird so lange wiederholt, bis die Subsimulation des Tages fehlerfrei war.

Im EMS-Agenten offenbarte sich ein weiteres Jadex-Problem. Änderungen an Beliefs aus Serviceaufrufen heraus werden im Agenten nicht als Event detektiert, wodurch Goals, die auf Belief-Änderungs-Events warten, nicht gestartet werden. Als Beispiel sollte das *AchieveDayAheadGoal* ausgeführt werden, nachdem das *systemRunning*-Belief durch den *systemReady*-Service-Aufruf auf *true* gesetzt wurde. Eine Möglichkeit wäre es, das Goal direkt aus dem Service heraus zu starten mit dem im Listing 5.4 A gezeigten Code. Das sequenzielle Erzeugen von Goals steht aber dem BDI-Paradigma gegenüber, in dem eigentlich Agenten auf Umweltänderungen reagieren sollen. Wird das Belief des Agenten in dem Service-Aufruf direkt geändert, wie in Listing 5.4 B gezeigt, löst Jadex jedoch im Agenten kein Event aus.

```
final EMSBDI iAgent =(EMSBDI) ((PojoBDIAgent)agent).getPojoAgent();

A: iAgent.agent.dispatchTopLevelGoal(iAgent.new AchieveDayAheadGoal());

B: iAgent.systemRunning = true;

C: iAgent.changeSystemRunningBelief(true);
```

Listing 5.4: Starten eines Goals

Zur Lösung dieses Problems muss eine Methode zur Änderung des Beliefs im Agenten selbst erstellt werden (Listing 5.5), die dann mit dem Code C in Listing 5.4 aus dem Service heraus aufgerufen wird.

```
public void changeSystemRunningBelief(boolean input)
{
        systemRunning = input;
}
```

Listing 5.5: Ändern eines Beliefs aus einem Service heraus

5.2. Implementierung der Subsimulationsfunktionalität

Nachdem die Optimierung abgeschlossen und die Device-Agenten mit Startzeiten parametrisiert wurden, simulieren die Device-Agenten ihren Tagesverlauf. In dieser Subsimulation findet eine Event-basierte Simulation statt, was bedeutet, dass grundsätzlich davon ausgegangen wird, dass der vorher erstellte Plan eingehalten werden kann. Im Falle einer Abweichung senden der betreffende Agent die Fehlermeldung an das EMS, wo sie behandelt wird.

Die zwei in der Analyse bereits definierten interessanten Simulationsfehler wurden im System umgesetzt und müssen händisch zu einer Device-XML ergänzt werden. Ein sogenannter *SimulationIncident* besteht aus drei Parametern: Dem Auftrittszeitpunkt *atTime*, dem Fehlertypen *type* und dem Wert *value*, der vom Fehlertyp abhängig ist.

Das erste Simulationsevent umfasst eine plötzliche Temperaturveränderung eines Vectordevices. Eine beispielhafte XML-Beschreibung ist in Listing 5.6 dargestellt. Wählbare Parameter sind der Auftrittszeitpunkt *atTime* und die zu diesem Zeitpunkt auftretende Temperatur in der Variable *value*. Der Parameter *type* muss mit dem Wert *temperature* angegeben werden.

```
<simulationIncident>
        <atTime>30</atTime>
        <type>temperature</type>
        <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xsi:type="xs:double">18.5</value>
</simulationIncident>
```

Listing 5.6: XML-Struktur des Temperatur-Simulationsfehlers

Das zweite Simulationsevent beschreibt eine spontane Änderung der Marktdaten (Listing 5.7). Der *type* muss hierbei als *EEX* angegeben werden. Der *value*-Wert gibt an, welches Marktszenario aus der Datenbank geladen werden soll. Mit diesem Simulationsfehler können eine Vielzahl von Situationen nachgebildet werden. So können zum Beispiel Unwägbarkeiten von regenerativen Energieerzeugern simuliert werden, wenn beispielsweise eine Verfügbarkeitsprognose revidiert werden muss. Auch kann dieses Ereignis dazu genutzt werden, um Intradayhandel und Tauschgeschäfte zu simulieren.

```
<simulationIncident>
    <atTime>2</atTime>
    <type>EEX</type>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs
       ="http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">2</value>
</simulationIncident>
```

Listing 5.7: XML-Struktur des EEX-Simulationsfehlers

5.3. Nutzerschnittstellen der Simulation

Um eine komfortable Nutzerinteraktion zu ermöglichen, wurden mehrere graphische Oberflächen entworfen und umgesetzt. So ist ein Großteil der Simulationsumgebung über graphische Oberflächen konfigurierbar und ein Teil der Ergebnisauswertung kann bereits in der Simulationsumgebung vorgenommen werden. Zu Ablage von Simulationsergebnissen wurde eine MySQL-Datenbank aufgesetzt, die ebenfalls umfangreiche Eingabeparamter und Modelle, wie zum Beispiel das Marktmodell oder Außentemperaturkurven, für die Simulation bereitstellt.

Praktisch alle Simulationsparameter sind durch die Datenbank oder freie XML-Dateien zugänglich und können daher auch ohne die bereitgestellten Tools separat modifiziert werden. Die graphischen Oberflächen wurden mit dem Eclipse WindowBuilder-Plugin[3] erstellt.

5.3.1. Hauptinterface

Die Hauptoberfläche dient dazu, Konfigurationsdateien zu öffnen, zu bearbeiten und zu speichern, sowie um Simulationen zu starten und auszuwerten (Abb. 5.8). Nach dem Laden einer Konfigurations-XML können die Optimierungsalgorithmen ausgewählt und konfiguriert werden, sowie die Bezeichnung des gewünschten Marktmodells eingegeben werden. Ferner kann die Zahl der Simulationsläufe bestimmt werden.

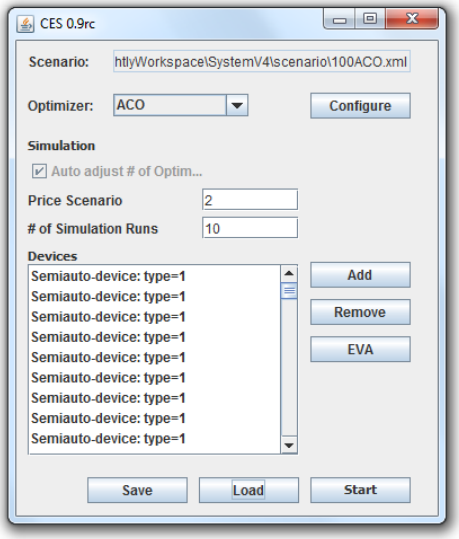

Abbildung 5.8.: Hauptbedienoberfläche der Software mit geladenem Simulationsdatensatz

[3] http://www.eclipse.org/windowbuilder/

Mittels Add- und Remove-Button können Devices hinzugefügt und auch wieder gelöscht werden. Der EVA-Button öffnet das Auswertungsfenster, welches in Abschnitt 5.3.3 näher beschrieben wird.

Save und Load speichern und laden Simulationskonfigurationen und der Start-Button startet die Simulation mit den angegebenen Parametern.

```
String[]        defargs = new String[]
{
        "-gui", "false",
        "-welcome", "false",
        "-cli", "false",
        "-printpass", "false"
};
String[]        newargs = new String[defargs.length];
System.arraycopy(defargs, 0, newargs, 0, defargs.length);

// Start the platform with the arguments.
IFuture<IExternalAccess>        platfut = Starter.createPlatform(newargs);

// Wait until the platform has started and retrieve the platform access.
ThreadSuspendable       sus     = new ThreadSuspendable();
IExternalAccess platform        = platfut.get(sus);
System.out.println("Started platform: "+platform.getComponentIdentifier());

// Get the CMS service from the platform
IComponentManagementService     cms     = SServiceProvider.getService(platform.getServiceProvider(),
        IComponentManagementService.class, RequiredServiceInfo.SCOPE_PLATFORM).get(sus);

// Starting the management-agent
@SuppressWarnings("deprecation")
IComponentIdentifier    cid     = cms.createComponent(null, ManagerAgent.class.getName()+".class", null, null).get(sus);

System.out.println("Started manager component: " + cid);

IManagementService ims = SServiceProvider.getService(platform.getServiceProvider(), cid, IManagementService.class).get(sus);

ims.initializeApplication(simConf);
```

Listing 5.8: Starten einer Jadex-Plattform aus der GUI heraus und Initialisierung des Management-Agenten

Nach dem Drücken des Start-Buttons wird eine Jadex-Plattform entsprechend dem offiziellen Jadex-Tutorial[4] gestartet. Nachdem die Plattform gestartet ist, wird ein Management-Agent erstellt und über die *initializeApplication*-Methode des IManagementServices die Startkonfiguration, die in dem Objekt *simConfig* vorliegt, übergeben. In Listing 5.8 ist der entsprechend angepasste Code aufgezeigt.

5.3.2. Optimizer GUI

Mit dem Drücken des Configure-Buttons auf der Hauptoberfläche (siehe Abschnitt 5.3.1) wird das Konfigurationsfenster für die Optimierungsalgorithmen geöffnet. Die beiden implementierten Algorithmen haben leicht unterschiedliche Konfigurationsinterfaces (siehe Abb. 5.9 und Abb. 5.10). Jedes der Eingabefelder ist mit einem Tooltip ausgestattet, der sich öffnet,

[4] siehe: http://activecomponents.org/bin/view/AC+Tutorial/09+Application+Integration

wenn mit dem Mauszeiger kurz auf dem jeweiligen Feld verweilt wird. Die Tooltips erklären in einer kurzen, prägnanten Beschreibung die Funktion des jeweiligen Eingabefeldes.

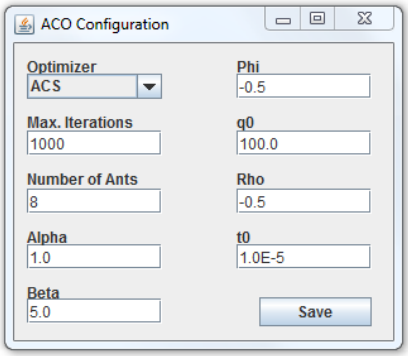

Abbildung 5.9.: ACO-Konfigurations-GUI

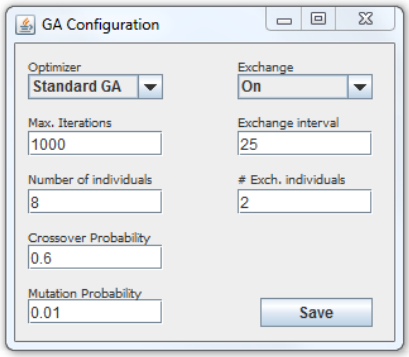

Abbildung 5.10.: GA-Konfigurations-GUI

Durch drücken des Save-Buttons werden die vorgenommenen Änderungen gespeichert. Sollen die Änderungen hingegegen verworfen werden, so muss das Fenster nur geschlossen werden. Die Konfigurationen für die Algorithmen müssen im Scenarios-Ordner als XML-Dateien (*acoConfig.xml* bzw. *gaConfig.xml*) vorliegen.

5.3.3. Auswertungs GUI

Die Auswertungs-GUI wird nach Abschluss eines Simulationslaufs automatisch geöffnet und stellt die in der Datenbank vorliegenden Messdaten dar. Die Graphen wurden mit JFreeChart[5] erstellt und können verschiedene Messdaten aus der Datenbank aufbereiten. Es stehen drei Anzeigemodi zur Verfügung. Im ersten Modus werden die Lastgänge dargestellt und es können alle Revisionen und Simulationsläufe in einer Grafik angezeigt werden (Abbildung 5.11). Die anderen zwei Modi zeigen einen Kostenvergleich zwischen allen Simulationsläufen und die benötigte Simulationszeit an.

[5] http://www.jfree.org/jfreechart/

Abbildung 5.11.: Auswertungsoberfläche mit Vergleich von 10 Simulationsläufen.

5.4. Datenbank

Zur Aufnahme von Simulationsdaten und -ergebnissen und der Speicherung der Marktmodelle und der Außentemperaturkurven für die Heizungssimulation wurde eine MySQL-Datenbank erstellt und in die Software eingebunden. Das Simulationssystem erwartet die Datenbank ”systemv1” auf der lokalen Plattform unter der IPv4-Adresse: 127.0.0.1 und dem Port 3306. Der Benutzername ist ”cesdbuser” und das Passwort ”ces2014”.

Die Datenbank besteht aus vier Tabellen. Zwei Tabellen halten Daten für die Simulation bereit. Die Tabelle ”EPEX” (Tabelle 5.1) enthält das oben bereits vorgestellte Tarifmodell. Der Wert für *profile* gibt an, zu welchem Marktmodell der Tabellenwert gehört und der Wert in dem Feld *time* den diskreten Zeitpunkt, welcher Werte von 0 bis 95 annehmen kann. *low* gibt den Preis für den günstigen Tarif zum Zeitpunkt *time* an und *amount* welche Menge davon verfügbar ist. *high* ist der Preis für den Konsum oberhalb des *amount*-Wertes.

Die zweite Datentabelle ist die ”Temperatur”-Tabelle (Tabelle 5.2). Diese hält Daten über Temperaturkurven von spezifischen Tagen in 15-minütigen Zeitabschnitten. Das Feld *Day* kann beliebige Zahlenwerte annehmen, aber es wird empfohlen, diese an die Tage eines Jahres zu koppeln (Werte zwischen 0 und 364). Das Feld *Time* enthält entsprechend die Tageszeit auf 15-Minuten-Basis. Für jeden angegeben Tag müssen exakt 96 Werte mit Feldwerten von 0 bis 95 vorliegen.

Für die Auswertung und Speicherung der Ergebnisse sind zwei Tabellen in der Datenbank

Tabelle 5.1.: EPEX-Tabellenstruktur

ID	Typ	Beschreibung
id	int	Primary-Key
profile	int	Profil
time	int	Zeit
amount	decimal	Menge
low	decimal	Niedriger Preis
high	decimal	Hoher Preis

Tabelle 5.2.: Temperatur-Tabellenstruktur

ID	Typ	Beschreibung
PRIM	int	Primary-Key
Day	int	Tag
Time	int	Zeitpunkt
Temp	decimal	Temperatur

vorgesehen. In der ”Results”-Tabelle (Tabelle 5.3) werden die wichtigsten Ergebnisse von jedem Simulationslauf gespeichert. Dazu gehören die Nummer des Simulationslaufs, die Revisionsnummer, die für jede vorgenommene Neuoptimierung durch die Subsimulation vergeben wird, die ermittelten Kosten, sowie der gewählte Fahrplan. Ebenfalls wird die Berechnungszeit registriert, um die Algorithmen auch zeitlich vergleichen zu können.

Die zweite Auswertungstabelle (Tabelle 5.4) speichert den Lastgang, also die Leistung pro Zeiteinheit für jeden Simulationslauf und jede Unterrevision.

Tabelle 5.3.: Results-Tabellenstruktur

ID	Typ	Beschreibung
run	int	Lauf
rev	int	Revision
price	double	Kosten
schedule	text	Fahrplan
runtime	int	Rechenzeit

Tabelle 5.4.: Lastgang-Tabellenstruktur

ID	Typ	Beschreibung
run	int	Lauf
rev	int	Revision
time	int	Zeit
power	double	Leistung

6. Simulation und Validierung

In diesem Kapitel sollen die Funktion des Simulationssystems validiert und die Simulationsergebnisse evaluiert werden. Dazu wird das System parametrisiert und die Funktion anhand der definierten Testcases geprüft. Danach werden drei Use-Cases vorgestellt, die die Anwendungen des Systems demonstrieren und auch die Möglichkeiten und Grenzen des gewählten Ansatzes aufzeigen.

6.1. Parametrisierung des Simulationssystems

Bei der statistischen Parametrisierung der Agenten wurde sich auf die Smart-A-Projektstudie von Stamminger u. a. (2008) herangezogen. Dabei wurden für den Semiautomatic-Agenten zwei Anwendungsfälle aus dem Haushaltsumfeld (Geschirrspüler & Waschmaschine) extrahiert. Die europaweite, durchschnittliche Laufzeitwahrscheinlichkeit der beiden Geräte, basierend auf Studien der Universität Bonn, wurde ausgewertet und in den Modellen umgesetzt (siehe Abbildung 6.1).

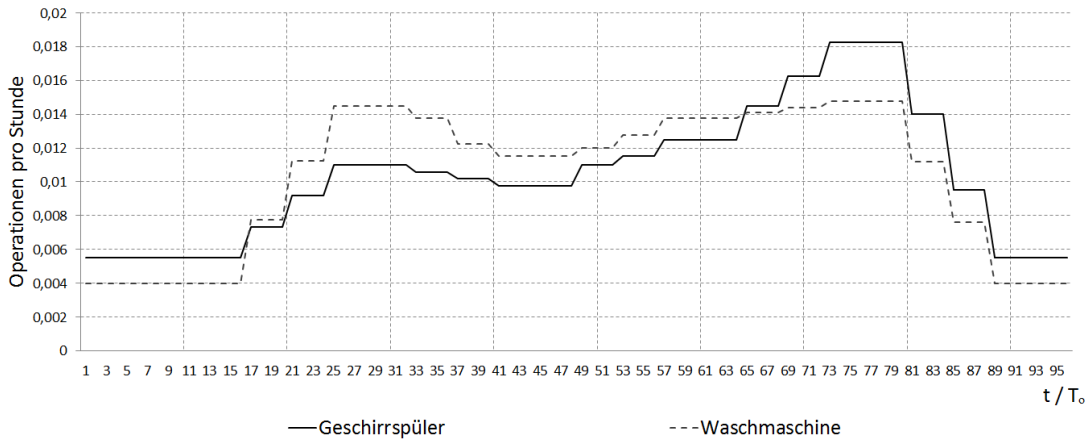

Abbildung 6.1.: Durchschnittliche Laufzeitwahrscheinlichkeiten von Geschirrspülern und Waschmaschinen in der EU

Die Laufzeitflexibilität, ausgehend von der in Abb. 6.1 aufgetragenen Laufzeitwahrscheinlichkeit, wird anhand der, in der Smart-A-Studie beschriebenen, Nutzung der Einschaltverzögerung an den jeweiligen Geräten berechnet. Zugrunde gelegt wird dabei der Zeitraum bis zu sieben Stunden nach der Benutzereingabe. Dieser Zeitraum wird dann auf die von den Nutzern durchschnittlich gewählte Dauer der Anschaltverzögerung skaliert. 56% der Nutzer wählen beispielsweise bei einer Waschmaschine einen Zeitraum zwischen 0 und 3 Stunden und 28% zwischen 4 und 6 Stunden. Bei der Berücksichtigung der Flexibilität von bis zu sieben Stunden werden dementsprechend ca. 84% der Nutzerpräferenzen berücksichtigt.

Die Studie zeigt auch, dass bei den Nutzern dieser Gerätetypen grundsätzlich von einer Akzeptanz für verzögertes Einschaltverhalten auszugehen ist. Die Spannbreite des Nutzungsgrades dieser Funktion bei Waschmaschinen bewegt sich zwischen etwa 38% in Schweden bis zu 81% in Italien (Deutschland: 45%), wobei die Nutzung der Einschaltverzögerung bei jedem Waschgang der Waschmaschine in Italien bei etwa 43% liegt (Deutschland: 12%). Die Verbreitung einer Zeitverzögerung lag EU-weit im Jahr 2007 bei rund 32%.

Bei Geschirrspülern ist die Situation vergleichbar. Hier waren 39% EU-weit mit Zeitverzögerungen ausgestattet, wobei etwa 27% der Nutzer angaben jedes mal die Einschaltverzögerung zu nutzen und 15% diese Funktion mindestens einmal die Woche einsetzten. Dem gegenüber steht die durchschnittliche Anzahl von 44% der Nutzer, die diese Funktion niemals nutzen. Etwa 66% der Nutzer wählen im europäischen Durchschnitt eine Startverzögerung von 0 bis 3 Stunden und 24% eine Startverzögerung von 4 bis 6 Stunden.

Tabelle 6.1.: Typische Verbrauchswerte von Geschirrspülern und Waschmaschinen in der EU im Jahr 2007

t in 15 min Intervallen	Geschirrspüler (P/W)	Waschmaschine (P/W)
0	80	100
1	2000	2000
2	80	900
3	80	100
4	80	100
5	2000	300
6	300	50
7	150	0

In Tabelle 6.1 ist die durchschnittliche Leistung von den im Jahr 2007 verfügbaren Geräten in 15-Minuten Intervallen dargestellt. Die Smart-A-Studie geht davon aus, dass bis 2020 die energieintensiven Aufheizvorgänge deutlich weniger Leistung benötigen werden. Da jedoch keine aktuelleren Daten vorliegen, werden die Smart-A Daten aus dem Jahr 2007 als Arbeitsgrundlage genutzt.

Das Vector-Device, welches eine Stromheizung repräsentiert, wurde mit einem Temperaturkorridor von $\pm$1,0 °C um die Standardraumtemperatur von 21 °C parametrisiert. Die Heizung kann in 15-Minuten-Intervallen ein- oder ausgeschaltet werden. Die Außentemperatur wurde anhand von Temperaturdaten des Hamburger Luftmessnetzes[1], Station Finkenwerder West, bestimmt. Grundlage für alle Simulationen mit Vector-Devices ist der Temperaturvektor des 15.09.2013, dem 259 Tag des Jahres 2013. Die betreffende Temperaturkurve ist in Abbildung 6.2 dargestellt.

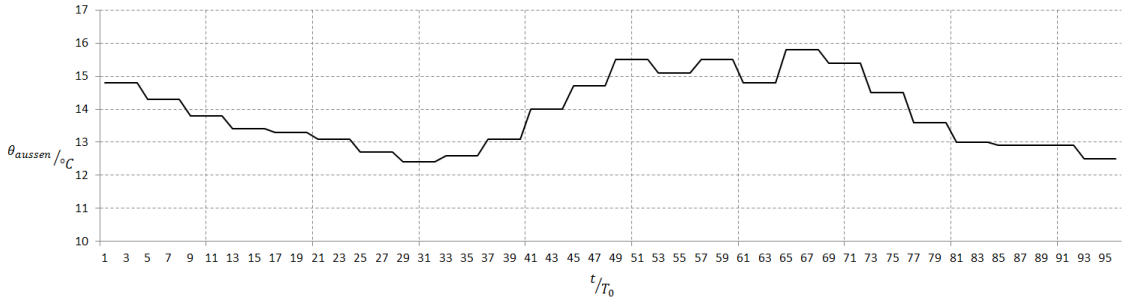

Abbildung 6.2.: Gemessene Außentemperatur am 15.09.2013 an der Station Finkenwerder West [Datenquelle: Hamburger Luftmessnetz]

Referenzszenario

Soweit bei den einzelnen Testcases oder Anwendungsfällen nicht anders angegeben wird eine kleine Liegenschaft mit 100 aktiven Haushalten betrachtet. In dieser Liegenschaft sind entsprechend der durchschnittlichen Verbreitung in Deutschland 95% Waschmaschinen, 62% Geschirrspüler und 4% E-Heizungen vorhanden, folglich 157 semiautomatische Agenten und 4 vollautomatische Agenten. Diese sind entsprechend obiger Beschreibung parametrisiert.

Wirtschaftsmodell

Das System kann eine Vielzahl von Betriebsmodellen mit dem implementierten Zwei-Tarife-Modell abbilden. Das Modell entspricht dabei einer Flexibilisierung des bei vielen Energiegrundversorgern[2] bereits vorhandenen Hoch- und Niedertarif-Modells. Dieser Anwendungsfall wurde für eine exemplarische Simulation weiterentwickelt. Es müsste in diesem Fall der Einkäufer eines kleinen oder mittleren Energieversorgungsunternehmens nicht mehr nach

[1] http://www.hamburger-luft.de/
[2] Als Beispiel: Arbon-Energie, Schweiz (http://www.arbonenergie.ch)

Standardlastprofil an der Börse einkaufen, sondern könnte einen Teil des Energiebedarfs der Liegenschaft anhand der Marktsituation decken, da die flexiblen Verbraucher auf diese Situation reagieren können. Der Einkäufer kann zwei Preise bilden. Einen niedrigen Preis für die eingekaufte Leistung und einen höheren Preis für Leistung, die außerhalb dieses Kontingents bezogen werden muss. Die Rolle des Einkäufers ist dabei die Bündelung der vorhandenen Angebote einer großen Börse (zum Beispiel der EEX) in ein einfacheres Tarifmodell für die betreffende Liegenschaft, welches aber durch die Marktbetrachtung ähnlichen Preisschwankungen wie die übergeordnete Börse unterliegt. In dieser Arbeit wurden daher reale Marktdaten verwendet, die qualitativ auf die Größe der zu simulierenden Einheit heruntergerechnet wurden.

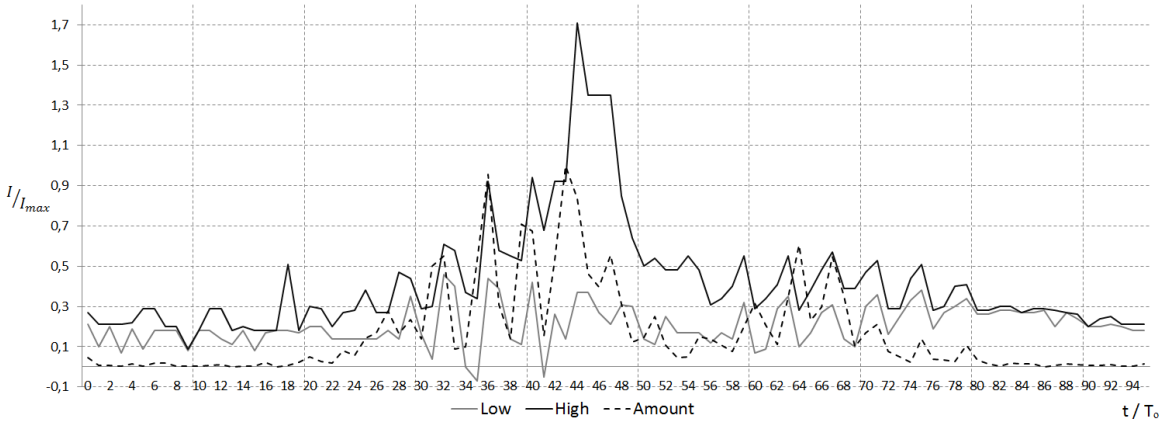

Abbildung 6.3.: Qualitative Auswertung der EPEX-Intraday-Daten vom 07.08.2013. Preise bezogen auf den höchsten Durchschnittspreis eines Slots [Datenquelle: epexspot.com]

Die Marktdaten wurden aus dem EPEX-Spotmarkt entnommen. Es handelt sich dabei um Intraday-Daten vom 07.08.2013. Verfügbar waren der Durchschnittspreis eines 15-minütigen Handelszeitraums, dem Höchst- sowie Niedrigstpreis und der insgesamt gehandelten Menge. Basierend auf dem Durchschnittspreis wurde die gehandelte Strommenge auf den Höchst- und den Niedrigstpreis aufgeteilt, sodass die zwei Tarife (Hoch- und Niedertarif) gebildet wurden. Der Strompreis wird dabei relativ zum Höchstpreis des durchschnittlichen Slot-Preises des Tageszeitraums angegeben, während die Strommenge relativ zur mittleren verfügbaren Menge des Tages angegeben wird. Um ein kompetitives Verhalten zu provozieren, wird die verfügbare Menge des Niederstromtarifs auf die oben berechnete, zwischen Hoch- und Niedertarif aufgeteilte Menge begrenzt. Der darüber hinausgehende Stromverbrauch wird mit dem Hochtarif berechnet und gilt als unendlich verfügbar. In Abbildung 6.3 ist die Low-Tarif-Kurve dementsprechend in Zusammenhang zu der verfügbaren Menge (Amount) zu sehen.

Ausgangswerte für die Optimierung

Die Parametrisierung der Optimierungsalgorithmen anhand von den Literaturangabe war deutlich aufwendiger, da im Bereich der Evolutionären Algorithmen unter den Autoren weitgehende Einigkeit besteht, dass die Systemparameter dem Anwendungsfall angepasst werden müssen (siehe de Castro (2006), Colorni u. a. (1994) und insbesondere Cantú-Paz (1998)). Zusätzlich bestand die Schwierigkeit, dass die Datenlage zu verteilten Implementierungen von ACO und GA relativ schwach war. Aufgrund der Ähnlichkeit Graphen-bezogener Problemstellungen könnten als Startparameter gängige TSP-Parameter verwendet werden, die dann in Abschnitt 6.2.4 näher auf die vorliegende Problemstellung spezifiziert werden.

Als Parameter-Grundlage für den GA werden die Daten von Nissen (1994) verwendet (siehe Tabelle 6.2). Für die Migrationsparameter in der hier definierten Problemdomäne, mit relativ wenigen Individuen pro Population und daher einer hohen Konvergenzrate, könnten laut Belding (1995) und Dethlefs u. a. (2013) hohe Austauschfrequenzen bessere Ergebnisse liefern. Belding (1995) erzielte mit einem Austauschverfahren, bei dem eine zufällige, andere Population für den Austausch ausgewählt wurde, die besten Ergebnisse.

Tabelle 6.2.: Ausgangswerte für den GA-Optimierer

Parameter	Wert	Bemerkung
Iterationen	100 - 1000	Wert pro Population
Populationsgröße	8	Quelle: Nissen (1994)
$p_{crossover}$	0,6	Quelle: Nissen (1994)
$p_{mutation}$	0,005	$p_{mutation} \ll 1$ Quelle: Nissen (1994)
$f_{exchange}$	5 - 25	Quelle: Dethlefs u. a. (2013) & Belding (1995)
Austausch	beste 2	Quelle: Nissen (1994)
Austauschmodus	verlustbehaftet	Quelle: Nissen (1994)
Austauschziel	Zufällige Population	Quelle: Belding (1995)

Für die Startwerte des ACS werden die Ausgangswerte von Dorigo und Gambardella (1997) genutzt (Tabelle 6.3), da diese, abgesehen von dem t_0-Parameter, relativ unabhängig von der Problemstellung sein sollen. Für die Anzahl von Ant-Agenten schlägt Dorigo u. a. (2006) vor, *N = Anzahl der Knoten* zu wählen. Es sei erwähnt, dass sich auf den Testsystemen zeigte, dass eine Anzahl von acht Ant-Agenten pro Kolonie bei vier optimierenden Agenten (= 32 Ant-Agenten) ein Achtkern-System voll auslasten konnten.

Ein wichtiger Aspekt ist die Vergleichbarkeit beider Ansätze. Ein Maß dafür könnte die Anzahl der (theoretisch) evaluierten Pfade sein. Bei einem Optimierer, der 1000 Iterationen mit einer Kolonie von acht Ameisen durchläuft, werden 8000 Pfade traversiert. Eine GA-Population von acht Individuen tausend mal iteriert, würde ebenfalls 8000 Pfade traversieren. Aufgrund der

Tabelle 6.3.: Ausgangswerte für den ACS-Algorithmus von Dorigo und Gambardella (1997)

Parameter	Wert	Bemerkung
Iterationen	100 - 1000	Wert pro Kolonie
N	8	Siehe Text
α	1,0	Pfadlängengewichtung
β	2,0	Pheromongewichtung
q_0	0,9	Exploitation-Wahrscheinlichkeit
ρ	0,1	Globale Pheromonverlustrate
t_0	10^{-6}	Initialer Pheromongehalt, Quelle Dorigo u. a. (1996)
φ	0,1	Lokale Pheromonverlustrate, an ρ angelehnt

speziellen Eigenschaften der Heuristiken handelt es sich dabei jedoch nicht um die absolute Anzahl von traversierten Pfaden, da Pfade aufgrund der Konvergenzeigenschaften beider Algorithmen mehrfach vorkommen können. Es sollte sich dabei aber dennoch um ein geeignetes Maß zum Leistungsvergleich handeln, da Pfaddopplungen bei beiden Algorithmen vorkommen können.

6.2. Testcases

Die im Analyseteil definierten Testcases sollen die korrekte Funktionalität des Systems nachweisen. Dazu wird zu Beginn geprüft, ob das System die Agenten korrekt parametrisiert. Ferner wird das Verhalten der Vectordevices anhand ihrer Temperaturkurve untersucht. Ist das Verhalten der Agenten im Sinne der Eingangsparametrisierung validiert, werden Tests zur Funktion der Subsimulation durchgeführt. Zuletzt werden die Optimierungsalgorithmen eingehend auf ihre Funktionalität hin untersucht und es wird versucht, domänenspezifische Startparameter zu bestimmen.

6.2.1. Validierung der statistischen Modelle

In diesem Abschnitt soll die Parametrisierung der Semiautomatic-Devices mittels Massensimulation validiert werden. Dazu werden 1000 Semiautomatic-Agenten gestartet und ihre Startparametrisierung aufgenommen. Das Verfahren wurde zehn mal wiederholt, sodass für die Auswertung der Startzeitwahl 10.000 Geräte simuliert wurden. Erwartet wird, dass sich das resultierende Startzeitenhistogramm an den Erwartungswert annähert. In Abbildung 6.4 ist dies für Waschmaschinen vorgenommen worden. In der Abbildung sind tatsächliche Startzeiten, Erwartungswert und Residuenquadrate abgebildet. Die durchschnittliche Abweichung

beträgt 0,97. Die größte Abweichung beträgt 3,30 zum Zeitpunkt 24. Insgesamt zeigt sich der Verlauf der simulierten Startzeiten ähnlich dem Erwartungswert, sodass der verwendete Java-Zufallszahlengenerator *Math.random()* für diese Anwendung ausreichend ist. Dieser wird laut Java-Dokumentation von Oracle[3] auch für einfache Anwendungen empfohlen, in denen keine weiterführenden Anforderungen nötig sind.

Abbildung 6.4.: Statistische Auswertung der Startzeiten von 1000 Waschmaschinen

Die Simulation von Geschirrspülern zeigt, ähnlich der vorigen Simulation, einen vergleichbaren Verlauf wie die Kurve des Erwartungswertes. Die mittlere Abweichung ist hier etwas höher mit 1,13. Auch hier kann davon ausgegangen werden, dass der von Oracle empfohlene Zufallszahlengenerator für diese Experimente ausreichend ist.

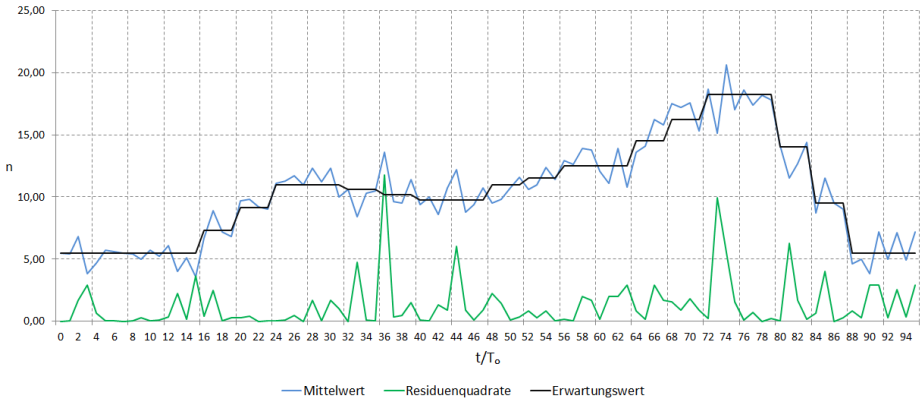

Abbildung 6.5.: Statistische Auswertung der Startzeiten von 1000 Geschirrspülern

[3] http://docs.oracle.com/javase/7/docs/api/java/util/Random.html (Aufgerufen am 23.02.14)

6.2.2. Validierung des Heizungsmodells

In diesem Testcase soll gezeigt werden, dass das Heizungsmodell unter Einhaltung der Temperaturgrenzen eine Optimierung der Heizpulse vornimmt. Zugrunde liegt das eingangs beschriebene EPEX-Spotmarkt Preismodell. Da nur eine einzige Heizung betrachtet wird, brauchen die Mengen für den niedrigen Preis nicht beachtet zu werden. Der erlaubte Temperaturbereich wurde, um eine bessere Übersicht zu ermöglichen, sehr breit gewählt und liegt zwischen 19,5 °C und 22,5 °C. Als Außentemperatur wurde der Temperaturvektor des 15.09.2013 genutzt. Nebeneffekte, wie Sonneneinwirkung, wurden nicht beachtet. Der Versuch wurde mit beiden Optimierungsalgorithmen mit jeweils 1000 Iterationen auf vier Optimierungsagenten mit den oben bereits definierten Ausgangswerten durchgeführt.

Wird die Preiskurve in drei gleiche Abschnitte geteilt, ergeben sich drei unterschiedliche Preisniveaus. Im ersten Tagesdrittel beträgt der Durchschnittspreis 0,16 Preiseinheiten (PE), im zweiten Tagesdrittel 0,22 PE und im dritten Tagesdrittel 0,24 PE. Erwartet wurde, dass eine optimierende Heizung dementsprechend im ersten Tagesdrittel eine etwas höhere Durchschnittstemperatur anstrebt, um die Heizung in den anderen Tagesdritteln etwas weniger betreiben zu müssen.

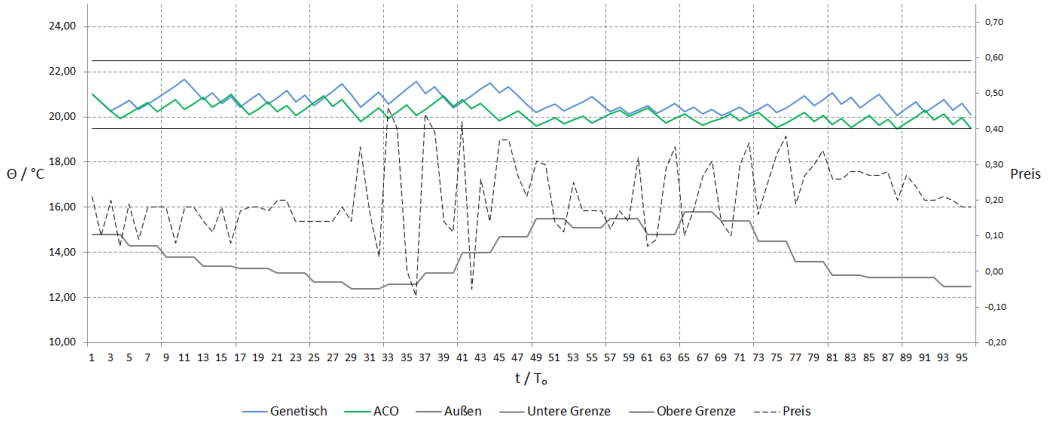

Abbildung 6.6.: Auswertung eines Vector-Devices mit den verschiedenen Optimierungsalgorithmen

Das Versuchsergebnis ist in Abbildung 6.6 dargestellt. Es zeigt sich, dass die ACO-Optimierung näher an der unteren Temperaturgrenze (Durchschnitt: 20,16 °C) ist, während der Genetische Algorithmus eine etwas höhere Durchschnittstemperatur von 20,68 °C aufweist. Entsprechend der Erwartung beträgt im ACO-Fall die Durchschnittstemperatur im ersten Drittel 20,46 °C, im zweiten Drittel 20,15 °C und im letzten Tagesdrittel bei dem höchsten Durchschnittspreis 19,88 °C. Beim GA ist das Verhalten analog.

6.2.3. Validierung der Subsimulation

In der eventbasierten Subsimulation wurden zwei Abweichungstypen implementiert. Der erste Abweichungstyp ist die Umsetzung einer spontanen Temperaturänderung eines thermischen Verbrauchers. In dem Testszenario wurden die Heizungsparameter und die EPEX-Kurve aus dem vorigen Testfall genutzt. Simuliert wurde mit dem ACO-Algorithmus mit den Ausgangsparametern und 1000 Iterationen. Zum Zeitpunkt 30 wurde die Raumtemperatur auf 18,5 °C gesetzt. Erwartet wurde eine anhaltende Heizphase, um die Temperatur wieder in den erlaubten Temperaturkorridor anzuheben und diese sich danach in der Nähe der unteren Temperaturgrenze einpendelt.

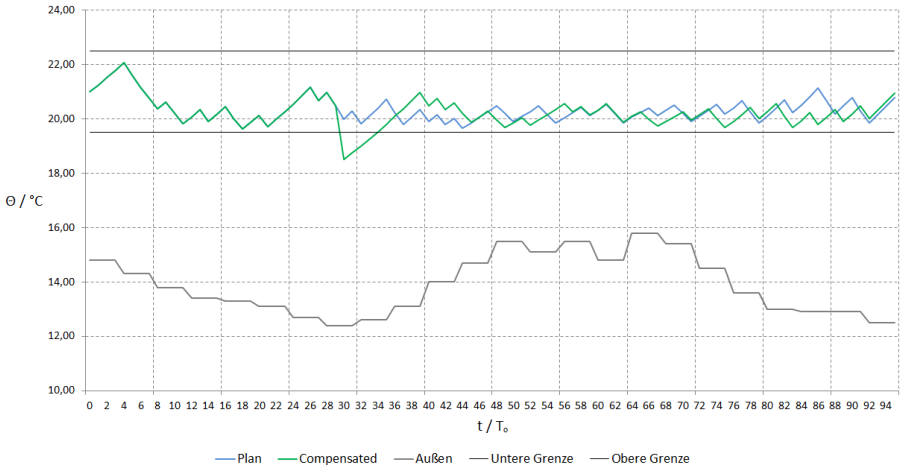

Abbildung 6.7.: Kompensation eines abrupten Temperaturabfalls

In dem Versuch (Abb. 6.7) zeigte sich, dass dieses Verhalten durch den Optimierungsalgorithmus erreicht wurde. Nach der abrupten Temperaturänderung gewährleisten der Optimierungsalgorithmus und das interne Simulationsmodell ohne weiteres Zutun, dass die Temperatur wieder über die Temperaturgrenze steigt.

Der zweite implementierte Abweichungstyp realisiert eine Änderung des Marktmodells. An einem definierten Zeitpunkt wird in der Subsimulation ein anderes Marktmodell gewählt, woraufhin das System ab diesem Zeitpunkt eine partielle Neuoptimierung vornimmt. Ein Anwendungsfall wäre, wenn beispielsweise Verbraucher auf die Prognose einer Windkraftanlage am Vortag optimiert wurden und ab einem bestimmten Zeitpunkt des betreffenden Tages die Prognose nicht mehr zutrifft und sich die Verbraucher somit auf eine neue (angepasste) Prognose optimieren müssen.

Zur Verifikation dieses Verhaltens wurde ein Versuch mit der Standardliegenschaft durchgeführt. Im ersten Marktmodell ist der Preis in jedem Zeitabschnitt konstant eins, außer in

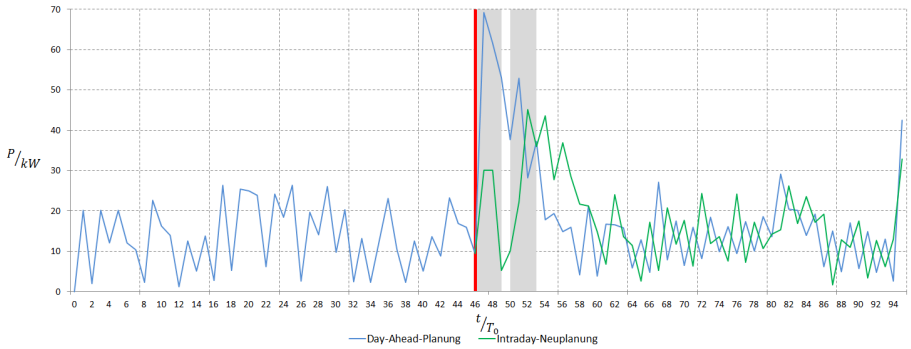

Abbildung 6.8.: Simulation einer Preissignalverschiebung

dem Zeitabschnitt $t_1 = \{46, \ldots, 49\}$ in dem der Preis 0,01 PE beträgt. In Abbildung 6.8 ist dieser Zeitraum durch den linken grauen Bereich gekennzeichnet. Wie erwartet zeigt sich durch die Optimierung eine deutliche Lastspitze in diesem Zeitkorridor (blaue Kurve). In der Subsimulation wird zum Zeitpunkt $t = 46$ der günstige Korridor auf $t_2 = \{50, \ldots, 53\}$ verschoben (rechter grauer Bereich in Abbildung 6.8). Die Lastspitze verlagert sich jedoch nur teilweise, da einige Heizungen, aufgrund ihrer vorhergehenden Planung, die Heizpulse nicht weiter aufschieben können (grüne Kurve).

6.2.4. Vergleich der Optimierungsalgorithmen

In diesem Abschnitt sollen die Optimierungsalgorithmen validiert und deren Parameter überprüft werden. Dazu wurden Daten des EPEX-Szenarios an die Verbrauchsdaten der Liegenschaft angepasst und, entsprechend der oben beschriebenen Parametrisierung der Standardliegenschaft, eine Liegenschaftsgröße von 100 aktiven Haushalten angenommen, mit 95 Waschmaschinen, 62 Geschirrspüler und vier E-Heizungen als Verbraucheragenten.

Evaluation des Genetischen Algorithmus

Beim Genetischen Algorithmus zeigte sich bereits in ersten Simulationen, dass die Parameter von Nissen (1994) nur unzureichende Ergebnisse im Vergleich zu denen des ACS brachten. Daher wurden grundlegende Daten aus der Arbeit von Belding (1995) zurate gezogen. Belding (1995) nutzt eine Zahl von 20 Individuen pro Population. Da aus diversen Arbeiten hervorgeht, dass mehr Individuen pro Population vorteilhaft sein könnten, wird auch die doppelte Anzahl (= 40) Individuen pro Population untersucht. Die Crossover-Wahrscheinlichkeit wird auf $P_c = 1.0$ geändert und alle fünf Iterationen werden 50% der Individuen migriert,

entsprechend der Ergebnisse von Belding (1995). Eine weitere vielversprechende Verbesserung ist das sogenannte Sigma Scaling. Dabei werden die Fitnesswerte auf einen gewissen Bereich abgebildet, um eine vorzeitige Konvergenz zu vermeiden. Genutzt wird die von Mitchell (1998) beschriebene Skalierungfunktion:

$$ExpVal(i,t) = \begin{cases} 1 + \frac{f(i) - \bar{f}(t)}{2\sigma(t)} \text{ wenn } \sigma(t) \neq 0 \\ 1.0 \text{ wenn } \sigma = 0, \end{cases} \tag{6.1}$$

mit ExpVal(i,t) als skalierten Fitnesswert des Individuums i in der Iteration t, $f(i)$ als Fitnesswert des Individuums, $\bar{f}(t)$ als dem Mittelwert der Fitness der Iteration t und $\sigma(t)$ als Standardabweichung.

Für die Optimierung wurden vier Optimierungsagenten genutzt und für jede der drei Populationsgrößen 8, 20 und 40 wurden Werte zwischen 800 und 80.000 traversierten Pfaden untersucht. Bei einer Populationsgröße von 40 Individuen bei vier Optimierungsagenten bedeuten 80.000 traversierte Pfade 500 Optimierungsiterationen pro Population ($4 \cdot 40 \cdot 500 = 80000$). Es wurden 50 Simulation pro Datenpunkt mit dem oben definierten Referenzszenario durchgeführt und der Mittelwert gebildet.

Mit dem Ziel, gute Startwerte für den Genetischen Algorithmus zu erzeugen, wurden zuerst akzeptable Werte für eine Populationsgröße und eine Iterationsanzahl bestimmt. Akzeptabel bedeutet in diesem Zusammenhang ein Kompromiss zwischen Ergebnis und Zeitbedarf. Daraufhin sollte der Einfluss der Mutationsrate untersucht werden. Mit der besten Mutationsrate wurde danach der Einfluss des Crossovers evaluiert und daraufhin die für die Domäne beste Migrationsrate bestimmt. Abschließend soll die Wirkung des Sigma Scalings näher untersucht werden.

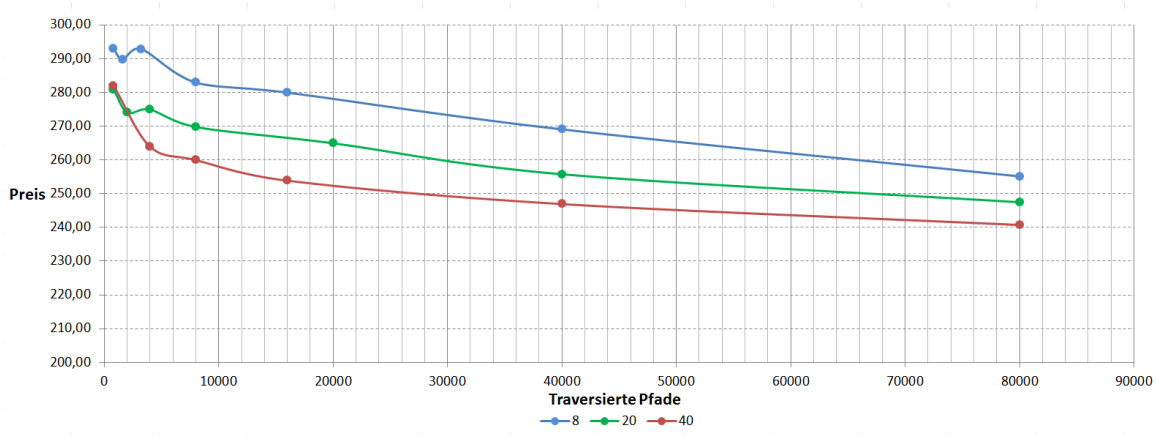

Abbildung 6.9.: Untersuchung des Einflusses der Populationsgröße auf die Fitness, Durchschnitt aus 50 Experimenten

In Abbildung 6.9 sind die Ergebnisse der Populationssimulation über die Anzahl der traversierten Pfade aufgetragen. Der erwartete Verlauf, zu Beginn ist mit wenigen Iterationen eine große Verbesserung zu erzielen, später muss für kleine Verbesserungen ein großer Aufwand betrieben werden, konnte gezeigt werden. Es ist ferner zu entnehmen, dass die Optimierung mit 40 Individuen pro Population die besten Ergebnisse erzielt hat.

Für weitere Untersuchungen wurden 40 Individuen bei 100 Iterationen genutzt. Dieser Punkt bei 16.000 traversierten Pfaden ($4 \cdot 40 \cdot 100 = 16000$) ist der Bereich, ab dem sich die Optimierung relativ stabil verhält, also die anfängliche Phase mit dem starken Abstieg abgeschlossen ist. Dieser Bereich wurde auch deswegen gewählt, weil der Zeitbedarf mit etwa 3 Minuten für eine Optimierung akzeptabel ist.

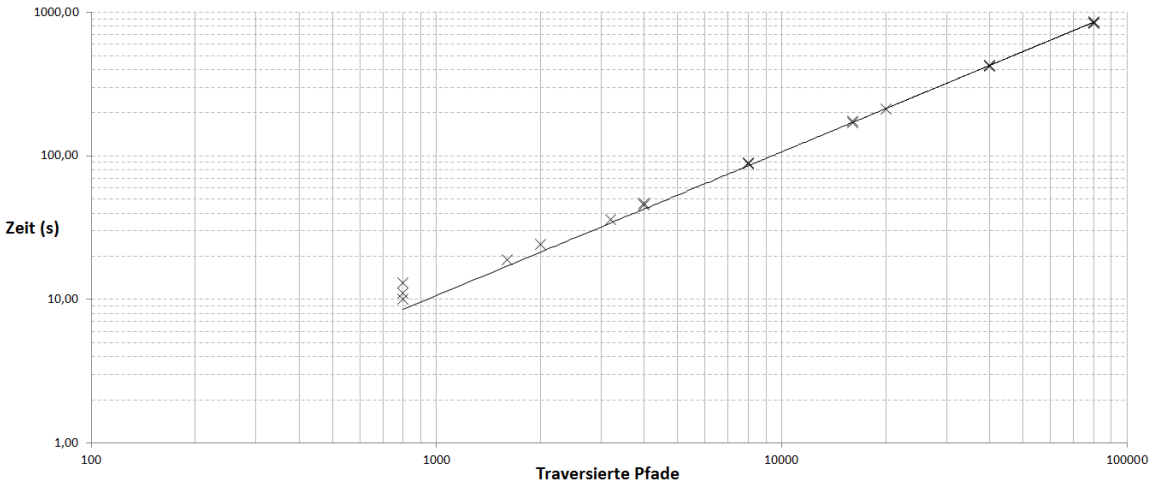

Abbildung 6.10.: Zeitbedarf des Genetischen Algorithmus zur Optimierung (log/log-Skala)

Werden die benötigten Zeiten über die traversierten Pfade aufgetragen, wird erwartet, dass in der Anfangsphase der Zeitbedarf aufgrund des grundsätzlichen Zeitbedarfs für die Initialisierung der Optimierung leicht erhöht sein sollte und sich daraufhin einer Grade annähert. In der Abbildung 6.10 wurden diese Daten aus dem obigen Versuch entnommen und aufgrund der großen Zeitdifferenzen auf einer Log/Log-Skala aufgetragen. Es zeigt sich, dass der Zeitbedarf nahezu linear von der Anzahl der traversierten Pfade abhängt. Dabei ist es unerheblich, ob die Populationsgröße höher ist, oder mehr Iterationen gewählt wurden.

Die Anpassung der Mutationsrate (siehe Abb. 6.11) kann zu einer signifikanten Verbesserung des Ergebnisses führen, in diesem Fall um rund 18 Preiseinheiten vom ursprünglichen Wert $P_m = 0,005$ zum besten Wert $P_m = 0,5$. Danach steigt der Preis wieder an. Dies zeigt, dass eine zu häufige Mutation dem Konvergenzverhalten und der Stabilität des Algorithmus schadet. Die Überprüfung der Crossoverwahrscheinlichkeit zeigte, dass es sich dabei um den wichtigsten Operator handelt. Abgeschaltet beträgt die Durchschnittspreis nur

noch 305,00 PE und ist damit nur noch etwa 10% besser als eine Liegenschaft ohne Optimierung. Die optimale Crossoverwahrscheinlichkeit für diese Anwendungsdomäne ist der bereits vorher genutzte Wert von $P_c = 1.0$.

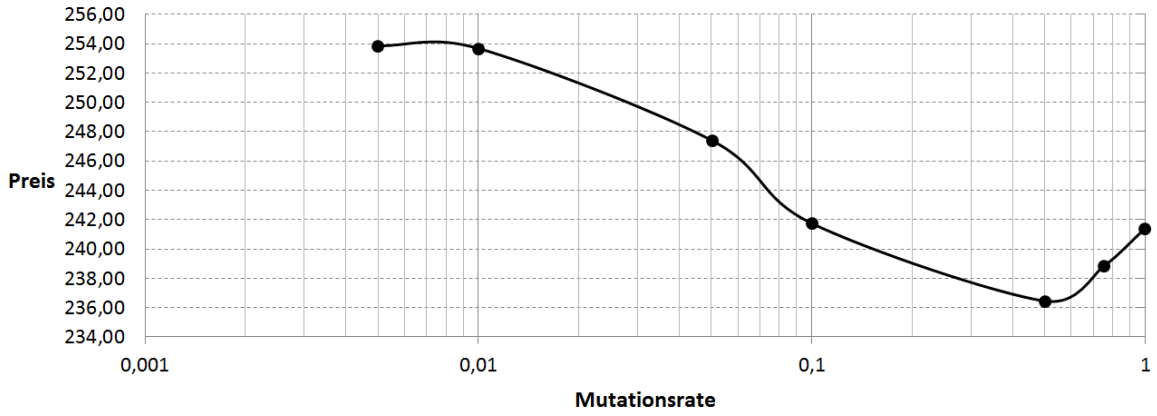

Abbildung 6.11.: Wirkung unterschiedlicher Mutationsraten von 0,005 bis 1 auf die Fitness

Bei der Untersuchung der Migration wurden drei Migrationsraten getestet. Zuerst wurde der von Belding (1995) bestimmte Bestwert des Austauschintervalls von 5 getestet, dann die Verdopplung auf 10 und eine weitere Verdopplung auf 20. Migriert wurden jeweils 25%, 50% (Bestwert bei Belding (1995)) und 75% der jeweiligen Population. Erwartet wurden ähnliche Ergebnisse wie bei Belding (1995), ein Optimum bei einem Migrationsintervall von 5 und einem Migrationsanteil von 0,5, was 50% der Population entspricht. Die Notation ist im Ergebnisbild (Abb. 6.12) wie folgt: (Migrationsintervall; Anteil der auszutauschenden Individuen an der Gesamtpopulation)

Die in Abbildung 6.12 aufgetragenen Ergebnisse des Experimentes zeigen, entgegen der Erwartung, dass eine abgeschaltete Migration die besten Ergebnisse der Fitness bei der zweitniedrigsten Standardabweichung erzielt. Eine genaue Untersuchung der einschlägigen Literatur zeigte, dass dieses Phänomen für bestimmte Problemstellungen bekannt ist und bei diesen auch konventionelle GA überlegen sein können (siehe Cantú-Paz (1998), Belding (1995)).

Beim letzten Experiment zeigte sich, dass das Sigma-Scaling durch die Skalierung der Fitnesswerte einen deutlichen Einfluss auf das Ergebnis hat. Mit Sigma-Scaling beträgt der Durchschnittswert 233,66 PE, ohne 276,38 PE.

Die Ergebnisse der sequentiellen Bestimmung der Optimierungsparameter sind in Tabelle 6.4 eingetragen. Mit diesen werden die folgenden Anwendungsfälle untersucht.

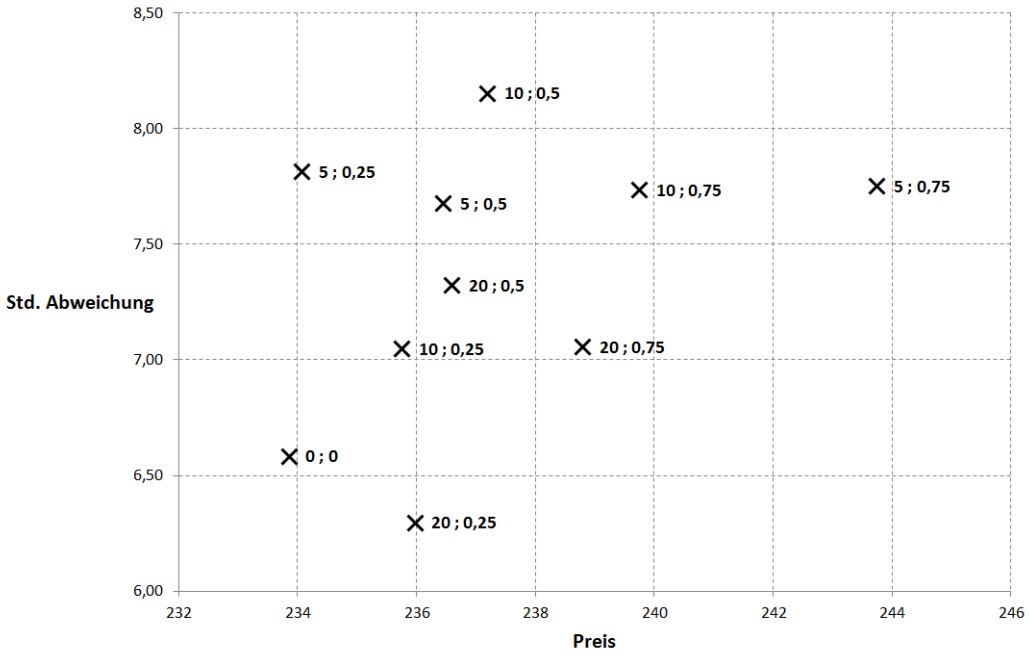

Abbildung 6.12.: Untersuchung der Migrationsraten 5, 10, 20, jeweils mit Anteil der migrierten Individuen, im Vergleich mit abgeschalteter Migration (0;0)

Tabelle 6.4.: Optimierte Werte des Genetischen Algorithmus

Parameter	**Wert**	**Bemerkung**
Iterationen	> 100	Wert pro Population
Populationsgröße	40	pro Optimierer
Optimierer	4	
$p_{crossover}$	1,0	
$p_{mutation}$	0,5	
Austausch	Abgeschaltet	

Evaluation der Ant Colony Optimization

In das Simulationssystem wurde eine verteilte ACS-Optimierung implementiert und nach Dorigo und Gambardella (1997) parametrisiert. Zur Verfeinerung der Parametrisierung wurde die Optimierung mit acht Ant-Agenten pro Kolonie in vier Kolonien durchgeführt und die Anzahl der Iterationen zwischen 25 und 1000 pro Kolonie variiert, sodass $4 \cdot 8 \cdot 25 = 800$ bis $4 \cdot 8 \cdot 1000 = 32.000$ Pfade traversiert wurden. Dabei wurden auch verschiedene Modi des ACS evaluiert. Wird das Pheromongewicht $\beta = 0$ gesetzt, wählt der Algorithmus den kürzesten Pfad zwischen zwei Nachbarn nach der *Nearest-Neighbor-Heuristik* und verhält sich somit wie ein Greedy-Algorithmus. Werden die lokale Pheromonenanpassung und die Exploitation q_0 deaktiviert, verhält sich der ACS wie das klassische Ant-System (AS).

Für jeden Iterationswert wurden 50 Simulationen mit dem Referenzszenario durchgeführt und der Mittelwert gebildet. Es wird erwartet, dass der Greedy-Ansatz die schlechtesten Ergebnisse liefert, da dieser aufgrund der Auswahlheuristik anfällig für suboptimale Pfade ist. AS sollte dementsprechend deutlich bessere Ergebnisse liefern und der ACS aufgrund der, laut Dorigo u. a. (2006) verbesserten Eigenschaften, die besten Ergebnisse.

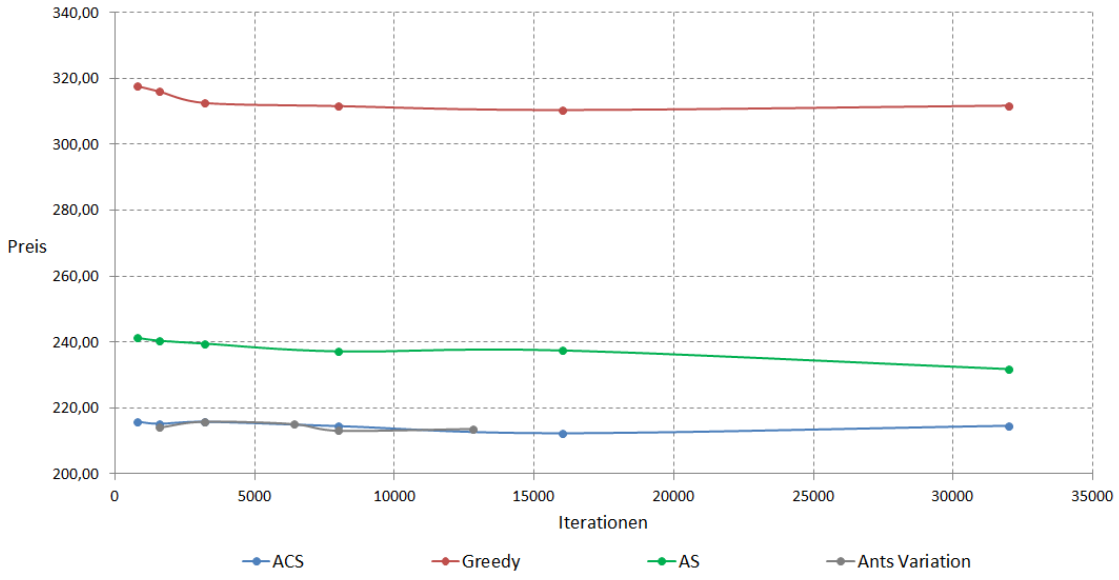

Abbildung 6.13.: Untersuchung verschiedener Ant-Colony-Optimierungsverfahren (Greedy, AS, und ACS)

In Abbildung 6.13 sind die Ergebnisse der Versuche aufgetragen. Die Algorithmen verhalten sich wie erwartet. Hervorzuheben ist, dass der ACS-Algorithmus bereits nach wenigen Iterationen sehr gute Ergebnisse liefert, die sich durch eine Erhöhung der Iterationszahl nur noch unwesentlich verbessern lassen. Um zu prüfen, ob die Anzahl der Ant-Agenten, oder

die Anzahl der Iterationen größeren Einfluss auf das Ergebnis hat, wurde bei einer konstanten Zahl von 100 Iterationen die Anzahl der Ant-Agenten zwischen 4 und 32 variiert und den vorigen Ergebnissen gegenübergestellt (graue Linie in Abb. 6.13). Die Unterschiede sind jedoch gering, was darauf hinweist, dass das globale Pheromonenupdate, dessen Häufigkeit durch die Variation der Ant-Agentenzahl ebenfalls fluktuiert, keinen bedeutenden Einfluss auf das Ergebnis hat. Der AS-Algorithmus erzielt ähnliche Werte wie der optimierte Genetische Algorithmus und ist damit im Durchschnitt etwa 10% schlechter als die ACS-Optimierung. Verglichen mit den Resultaten des Genetischen Algorithmus liefern die Startwerte von Dorigo und Gambardella (1997) bereits sehr gute Ergebnisse.

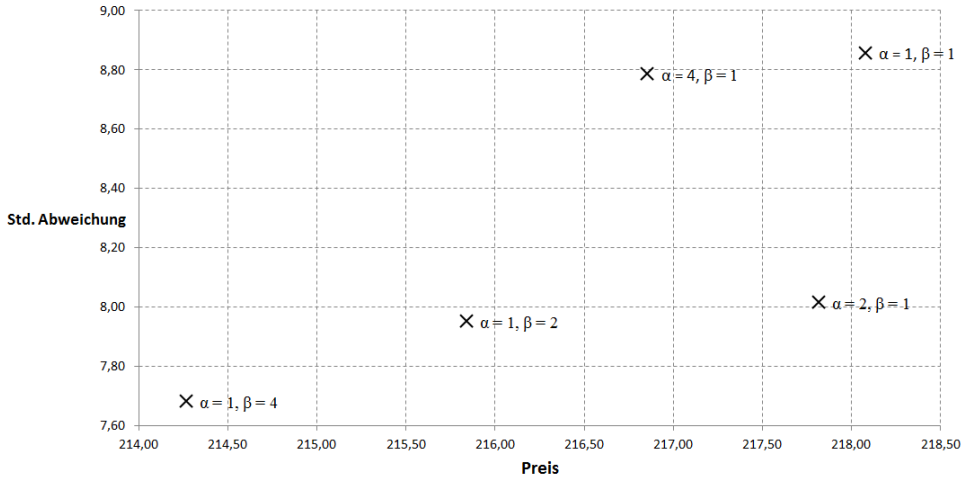

Abbildung 6.14.: Vergleich zwischen verschiedenen α und β Parametern des ACS

Ausgehend von den obigen Ergebnissen sollte der Einfluss des Verhältnisses von Pfad- (α) zu Pheromonengewichtung (β) beim ACS untersucht werden. Um den Zeitbedarf zu begrenzen, wurde die Zahl der Iterationen pro Kolonie auf 100 begrenzt, sowie die Anzahl der Ant-Agenten pro Kolonie auf acht, sodass bei vier Optimierern 3.200 Pfade traversiert werden. Es wurde erwartet, dass ein höheres β bessere Ergebnisse liefert, da dies die Wirkung der spezielle ACS-Heuristik gegenüber der Nearest-Neighbor-Heuristik verstärkt. Die in Abbildung 6.14 dargestellten Ergebnisse unterstützen die Erwartung. Jeder Datenpunkt entspricht dem gemittelten Wert aus 50 Simulationen. Es zeigt sich, dass eine stärkere Gewichtung von β zu deutlich besseren Ergebnissen, sowohl in Bezug auf den Preis, als auch der Standardabweichung, führt.

Zuletzt soll der Einfluss der Exploitation-Wahrscheinlichkeit q_0 untersucht werden. Dorigo und Gambardella (1997) empfehlen $q_0 = 0.9$. Ausgehend von der vorigen Evaluation wurde der Algorithmus wieder mit den Startwerten und 100 Iterationen bei acht Ant-Agenten, verteilt auf vier Kolonien parametrisiert und die Werte über 50 Experimente gemittelt. Es zeigt sich,

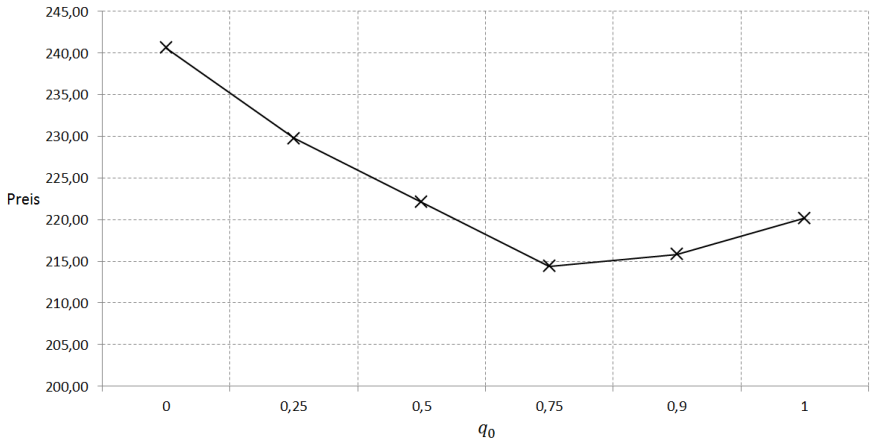

Abbildung 6.15.: Untersuchung des q_0 Parameters bei ACS

dass höhere q_0 Werte die Ergebnisse des Algorithmus deutlich verbessern. Das Optimum liegt jedoch in diesem Falle nicht, wie bei Dorigo und Gambardella (1997) angegeben, bei $q_0 = 0,9$, sondern etwas niedriger bei $q_0 = 0,75$.

Insgesamt zeigt sich, dass die von Dorigo und Gambardella (1997) angegebenen Werte sehr gute Ausgangsergebnisse liefern und wenig domänenspezifisches Verbesserungspotential bieten. Die in Tabelle 6.5 aufgetragenen Werte für die weiteren Versuche sind demnach sehr nahe an den Startparametern. Im Vergleich zum Genetischen Algorithmus liefert der ACS-Algorithmus deutlich bessere Werte, benötigt aber für die gleich Anzahl an traversierten Pfaden um den Faktor 3 - 4 mehr Zeit. Selbst bei nur 800 traversierten Pfaden erzielt der ACS-Algorithmus jedoch Ergebnisse, die beim Genetischen Algorithmus auch bei über 80.000 traversierten Pfaden nur in Ausnahmefällen erreicht werden, wodurch sich der Zeitunterschied wieder relativiert.

Tabelle 6.5.: Optimierte Werte des ACS-Algorithmus

Parameter	**Wert**	**Bemerkung**
Iterationen	100	Wert pro Kolonie
Koloniegröße	8	pro Optimierer
Optimierer	4	
α	1	Pfadlängengewichtung
β	4	Pheromonengewichtung
q_0	0,75	Exploitation-Wahrscheinlichkeit
ρ	0,1	Globale Pheromonenverlustrate
t_0	10^-6	Initialer Pheromongehalt
φ	0,1	Lokale Pheromonenverlustrate

6.3. Day-Ahead Optimierung

Eine klassische Anwendung für die Optimierung von Verbrauchern ist die Day-Ahead-Optimierung. Dabei werden plan- und steuerbare Lasten auf einen vorgegeben Ziellastgang optimiert. Die Day-Ahead-Optimierung soll hier nur als Beispiel für eine Vielzahl von Anwendungsfällen aus der Verbrauchsplanung sein, bei denen unterschiedliche Zeitskalen oder Anforderungen im Vordergrund stehen, so zum Beispiel bei der Planung der Minutenreserve oder der im nächsten Abschnitt behandelten Lastverlagerung durch Verbundoptimierung.

In der Praxis ist es häufig erforderlich, den Stromverbrauch so zu planen, dass systemimmanente volatile DERs möglichst optimal genutzt werden können. Die Erzeugungsleistung von Windkraftanlagen ist anhand von Wettervorhersagen inzwischen bereits am Vortag in der Regel mit hinreichender Genauigkeit planbar. Da für den Betrieb der Anlage keine Ressourcen erforderlich sind, ist die erzeugte Energie günstig verfügbar. Wird der Verbrauch anhand der Erzeugung geplant, muss in geringerem Maße externe (und damit häufig teure) Energie dazugekauft werden und die Nutzer sparen Geld bei gesteigerter Effizienz.

In diesem Fall soll die erzeugte Energie einer 330 kW Windkraftanlage vom Typ Enercon E-33 dazu dienen, den Bedarf der flexiblen Verbraucher in der Referenzliegenschaft zu decken. Dazu wurde die Stromerzeugung der Anlage mit den Wetterdaten vom 15.09.2013 und dem Greenius-Tool[4] des DLR simuliert und auf den steuerbaren Strombedarf der 100 Haushalte umfassenden Referenzliegenschaft skaliert, sodass 10% der Nennleistung der Anlage zur Deckung des Energiebedarfs der flexiblen Verbraucher zur Verfügung steht.

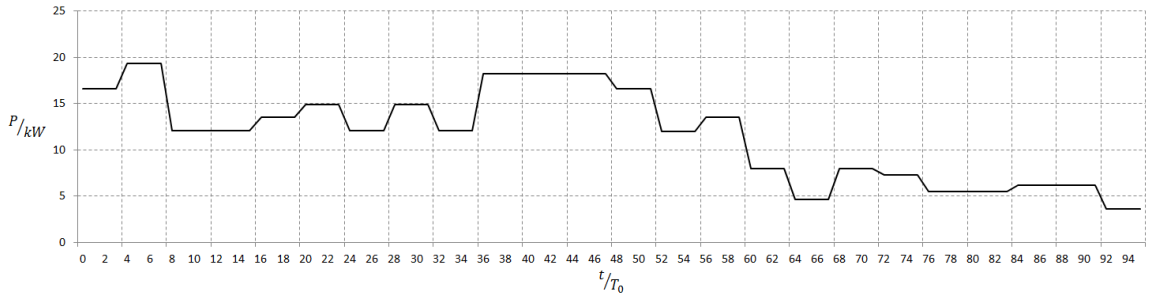

Abbildung 6.16.: Simulation einer Enercon E-33 mit Winddaten vom 15.09.13, Skaliert auf 10% der Nennleistung

Die hier simulierte Windenergieerzeugung ist extrem atypisch für die Liegenschaft, da diese durch die hohe Anzahl von programmgetriebenen Verbrauchern erst in den Abendstunden mehr Energie benötigt. Dementsprechend kann auch von dem optimierten Fahrplan nicht erwartet werden, dass dieser die zur Verfügung stehende Energie vollständig ausnutzen kann.

[4] http://freegreenius.dlr.de , aufgerufen am 13.03.14

Die verbesserte Ausnutzung der Freiheitsgrade der vier vollautomatischen Verbraucher sollte jedoch für signifikante Einspareffekte sorgen.

Es wurden zehn Simulationen im gesteuerten Betrieb mit dem ACS und zehn Simulationen im ungesteuerten Betrieb durchgeführt und die Mittelwerte gebildet, um die durchschnittlichen Einsparpotentiale zu ermitteln. In Abbildung 6.17 sind die Verbrauche vom gesteuerten und freien Betrieb gegenübergestellt.

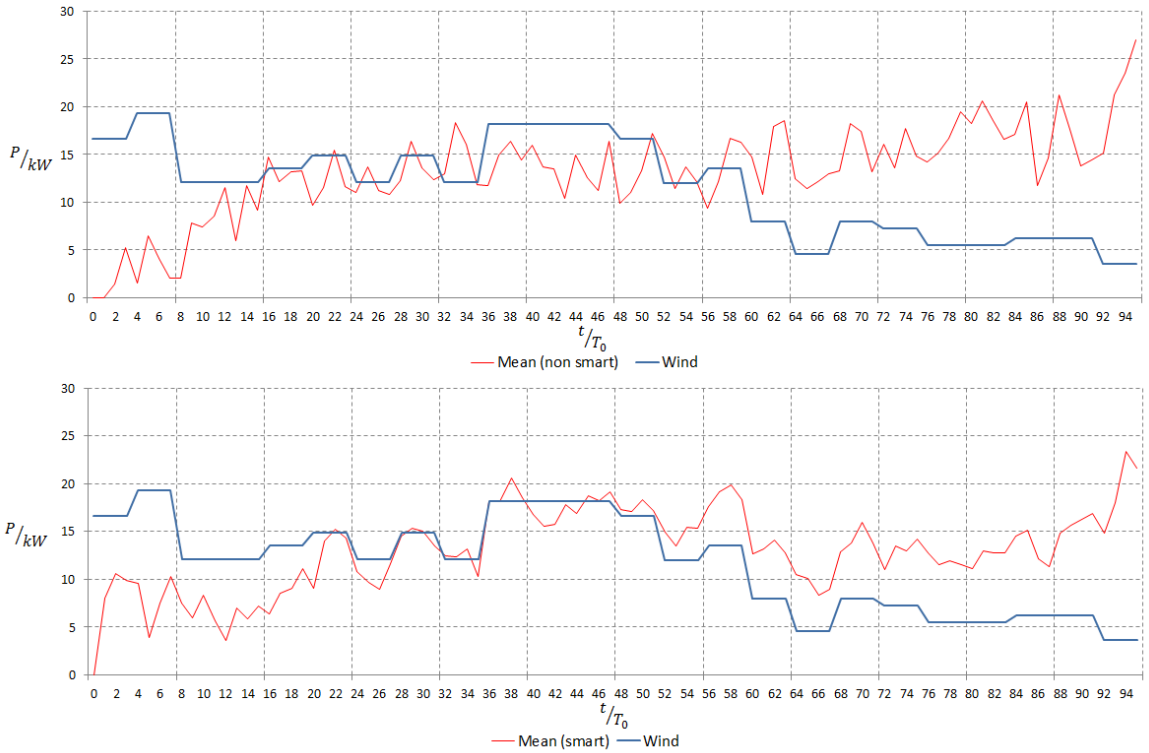

Abbildung 6.17.: Vergleich zwischen und freiem Energieverbrauch (oben) und smarter Laststeuerung (unten)

Wie erwartet übersteigt in beiden Fällen der Verbrauch in den späteren Zeitabschnitten die Erzeugung. Im optimierten Fall jedoch deutlich geringer und auch die frühen Zeitabschnitte werden besser ausgenutzt. Im ungesteuerten Fall werden 78,5% der erzeugten Leistung genutzt. Im gesteuerten Fall werden 85,6% genutzt (+ 7,1%). Dies bedeutet, dass im ungesteuerten Fall 100,0 kWh an externer Energie eingekauft werden müssen, während es im optimierten Szenario im Durchschnitt nur 78,5 kWh sind (- 21,5 kWh).

Die Einbindung weiterer vollautomatischer Verbraucher, wie zum Beispiel Kühlschränke oder Gefriertruhen, sollte das Optimierungsergebnis nochmals deutlich verbessern können. In

diesem Fall konnte anhand der simulierten Winddaten auch gezeigt werden, dass das Optimierungspotential der halbautomatischen Verbrauchern immer noch im hohen Maße von den Nutzungszeiträumen abhängig ist.

6.4. Gezielte Lastverlagerung durch Verbundoptimierung

Analog zu Lünsdorf und Sonnenschein (2009) soll gezeigt werden, dass ein gemeinsames Schaltverhalten auch mit weichen Bedingungen, wie zum Beispiel einem Preissignal, angeregt werden kann. In der zitierten Arbeit wurden die Geräte mittels einer harten Einschaltbedingung, eines Schaltimpulses, dazu gebracht, in einem bestimmten Zeitraum zu starten. Die Hypothese ist in diesem Fall, dass ein signifikanter Einfluss des Preissignals erkennbar ist, dieser aber aufgrund der weichen Preisbedingung und der begrenzten Optimierung nicht so effektiv ist, wie der direkte Schaltpuls in der Arbeit von Lünsdorf und Sonnenschein (2009).

In Abbildung 6.18 beschreibt die rote Kurve den mittleren Stromverbrauch von zehn Simulationen (blaue Kurven) mit 500 Geschirrspülern mit einem konstanten Stromtarif. Diese entspricht in ihrem Verlauf größtenteils dem Erwartungswert (siehe auch Abb. 6.1 und Abb. 6.5).

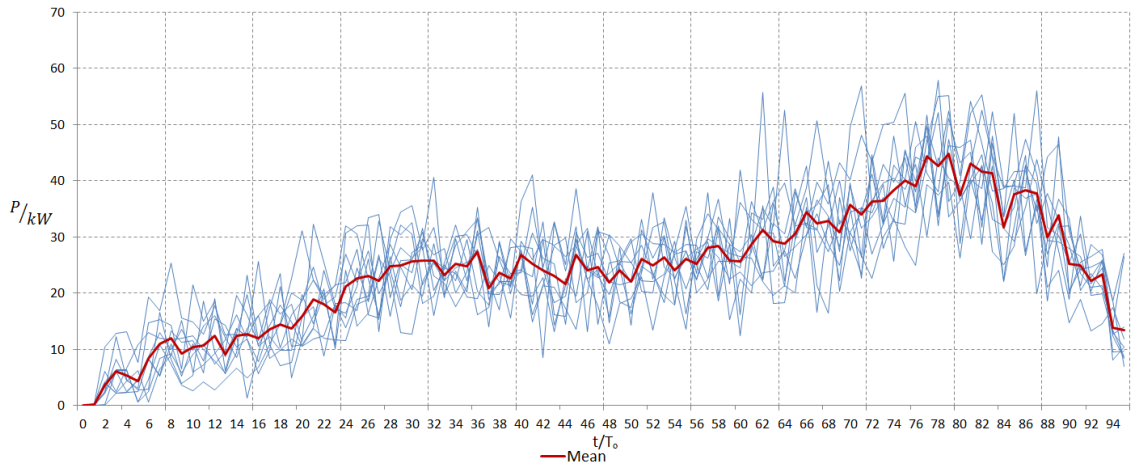

Abbildung 6.18.: Stromverbrauch von 500 Geschirrspülern mit konstantem Preis

In dem Experiment wurde daraufhin ein Steuersignal um 13:00 Uhr für eine Stunde eingesetzt. Der Preis beträgt in diesem Zeitraum 0,001 PE, außerhalb des Zeitraums 1 PE, bei unbegrenzter Leistung. Optimiert wurde mit einem ACO mit 1000 Iterationen bei acht Ameisen pro Kolonie und vier Kolonien und den oben bestimmten Parametern für diesen Optimierungsalgorithmus. Die in Abbildung 6.19 dargestellten Simulationsergebnisse zeigen,

dass in dem Zeitraum, in dem nicht die gesamte Operation des Geschirrspülers durchgeführt werden kann, die hintere der beiden Lastspitzen platziert wird, da um diese Lastspitze der Verbrauch höher ist (vgl. dazu die Tab. 6.1). Dadurch beginnen die Geräte größtenteils um 12:00 Uhr mit ihrer Operation, was die direkte Vergleichbarkeit mit den Ergebnissen von Lünsdorf und Sonnenschein (2009) ermöglicht, da bei diesen ein hartes Einschaltsignal um 12:00 Uhr angewendet wurde.

Im direkten Vergleich konnten, ebenso wie bei Lünsdorf und Sonnenschein (2009), Lastspitzen von bis zu 90 kW erzeugt werden, wodurch gezeigt werden kann, dass ein Preissignal in diesem Kontext eine vergleichbare Wirkung wie ein hartes Einschaltsignal haben kann.

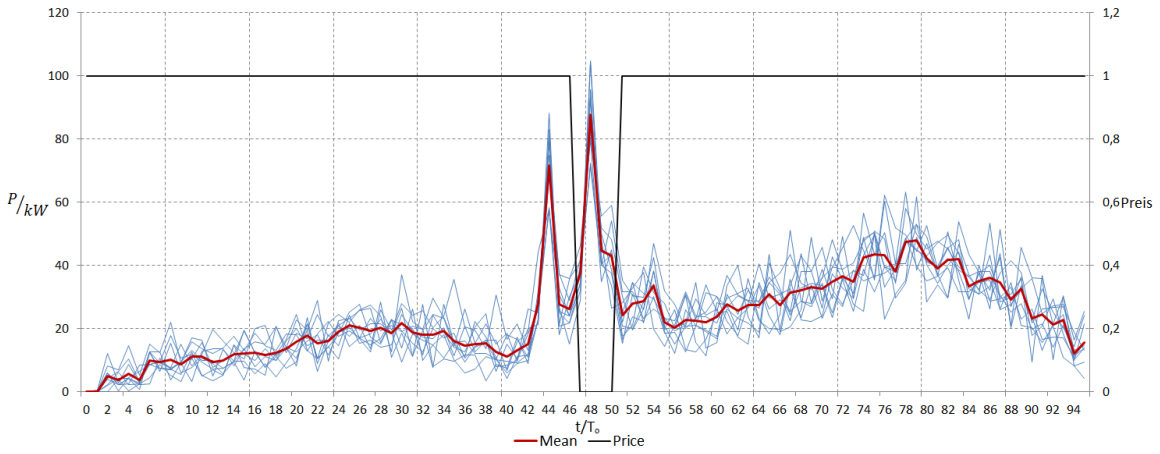

Abbildung 6.19.: Stromverbrauch von 500 Geschirrspülern mit Preissignal

Wird das Differenzsignal (siehe Abb. 6.20) zwischen dem Verbrauch mit konstantem Tarif und dem Verbrauch mit Preissignal betrachtet, ist zu erkennen, dass ein Großteil der Leistung durch eine Einschaltverzögerung abgerufen wird und folglich nur verlagert wird. Der Effekt der scheinbaren Einsparung, die im Vorwege stattfindet und später abgerufen wird, ist als *Rebound-Effekt* bekannt (siehe Palensky und Dietrich (2011)). In Bezug auf den Zeitraum, aus dem die Energie abgerufen wird, besteht ein fundamentaler Unterschied zu der Arbeit von Lünsdorf und Sonnenschein (2009), da diese den Verbrauch vorziehen. Dieser Unterschied basiert auf dem unterschiedlichen Einschaltparadigma. In dem hier vorgestellten Planungsszenario wird das Gerät eingeschaltet und würde ohne Optimierung zum frühestmöglichen Zeitpunkt starten, wodurch die Ähnlichkeit zwischen Abb. 6.1 und Abb. 6.18 erklärt wird. Der Verbrauch kann dadurch nur in spätere Zeiträume verlagert werden. Bei Lünsdorf und Sonnenschein (2009) werden die Geräte eingeschaltet und starten ohne Signal erst zum spätest möglichen Zeitpunkt, wodurch der Verbrauch durch ein Startsignal nur noch vorgezogen werden kann. Das hier implementierte System kann mit beiden Simulationsarten betrieben werden und somit sowohl den Anwendungsfall der Energieplanung, als auch den Anwendungsfall des kurzfristigen Energieabrufs umsetzen. Es konnte dadurch demonstriert

werden, dass längerfristige Energieplanung die gleichen Effekte wie das kurzfristige Abrufen hervorrufen kann und beide Szenarien sich darin unterscheiden, ob die Einsparung vor oder nach der Lastspitze auftritt.

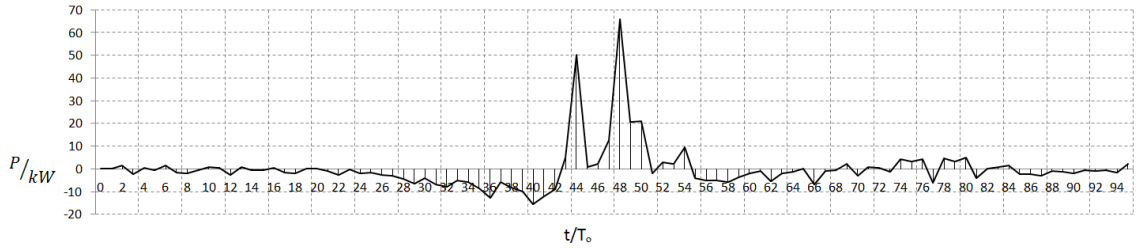

Abbildung 6.20.: Differenzsignal aus konstanter Preissimulation und Einsatz des Preissignals

Es ist unschwer zu erkennen, dass solche kurzen Signale ohne explizite Leistungsbegrenzung durch die auftretenden Lastspitzen negative und destabilisierende Effekte auf das Netz haben könnten. In der Anwendung müssten daher eindeutige Richtlinien getroffen werden, welche Größenordnungen über kurze Zeiträume abgerufen oder nicht abgerufen werden dürfen.

6.5. Untersuchung des Grenzverhaltens der Optimierungsalgorithmen

In diesem Szenario soll das Grenzverhalten der Optimierungsalgorithmen untersucht werden. Von de Castro (2006) wurde die Hypothese formuliert, dass die Ant Colony Optimization dem Genetische Algorithmus bei einfachen Problemen überlegen ist, während sich dieses Verhältnis mit zunehmender Problemkomplexität umkehrt. Es soll untersucht werden, ob dieses Verhalten auch in dieser Domäne nachgewiesen werden kann. Insbesondere interessant ist dieser Sachverhalt daher, da im Abschnitt 6.2.4 durch die Migration gezeigt werden konnte, dass die Problemstellung für den Genetische Algorithmus in einer besonderen Problemklasse angesiedelt ist.

Für die Untersuchung wurde die erwartete benötigte Leistung anhand des Erwartungswertes von 250 Waschmaschinen und 250 Geschirrspülern über einen Tag in 15 Minuten Abschnitten berechnet. Im Ausgangsszenario wurden die 500 Verbraucher auf den Erwartungswert optimiert. Danach wurden Simulationen mit 500 Verbraucher bis zum halben Erwartungswert nach unten und bis zum doppelten Erwartungswert nach oben durchgeführt. Erwartet wurde, dass im Bereich vom doppelten bis zum einfachen Erwartungswert der ACS-Algorithmus bessere Ergebnisse erzielt und im Bereich des einfachen bis zum halben Erwartungswert der Genetische Algorithmus unter Umständen ein besseres Verhalten zeigen könnte. Dabei muss beachtet werden, dass hier der Genetische Basisalgorithmus mit dem besten ACO-Ansatz verglichen wird. Daher wurde auch der ACO-Basisalgorithmus Ant-System (AS) mit in die Betrachtungen einbezogen.

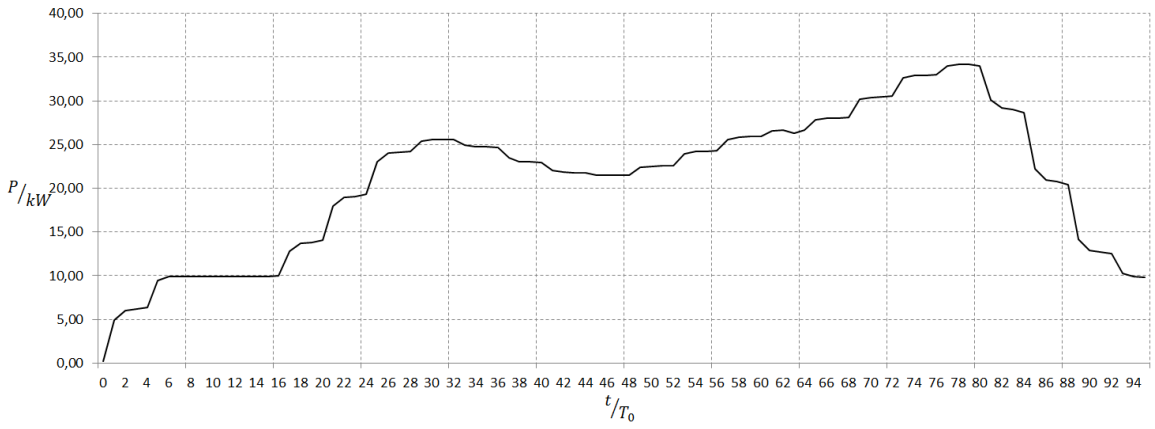

Abbildung 6.21.: Grenzszenario

Zur besseren Vergleichbarkeit und Gleichwertigkeit wurden mit jedem Algorithmus, unter Berücksichtigung der optimierten Parameter für die Optimierungsalgorithmen, 3200 Pfade traversiert. In Abbildung 6.22 ist das Ergebnis der Versuche dargestellt. Es zeigt sich, dass die Ergebnisse der Evaluation und Parametrisierung der Optimierungsalgorithmen wiederholt wurden konnten (siehe Abschnitt 6.1). Alle drei Algorithmen zeigen für die verschiedenen Erwartungswerte ein ähnliches Verhalten. Für diese Problemstellung konnten die ACO-Ansätze immer bessere Ergebnisse als der Genetische Algorithmus erzielen, wobei ACS gegenüber AS nochmals besser abschnitt.

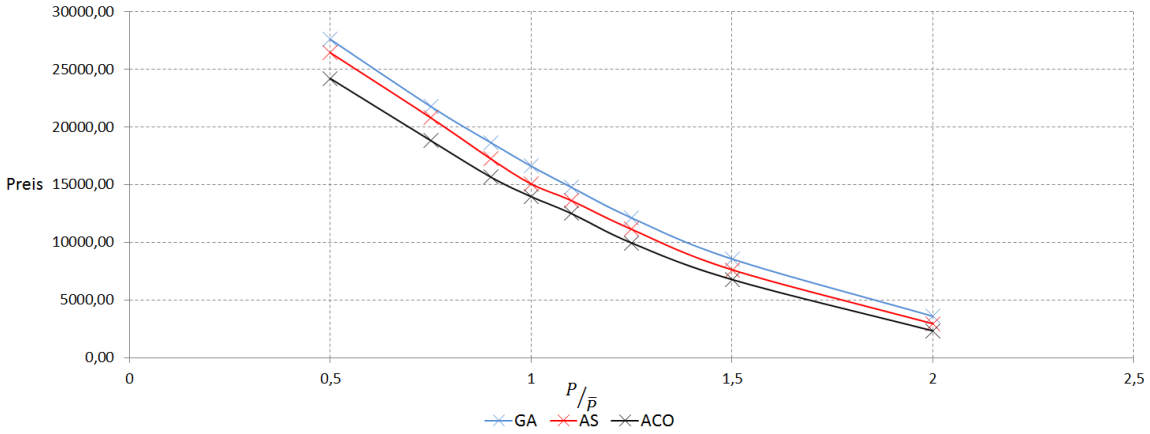

Abbildung 6.22.: Vergleich des Verhaltens der drei Optimierungsalgorithmen

Die Ergebnisse lassen folgende mögliche Schlussfolgerungen zu:

1. Das Problem ist im Sinne von de Castro (2006) nicht komplex genug, wobei die Definition des Begriffes *komplex* in diesem Zusammenhang nicht weiter präzisiert wurde.
2. Das Versuchsdesign war unzureichend. Hier zeichnet sich jedoch kaum Verbesserungspotential ab, da die Definition der *zunehmenden Komplexität* unklar ist (siehe 1. Punkt).
3. ACO-Ansätze sind dem Genetischen Basisalgorithmus überlegen. Es wäre nun an weiterführenden Arbeiten, komplexere und fortgeschrittenere GA zu implementieren und zu untersuchen (bspw. Sigma-Scaling und andere Skalierungsfunktionen).

Historisch betrachtet wurden die ersten ACO-Ansätze verfolgt, als Genetische Algorithmen bereits etabliert waren. Dementsprechend wäre es plausibel, dass AS gegenüber dem Genetischen Basisalgorithmus eine bessere Leistung zeigt, jedoch gegenüber weiterentwickelten Ansätzen, durch den Entwicklungsvorsprung der GA, schlechter abschnitt.

7. Zusammenfassung und Ausblick

In diesem Kapitel sollen die gewonnenen Ergebnisse zusammengefasst werden und ein Ausblick auf Weiterentwicklungspotentiale und Verbesserungsmöglichkeiten für zukünftige Forschungsthemen gegeben werden.

7.1. Zusammenfassung

Demand-Side-Management ist eine der Schlüsseltechnologien für das zukünftige Smart Grid, um einerseits durch langfristige Maßnahmen den zunehmenden Stromverbrauch zu senken (siehe Abschnitt 2.1.5) und andererseits durch kurzfristige Steuerung besser auf die Volatilität der erneuerbaren Energieerzeuger reagieren zu können und das Energienetz und die Infrastruktur effizienter zu nutzen. In dieser Arbeit wurde ein Ansatz entwickelt, um die Flexibilität der Verbraucher automatisiert und für eine Vielzahl von Anwendungsmöglichkeiten nutzbar zu machen. Dazu wurde eine Architektur definiert, die die dafür notwendigen Komponenten spezifiziert. Neben Managementkomponenten, welche die Geschäftsanwendungen in das System eingeben, und infrastrukturellen Komponenten, wie einer Registry und den Kommunikationspfaden, wurde auch ein generisches Verbrauchermodell entwickelt. Das generische Verbrauchermodell abstrahiert durch ein iteratives Planungsmodell die darunterliegende physikalische Schicht für das Optimierungssystem und ermöglicht so die Optimierung von physikalischen Geräten und aggregierten Verbänden gleichermaßen. Durch das Planungsmodell ist es darüber hinaus auf Lastebene möglich, die beiden, in der Literatur diskutierten und für das DSM relevanten Verbrauchertypen (programmgetrieben und vollautomatisch) in einem einzigen Modell abzubilden.

Das entwickelte Marktmodell realisiert eine einfache, mehrstufige Börse, in der Anbieter ihre Tarife mit den dazugehörigen Erzeugungskapazität anbieten können. Im Gegensatz zu einer direkten Steuerung ist damit kein spezifisches Wissen über die Lasten erforderlich, da dieses lokal verbleibt. Dies unterstützt auch datenschutzrechtliche sowie psychologische Aspekte, die sich aus einer Fremdsteuerung durch den Energieversorger ergeben. Gegenüber reinen Preissignalen sind keine Echtzeitüberwachung, regelungstechnische Ansätze zur Gegensteuerung oder Heuristiken zur Abschätzung der Wirkung der Preissignale auf die schätzungsweise vorhandenen Verbraucher, zur Vermeidung von Lawineneffekten, erforderlich

(siehe Abschnitt 6.4). Bei den Auktionsansätzen im DSM wird die Preisgestaltung meist bilateral vorgenommen. Während es jedoch für den Erzeuger in der Regel durch geeignete betriebswirtschaftliche Maßnahmen einfach ist, Preisniveaus und Grenzkosten zu berechnen, müssen Kunden regelmäßig neue akzeptable Preise definieren. Ein unilateraler Preisansatz sollte daher der Automatisierung des Marktes förderlich sein. Aufgrund der beschriebenen Eigenschaften und deren Nachweis durch die Anwendungsfälle ist das hier definierte Marktmodell anderen Modellen zumindest ebenbürtig. Bislang ungeklärte Fragen in Bezug auf das in dieser Arbeit entwickelte Marktmodell sind die Echtzeitfähigkeit der Märkte, deren Verteilung und Granularisierung, sowie die Überwachung durch Kontrollinstanzen bzw. Zertifizierungsstellen. Gleichzeitig darf die psychologische Wirkung des Kontrollverlustes für Energieversorger und Service Provider durch die Aufgabe der traditionellen Top-Down-Planung des Bedarfes nicht unterschätzt werden. Dies ist vermutlich einer der Hauptgründe dafür, dass dezentralisierte Ansätze, trotz ihrer vorteilhaften Eigenschaften, in der Praxis bislang geringe Verbreitung fanden.

Die entwickelte Architektur eines Energiesystems wurde in Jadex implementiert und an die Eigenschaften des Simulationssystems angepasst. Die Simulation umfasst die Umsetzung von zwei Verbrauchertypen (programmgetriebener und thermischer Verbraucher) als Referenzimplementierungen des generischen Verbrauchermodells, sowie einer grundlegenden Subsimulationsfunktionalität. Es wurden zwei verteilte Optimierungsalgorithmen (Genetische Algorithmen und Ant Colony System) implementiert und Parameter für diese bestimmt. Die Tests der Komponenten des Systems in Abschnitt 6.2 wiesen dessen korrekte Funktionalität nach.

Im Day-Ahead-Anwendungsszenario (siehe Abschnitt 6.3) wurden 157 Haushaltsgeräte (95 Waschmaschinen und 62 Geschirrspüler), sowie 4 Stromheizungen auf 10% der Nennleistungsvorhersage einer Windkraftanlage optimiert. Dabei konnten durch die Optimierung im Durchschnitt 21,5 kWh gegenüber dem unkoordinierten Fall eingespart werden. Die Erzeugung war in dem Szenario gegenläufig zum erwarteten Verbrauch, wodurch die durch die begrenzte Flexibilität der programmgetrieben Verbraucher eine besondere Schwierigkeit für die Optimierung ergab. Das implementierte Marktmodell ist damit in der Lage, die Verfügbarkeit von knappen Ressourcen adäquat abzubilden, während sich das Optimierungssystem sich dementsprechend verhält und die Effizienz des Gesamtsystems steigert.

Es konnte ferner in Abschnitt 6.4 nachgewiesen werden, dass eine gezielte Lastverlagerung durch Preissignale möglich ist und einen vergleichbaren Effekt hatte, wie bei der durch Lünsdorf und Sonnenschein (2009) beschriebenen direkten Steuerung. Gleichzeitig bietet das Preissignal einige Vorteile gegenüber der direkten Steuerung. So muss dem Energieversorger nicht bekannt sein, welche spezifischen Geräte an der Lastverlagerung teilgenommen haben, was einerseits aus technischer Sicht wünschenswert ist, da nur wenig domänenspezifisches Wissen über die beteiligten Geräte vorhanden sein muss und andererseits die

Kunden nicht die Fremdsteuerung durch den Energieversorger akzeptieren müssen, beziehungsweise ihre Daten dem Energieversorger zur Verfügung stellen müssen (siehe auch Sonnenschein u. a. (2010)).

Im Zuge der Arbeit wurde die Metaheuristik des Ant Colony Systems von Dorigo und Gambardella (1997) zu einem verteilten Ansatz erweitert. Dabei erzeugt jeder optimierende Agent als Master eine lokale Kolonie von Ant-Agenten, die den verteilten Graphen traversieren. Der Lösungsaustausch findet, wie in der Natur, passiv über die global platzierten und lokal in jedem Knoten gespeicherten Pheromone statt. Die von Dorigo und Gambardella (1997) beschriebenen, generischen Optimierungsparameter konnten größtenteils bestätigt werden. Es konnte gezeigt werden, dass der verteilte Ansatz, im Gegensatz zu dem verteilten Genetischen Algorithmus und dem verteilten Greedy-Algorithmus, deutlich bessere Lösungen nach nur wenigen Optimierungsschritten erzeugt. Versuche mit zunehmend verknappenden Ressourcen (Abschnitt 6.5) zeigten, dass der verteilte ACS-Ansatz fortwährend bessere Ergebnisse liefert, als der verteilte Genetische Basisalgorithmus. Einschränkend muss festgehalten werden, dass der Versuch mit dem Genetischen Basisalgorithmus durchgeführt wurde und modernere Implementierungen deutlich bessere Ergebnisse liefern dürften.

Folgende Nebenresultate wurden im Zuge des Entwurfs und der Entwicklung des verteilten Systems erzielt:

Um die internen Abläufe und Prozesse in den einzelnen Agenten zu verdeutlichen, wurden diese in CTL nach Xing und Singh (2003) visualisiert. Um die Interaktionen zwischen den Agenten zu artikulieren, wurde das Visualisierungswerkzeug in Abschnitt 2.3.3 um die formale Definition von Kommunikationspfaden und Agentendomänen erweitert.

Zur Visualisierung der internen Struktur der BDI-Agenten wurde in Anhang B eine an Tropos angelehnte Agentenbeschreibung entwickelt. Mit ihr lassen sich Ziele, Pläne und die Belief-Base eines Agenten und deren Relationen zueinander abbilden. Ebenso ist es möglich, Ausführungsbedingungen und Services zu definieren und diese verständlich und übersichtlich darzustellen. Die Anwendbarkeit wurde sowohl an einem Beispielagenten zur Heizplanung, als auch an der Implementierung des Base- und des EMS-Agenten, demonstriert.

Die Ergebnisse dieser Arbeit plädieren für eine weitergehende Diskussion über die Ablösung des Paradigmas der flexiblen Tarife, hin zu einem betriebswirtschaftlich fundierten Marktmodell für ein intelligentes Smart Grid. Der Einsatz dieses hier entwickelten, weitergehenden Marktmodells, welches Nebenbedingungen und Netzrestriktionen über Strafpreise, aber auch Pflichten des Energieversorgers, wie die Versorgungssicherheit, implizit abbilden kann, würde eine Reihe von technischen Problemstellungen, die sich durch den Einsatz des flexiblen Tarifkonzepts ergeben, obsolet machen (z.B. Desynchronisierungsalgorithmen bei Lünsdorf und Sonnenschein (2009), Echtzeitüberwachung bei Ramchurn u. a. (2011)).

7.2. Ausblick

Die hier spezifizierte Architektur für ein verbraucherorientiertes Energiesystem bietet deutliches Erweiterungspotential. So kann die Beschreibung der Schnittstellen und Kommunikationsprotokolle in folgenden Arbeiten genauer spezifiziert werden. Ebenfalls könnten neue EMS und Registry-Komponenten entwickelt werden und sich aus der weiterentwickelten Architektur neue Nutzungspotentiale und Betriebsmodelle ableiten lassen. Ebenso ist es denkbar, dass im Zuge einer funktionierenden Architektureinbindung die Verbrauchsgeräte um diesbezügliche Funktionalitäten erweitert werden, woraus sich neue Forschungsfragen für den Betrieb und die Potentiale des DSM ergeben könnten. Ein Beispiel wäre die Einbindung und Untersuchung von programmgetriebenen Haushaltsgeräten, die automatisiert eine Stunde Flexibilität anbieten. Für den Endverbraucher wäre dies nur ein minimaler Komfortverlust, der durch eine zuschaltbare Express-Funktion für Situationen, in denen das Gerät schnell arbeiten muss, noch abgemildert werden könnte.

Im Bereich der Implementierung und Simulation könnten weitergehende Simulationen von Betriebskonzepten zur Verifikation und Weiterentwicklung beitragen. Potential bietet hier die weitergehende Untersuchung der Optimierungsalgorithmen. In modernen Operation Research Anwendungen werden Kombinationen von *Branch and Cut*-Algorithmen und Metaheuristiken eingesetzt, um schneller und effizienter kombinatorische Probleme zu lösen. Die Referenzimplementierung mit Jadex kann ebenfalls realitätsnäher gestaltet werden, indem genauere Simulationsmodelle für thermische Verbraucher eingebunden werden (z.B. MATLAB/Simulink-Modelle). Die darauf folgende Stufe wäre die Implementierung der hier beschriebenen dezentralen Kapazität und des Marktmodells in bestehende Energy Service Interfaces wie OGEMA oder OpenADR und die Durchführung von Feldtests und Messungen auf physischen Geräten und Hardware. Im Zuge dazu müsste parallel auch eine Infrastruktur zum Registrieren und Auffinden von Energy-Services entwickelt werden.

Abschließend muss festgehalten werden, dass die dynamische Laststeuerung nur ein Teil des Smart Grids der Zukunft sein kann. Daneben gilt es langfristige DSM-Bemühungen weiterhin aufrecht zu erhalten. Darüber hinaus ist die physikalische Demand-Response als zweite Säule notwendig. Nur in Zusammenhang mit diesen Maßnahmen kann eine vollständige Integration des Demand-Side-Managements im Smart Grid gelingen.

Literaturverzeichnis

[Androulakis und Floudas 1999] ANDROULAKIS, Ioannis P. ; FLOUDAS, Christodoulos A.: Distributed branch and bound algorithms for global optimization. In: *Parallel processing of discrete problems*. Springer, 1999, S. 1–35

[Axelrod 1984] AXELROD, Robert: *The Evolution of Cooperation*. Basic Books, New York, 1984

[BDEW 2012] BDEW: Netto-Stromverbrauch der Verbrauchergruppen / Bundesverband der Energie- und Wasserwirtschaft. 2012. – Forschungsbericht

[Beer u. a. 2011] BEER, Sebastian ; SONNENSCHEIN, Michael ; APPELRATH, Hans-Jürgen: Towards a self–organization mechanism for agent associations in electricity spot markets. In: *Informatik* (2011)

[Belding 1995] BELDING, Theodore C.: The Distributed Genetic Algorithm Revisited. In: *ICGA* Citeseer (Veranst.), 1995, S. 114–121

[Bergmann u. a. 2010] BERGMANN, Johannes ; GLOMB, Christian ; GOTZ, J ; HEUER, Jörg ; KUNTSCHKE, Richard ; WINTER, Martin: Scalability of smart grid protocols: Protocols and their simulative evaluation for massively distributed DERs. In: *Smart Grid Communications (SmartGridComm),2010 First IEEE International Conference on* IEEE (Veranst.), 2010, S. 131 – 136

[Bremer und Sonnenschein 2013] BREMER, Jörg ; SONNENSCHEIN, Michael: Model-based integration of constrained search spaces into distributed planning of active power provision. In: *Computer Science and Information Systems* 10 (2013), Nr. 4, S. 1823–1854

[Bresciani u. a. 2004] BRESCIANI, Paolo ; PERINI, Anna ; GIORGINI, Paolo ; GIUNCHIGLIA, Fausto ; MYLOPOULOS, John: Tropos: An agent-oriented software development methodology. In: *Autonomous Agents and Multi-Agent Systems* 8 (2004), Nr. 3, S. 203–236

[Breuer 2013] BREUER, A.: Dezentrale Systeme für die sichere Versorgung. In: *eEnergy-Abschlusskongress* (2013)

[Bruinenberg u. a. 2012] BRUINENBERG, Jan ; COLTON, Larry ; DARMOIS, Emmanuel ; DORN, John ; DOYLE, John ; ELLOUMI, Omar ; ENGLERT, Heiko ; FORBES, Raymond ; HEILES, Jürgen ; HERMANS, Peter u. a.: Smart grid coordination group technical report reference architecture for the smart grid version 1.0 (draft) 2012-03-02. In: *CEN, CENELEC, ETSI, Tech. Rep* (2012)

[Bullnheimer u. a. 1997] BULLNHEIMER, Bernd ; KOTSIS, Gabriele ; STRAUSS, Christine: Parallelization strategies for the ant system. (1997)

[Cantú-Paz 1998] CANTÚ-PAZ, Erick: A survey of parallel genetic algorithms. In: *Calculateurs paralleles, reseaux et systems repartis* 10 (1998), Nr. 2, S. 141–171

[de Castro 2006] CASTRO, Leandro N. de ; SAHNI, Sartaj (Hrsg.): *Fundamentals of natural computing*. Chapman & Hall/CRC, 2006

[Colorni u. a. 1994] COLORNI, Alberto ; DORIGO, Marco ; MANIEZZO, Vittorio: Distributed Optimization by Ant Colonies. In: *ECAL91 - EUROPEAN CONFERENCE ON ARTIFICIAL LIFE*, 1994

[Cramer u. a. 2009] CRAMER, B ; ANDRULEIT, H ; REMPEL, H ; BABIES, HG ; SCHLÖMER, S ; SCHMIDT, S ; SCHWARZ-SCHAMPERA, U ; OCHMANN, N ; MESSNER, J ; REHDER, S: Energierohstoffe 2009: Reserven, Ressourcen, Verfügbarkeit. In: *Erdöl, Erdgas, Kohle, Kernbrennstoffe, Geothermische Energie. Bundesanstalt für Geowissenschaften und Rohstoffe (BGR), Hannover, 2009. www. bgr. bund. de* (2009)

[Darwin 1859] DARWIN, Charles: *On the origin of species by means of natural selection: or, the preservation of favored races in the struggle for life*. John Murray, 1859

[Dethlefs und Renz 2013] DETHLEFS, Tim ; RENZ, Wolfgang: A distributed registry for service-based energy management systems. In: *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE* IEEE (Veranst.), 2013, S. 4710–4714

[Dethlefs u. a. 2013] DETHLEFS, Tim ; RENZ, Wolfgang ; BALTHASAR, Gregor ; PREISLER, Thomas: *A Service-based Multi-Agent approach on Distributed Demand-Side-Optimization*. 2013. – Zur Veröffentlichung eingereicht

[Dorigo u. a. 2006] DORIGO, Marco ; BIRATTARI, Mauro ; STUTZLE, Thomas: Ant colony optimization. In: *Computational Intelligence Magazine, IEEE* 1 (2006), Nr. 4, S. 28–39

[Dorigo und Gambardella 1996] DORIGO, Marco ; GAMBARDELLA, Luca M.: A study of some properties of Ant-Q. In: *Proceedings of PPSN* (1996), S. 656 – 665

[Dorigo und Gambardella 1997] DORIGO, Marco ; GAMBARDELLA, Luca M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. In: *Evolutionary Computation, IEEE Transactions on* 1 (1997), Nr. 1, S. 53–66

[Dorigo u. a. 1996] DORIGO, Marco ; MANIEZZO, Vittorio ; COLORNI, Albert: Ant System: optimization by a Colony of Cooperation Agents. In: *IEEE Transactions on Systems Man and Cybernetics* 26 (1996), S. 29 – 41

[Energieagentur NRW 2006] ENERGIEAGENTUR NRW: *Prozentuale Anteile der 12 Stromverbrauchsbereiche in den verschiedenen Haushaltsgrößen*. 03 2006

[ENTSOE 2004] ENTSOE: *Load-Frequency Control and Performance*. 2004

[Federal Energy Regulatory Commission 2009] FEDERAL ENERGY REGULATORY COMMISSION: *A National Assessment of Demand Response Potential*. 2009

[Frey u. a. 2007] FREY, Günther ; SCHULZ, Wolfgang ; HORST, Juri ; LEPRICH, Uwe: *Studie zu den Energieeffizienzpotentialen durch den Ersatz von elektrischem Strom im Raumwärmebereich*. IZES gGmbH, Saarbrücken, 2007

[Frey 2013] FREY, H.: *MeRegio Abschlussbericht*. 2013

[Fuxman u. a. 2004] FUXMAN, Ariel ; LIU, Lin ; MYLOPOULOS, John ; PISTORE, Marco ; ROVERI, Marco ; TRAVERSO, Paolo: Specifying and analyzing early requirements in Tropos. In: *Requirements Engineering* 9 (2004), Nr. 2, S. 132–150

[Gabaldon u. a. 2003] GABALDON, A ; MOLINA, A ; ROLDAN, C ; FUENTES, JA ; GOMEZ, E ; RAMIREZ-ROSADO, IJ ; LARA, P ; DOMINGUEZ, JA ; GARCIA-GARRIDO, E ; TARANCON, E: Assessment and simulation of demand-side management potential in urban power distribution networks. In: *Power Tech Conference Proceedings, 2003 IEEE Bologna* Bd. 4 IEEE (Veranst.), 2003, S. 5–pp

[Gambardella und Dorigo 1995] GAMBARDELLA, Luca M. ; DORIGO, Marco: Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. In: *Proceedings of ML-95* (1995), S. 252–260

[Gellings 1985] GELLINGS, Clark W.: The concept of demand-side management for electric utilities. In: *Proceedings of the IEEE* 73 (1985), Nr. 10, S. 1468–1470

[Gerdts 2013] GERDTS, Matthias: *Einführung in die lineare und nichtlineare Optimierung*. 2013

[Giorgini u. a. 2005] GIORGINI, Paolo ; MYLOPOULOS, John ; SEBASTIANI, Roberto: Goal-oriented requirements analysis and reasoning in the tropos methodology. In: *Engineering Applications of Artificial Intelligence* 18 (2005), Nr. 2, S. 159–171

[Huber u. a. 2011] HUBER, Dennis ; TAYLOR, Zephyr ; KNUDSEN, Steven: Environmental Impacts of Smart Grid / National Energy Technology Laboratory, US-Department of Energy. 2011. – Forschungsbericht

[Hübner 2012] HÜBNER, Christian: VDE/ITG-Positionspapier Energieinformationsnetze und-systeme: Teil A "Verteilungsnetzautomatisierung im Smart Grid". In: *VDE-Kongress 2012* VDE VERLAG GmbH (Veranst.), 2012

[IEA 2007] IEA: *Key World Energy Statistics 2007*. OECD/IEA, 2007

[IEC 60038 2009] IEC 60038: *CENELEC-Normspannungen*. 2009

[IPCC 2013] IPCC: Climate Change 2013: The physical science basis. (2013)

[Kamper 2010] KAMPER, Andreas: *Dezentrales Lastmanagement zum Ausgleich kurzfristiger Abweichungen im Stromnetz*. KIT Scientific Publishing, 2010

[Kok u. a. 2010] KOK, JK ; SCHEEPERS, MJJ ; KAMPHUIS, IG: Intelligence in electricity networks for embedding renewables and distributed generation. In: *Intelligent infrastructures*. Springer, 2010, S. 179–209

[Kok u. a. 2005] KOK, JK ; WARMER, CJ ; KAMPHUIS, IG: PowerMatcher: multiagent control in the electricity infrastructure. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* ACM (Veranst.), 2005, S. 75–82

[Linnenberg u. a. 2011] LINNENBERG, T ; WIOR, I ; SCHREIBER, S ; FAY, A: A Market-Based Multi-Agent-System for Decentralized Power and Grid Control. In: *IEEE EFTA 2011*, 2011

[Logenthiran und Srinivasan 2011] LOGENTHIRAN, T ; SRINIVASAN, D: Multi-Agent System for Demand Side Management in smart grid. In: *Power Electronics and Drive Systems (PEDS),2011 IEEE Ninth International Conference* (2011)

[Logenthiran u. a. 2012] LOGENTHIRAN, Thillainathan ; SRINIVASAN, Dipti ; SHUN, Tan Z.: Demand side management in smart grid using heuristic optimization. In: *IEEE Transactions on Smart Grid* 3 (2012), Nr. 3, S. 1244–1252

[Lünsdorf und Sonnenschein 2009] LÜNSDORF, Ontje ; SONNENSCHEIN, Michael: Lastadaption von Haushaltsgeräten durch Verbundsteuerung. In: *Tagungsband zum 3. Statusseminar des FEN*, Forschungsverbund Energie Niedersachsen, 09 2009

[Matallanas u. a. 2012] MATALLANAS, E ; CASTILLO-CAGIGAL, Manuel ; GUTIÉRREZ, A ; MONASTERIO-HUELIN, F ; CAAMAÑO-MARTÍN, Estefanía ; MASA, D ; JIMÉNEZ-LEUBE, J: Neural network controller for Active Demand-Side Management with PV energy in the residential sector. In: *Applied Energy* 91 (2012), Nr. 1, S. 90–97

[Merz 2003] MERZ, P.: *Moderne heuristische Optimierungsverfahren: Meta-Heuristiken*. Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, 2003

[Mitchell u. a. 2009] MITCHELL, Cynthia ; DEUMLING, Reuben ; COURT, Gill: Stabilizing California's Demand. In: *Fortnightly Magazine* 03 (2009), S. 10

[Mitchell 1998] MITCHELL, Melanie: *An Introduction to Genetic Algorithms*. MIT Press, 1998

[Mockapetris 2003] MOCKAPETRIS, Paul: RFC 1034: Domain names: concepts and facilities (November 1987). Status: Standard (2003)

[Nestle u. a. 2010] NESTLE, David ; RINGELSTEIN, Jan ; WALDSCHMIDT, Heiko: Open Energy Gateway Architecture for Customers in the Distribution Grid. In: *Information Technology, Oldenbourg Verlag, Munich* (2010), S. 83–88

[Nissen 1994] NISSEN, Volker: *Evolutionäre Algorithmen: Darstellung, Beispiele, Betriebswirtschaftliche Anwendungsmöglichkeiten*. Deutscher Universitäts-Verlag, 1994

[NIST 2010] NIST: Roadmap for Smart Grid Interoperability Standards. In: *NIST special publication* 1108 (2010)

[Odell 2000] ODELL, Peter R.: The global energy market in the long term: the continuing dominance of affordable non-renewable resources. In: *Energy, Exploration & Exploitation* 18 (2000), Nr. 5, S. 599–613

[Paerschke 2005] PAERSCHKE, H.: Versuchsanleitung: Modellbildung mit Matlab/Simulink in der Gebäudetechnik. (2005)

[Palensky und Dietrich 2011] PALENSKY, Peter ; DIETRICH, Dietmar: Demand side management: Demand response, intelligent energy systems, and smart loads. In: *Industrial Informatics, IEEE Transactions on* 7 (2011), Nr. 3, S. 381–388

[Pettey u. a. 1987] PETTEY, Chrisila B. ; LEUZE, Michael R. ; GREFENSTETTE, John J.: A parallel genetic algorithm. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application* L. Erlbaum Associates Inc. (Veranst.), 1987, S. 155–161

[Piette u. a.] PIETTE, Mary A. ; GHATIKAR, Girish ; KILICCOTE, Sila ; PALENSKY, Peter ; MCPARLAND, Charles ; KOCH, Ed ; HENNAGE, Dan: Open Automated Demand Response Communications Specification (Version 1.0).

[Pokahr u. a. 2005] POKAHR, Alexander ; BRAUBACH, Lars ; LAMERSDORF, Winfried: Jadex: A BDI reasoning engine. In: *Multi-agent programming*. Springer, 2005, S. 149–174

[Potsdam Institute for Climate Impact Research and Climate Analytics 2012] POTSDAM INSTITUTE FOR CLIMATE IMPACT RESEARCH AND CLIMATE ANALYTICS: *Turn Down the Heat: Why a 4° C World Must Be Avoided*. The World Bank, 2012

[Ramchurn u. a. 2011] RAMCHURN, S ; VYTELINGUM, Perukrishnen ; ROGERS, Alex: Agent-based control for decentralised demand side management in the smart grid. In: *The Tenth International Conference on Autonomous Agents and Multiagent Systems AAMAS 2011 (2011)*, 2011, S. 5–12

[Rusitschka u. a. 2009] RUSITSCHKA, Sebnem ; GERDES, Christoph ; EGER, Kolja: A low-cost alternative to smart metering infrastructure based on peer-to-peer technologies. In: *Energy Market, 2009. EEM 2009. 6th International Conference on the European* IEEE (Veranst.), 2009, S. 1–6

[Ruthe u. a. 2013] RUTHE, Sebastian ; REHTANZ, Christian ; LEHNHOFF, Sebastian: A market-oriented stochastic optimization framework and its application in the energy domain. In: *Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE* IEEE (Veranst.), 2013, S. 4773–4778

[Sonnenschein u. a. 2010] SONNENSCHEIN, M. ; RAPP, B. ; J., Bremer: Demand Side Management und Demand Response. In: *Handbuch Energiemanagement*. EW Medien und Kongresse GmbH, 2010

[Stamminger u. a. 2008] STAMMINGER, Rainer ; BROIL, Gereon ; PAKULA, Christiane ; JUNGBECKER, Heiko ; BRAUN, Maria ; RÜDENAUER, Ina ; WENDKER, Christoph: Synergy potential of smart appliances. In: *Report of the Smart-A project* (2008)

[Statistisches Bundesamt 2013] STATISTISCHES BUNDESAMT: Wirtschaftsbereich Energie - Erzeugung / Statistisches Bundesamt. 2013. – Forschungsbericht

[Tanenbaum und Van Steen 2002] TANENBAUM, Andrew S. ; VAN STEEN, Maarten: *Distributed systems*. Bd. 2. Prentice Hall, 2002

[Tanese 1987] TANESE, Reiko: Parallel genetic algorithms for a hypercube. In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application* L. Erlbaum Associates Inc. (Veranst.), 1987, S. 177–183

[Ulbig und Andersson 2010] ULBIG, A. ; ANDERSSON, G.: Towards variable end-consumer electricity tariffs reflecting marginal costs: A benchmark tariff. In: *Energy Market (EEM), 2010 7th International Conference on the European*, June 2010, S. 1–6

[U.S. Energy Information Administration 2013] U.S. ENERGY INFORMATION ADMINISTRATION: Annual Energy Outlook 2013 with Projections to 2040. (2013)

[Uslar u. a. 2012] USLAR, Mathias ; SPECHT, Michael ; ROHJANS, Sebastian: *The Common Information Model CIM: IEC 61968/61970 and 62325-A practical introduction to the CIM*. Bd. 66. Springer, 2012

[Varian 2010] VARIAN, Hal ; REPCHECK, Jack (Hrsg.): *Intermediate Microeconomics - A modern approach*. 8. W. W. Norton & Company, 2010

[Verbeeck u. a. 2014] VERBEECK, Jef ; KUIJPER, Fred ; WELLBROCK, Philipp ; GRAY, David ; SKOG, Daniel ; SCHREUDER, Jan ; KOOISTRA, Sander: *The e-habours Journey*. February 2014

[Verbraucherzentrale Sachsen 2012] VERBRAUCHERZENTRALE SACHSEN: Smart Meter überzeugen noch nicht / Verbraucherzentrale Sachsen e.V. 2012. – Forschungsbericht

[Von Martial 1990] VON MARTIAL, Frank: Interactions among autonomous planning agents. In: *Decentralized AI* (1990), S. 105–119

[Wooldridge 2004] WOOLDRIDGE, Michael: *An introduction to multiagent systems*. Wiley, 2004

[Xing und Singh 2003] XING, Jie ; SINGH, Munindar P.: Engineering commitment-based multiagent systems: a temporal logic approach. In: *Proceedings of the second international joint conference on Autonomous agents and multiagent systems* ACM (Veranst.), 2003, S. 891–898

A. XML-Definitionen

A.1. XML-Definition des Semiautomatic-Devices

Tabelle A.1.: XML-Struktur des Semiautomatic-Devices

Element	**Beschreibung**
<ConsumerProfile>	
<device>	Gerätebeschreibung
<type>	Gerätetyp (sdevice)
<powerProfile>	Vektor mit dem Lastprofil
<data>	Leistung in kW zum Zeitpunkt 0
...	
<data>	Leistung in kW zum Zeitpunkt n
</powerProfile>	
<flexibilityVector>	kumulierte Wahrscheinlichkeit für 0 Zeitslots Flexibilität
...	
<flexibilityVector>	kumulierte Wahrscheinlichkeit für n Zeitslots Flexibilität
<runtimeVector>	kumulierte Wahrscheinlichkeit für Startzeitpunkt 0
...	
<runtimeVector>	kumulierte Wahrscheinlichkeit für Startzeitpunkt 95
</ConsumerProfile>	

A.2. XML-Definition des Vector-Devices

Tabelle A.2.: XML-Struktur des Vector-Devices

Element	Beschreibung
<VectorInfo>	
<day>	Tag des Jahres
<powerProfile>	Vektor mit dem Lastprofil, sofern kein Modell angegeben wurde
<data>	Leistung in kW zum Zeitpunkt 0
...	
<data>	Leistung in kW zum Zeitpunkt n
</powerProfile>	
<hA>	Verlustkoeffizient
<CCoeff>	Raumkoeffizient
<obereSchwelle>	Obere Temperaturschwelle
<untereSchwelle>	Untere Temperaturschwelle
<raumtemperatur>	Raumtemperatur zum Simulationstart
</VectorInfo>	

B. Graphische Beschreibung von Service-orientierten Agentenystemen

Verfasst von Tim Dethlefs
Entwickelt in Zusammenarbeit mit Thomas Preisler, Gregor Balthasar und Wolfgang Renz
HAW-Hamburg, Fakultät Technik und Informatik, Departement für Informations- und Elektrotechnik, Labor für Multimediale Systeme

B.1. Einleitung

Durch die zunehmende Verbreitung von vernetzen Geräten und dem steigenden Automatisierungsgrad werden immer mehr proaktive, autonome und interagierende Softwaremodule Teil dieser Infrastruktur. Wooldridge (2004) definiert diese Softwaremodule als *Agenten*, welche reaktiv oder eigenständig, anhand von Zielen und Plänen, mit der Umwelt interagieren. Neben der direkten Umwelt, mit der der Agent wechselwirkt, spielt auch die Interaktion zwischen Agenten zwecks Informationsaustausch oder Koordination eine zunehmende Rolle.

In Projekten mit Agententechnologie muss dementsprechend die zunehmende Komplexität der Komponenten an sich, als auch deren Wechselwirkungen, konsequent durch Planungen und Beschreibungen verwaltet werden. Um den Entwicklungsprozess zu unterstützen, entwickelten Bresciani u. a. (2004) und Giorgini u. a. (2005) *TROPOS* als graphisches Modellierungswerkzeug für Agenten.

Tropos positioniert sich bereits früh im Entwicklungsprozess, beginnend mit der Anforderungsanalyse (siehe Fuxman u. a. (2004)) und dem Design und kann somit Wasserfallmodelle im Softwareprojektmanagement optimal bis zur Implementierung unterstützen. In der Implementierungsphase und darüber hinaus ist Tropos aufgrund der eingeschränkten und unspezifischen Sprachelemente jedoch nicht ausreichend definiert. Dabei ist die genaue Dokumentation und Beschreibung der Elemente in den zunehmend iterativen Entwicklungsprozessen, wie *extreme programming* oder *SCRUM*, essentieller Bestandteil für die effiziente Fortentwicklung des Softwaresystems.

In dieser Arbeit sollen daher graphische Beschreibungselemente definiert werden, die der Dokumentation von Agenten, insbesondere in Jadex[1], dienen sollen. Dabei sollen bekannte Elemente von Tropos übernommen werden und den Anforderungen angepasste werden, sodass eine einfache graphische Beschreibung von Agenten für Dokumentationen ermöglicht wird. Dadurch sollte das Agentenparadigma in moderne Vorgehensmodellen zum iterativen Softwaredesign besser einsetzbar sein.

Die Arbeit ist wie folgt strukturiert: Nach der Einleitung werden die grundlegenden Designelemente näher beschrieben und spezifiziert. Daraufhin wird anhand eines einfachen Beispiel deren Einsatz demonstriert. Im letzten Abschnitt werden Verbesserungs- und Weiterentwicklungspotentiale identifiziert.

B.2. Beschreibung des graphischen Grundelemente

Die Grundelemente umfassen nach derzeitigem Stand acht Elemente, die in Abbildung B.2 dargestellt werden. Ähnlich wie bei UML wird der Typ des jeweiligen Elements, umschlossen von Doppelpfeilen, als oberstes Textelement angegeben (Beispielsweise: «Belief»). Es folgt die eindeutige Bezeichnung des Elements in Fettschrift, sowie optional nach einem Doppelpunkt in kursiver Schrift der Datentyp des jeweiligen Elements.

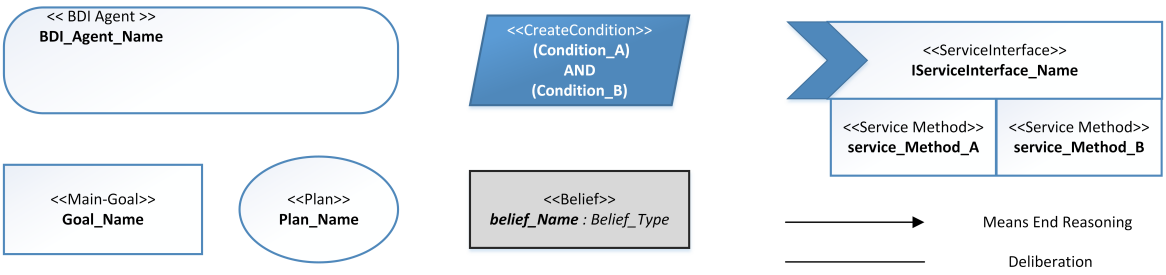

Abbildung B.1.: Graphische Beschreibungselemente

Das zentrale Element ist die Domäne («Agent», «BDI-Agent» oder «Capability»), die den Umfang des Agenten beschreibt. Sie umschließt sämtliche Elemente, die Teil des Agenten sind.

Bestandteile von BDI - Agenten sind hauptsächlich *Goals* und *Plans*, die durch ein Rechteck, beziehungsweise durch ein Oval, darstellt werden. Pläne werden mit Goals durch *Means-End*-Pfeile verbunden, wobei die Pfeilspitze vom Plan zu dem verbundenen Goal führt. Goals können auf drei Arten aufgerufen werden: Durch Servicemethodenaufruf, durch *Conditions*

[1] www.activecomponents.org

und durch andere Goals. In jedem Fall wird das entsprechende Element durch einen *deliberation*-Strich mit dem Goal verbunden. Goals, die durch Umweltbedingungen aufgerufen wurden, also entweder durch Serviceaufrufe, oder durch *Belief*-Änderungen und dadurch erfüllte *Conditions*, sollten als *«Main-Goal»* klassifiziert werden. Goals, die Teilziele von *Main-Goals* darstellen als *«Sub-Goals»*.

Der *Condition*-Block soll Bedingungen darstellen, die sich aus den Beliefs ergeben und damit zur Erzeugung eines Goals führen. Die Bedingungen können gemäß der Booleschen Algebra formuliert werden.

Beliefs stellen das Umweltwissen des Agenten über Variablen dar und sollten dementsprechend wie Variablen in UML unter Angabe des Datentyps beschrieben werden.

Services, die der Agent anbietet, werden durch einen Winkel repräsentiert, welcher aus der Agentendomäne herausragen sollte. Die Methoden, die durch den Service bereitgestellt werden, sind unter dem Beschreibungsrechteck angeordnet.

B.3. Anwendungsfall: Automatisierte Heizplanung in einem Haushalt

Zur Demonstration der im vorigen Abschnitt definierten Elemente dient ein einfaches Agentenszenario. Dabei soll ein BDI-Agent beschrieben werden, der in einem Haus mit zwei Räumen anhand von Tarifsignalen und Wetterdaten automatisiert einen Heizplan für eine Stromheizung erstellt. Der Agent ist in Abbildung B.2 mit den beschriebenen Elementen dargestellt. Entsprechend der Smart-Meter Richtlinie in Deutschland muss das Energy Service Interface (ESI) dem Stromanbieter einen Service zur Eingabe eines Tarifsignals bereitstellen. Die Wetterdaten werden, der Einfachheit halber, ebenfalls über das ESI gesendet. Der Hausagent erzeugt nun einen Heizplan mittels des *GenerateHeatingForecast*-Goals, sofern keiner vorliegt, oder wenn das Tarifsignal oder die Wetterdaten aktualisiert wurden.

Wenn entweder bestimmte Temperaturgrenzen verletzt sind, oder die Überprüfung des Heizplan durch die *check(heatingForecast, now)*-Methode ergibt, dass geheizt werden muss, wird das entsprechende *KeepTemperatureGoal* ausgeführt, welches dafür sorgt, dass in einem der Räume oder in allen durch die Ausführung des entsprechenden Plans geheizt wird.

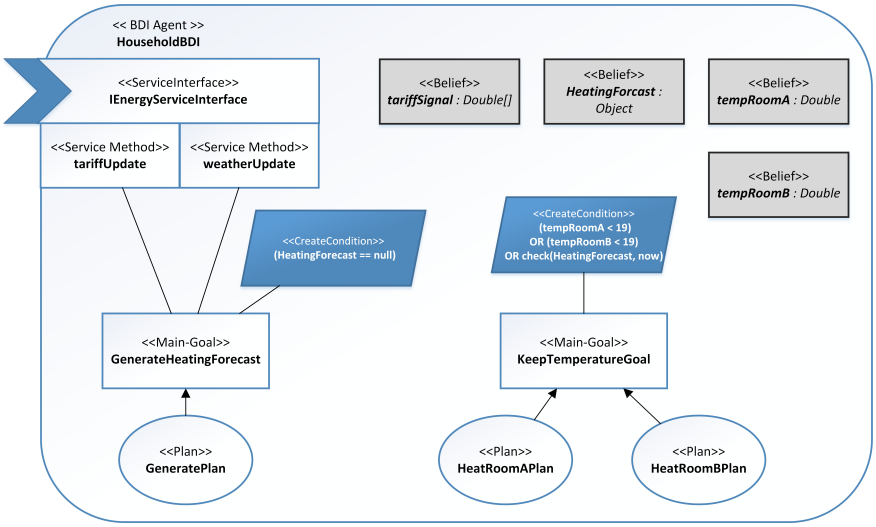

Abbildung B.2.: BDI-Agentenentwurf für eine automatisierte Heizungssteuerung anhand von externen Preissignalen und der Wettervorhersage

B.4. Fazit

Die Fortentwicklung von Tropos für die Implementierungsphase und die Softwaredokumentation steht noch am Anfang. Die Beschreibungselemente sind daher bislang stark an den Möglichkeiten von Jadex orientiert. Für eine Weiterentwicklung müssten daher mehr generische Beschreibungselemente entwickelt werden. Ebenfalls muss zu diesem Thema die Literatur eingehender untersucht werden, um mögliche Synergieeffekte und parallele Entwicklungen zu identifizieren.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 22. Mai 2014	
Ort, Datum	Unterschrift