Universität Lüneburg
Fakultät Umwelt und Technik
Bereich Informatik

Diplomarbeit

zur Erlangung des akademischen Grades

Diplom-Informatiker (FH)

Ant Colony Optimization zur Optimierung der Energieeffizienz in verteilten Netzen

Sebastian Schildt

Erstprüfer: Prof. Dr.-Ing. Ralph Welge

Zweitprüfer: Prof. Dr.-Ing. Dipl.-Inform. Eckhard C. Bollow

Angefertigt am Institut für Systems Engineering

Fachgebiet System- und Rechnerarchitektur

Leibniz Universität Hannover

08. September 2006

Kurzfassung

Der weltweite Energieverbrauch nimmt stetig zu. Statistiken zeigen, dass ein wesentlicher Anteil dieses Anstiegs nicht von der Industrie ausgeht, sondern auf Privathaushalte und öffentliche Einrichtungen zurückzuführen ist. Die Anzahl der elektrischen Verbraucher in einem durchschnittlichen Privathaushalt steigt. Es wird für den Menschen schwieriger zu beurteilen, was in seiner persönlichen Umgebung welche Menge Energie verbraucht. Diese Mannigfaltigkeit an Verbrauchern manuell im Sinne eines effizienten Umgangs mit Energie zu managen ist kaum möglich. In dieser Arbeit wird ein System skizziert, welches es einem Gebäude erlaubt seine Energieströme in autonomer Weise selbst zu verwalten. Ein hochdynamisches Netz verteilter Netzwerkknoten soll in der Lage sein elektrische Verbraucher entsprechend den Bedürfnissen der Benutzer zu verwalten mit dem Ziel, den Gesamtenergiebedarf des Netzes zu minimieren. Es ist eine besondere Herausforderung, dass in diesem Netzwerk gleichberechtigter Knoten auf Grund von technischen Randbedingungen kein Knoten globales Wissen über das Gesamtnetzwerk akkumulieren kann. Für diese Optimierungsaufgabe wurde die Ant Colony Optimization (ACO) ausgewählt. Damit innerhalb des hochdynamischen Netzwerks eine Optimierung nur auf Basis der lokalen Informationen der Netzteilnehmer durchgeführt werden kann, wird eine angepasste ACO Variante, das das Asynchronous Ant System, eingeführt. Diese Arbeit stellt eine Java basierte Simulation des AAS Algorithmus' im verteilten Netz sowie eine ANSI C Implementierung des AAS Systems auf einer Embedded Plattform vor. Es zeigt sich, dass es mit Hilfe des AAS möglich ist, Optimierungsaufgaben im verteilten Netz zu lösen.

Abstract

The total use of energy worldwide is increasing every year. Statistics show that the most significant part of this rise is not caused by the industry, but it's due to increased energy usage in the residential, commercial & public service sectors. The number of electric consumers in an average household is increasing. Assessing the amount of energy every device or appliance consumes is getting more difficult. It is close to impossible to manually manage the great number and diversity of energy consumers in an average household in an energy efficient manner. This thesis presents a system which allows a buildung to manage its inherent energy flows in an autonomous manner. A highly distributed network of embedded network nodes will be able to control and manage electrical consumers in such a way, that the user's needs and wishes are taken care of while in the same time following the goal of using the minimum possible energy to accomplish this task. Due to technical constraints pointed out in this thesis no network node is able to accumulate global knowledge of the state of the network and it's participants. The Ant Colony Optimization (ACO) was selected for this optimization problem. In order to be able to conduct an optimization based on each node's local information only, a special ACO variant called Asynchronous Ant System (AAS) will be introduced. This thesis presents a Java simulation of the AAS system and an actual ANSI C implementation on an embedded platform. It is shown, that the AAS is a feasible solution to the problem of performing optimizations in highly distributed, loosely coupled systems.

Inhaltsverzeichnis

1. Einleitung **1**
- 1.1. Motivation . . . 1
- 1.2. Szenario . . . 2
- 1.3. Ziel . . . 3

2. Stand der Technik **5**
- 2.1. Verwendete Embedded Komponenten . . . 5
  - 2.1.1. Infineon XC167 . . . 5
  - 2.1.2. VSM Runtime System und VISE IP Stack . . . 6
  - 2.1.3. Lastrelais . . . 6
- 2.2. Verwendete IT Komponenten . . . 7
  - 2.2.1. Repast . . . 7
  - 2.2.2. XML Schema & Xerces XML Parser . . . 7
  - 2.2.3. WireShark Netzwerk Analyser . . . 8
- 2.3. PDA Umgebung . . . 8
  - 2.3.1. IBM WebSphere Everyplace Micro Environment . . . 9
  - 2.3.2. Windows Mobile PDA . . . 9
- 2.4. Netzwerktechnologien für Wohn- und Zweckgebäude . . . 10
- 2.5. L3 Middleware . . . 10
  - 2.5.1. Abgrenzung zu anderen Techniken . . . 13
- 2.6. Metaheuristiken . . . 14
  - 2.6.1. Local Search . . . 14
  - 2.6.2. Simulated Annealing . . . 15
  - 2.6.3. Evolutionäre Algorithmen . . . 15
  - 2.6.4. Ant Colony Optimization . . . 15

3. Ant Colony Optimization **17**
- 3.1. Natürliches Vorbild der ACO . . . 17
  - 3.1.1. Informationen aus der Umgebung - Stigmergie . . . 19
- 3.2. Modellierung . . . 20
  - 3.2.1. Verhalten der Ants . . . 22
- 3.3. ACO Varianten . . . 23
  - 3.3.1. Ant System . . . 25
  - 3.3.2. Elitist Ant System . . . 28
  - 3.3.3. Rank Based Ant System . . . 29
  - 3.3.4. MAX-MIN Ant System . . . 30

3.3.5. Ant Colony System 31
3.4. ACO zur Energieeffizienzoptimierung 32

4. Asynchronous Ant System **35**
4.1. Das Energieproblem 36
4.2. Modellierung 37
4.2.1. Verhalten der Ants 37
4.3. Auswahl des nächsten Peers 39
4.4. Pheromonverteilung 39
4.5. Pheromon Verdampfung 40
4.6. Beispiel 41
4.7. Erweiterungen & Performance 44

5. Implementierung **45**
5.1. Testumgebung 45
5.2. Datenstrukturen 46
5.2.1. ANT 47
5.2.2. Problem Repräsentation 48
5.2.3. Lösungen 49
5.2.4. Pheromon Storage 50
5.3. Verhalten bei Fehlern 51
5.4. Repast Java Simulation 52
5.4.1. Benutzeroberfläche 52
5.4.2. Topologiekonfiguration 57
5.4.3. ACO Datenstrukturen 59
5.5. L3 Services 61
5.5.1. ACO Service 61
5.5.2. Interface Service 62
5.5.3. Trade Service 62
5.6. PDA Anwendung 63
5.7. Ant Verarbeitung 65
5.7.1. Forward Ants 65
5.7.2. Backward Ants 67

6. Zusammenfassung und Ausblick **71**

A. Anhang **75**
A.1. XML Schema für Toplogiebeschreibungen 75
A.2. Topolgiedaten 76
A.2.1. TestNetwork.xml 76
A.2.2. LongLine.xml 78
A.2.3. FullyConnected.xml 80
A.2.4. Bridge.xml 82
A.3. Erklärung zur Diplomarbeit 85

Abbildungsverzeichnis **87**

Literaturverzeichnis **89**

1. Einleitung

1.1. Motivation

Der weltweite Energieverbrauch nimmt stetig zu. Die International Energy Agency prognostiziert, dass der weltweite Energieverbrauch bis 2030 um 50% ansteigen wird. Die Mehrheit dieses Anstiegs wird von den fossilen Energieträgern Öl und Gas getragen werden. Die IEA geht davon aus, dass der Anteil der regenerativen Energiequellen am Primärenergieverbrauch bis 2030 nur 2% erreichen wird. Dieser Anstieg im Energieverbrauch wird zu einer erheblichen Steigerung des Ausstoßes von Treibhausgasen führen. Der weltweite CO_2 Ausstoß stieg von 15622Mt 1973 auf 24983Mt im Jahr 2003.

Ein wesentlicher Teil dieser Energie wird von elektrischen Verbrauchern benötigt. Der weltweite Verbrauch elektrischer Energie stieg von ca. 439 Mtoe[1] 1973 auf fast 1175 Mtoe 2003. Das entspricht einem weltweiten Elektrizitätsumsatz von 15.223 TWh. Dabei sank der Anteil der Industrie zwischen 1973 und 2003 von 51,3% auf 42,2% ab. Im selben Zeitraum stieg der Anteil aus den Sektoren Agrarwirtschaft, Öffentliche Einrichtungen, Gewerbe & Haushalte, von 46,3% auf 56%. Der Anstieg im Energiebedarf geht also nicht primär von Großverbrauchern aus, sondern zu einem wesentlichen von vielen einzelnen Haushalten und Gewerben[1].

Diese Zahlen bedeuten, dass die Menschheit sich in Zukunft dem Problem des wachsenden Energiebedarfs und den damit verbundenen ökologischen Folgen stellen muss. Ansatzpunkte sind

- den Energieverbrauch mindern
- den wachsenden Energiebedarf aus ökologisch unbedenklichen Quellen decken
- den Menschen diese Problematik bewusst machen, und so jeden Einzelnen zum Handeln bewegen

Der umsichtige und bewusste Umgang mit den natürlichen Ressourcen des Planeten wird oft unter dem Schlagwort „Sustainable ...“ zusammengefasst. Die „World Commission on Environment and Development“ definierte 1987 „Sustainable Development“ als

> „development that meets the needs of the present without compromising the ability of future generations to meet their own needs“[12]

[1] Mtoe=Megatonne Öleinheiten, vergleichende Maßeinheit für den Energieverbrauch

Längst schon handelt es sich bei Überlegungen zur Energieeffizienz nicht mehr um ideelle Gedankenspielereien. Konzepte und Produkte zum Energie sparen und nachhaltigem Handeln spielen heute in Wirtschaft und Industrie eine große Rolle. Gerade auch die Endverbraucher schauen auf Grund gestiegener und weiter steigender Energiepreise bei Neuanschaffungen auch auf die Energieeffizienz. Bekannte Beispiele sind die Einteilung bei Kühlgeräten in Energieeffizienzklassen und die, sich in den letzten Jahren verstärkt durchsetzenden, Kompaktleuchtstofflampen („Energiesparlampen"). Nicht zuletzt ist ein „grünes Image" heute ein wichtiger Wettbewerbsfaktor. Beispielsweise hat sich der japanische Elektronikkonzern Toshiba in seiner „Umwelt Vision 2010"[2] vorgenommen die „Umwelteffizienz" seiner Produkte gegenüber denen von 2000 bis 2010 um 120% zu steigern. Hierbei wird der gesamte Lebenszyklus eines Produktes von der Rohstofferzeugung bis zur Verschrottung berücksichtigt.

Die vorliegende Arbeit entstand im Rahmen des Forschungprojektes VAUST (Verteilte autonome Systeme und Technologien) an der Universität Lüneburg. Es soll gezeigt werden, wie ein verteiltes technisches System entwickelt werden kann, dass in autonomer Weise den Energieverbrauch seiner konstituierenden Teilnehmer minimiert und den Menschen dabei unterstützt sein Ziel, einen möglichst minimalen Energieverbrauch zu verursachen, zu erreichen.

1.2. Szenario

Die Anzahl der elektrischen Verbraucher in Privathaushalten ist in den letzten Jahren enorm angestiegen. Heute haben beispielsweise viele Haushalte nicht nur einen PC, sondern zwei oder drei. Der Leistungsbedarf der Rechner ist in den letzten Jahren kontinuierlich angestiegen. Nicht nur der Rechner verbraucht Energie, neben dem Rechner finden sich vermutlich Steckernetzteile für den Drucker, einen Scanner, USB- oder Ethernet Hubs. Moderne TV Geräte sind auf Grund neuer Technik vielleicht energiesparender als ein vergleichbares Gerät vor einigen Jahren, dafür wächst die durchschnittliche Bildschirmdiagonale und die Anzahl der Geräte pro Haushalt. Viele dieser Geräte verbrauchen auch im ausgeschalteten Zustand Energie. Es gibt Geräte die werden niemals ausgeschaltet, wie DSL Router oder WLAN Accesspoints. Dazu kommt eine große Zahl von Akkuladegeräten für mobile Verbraucher, wie das Handy, das PDA oder den Laptop.

Reichte es früher das Licht abzuschalten wenn es nicht benötigt wird, um Energie einzusparen, sieht sich der Mensch heute von einer großen Anzahl und Mannigfaltigkeit von Geräten umgeben. Diese sind manuell kaum im Sinne eines energieeffizienten Verhaltens zu managen, auch weil der Energiebedarf der einzelnen Verbraucher dem Menschen vielleicht gar nicht bewusst ist. Es ist immer noch eine gute Idee, das Licht auszuschalten, aber in einem durchschnittlich ausgestattetem Haushalt ist das heute unter Umständen nur ein Tropfen auf den heißen Stein.

[2] `http://de.computers.toshiba-europe.com/cgi-bin/ToshibaCSG/generic_content.jsp?service=EU&ID=ENVIRONMENT_SECTION_VISION_TGA_1205`

1.3. Ziel

Wir schlagen zur Kontrolle der Gebäudetechnik ein System von dezentralen autonomen Einheiten vor. Die einzelnen Geräte im Feld sind im Verbund ohne die Hilfe einer übergeordneten zentralen Instanz in der Lage die vitalen Funktionen eines Gebäudes im Sinne der Benutzer zu beeinflussen und dabei den Energieverbrauch zu minimieren. Mit Hilfe der an der Universität Lüneburg entwickelten Kommunikations Middleware L3 ist es möglich aus den einzelnen Verbrauchern im Haus ein dynamisches Netzwerk zu bilden, ohne dass man die einzelnen Knoten separat konfigurieren muss und ohne dass spezielle Infrastruktur wie z.B. Adressvergabedienste zur Verfügung stehen müssen. Die automatischen Konfigurationsfähigkeiten von L3 insbesondere in Verbindung mit Funktechnologien wie ZigBee oder die Datenübertragung über das Stromnetz (PLC) werden die Installation und Nachrüstung des Systems in vorhandene Gebäude extrem einfach machen, da weder neue Buskabel verlegt, noch ein kompliziertes Software Setup vorgenommen werden muss.

Damit ist die Kommunikationsgrundlage für das Netz gegeben. Wenn das Gesamtsystem jetzt Energie sparen soll, bedeutet das, dass entschieden werden muss wann welche Verbraucher ausgeschaltet oder heruntergefahren werden können. Die Entscheidung über das weitere Vorgehen muss von den einzelnen Knoten im Feld getroffen werden, da es eine zentrale Instanz wie einen Gebäudeleitrechner im System nicht gibt. Mehrere Faktoren haben Einfluss auf das Systemverhalten:

- Der Zustand der Umgebung: *Ist jemand im Raum?*
- Den Prioritäten der Verbraucher: *Wie kalt/dunkel darf es werden?*
- Den Vorlieben & Wünschen der Benutzer: *Welche Leistung möchte der Benutzer in Anspruch nehmen?*

Hier zeigt sich schon die zentrale Schwierigkeit, die bei dem Versuch Energie zu sparen auftaucht: Es muss immer einen Kompromiss zwischen Energieeinsparung & Komfort gefunden werden. Energie einsparen bedeutet Kompromisse eingehen. Dies muss aber in einer für den Menschen nachvollziehbaren Art und Weise geschehen, damit das System angenommen wird.

Bei Minimierung des Energieverbrauches bei vorgegebenen Benutzerwünschen oder der Maximierung des Komforts bei vorgegeben Energiegrenzen agiert das Gesamtsystem immer im Spannungsfeld Komfort<->Energieeffizienz. Bei der Minimierung oder Maximierung einer Komponente unter Einhaltung vorgegebener Randbedingungen handelt es sich um ein kombinatorisches Optimierungsproblem. Dieses Problem ist NP-Äquivalent. Das bedeutet, der Zeitaufwand dieses Problem mit deterministischen Mitteln zu lösen, wächst exponentiell mit der Größe des Problems. Daraus folgt, dass in Anbetracht der geringen zur Verfügung stehenden Rechenleistung nur nicht deterministische Algorithmen zur Lösung in Frage kommen. Ein weiteres Problem ist, dass die einzelnen Rechenknoten über lokales Wissen verfügen, welches den anderen Knoten nicht zugänglich ist. Die Akkumulation allen lokalen Wissens auf einem Knoten

ist aus Speicherplatzgründen nicht möglich. Damit muss als Grundlage einer Lösung auf jeden Fall ein konstruktiver Algorithmus verwendet werden, d.h. ein Alogrithmus der von Knoten zu Knoten „springen“ kann, um schrittweise eine Lösung zu konstruieren. In dieser Arbeit soll das Problem mit Hilfe der Ant Colony Optimization gelöst werden. Es handelt sich bei der ACO um eine Metaheuristik, d.h. ein standardisiertes Verfahren Probleme in NP zu lösen. Die ACO hat den Vorteil, dass die Lösung konstruktiv, d.h. schrittweise, aufgebaut wird und zu keinem Zeitpunkt, auch nicht zur Bewertung der Lösung, ein Knoten über alle Informationen aus dem Netzwerk verfügen muss. Informationen die im Verlaufe des Algorithmus zur Lenkung der Suche verwendet werden, sind ebenfalls über alle Knoten verteilt. Damit erscheint die ACO in der Lage zu sein den Rechen-, Speicher- und Informationsbedarf ideal auf die Knoten im Netzwerk zu verteilen.

Mit Hilfe der ACO soll ein System aus verteilten Rechenknoten das Ziel der Energieverbrauchsminimierung im Haushalt erreichen. Dieses Ziel steht immer in Konflikt mit den Komfortbedürfnissen des Menschen. Es soll gezeigt werden, dass ein System leistungsarmer Rechenknoten in der Lage ist mit Hilfe der Ant Colony Optimization dieses Problem im Verbund zu lösen.

Die Ergebnisse der Optimierung variieren mit den Randbedingungen, d.h. den a priori vorhanden Prioritäten der Verbraucher so wie den Vorgaben des Benutzers. Wie die Entscheidungen des Systems vom Menschen akzeptiert werden können, hängt also auch davon ab, wie transparent die Entscheidungen des Systems dem Menschen gemacht werden können. Hierfür können später andere Arbeiten integriert werden, wie die Zuordnung des Energieflusses zum einzelnen Menschen. Ebenso ist eine intuitive Parametrierbarkeit des Systems wichtig. Dies kann über ontologische Verfahren zur Beschreibung und Beeinflussung des Gesamtsystems erreicht werden.

Im Rahmen dieser Diplomarbeit soll zuerst eine Simulation der Optimierung für den PC erstellt werden. Dazu wird das Repast Agenten Simulations Framework benutzt, welches auf der Java Plattform aufsetzt. Nachdem die Simulationsphase abgeschlossen ist, soll der entwickelte verteilte Algorithmus als L3 Service auf dem XC167 implementiert werden, und seine Funktionsfähigkeit im realen Beispiel erprobt werden. Es soll möglich sein, im Rahmen eines definierten Gesamtenergiebudgets, Verbraucher ein- und auszuschalten. Hierbei werden die Verbraucher die benötigten Energiekontingente aus dem verteilten Netzwerk arbitrieren. Die Verbraucher können zugewiesene Energiekontingente untereinander in Form von Energiezertifikaten austauschen. Die Priorität der Verbraucher bestimmt im Falle eines Konflikts darüber, welcher Verbraucher zugunsten des anderen abgeschaltet wird.

Zum Schalten und Visualisieren des aktuellen Energiezustandes des Netzes wird eine PDA Applikation entwickelt. Der PDA ist über ein WLAN Gateway mit dem Netz der Embedded Peers verbunden. Es wird eine Testumgebung aus mehreren Embedded Peers und elektrischen Verbrauchern aufgebaut, um die die Applizierbarkeit des Algorithmus zu zeigen.

2. Stand der Technik

2.1. Verwendete Embedded Komponenten

In diesem Abschnitt sollen kurz wesentliche Hard- und Softwarekomponenten vorgestellt werden, die auf Embedded Seite zur Realisierung dieser Arbeit beigetragen haben. Es kann im Rahmen dieses Abschnitts nicht im Detail auf alles eingegangen werden. Vertiefende Informationen finden sich in den angegebenen Quellen.

2.1.1. Infineon XC167

Die Embedded Variante des ACO wurde auf einem Mikrocontroller der XC166 Familie aus dem Hause Infineon entwickelt. Es handelt sich um eine Familie von 16 Bit Prozessoren. Die XC166 Architektur ist eine Modernisierung der älteren C166 Architektur, welches ebenfalls weiter erhältlich ist. Im Vergleich zur älteren Variante ist die Pipeline von 4 auf 5 Stages angewachsen. Die Zykluszeit wurde verkürzt. Alle XC Derivate besitzen eine MAC Einheit und verfügen über Hardware Debugging Support (OCDS)[16].

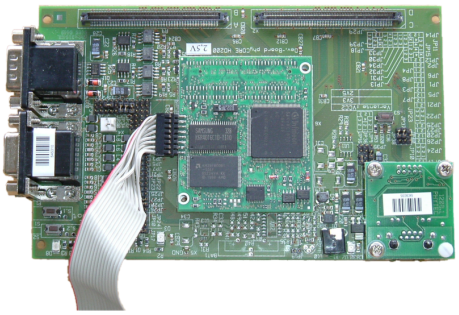


Abbildung 2.1.: phyCore XC167

Entwickelt wurde auf pyhCore XC167 Evaluation Kits (Abbildung 2.1) der Firma Phytec[1]. Das verwendete Evaluation Board ist mit dem Crystal CS8900 Ethernet Chip sowie einer RJ45 Buchse bestückt. Als C Toolchain kam Tasking von Altium[2] in der Version 8.5 zum Einsatz.

[1] `http://www.phytec.de/`
[2] `http://www.altium.com/`

2.1.2. VSM Runtime System und VISE IP Stack

Auf dem XC167 Controller läuft das VSM (Virtual State Machine) Runtime System[26]. Bei der VSM handelt es sich um ein nicht preemptives, nachrichtenorientiertes Embedded Betriebssystem für den XC167 Controller. Die VSM enthält den IP Stack VISE (Versatile IP Stack for Embedded Systems)[25]. Der VISE Stack ist ein kleiner hochmodularer Embedded IP Stack der an der Universität Lüneburg entwickelt wurde. VISE unterstützt UDP, TCP und HTTP. Die in dieser Arbeit verwendete L3 Middleware (siehe Abschnitt 2.5) setzt auf VISE auf.

2.1.3. Lastrelais

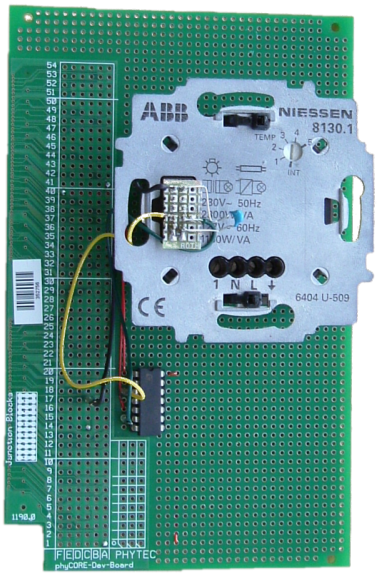


Abbildung 2.2.: Relais auf einem Phytec Expansionboard

Um elektrische Verbraucher vom Mikrocontroller aus zu schalten wurden Relais der Firma ABB Niessen[3] verwendet. Die Relais wurden von der Firma Busch-Jaeger[4] (ebenfalls zur ABB Gruppe gehörend) zur Verfügung gestellt. Es handelt sich um den Typ 8130.1. Dieses Relais ist für einen maximalen Strom von 10A ausgelegt und kann über zwei potentialfreie Eingänge vom XC167 Controller geschaltet werden. Abbildung 2.2 zeigt das Relais auf einem Expansionsboard für den XC167. Das Expansionsboard kann an das phyCore Evlaluationsboard (Abbildung 2.1) gesteckt werden.

[3] http://www.abb.es/niessen/
[4] http://www.busch-jaeger.de/

2.2. Verwendete IT Komponenten

In diesem Abschnitt sollen kurz wesentliche Hard- und Softwarekomponenten vorgestellt werden, die auf IT Seite zur Realisierung dieser Arbeit verwendet wurden. Eine klassische IT Umgebung (PC mit Java Plattform) kam bei der Entwicklung der Simulation und zum Monitoring des Embedded Systems zum Einsatz. Es kann im Rahmen dieser Arbeit nicht im Detail auf alles eingegangen werden. Vertiefende Informationen finden sich in den angegebenen Quellen.

2.2.1. Repast

Das Repast Toolkit[5] wurde verwendet um die ACO Simulation zu erstellen. Repast ist ein Akronym für „The Recursive Porous Agent Simulation Toolkit“. Es wurde an der Universität von Chicago entwickelt und war ursprünglich als Ersatz für Swarm gedacht: Swarm war damals bereits eine sehr bekannte ABM Plattform, welche allerdings auf ObjectiveC basiert. Repast sollte ein Swarm ähnliches Framework unter Java bieten. Später bekam Swarm Java Bindings, welche aber immer noch stark erkennen lassen, dass Swarm eigentlich ein ObjectiveC Framework ist. Repast wurde unabhängig von Swarm weiterentwickelt, aber es es sind immer noch Swarm Konzepte vorhanden. Ursprünglich war Repast hauptsächlich für sozialwissenschaftliche Simulationen ausgerichtet[19], inzwischen handelt es sich aber um ein umfangreiches Paket, dass sich allgemein anwenden lässt. Es existieren ebenfalls Repast Implementierungen für Python und .NET, welche alle vom selben Team gepflegt werden. Ein Vergleich verschiedener ABM Systeme ist in [19] zu finden. Eine ausführliche Vorstellung von Repast findet sich in [3].

2.2.2. XML Schema & Xerces XML Parser

Für die Konfiguration der Simulation wird ein XML File verwendet. Zum Parsen der XML Daten wurde der Xerces2 Java Parser[6] des Apache Projekts verwendet. Dieser Parser bietet volle Unterstützung für XML Schema. Ein XML Schema wurde benutzt, die Struktur der Konfiguration formal zu beschreiben. Bei XML Schema handelt es sich um eine W3C Empfehlung[7], die es erlaubt bestimmte Klassen von XML Dokumenten zu beschrieben. Es kann festgelegt werden, welche Tags enthalten sein können und wie diese im Dokument angeordnet sein müssen. Es können die Datentypen der einzelnen Attribute und Elemente festgelegt werden, sowie neue Datentypen definiert werden. XML Schemas lösen die älteren, weniger leistungsfähigen Document Type Definitions (DTD) ab. Ein XML Dokument dessen Struktur der Beschreibung in einem XML Schema entspricht, wird als Instanzdokument des Schemas betrachtet. Ein Schema fähiger XML Parser kann überprüfen, ob ein bestimmtes XML Dokument alle Constraints der zugehörigen Schema Beschreibung erfüllt. Das kann bei der wei-

[5] `http://repast.sf.net/`
[6] `http://xerces.apache.org/xerces2-j/`
[7] `http://www.w3.org/TR/xmlschema-0/`

teren Verarbeitung viele Überprüfungen sparen, da die Applikation sicher sein kann, dass der erzeugte DOM Tree der im Schema beschriebenen Struktur entspricht. Eine ausführliche Einführung in XML Schema findet sich in [24].

2.2.3. WireShark Netzwerk Analyser

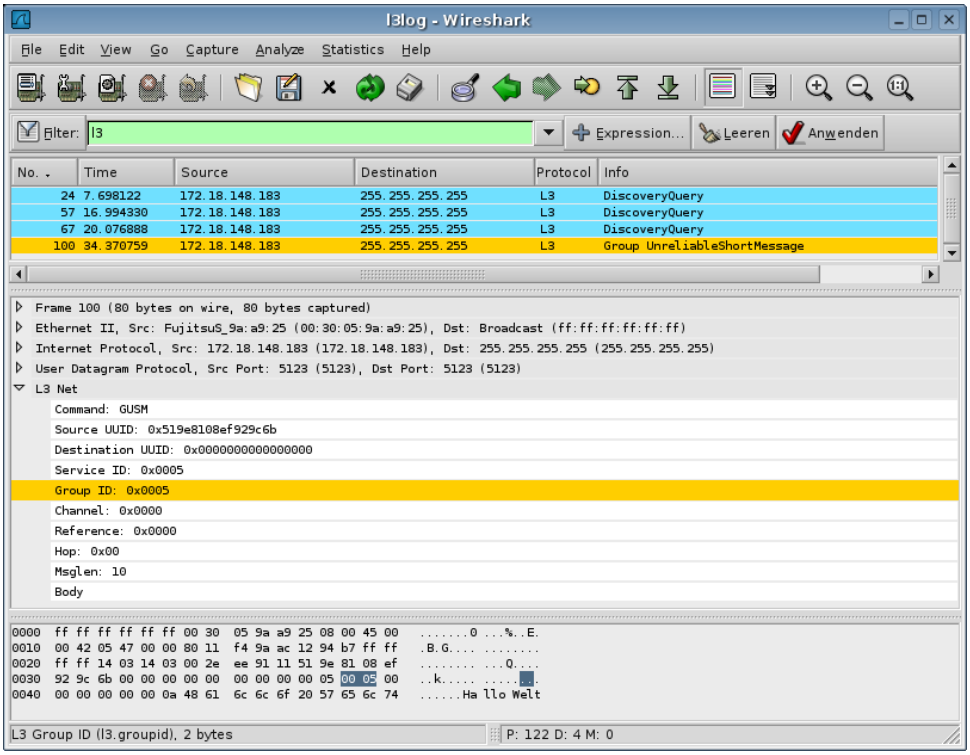


Abbildung 2.3.: WireShark Analyser mit L3 Plugin

Bei der Entwicklung wurde der WireShark[8] Multiprotokoll Netzwerk Analyser eingesetzt. Das Tool kann sämtlichen Datenverkehr an einem Netzwerkinterface mitschneiden, und die Daten übersichtlich nach Protokollschichten geordnet anzeigen. Da für die verwendete L3 Middleware (2.5) ebenfalls ein WireShark Plugin existiert, war es möglich den Datenverkehr zwischen den Embedded Peers zu analysieren. Damit konnten in der Entwicklungsphase Implementierungsfehler aufgedeckt und das Verhalten des Gesamtsystems beobachtet werden.

2.3. PDA Umgebung

In diesem Abschnitt soll die PDA Umgebung vorgestellt werden. Der PDA kommt als Benutzerinterface zum Einsatz und ist die Schnittstelle zwischen Mensch und

[8] `http://wireshark.org/` (früher unter dem Namen „Ethereal“ bekannt)

Embedded System.

2.3.1. IBM WebSphere Everyplace Micro Environment

Die PDA Anwendung zur Steuerung des Systems basiert auf Java. Sie läuft auf der J9 VM aus dem IBM WebSphere Everyplace Micro Environment[9]. Die J9 VM ist auf dem Stand von Java 1.3. Es wurde auf die APIs des J2ME Personal Profiles[10] zurückgegriffen. Die Anwendung hat eine grafische Oberfläche, die mit Hilfe der im Personal Profile vorhandenen AWT[11] Bibliothek erstellt wurde.

2.3.2. Windows Mobile PDA

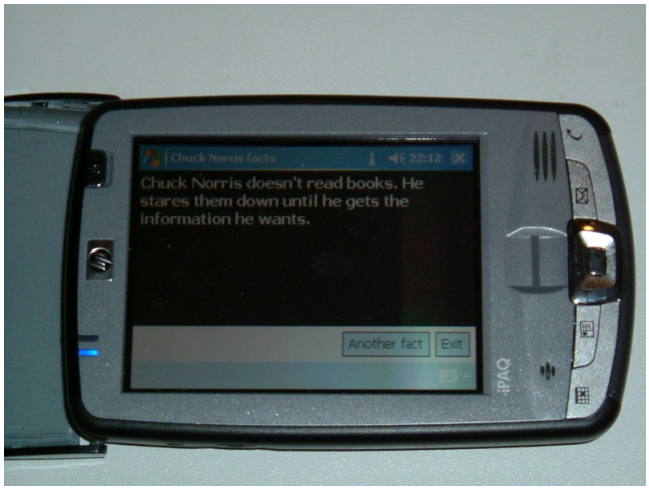


Abbildung 2.4.: iPaq HX2790

Die Anwendung wurde auf zwei Windows Mobile 5[12] PDAs entwickelt. Windows Mobile ist eine Distribution aus einem schmalen Betriebssystem mit entsprechenden Anwendungen von Microsoft, das speziell für Geräte wie Smartphones oder PocketPCs entwickelt wurde. Das Betriebssystem stellt ein Subset der Win32 APIs zur Verfügung und auch die Benutzeroberfläche gleicht dem der Windows Betriebssysteme auf PCs. Als Geräte kamen ein HP iPaq HX2790 (Abbildung 2.4) und einem Dell Axim 51v zum Einsatz. Beide Geräte verfügen über einen 624MHz Intel XScale Prozessor sowie ein integriertes WLAN Modul.

[9] `http://www-306.ibm.com/software/wireless/weme/`
[10] `http://jcp.org/en/jsr/detail?id=216`
[11] Abstract Windowing Toolkit
[12] `http://www.microsoft.com/windowsmobile/default.mspx`

2.4. Netzwerktechnologien für Wohn- und Zweckgebäude

Es gibt heute bereits einen etablierten Markt für Gebäudeautomatisierung und eine Vielzahl an anerkannten Standards für geeignete Bussysteme. Zu den verbreiteten Bussystemen zählen der European Installation Bus EIB (EN50090) oder der LON Bus (Local Operating Network Bus, ANSI/EIA-709.x, EIA-852 und EN14908). Beide Busse setzen auf eine separate Busverkabelung. Als Alternative werden Verfahren benutzt, die Daten per PLC über das vorhandene Stromnetz zu übertragen. Im Gegensatz zu den im EDV Bereich eingesetzten PLC Verfahren (Homeplug u.ä.) handelt es sich hier um wesentlich schmalbandigere Verfahren. Beispiele für in der Gebäudetechnik eingesetzte PLC Verfahren sind das alte X10 Protokoll oder das neuere EHS (European Home Systems). In den letzten Jahren sind verstärkt energiesparende Funkverfahren, die sich für den Einsatz in der Gebäudeautomatiserung eignen, entwickelt worden. Mit ZigBee scheint sich ein Standard in letzter Zeit besonders durchzusetzen. ZigBee setzt auf dem IEEE 802.15.4 Standard[15] auf. In dem IEEE 802.15.4 Standard sind die unteren Schichten des OSI Referenzmodells[17], nämlich einen Physical und einen Mac Layer, spezifiziert. ZigBee setzt darauf noch einen Network Layer und einen sogenannten „Application Support Layer“ auf[27].

Damit gibt es für die unteren Protokollschichten zum Transport von Daten zur Gebäudeautomatisierung bereits erprobte Standards. Bei der klassischen Gebäudeautomatisierung ist das Gesamtsystem sehr ähnlich aufgebaut wie die Automatisierung in Fabriken: Auf unterster Ebene befinden die „dummen“ Feldgeräte (Schalter, Temperatursensoren, etc.) welche über einen der o.g. Busse verbunden sind. Die Steuerung erfolgt eine Ebene höher in der Gebäudeleittechnik. Hier wird üblicherweise ein PC Programm eingesetzt, welches die Vorgänge auf der Feldebene visualisieren und steuern kann. Solche komplexen Systeme werden in der Industrie bzw. in großen Zweckgebäuden eingesetzt. Ihr Einsatz ist mit einem erheblichen Administrations- und Wartungsaufwand verbunden, so dass ein derartiges System für Privathaushalte ebensowenig in Frage kommt wie für viele kleine Gewerbe. Für private Haushalte werden einfachere Produkte angeboten. Es können hier primär die relevanten Verbraucher wie Lampen oder Rollläden gesteuert werden. Viele Automatisierungslösungen in diesem Bereich beschränken sich auf das reine Fernsteuern der Verbraucher. Sofern die Systeme selbstständig agieren können, handelt es sich im wesentlich um Zeitschaltuhren, die zu fest programmierten Zeiten bestimmte Verbraucher schalten, oder auf Knopfdruck eine vorher festgelegte Konfiguration („Beleuchtungsszenarien“) herstellen. Diese Systeme haben sich allerdings nicht auf breiter Front durchsetzen können. Sie werden teilweise eher als „Spielerei“ gesehen, da für die meisten Anwender der erzielbare Nutzen den Administrations- und Installationsaufwand nicht aufwiegt.

2.5. L3 Middleware

L3 ist eine Kommunikations Middleware, die von den Embedded Peers zur Datenübertragung untereinander genutzt wird. Die PDA Anwendung kommuniziert ebenfalls

über L3 mit dem verteilten Netz der Embedded Peers. L3 wurde an der Universität Lüneburg entwickelt. Es ist ein sehr platzsparendes, Bandbreiten schonendes, einfach zu parsendes Datenformat (Abbildung 2.5) und ein darauf aufbauendes P2P AdHoc

Command	Source	Destination	ServiceId	GroupId
1 Byte	8 Byte	8 Byte	2 Byte	2Byte

Channel	Reference	Hop	Message length	Message payload
2 Byte	2 Byte	1 Byte	2 Byte	[message length] Byte

Abbildung 2.5.: L3 Nachrichtenformat

Netzwerkprotokoll. Die 3 L stehen dabei für Low Power, Low Cost, Low Datarate. Es existiert eine Java sowie eine ANSI C Implementierung von L3. Die Kommunikation der Embedded Peers wird in diesem Projekt über die L3 Middleware realisiert. Für die PDAs wird eine J2ME Version des Java L3 Cores verwendet.

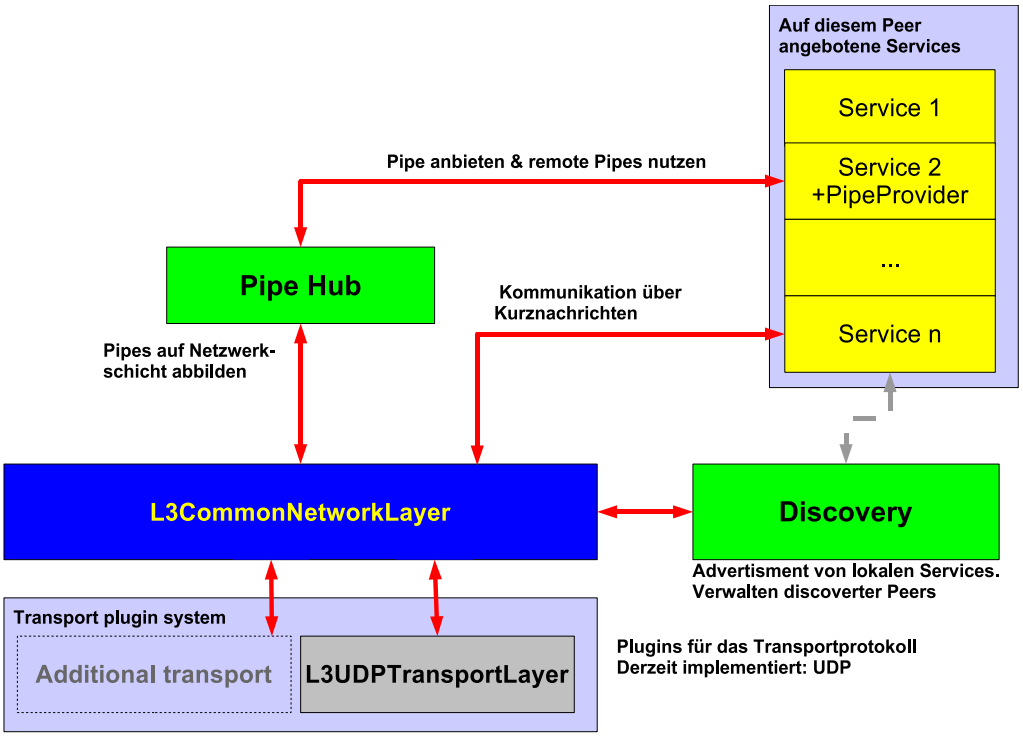


Abbildung 2.6.: L3 Architektur Übersicht[21]

An Hand der Java Referenzimplementierung soll hier kurz auf die wesentlichen Komponenten von L3 eingegangen werden[21][22]. Abbildung 2.6 zeigt eine Übersicht der L3 Architektur. L3 ist eine Service basierte Middleware, d.h. Peers die bestimmte Dienstleistungen anbieten, tun dies in Form von Services. L3 besitzt eine Discovery

Komponente, die die Präsenz von Services im Netz distributiert. Mit Hilfe des Discoveries entdecken L3 Peers andere L3 fähige Geräte im Netzwerk und können in Erfahrung bringen, welche Services unterstützt werden.

L3 unterstützt 2 Arten der Datenübertragung: Kurznachrichten und Pipes. Kurznachrichten kann man mit UDP vergleichen: Es werden einzelne unabhängige Datenpakete versendet. In Form der USM (Unreliable Short Message) handelt es sich wie bei UDP Datagrammen um eine nicht zuverlässige Kommunikation. Der Sender hat keine Möglichkeit festzustellen, ob die Nachricht angekommen ist oder nicht. Bei der RSM (Reliable Short Message) ist das anders: Hier bekommt der Sender ein Acknowledge wenn die Nachricht empfangen wurde. Wenn kein Acknowledge empfangen wurde, unternimmt L3 mehrere Zustellversuche bevor das Datenpaket aufgegeben wird.

Die zweite Variante, Daten zu übertragen sind die Pipes. Pipes simulieren stehende Verbindungen. Sie sind zum sicheren Versenden größere Datenmengen gedacht. Eine Pipe Verbindung bietet folgende Möglichkeiten

1. Es können größere Datenmengen versendet werden, als in eine Nachricht passen.

2. Die Übertragung ist zuverlässig: Es werden Acknowledges für versandte Datenpakete erwartet und es werden Übertragungen wiederholt sofern kein Acknowledges empfangen wurde.

3. Die korrekte Reihenfolge der Datenpakete wird gesichert. Wenn die zu übertragenen Daten aus mehreren Nachrichten bestehen, kann es passieren, dass Pakete evtl. über verschiedene Routen übertragen werden und daher nicht in der Reihenfolge ankommen, wie sie versendet wurden. Auch wird verhindert das Daten zweimal empfangen werden falls der Empfänger eine Nachricht zwar empfangen hat, aber das Acknowledge erst in dem Moment absetzt wo der Sender seine Übertragung wiederholt.

Die Punkte 1 und 2 können auch von mehreren hintereinander versendeten RSMs erfüllt werden. Die korrekte Reihenfolge der Datenpakete und die Vermeidung von Dubletten wäre mit RSM aber nur mit zusätzlichem Aufwand möglich.

Der L3CommonNetworkLayer ist eine wesentliche Schnittstelle um L3 nutzen zu können. Aus Programmiersicht könnte man sagen, dass der L3CommonNetworkLayer für L3 das ist, was das BSD Socket Interface für IP ist. Der L3CommonLayer zeigt, dass L3 unabhängig vom verwendeten Transportprotokoll ist. Über den L3CommonNetworkLayer können L3 Kurznachrichten an bestimmte Services auf durch ihre UUID identifizierte Peers gesendet werden. Damit stellt der L3CommonNetworkLayer im wesentlichen die Funktionen zur Verfügung, für die im IP Netzwerk UDP zuständig ist. Der L3CommonNetworkLayer ist von den Protokollen des darunterliegenden Netzwerks unabhängig, d.h. beim Programmieren arbeitet man nur auf L3 UUID Ebene und bekommt nichts davon mit, ob das darunterliegende Transportprotokoll beispielsweise UDP, Bluetooth oder irgend etwas völlig anderes ist. Erst das Transport Plugin System überträgt die eigentlichen Daten. Es liegt direkt unter dem

L3CommonNetworkLayer. Die Zuordnung von UUIDs zu den entsprechenden Transport spezifischen Adressen wird transparent in den entsprechenden Transport Plugins vorgenommen.

2.5.1. Abgrenzung zu anderen Techniken

Im Bereich der Service basierten Ad-Hoc Netzwerke existieren bereits verschiedene Lösungsansätze und Ideen. Teilweise wird versucht mittels zusätzlicher Erweiterungen und Aufsätze einen Teil der Ad-Hoc Philosophie auf die gängigen infrastrukturbasierten Netzwerke abzubilden. Beispiel hierfür ist ZeroConf von der IETF[13] (eher bekannt unter dem Namen „Bonjour“, Apples Implementierung von ZeroConf). ZeroConf definiert Verfahren zur automatischen Vergabe von IP Adressen ohne DHCP Server, ein Konzept zur Namensauflösung ohne zentralen DNS Server (Multicast DNS, mDNS) sowie Mechanismen zum Auffinden von Services im lokalen Netzwerk (DNS Service Discovery, DNS-SD). ZeroConf ist damit keine eigenständige Lösung für Ad-Hoc Computing sondern versucht lediglich die Vorteile dieses Ansatzes in die infrastrukturbasierten Netze zu übertragen. Ähnlich ist auch UPnP[14] zu bewerten. Es basiert auf einer Reihe von standardisierten Netzwerkprotokollen und Datenformaten und dient zur herstellerübergreifenden Ansteuerung von Consumer Geräten (Stereoanlagen, Router, Drucker, Haussteuerungen) über ein IP basierendes Netzwerk, mit oder ohne zentrale Kontrolle durch sogenanntes Residential Gateway.

JINI [15] ist ein Java basiertes Protokoll, das von Sun Microsystems entwickelt wurde. JINI ermöglicht die Entwicklung verteilter Anwendungen. Zentrales Element bei JINI sind Dienste, die von Geräten im Netzwerk zur Verfügung gestellt werden. Konzeptuell ist JINI sehr umfangreich und mächtig, hat sich aber nicht durchsetzen können. Der Grund hierfür liegt vor allem darin, dass JINI auf Java angewiesen ist und die Entwicklung zu einer Zeit begonnen wurde als man bei Sun noch davon ausgegangen ist, dass sich Java auf breiter Front vor allem auch bei den Consumer Appliances durchsetzen würde.

Ein weiterer Ansatz stammt ebenfalls von Sun: JXTA[16]. JXTA selbst ist sehr breit angelegt und versucht alle Anforderungen von P2P Netzwerken in sich zu vereinen und umzusetzen. Auf der JXTA Website heißt es

> „JXTA technology is a set of open protocols that allow any connected device on the network ranging from cell phones and wireless PDAs to PCs and servers to communicate and collaborate in a P2P manner.“

Sun hat den Fehler von JINI sich auf ein vorgegebenes Runtime System zu stützen nicht wiederholt. Statt dessen setzt JXTA auf XML als zentrale Komponente. Zudem kennt JXTA das Konzept des Transports. Ein Transport ist ein Kommunikationskanal der JXTA Nachrichten transportieren kann. Die Referenzimplementierung enthält

[13] Internet Engineering Task Force, `http://www.ietf.org/`
[14] Universal Plug and Play, `http://upnp.org/`
[15] Java Intelligent Network Infrastructure, `http://www.jini.org/`
[16] `http://www.jxta.org/`

Transporte für TCP oder HTTP. Neue Transporte können so prinzipiell leicht ergänzt werden. Der umfassende Ansatz und die Verwendung von XML bringt ganz eigene Probleme mit sich: JXTA benötigt immens viel Ressourcen. Der Bandbreitenbedarf ist sehr hoch da ständig sehr ausführliche und redundante XML Nachrichten (schon für Kernfunktionen teilweise im zweistelligen KB Bereich) verschickt werden. Zudem ist die Verarbeitung von XML sehr rechenzeitaufwändig, so dass die erwähnten Mobiltelefone wohl derzeit die kleinsten Geräte sind auf denen sich JXTA noch sinnvoll implementieren lässt. Kleinere Embedded Systems, besonders wenn ein möglichst geringer Energieverbrauch gefordert ist, können nicht von JXTA profitieren.

L3 versucht allgemein zu sein wie JXTA, aber nicht so umfassend. L3 ist speziell für die Bedürfnisse schmalbandiger Kommunikationskanäle und geringem Energiebedarf zugeschnitten. L3 ist eine Protkollspezifikation und somit unabhängig von der verwendeten Programmiersprache oder anderen implementationsspezifischen Randbedingungen. Die Java Implementation von L3 hat von JXTA das Konzept der Transporte übernommen und ausgebaut.

2.6. Metaheuristiken

Metaheuristiken sind Algorithmen mit denen sich kombinatorische Optimierungsprobleme lösen lassen. Keine Metaheuristik kann jedoch garantieren, dass eine optimale Lösung gefunden wird. Es handelt sich hier immer um nicht deterministische Algorithmen. Sie werden daher auf Probleme angewandt die anders nicht handhabbar wären. Andere bekannte Metaheuristiken neben der ACO sind Local Search, Simulated Annealing oder Evolutionäre Algorithmen. Viele dieser Techniken lassen sich auch kombiniert anwenden. Einige Metaheuristiken sollen in diesem Abschnitt kurz vorgestellt werden

2.6.1. Local Search

Grundlage für die lokale Suche ist eine gegebene, gültige Lösung. Diese kann z.B. zufällig durch Raten erzeugt sein. Der Local Search Algorithmus zieht im nächsten Schritt benachbarte (lokale) Lösungen heran, indem die ursprüngliche Lösung modifiziert wird. Dazu muss definiert sein, was eine benachbarte Lösung im Kontext des Problems ist. Welche benachbarte Lösung als nächstes akzeptiert wird, hängt wieder von der LocalSearch Variante ab. Im einfachsten Fall wird eine benachbarte Lösung nur akzeptiert, wenn sie bezüglich der Bewertungskriterien für Lösungen dieses Problems besser ist. Dies wird dann „hill climbing“ genannt. Hierbei besteht allerdings die Gefahr, dass der Algorithmus bei einem lokalen Optimum stecken bleibt. Eine andere Local Search Variante ist Tabu Search. Beim Tabu Search werden nicht alle benachbarten Lösungen zugelassen, z.B. kann verhindert werden, dass bestimmte Lösungen mehrmals getestet werden.

2.6.2. Simulated Annealing

Simulated Annealing (Simuliertes Tempern) ist eine weitere Metaheuristik. Vorbild ist hier das abkühlen eines Metalls. In der Metallurgie werden Metalle erwärmt und dann wieder abgekühlt, was zur Ausbildung größerer Kristallstrukturen mit weniger Defekten führt. Metalle sind umso härter, je besser sich eine geordnete Kristallstruktur ausbildet. Bei zu schneller Abkühlung eines Metalls haben die Atome nicht ausreichend Zeit sich zu ordnen und es befinden sich zu viele einzelne Kristalle in der Struktur. Die Kristalle befinden sich dann in einer energetisch ungünstigen Konfiguration, sie stecken in einem lokalen Energieminimum. Um dieser Konfiguration zu entrinnen, müssen sie jedoch zunächst eine energetisch ungünstigere Konfiguration annehmen um danach eine günstigere zu erreichen. Im optimalen Fall befinden sie sich dann im globalen Energieminimum. Deshalb werden die Metalle gesteuert abgekühlt (Tempern). Durch das Tempern wird den Metallatomen noch einmal die Möglichkeit gegeben, sich so zu bewegen, dass sie beim Abkühlen einen energetisch günstigeren Platz finden können.

Dieses Verfahren wurde kopiert und in einen Algorithmus umgesetzt. Es wird eine zufällige Lösung erzeugt und bewertet. Die nächste Lösung entsteht durch eine zufällige Änderung der vorherigen Lösung. Der Grundgedanke dieses Verfahrens ist es, in der Anfangsphase der Optimierung auch schlechtere Lösungen zuzulassen, um nicht in lokalen Minima hängen zu bleiben. Im Verlauf des Verfahrens, also mit stetiger Absenkung der „Temperatur“, wird die Akzeptanz von Verschlechterungen immer unwahrscheinlicher. Ab einer bestimmten Temperatur werden dann nur noch Verbesserungen akzeptiert.

2.6.3. Evolutionäre Algorithmen

Bei den evolutionären Ansätzen handelt es sich um eine ganze Reihe von Verfahren. Allen gemein ist, dass sie die Prinzipien der Evolution in der Natur zum Vorbild nehmen. Evolutionäre Ansätze arbeiten mit Populationen von potentiellen Lösungen. Diese werden dann gemäß den Verfahren der Genetik, wie Reproduktion, Rekombination und Mutation, verändert. Je nach verwendeter Variante wird mal der Aspekt der Mutation (Evolutionsstrategien[20]) und mal der der Rekombination (genetische Algorithmen[14]) stärker betont. Wie in der Natur sorgt die Selektion dafür, dass nicht alle Individuen (Lösungen) überleben, bzw. sich in die nächste Generation fortpflanzen. Lösungen höherer Qualität haben bessere Überlebenschancen, d.h eine höhere Chance Teil der nächsten Generation zu werden.

2.6.4. Ant Colony Optimization

Die Ant Colony Optimization hat das Verhalten von Ameisen bei der Futtersuche zum Vorbild. Auf dem Weg zu einer Nahrungsquelle werden von den Ameisen um so mehr Duftstoffe verteilt, je kürzer der Weg ist. Diese Spuren animieren andere Ameisen denselben Weg zu wählen. Dadurch konvergiert das System zu einem Zustand, wo alle

Ameisen den kürzesten Weg nehmen. Dieses Prinzip wurde für die ACO Metaheuristik adaptiert.

Für diese Arbeit wurde der ACO als zu implementierende Metaheuristik gewählt, daher findet sich in Kapitel 3 eine umfassende Beschreibung.

3. Ant Colony Optimization

Die Ant Colony Optimization gehört zur Familie der Metaheuristiken. Vorbild für die Ant Colony Optimization ist das Verhalten von Ameisen bei der Futtersuche[11]. Die ACO wurde 1992 zuerst in Form des Ant Systems (AS) von dem Italiener Marco Dorigo in seiner Doktorarbeit eingeführt. 1999 wurde die ACO von Dorigo, Di Caro und Gambardella formal als Metaheuristik definiert, siehe auch [7] und [8]. Inzwischen gibt es viele ACO Varianten (z.B. Ant Colony System ACS [9] , MaxMin Ant System MMAS[23]).

In diesem Kapitel soll die ACO vorgestellt werden. In diesem und allen folgenden Kapiteln gilt folgender Sprachgebrauch: „Ameise“ meint das Insekt, während „Ant“ sich immer auf die künstlichen Ameisen, welche der ACO Algorithmus benutzt, bezieht. Als erstes wird das Verhalten der natürlichen Ameisen vorgestellt. Es folgt eine allgemeine Modellierung des ACO. Hier wird gezeigt, wie das Problem zu repräsentieren ist und welche Aktionen die Ants auf dem Modell ausführen. Dann werden einige bekannte ACO Varianten vorgestellt. Zuletzt wird gezeigt, warum die ACO am besten für das Problem der Energieeffizienzoptimierung geeignet zu sein scheint und warum andere in 2.6 vorgestellte Metaheuristiken nicht einsetzbar sind.

3.1. Natürliches Vorbild der ACO

Die ACO Metaheuristik ist angelehnt an das Verhalten von Ameisen bei der Nahrungssuche. Es kann beobachtet werden, dass sich zwischen Ameisennest und Futter sogenannte „Ameisenstraßen“ bilden, d.h. alle Ameisen nehmen den gleichen Weg zur Nahrungsquelle und zurück. Ferner zeigt sich, dass diese Straßen meist den kürzesten Weg zur Nahrungsquelle darstellen. Die Ameisen können sich ebenfalls an veränderte Umgebungsbedingungen anpassen. Die Abbildung 3.1 zeigt, wie die Ameisen den kürzesten Weg um ein Hindernis finden.

Es handelt sich hierbei um ein Beispiel von Selbstorganisation. Klar ist, dass es keine „Chefameise“ gibt, die alle Wege auskundschaftet und die anderen Ameisen auf dem kürzesten Weg dirigiert. Es kann ebenfalls ausgeschlossen werden, dass einzelne Ameisen eine Vorstellung von dem Weg haben, und sich bewusst für den kürzesten bekannten Weg entscheiden. Die Forschung hat gezeigt, dass die Ameisen ihren Weg mit Hilfe von Pheromonspuren finden. Pheromone sind chemische Substanzen, die eine Ameise in der Umgebung platzieren kann. Andere Ameisen können diese wahrnehmen und entsprechend darauf reagieren.

Das Beispiel in Abbildung 3.1 soll den Mechanismus verdeutlichen:

Im ersten Bild sieht man eine Ameisenstraße, die sich bereits zwischen Nest und Futter gebildet hat. Praktisch alle Ameisen nutzen diesen Weg, welcher die direkte

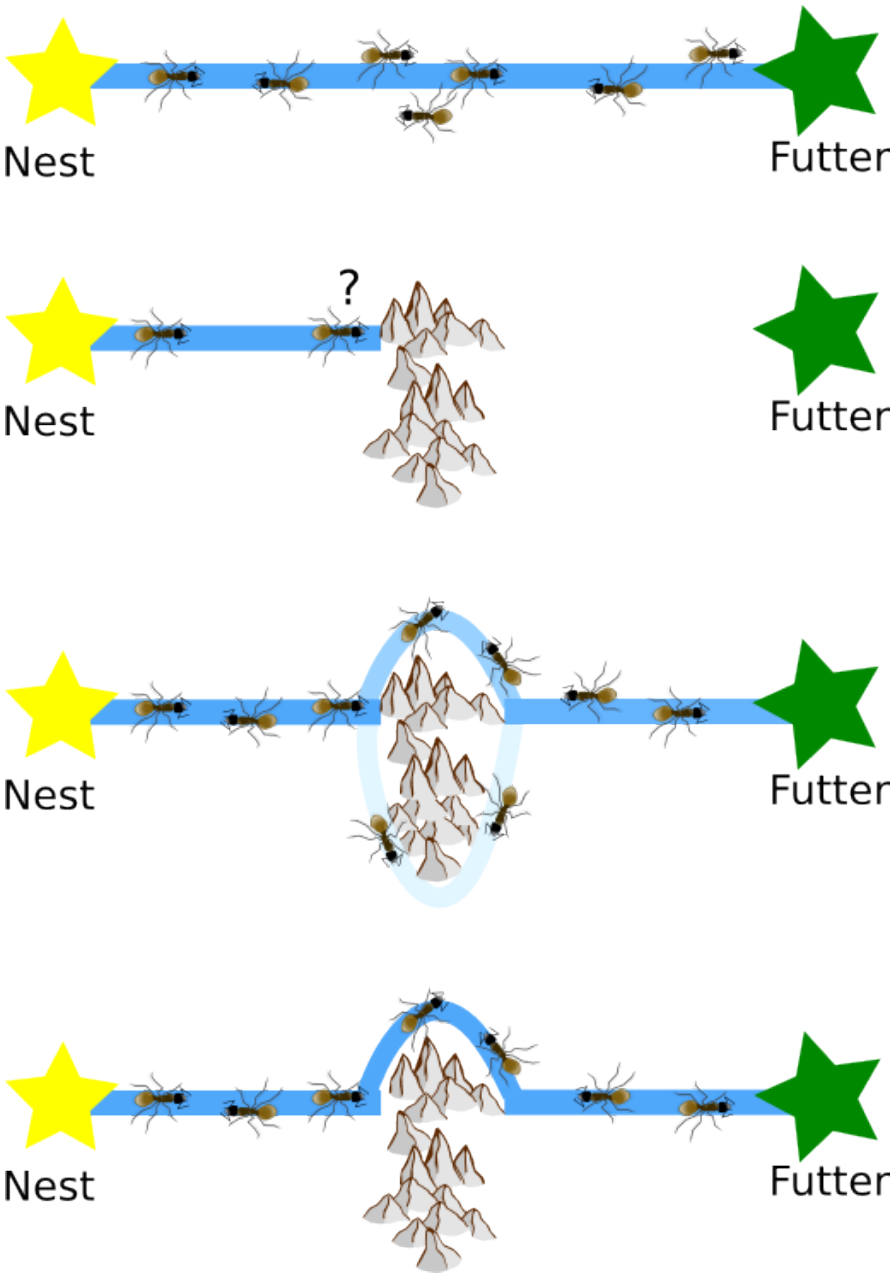


Abbildung 3.1.: Ameisen auf Nahrungssuche

und kürzeste Verbindung zwischen Nest und Futter darstellt.

In der zweiten Abbildung sieht man ein Hindernis auf dem direkten Weg zwischen Nest und Futter. Ameisen, die auf das Hindernis treffen, haben zwei Möglichkeiten: Sie können sich entweder nach links oder nach rechts wenden. Es werden daher näherungsweise 50% der Ameisen den rechten und 50% den linken Weg wählen.

Sowohl auf dem oberen als auch auf dem unteren Weg finden einige Ameisen die Futterquelle. Wenn eine Ameise die Futterquelle gefunden hat, macht sie sich auf dem Rückweg zum Nest. Auf diesem Weg verteilt sie Pheromone entlang des Weges. Da der obere Weg um das Hindernis kürzer ist, passieren die Ameisen ihn in höherer Frequenz. Daher werden auf dem oberen Weg mehr Pheromone platziert. Die Ameisen selber lassen sich derartig von den Pheromonspuren beeinflussen, dass die Wahrscheinlichkeit einem Weg zu folgen umso größer ist, je höher die Pheromonkonzentration ist. Eine Ameise, die vor dem Hindernis steht, wählt also mit steigender Wahrscheinlichkeit den oberen Weg, woraufhin die Pheromonkonzentration dort weiter steigt (Mitkopplung).

Auf diese Weise wählen nach einiger Zeit fast alle Ameisen den oberen Weg, auf dem die Pheromonkonzentration maximal ist, während auf dem unteren Weg keine neuen Pheromone verteilt werden und die alten langsam abgebaut werden.

Damit haben die Ameisen erneut den kürzesten Weg zwischen Nest und Futter gefunden. Es handelt sich hier um ein emergentes Muster, das allein aus dem Verhalten der einzelnen Ameisen (Verteilen und Reagieren auf Pheromonspuren) entsteht. Diese einfache Darstellung zeigt genau das Prinzip auf dem die ACO aufbaut. Eine umfassendere Darstellung der Wegfindung realer Ameisen ist in [5] zu finden.

3.1.1. Informationen aus der Umgebung - Stigmergie

Der Mechanismus, dass die Ameisen durch die Pheromone ihre lokale Umgebung verändern was wiederum das Verhalten der Ameisen beeinflusst, ist in der Biologie recht weit verbreitet. Man bezeichnet dieses Konzept als Stigmergie. Über die Veränderung der Umgebung durch das Verhalten des Systems findet eine indirekte Kommunikation zwischen den einzelnen Einheiten des Systems statt. Der Begriff Stigmergie wurde 1959 von dem französischen Biologen Pierre-Paul Grassé eingeführt als er die Entstehung von Termitenbauten untersuchte. Der aktuelle Status der entstehenden Struktur beeinflusst kontinuierlich das Verhalten der Termiten im weiteren Bauprozess. Grassé definiert Stigmergie als Stimulation von Arbeiterinnen durch das von ihnen Erschaffene[13]. Bei dem Beispiel mit den Termiten handelt es sich um eine sematektonsiche Stigmergie, d.h. der aktuelle Status des Werkes beeinflusst das weitere Verhalten der Individuen. Bei den Ameisen spricht man von einer markerbasierten Stigmergie. Hierbei erfolgt die Kommunikation über Marker, hier den Pheromonen, die nicht direkt mit der zu erfüllenden Aufgabe zusammenhängen.

Stigmergie Prozesse finden sich auch in anderen Bereichen. In der Technik werden bewusst Systeme aus der Natur kopiert und adaptiert, wie es bei der Ant Colony Optimization der Fall ist. Auch kann man das WorldWideWeb an sich als Beispiel für Stigmergie sehen: Eine Seite veröffentlicht eine Information, welche daraufhin von einigen Bloggern aufgegriffen wird, neue Webseiten entstehen und nach und nach

bildet sich eine Struktur vernetzter Inhalte.

3.2. Modellierung

Die ACO Metaheuristik kann auf jedes kombinatorische Optimierungsproblem angewendet werden, für das ein konstruktiver Lösungsalgorithmus definiert werden kann.

Damit die ACO angewendet werden kann, muss das Problem in eine Darstellung überführt werden, die es den künstlichen Ameisen erlaubt, Lösungen zu konstruieren. In diesem Abschnitt sollen alle Entitäten, die für die ACO geeignete Darstellung eines kombinatorischen Optimierungsproblems nötig sind, charakterisiert werden. Dieser und der nächste Abschnitt orientieren sich an der Darstellung in[11]. Die folgende Darstellung beschreibt nur die Metaheuristik. Die Beschreibung ist völlig unabhängig von dem zu lösenden Problem. Um diesen Abschnitt dennoch anschaulich zu gestalten, wird in den Beispielen auf den in Abbildung 3.2 dargestellten Graphen zurückgegriffen. Bei den Beispielen wird davon ausgegangen, dass es die Aufgabe ist in dem abgebildeten Graphen den kürzesten Weg von A nach D zu finden. Dabei darf man sich nur auf den Kanten des Graphen bewegen.

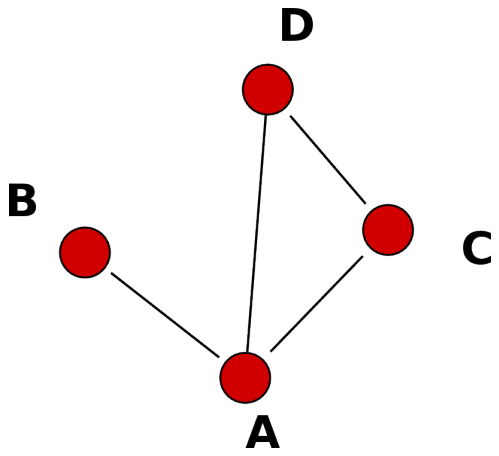


Abbildung 3.2.: Minimal Beispiel

Wir gehen von einem Optimierungsproblem (S, f, Ω) aus.

- S ist eine Menge von möglichen Lösungen
- f ist eine Bewertungsfunktion $f(s, t)$, die jeder Lösung $s \in S$ einen spezifischen Wert zuweist, der die Güte der Lösung quantifiziert.
- $\Omega(t)$ ist eine Menge von Randbedingungen. Im Beispiel gibt es die Randbedingung, dass alle Strecken bei A starten müssen. Außerdem darf jede Kante nur einmal innerhalb einer Lösung benutzt werden.

Der Parameter t zeigt, dass die Bewertungsfunktion und die Randbedingungen zeitabhängig sein können, wie es z.B. bei dynamischen Problemen wie dem Routing der Fall ist: Der Kostenfaktor eines Links ist abhängig von seiner Auslastung, welche wiederum zeitabhängig ist. Die Randbedingungen ändern sich, wenn z.B. ein Node ausfällt. Es ist das Ziel eine zulässige, global optimale Lösung s^* zu finden.

Um eine ACO durchführen zu können, muss die Modellierung des Problem folgende Bestandteile beinhalten:

- Eine endliche Menge $C = \{c_1, c_2, ..., c_{N_C}\}$ von Komponenten. N_C ist hierbei die Gesamtanzahl der Komponenten. Im Beispiel sind die Komponenten die Knoten A,B,C und D. Die Anzahl N_C ist 4.

- Die möglichen States des Problems sind definiert als Sequenzen endlicher Länge aus den Elementen von C: $x =< c_i, c_j, ...c_h >$. Die Menge aller möglichen States sei χ. Einige Beispiele für mögliche States im Beispiel sind $x_1 = \{A\}$, $x_2 = \{A, C, D\}$, $x_3 = \{B, D\}$ oder $x_4 = \{A, B\}$. Die Anzahl der Komponenten innerhalb einer Sequenz kann ausgedrückt werden als $|x|$. Im Beispiel: $|x_2| = 3$. Die maximale Länge der Sequenzen ist begrenzt: $n < +\infty$.

- Die Menge der potentiellen Lösungen S ist ein Subset von χ: $S \subseteq \chi$. Potentielle Lösungen im Beispiel sind $s_1 = \{A, D\}$ oder $s_2 = \{A, C, D\}$

- Eine Menge zulässiger States $\tilde{\chi}$, $\tilde{\chi} \subseteq \chi$. Es muss möglich sein, eine Sequenz $x \in \tilde{\chi}$ zu einer Lösung zu komplettieren, die die Randbedingungen Ω erfüllen kann. Beispiel für einen gültigen State im Beispiel ist x_1 oder x_2. Der State x_3 ist kein gültiger State, da er nicht zu einer Lösung komplettiert werden kann, da der Übergang B-D gar nicht erlaubt ist, da sich dort keine Kante im Graph befindet. Hierbei wird allerdings nicht garantiert, dass überhaupt eine Komplettierung s von x mit $s \in \tilde{\chi}$ existiert. Deshalb ist x_4 erstmal auch ein gültiger State, obwohl die Komplettierung dieses States zu einer Lösung nicht in $\tilde{\chi}$ liegt, da entweder der nicht gestattete Übergang BD genutzt werden müsste, oder die Kante AB ein zweites mal genutzt werden müsste, was die Randbedingungen verbieten.

- Eine nicht leere Menge S^* von optimalen Lösungen, wobei gilt $S^* \subseteq \tilde{\chi}$und $S^* \subseteq S$. Im Beispiel gibt es eine optimale Lösung $s_1 = \{A, D\}$.

- Mit jeder potentiellen Lösung $s \in S$ sind Kosten $g(s, t)$ verbunden. In den meisten Fällen gilt $g(s, t) \equiv f(s, t)$, $\forall s \in \tilde{S}$ wobei $\tilde{S} \subseteq S$ eine Menge von potentiellen Lösungen ist, die sich aus S mit Hilfe der Randbedingungen $\Omega(t)$ ermitteln lassen. Im Beispiel entsprechen die Kosten der Weglänge, d.h. die Anzahl der Kanten die zu einer Lösung gehören. $g(s_1, t) = 1$ und $g(s_2, t) = 2$. Je nach Definition kann im Beispiel gelten $g(s, t) \equiv f(s, t)$, oder wenn eine Bewertungsfunktion gewünscht ist, deren Ergebnis umso höher ist, je besser die Lösung ist, definiert man $f(s, t) = \frac{1}{g(s,t)}$.

Nach dieser Formulierung können Ants Lösungen konstruieren indem sie sich zufällig auf dem Graphen $G_C = (C, L)$ bewegen. Die Knoten des Graphen sind die Komponenten C, und die Menge L wird gebildet aus den Verbindungen der Komponenten C. Der Graph G_C wird Konstruktionsgraph genannt.

3.2.1. Verhalten der Ants

Ants konstruieren zufällige Lösungen bei ihrer Bewegung durch den Konstruktionsgraph $G_C(C, L)$. Dabei werden die Randbedingungen $\Omega(t)$ von den Ants beachtet. Für die meisten Anwendungen ist es sinnvoll, dass die Ants nur zulässige Lösungen konstruieren. Es kann aber Anwendungsfälle geben, wo es von Vorteil ist, wenn es auch möglich ist, dass die Ants eigentlich unzulässige Lösungen konstruieren.

Komponenten $c_i \in C$ und Verbindungen $l_{ij} \in L$ können Pheromonspuren τ_i bzw. τ_{ij} zugeordnet werden. Die Pheromonspuren kann man als globales Gedächtnis des Systems sehen. Die Ergebnisse der bisherigen Suche sind ihnen eingeprägt. Die Pheromonspuren werden von den Ants aktualisiert.

Etwas anderes sind heuristische Informationen. Diese beinhalten a priori Wissen über ein Problem oder Laufzeit Informationen, die aus einer anderen Quelle als den Ants selber stammen. Ein Beispiel wären zum Beispiel die Kosten, bzw. geschätzten Kosten η einer bestimmten Komponente oder Verbindung zur aktuell konstruierten Lösung hinzuzufügen. Diese Informationen werden von den Ants einbezogen wenn eine Entscheidung getroffen wird, wie sie sich auf dem Graph weiterbewegen.

Alle Ants haben folgende Eigenschaften:

- Sie wandern durch den Konstruktionsgraph $G_C(C, L)$ auf der Suche nach optimalen Lösungen $s^* \in S^*$
- Jede Ameise k hat ein Gedächtnis M^k. Das k ist hier kein Exponent, sondern ein Index, der ausdrückt, dass es sich um das Gedächtnis der Ant k handelt. Dies gilt auch für alle weiteren ^{k} in diesem Abschnitt. Das Gedächtnis kann benutzt werden, um Informationen über den Weg zu speichern, den die Ameise bis jetzt verfolgt hat. Dieses Gedächtnis kann genutzt werden
  - um akzeptable Lösungen zu erzeugen (Überprüfung der Randbedingungen)
  - zur Berechnung der heuristischen Informationen η
  - zur Bewertung der gefundenen Lösung
  - um den Weg der Ant zurück zu verfolgen
- Jede Ameise k hat einen Startzustand x_s^k sowie eine oder mehrere Abbruchbedingungen e^k. Normalerweise ist der Startzustand eine leere Folge oder eine Folge mit genau einer Komponente. Im Beispiel ist der Startzustand jeder Ant $x_s^k = \{A\}$, da festgelegt wurde, dass die Ants bei Knoten A starten. Die Abbruchbedingungen im Beispiel sind erfüllt, wenn eine Ant den Knoten D erreicht oder wenn sie keinen weiteren Zug machen kann ohne die Randbedingungen zu verletzen (siehe Zustand x_4 im vorherigen Abschnitt).

- Befindet sich die Ant im Zustand $x_r =< x_{r-1}, j >$ und keine der Abbruchbedingungen e^k trifft zu, so bewegt sich die Ant zu einem Knoten j in der Nachbarschaft $N^k(x_r)$ dieses Knotens. Das heißt, die Ant geht in den Zustand $< x_r, j > \in \chi$ über. Wenn mindestens eine der Abbruchbedingungen e^k erfüllt ist, stoppt die Ant.

- Während eine Ant eine potentielle Lösung konstruiert, sind in den meisten Applikationen Übergänge zu unzulässigen Zuständen verboten. Dies kann z.B mit Hilfe des Ant Gedächtnisses M^k oder mit entsprechenden heuristischen Informationen η sichergestellt werden. Im Minimalbeispiel gibt es keine heuristischen Informationen. Wäre der Graph aber gewichtet, d.h. wären die Längen der Kanten unterschiedlich, so wäre die Länge einer Kante eine heuristische Information.

- Eine Ant wählt ihren nächsten Schritt mit Hilfe einer probabilistischen Entscheidungsregel. Diese Regel ist eine Funktion, die von den lokal verfügbaren Pheromonspuren und heuristischen Informationen, dem Gedächtnis und dem aktuellen Status der Ant sowie den Randbedingungen abhängt. Die Pheromonspuren und heuristischen Informationen bestimmen die Wahrscheinlichkeit (dies ist probabilistische Teil der Regel), mit der ein bestimmter nächster Schritt ausgewählt wird.

- Wenn eine Komponente c_j zur aktuellen Lösung hinzugefügt wird, kann die Ant die Pheromonspur τ, die dieser Komponente oder dieser Verbindung zugeordnet ist, aktualisieren

- Sobald eine Lösung komplett ist, kann eine Ant ihren Weg zurückverfolgen und dabei die Pheromonspuren für die benutzten Komponenten aktualisieren

Die Ants arbeiten gleichzeitig und unabhängig voneinander. Schon eine einzelne Ant ist in der Lage eine Lösung zu konstruieren, wenn auch vermutlich eine relativ schlechte. Gute Lösungen sind das Resultat der Interaktion zwischen den Ants. Die Interaktion erfolgt indirekt über die Pheromonspuren, d.h. Informationen werden zwischen den Ants, über das Verteilen und Auslesen der Pheromonspuren, ausgetauscht. Im Prinzip handelt es sich um einen verteilten Lernprozess. Das besondere ist, dass sich in diesem Fall nicht die Ants an das Problem anpassen, sondern dass sie vielmehr die Art und Weise ändern, wie sich das Problem ihnen darstellt.

3.3. ACO Varianten

In diesem Abschnitt sollen einige bekannte ACO Varianten vorgestellt und die Unterschiede zwischen ihnen gezeigt werden. Die genaue Ausprägung der Algorithmen ist immer auch von dem zu lösenden Problem abhängig. Die exakte Formel für die Verteilung der Pheromone oder die Art und Weise der heuristischen Informationen können nicht immer auf generische Weise angegeben werden. Daher wird in diesem Vergleich davon ausgegangen, dass eine ACO auf das Traveling Salesman Problem

angewendet wird. Bei diesem Problem geht es darum, die kürzeste Verbindung einer Menge von Städten zu finden. Bedingung ist, dass jede Stadt genau einmal besucht wird. Bei dem Konstruktionsgraph $G = (C, L)$ sind die Komponenten C die Städte und L alle möglichen Verbindungen zwischen ihnen. Die Randbedingungen Ω stellen sicher, dass nur Lösungen konstruiert werden, die einer Permutation der Städte C entsprechen. Die Pheromonspuren sind den Kanten des Graphen zugeordnet, d.h. die Spur τ_{ij} ist ein Maß dafür, wie erstrebenswert es ist die Stadt j nach der Stadt i zu besuchen. Zudem existieren heuristische Informationen $\eta_{ij} = \frac{1}{d_{ij}}$, d.h. nach den heuristischen Informationen ist die Wahrscheinlichkeit die Stadt j direkt nach der Stadt i zu besuchen, umgekehrt proportional zum Abstand d_{ij} der beiden Städte (Abbildung 3.3).

Auch wenn es hier teilweise speziell um das TSP geht, hat sich gezeigt dass die Performance und das Verhalten der Algorithmen beim TSP durchaus auch auf andere Probleme übertragbar ist. Sofern nicht anders vermerkt, verhalten sich die vorgestellten Algorithmen wie in den Abschnitten 3.2 und 3.2.1 beschrieben. Als erstes wird das ursprüngliche Ant System[7] vorgestellt, das sozusagen die grundlegendste ACO Variante darstellt. Die anderen Varianten sind meist Erweiterungen des ursprünglichen Ant Systems. ACO Varianten, die das AS Verhalten grundlegend verändern, wie ANTS[18] oder Hyper-Cube AS[6], werden hier nicht betrachtet. Die Erläuterung der verschiedenen ACO Varianten folgt der Darstellung in [11].

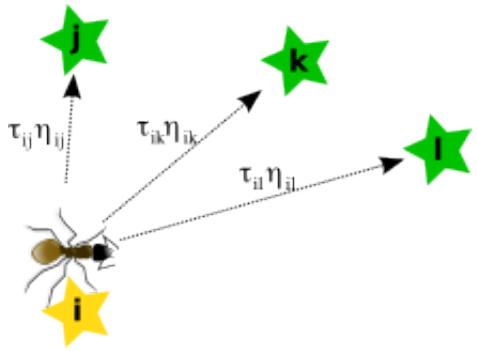


Abbildung 3.3.: Auswahl der nächsten Stadt

In Abbildung 3.4 ist ein einfaches TSP Problem skizziert. Die Städte A,B,C und D sollen besucht werden. Jede Stadt soll genau einmal besucht werden, so dass sich der kürzeste Weg ergibt. Die Entfernungen zwischen den Städten sind ebenfalls angegeben. Es handelt sich um ein symmetrisches TSP Problem, d.h. die Entfernung zwischen A und B ist genauso groß wie die Entfernung zwischen B und A: $d_{AB} = d_{BA}$. Da die Entfernung bekannt ist, lassen sich die heuristischen Informationen ebenfalls berechnen.

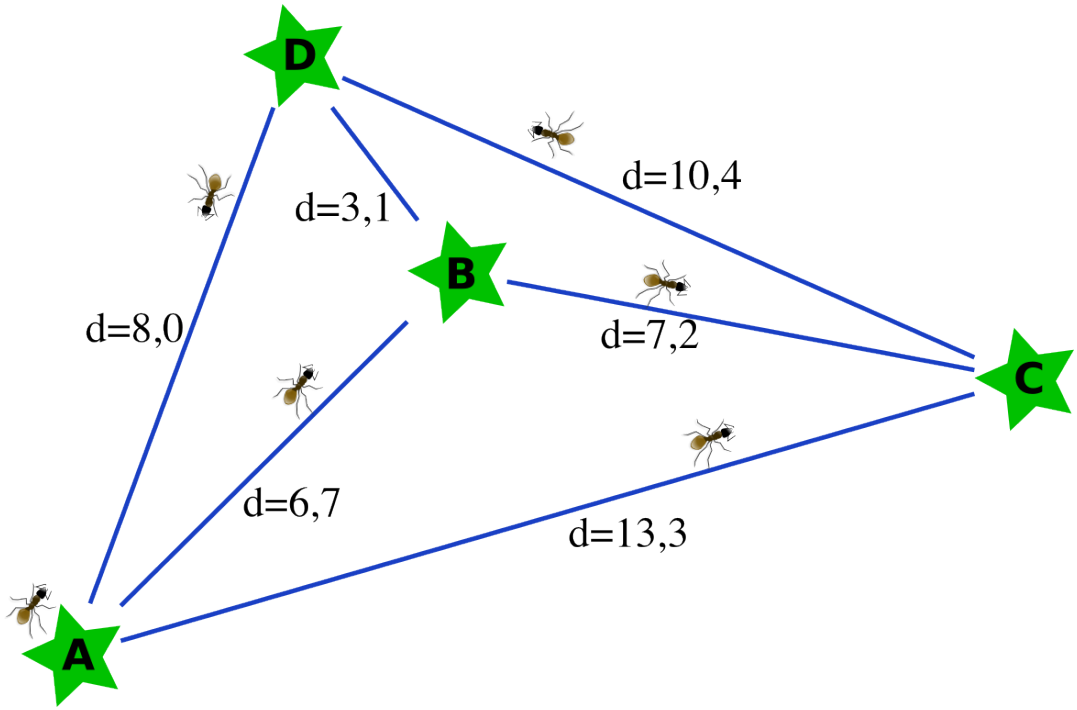


Abbildung 3.4.: TSP Instanz

$$\begin{aligned} \eta_{AB} &= \frac{1}{d_{AB}} \approx 0,149 \\ \eta_{AC} &= \frac{1}{d_{AC}} \approx 0,075 \\ \eta_{AD} &= \frac{1}{d_{AD}} \approx 0,125 \\ \eta_{BC} &= \frac{1}{d_{BC}} \approx 0,139 \\ \eta_{BD} &= \frac{1}{d_{BD}} \approx 0,323 \\ \eta_{CD} &= \frac{1}{d_{CD}} \approx 0,096 \end{aligned} \tag{3.1}$$

3.3.1. Ant System

Ursprünglich wurden 3 Ant System Versionen vorgestellt, die sich im Zeitpunkt der Pheromondeponierung unterscheiden. Die Varianten *ant-density* und *ant-qualitiy* aktualisierten die Pheromonspuren schon sobald eine Ant von einer Stadt zur nächsten gewandert ist, während die dritte Variante, *ant-cycle*, dies erst tut, wenn alle Ants ihren Touren komplettiert haben[10]. Weil sich gezeigt hat, dass die *ant-density* und *ant-quality* Varianten unterlegen waren, wird heute nur noch *ant-cycle* betrachtet. Beim AS gibt es 2 Phasen: Einmal die Konstruktion der Lösungen und dann die Verteilung der Pheromone.

Es hat sich als sinnvoll erwiesen, als Erstes alle Pheromonspuren so zu initialisieren, dass die Menge der Pheromone ein wenig höher ist, als die erwartete Menge der Pheromone, die die Ants in einem Durchgang verteilen würden. Dies kann erreicht werden,

indem die Pheromonspuren wie folgt initialisiert werden: $\forall(i,j), \tau_{ij} = \tau_0 = \frac{m}{C^{nm}}$. Hierbei ist m die Anzahl der Ants und C^{nm} die Länge einer Tour, die mittels einer einfachen Heuristik konstruiert wurde. Beim TSP bietet es sich an immer den nächsten Nachbarn zu wählen. Von einer Stadt ausgehend wählt man aus allen möglichen Folgestädten die aus, zu der die Entfernung d minimal ist. Auf Abbildung 3.4 bezogen, wäre die mit einer solchen Heuristik von A ausgehend konstruierte Tour: A-B-D-C, was eine Länge von $C^{nm} = 6,7 + 3,1 + 10,4 = 20,2$ ergibt. Es werden soviele Ants in einer Iteration losgeschickt, wie Städte vorhanden sind. Dann kann in jeder Iteration eine Ant aus jeder Stadt starten. Die initiale Pheromonkonzentration ergibt sich im Beispiel also zu $\tau_0 = \frac{4}{20,2} \approx 0,198$. Diese Menge wird zu Beginn der Optimierung jeder Kante zugeordnet. Wenn am Anfang keine, oder nur wenig Pheromone verteilt werden, kann es schnell passieren, dass sich die Ants im Laufe des ACO bei ihrer Suche sehr auf die zuerst generierten Touren konzentrieren, und somit eventuell nur ein lokales Optimum gefunden wird. Eine zu hohe initiale Pheromonkonzentration ist ebenfalls zu vermeiden, da es sonst sehr lange dauert bis die ursprünglichen Spuren verdampft sind und sich die Suche nach den von den Ants hinterlegten Pheromonspuren richtet.

Nachdem die Pheromonspuren initialisiert wurden, werden die Ants zufällig auf verschiedene Städte verteilt und beginnen dann mit der Konstruktion von Lösungen. Wenn die Ant die Entscheidung treffen muss von einer Stadt (Komponente) i zur nächsten Stadt zu wechseln, so berechnet sich die Wahrscheinlichkeit, dass die Ant k nach der aktuellen Stadt i die Stadt j besucht wie folgt[11]:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \tag{3.2}$$

Hierbei muss $j \in N_i^k$ sein. N_i^k ist die Nachbarschaft von i, d.h. eine Menge von Städten, die die Ant als nächstes unter Berücksichtigung aller Randbedingungen besuchen kann. Bei dem TSP ist das die Menge der Städte, die die Ant k bislang nicht besucht hat. Über die Parameter α und β kann die Gewichtung zwischen Pheromonspuren und heuristischen Informationen gesteuert werden. Wenn α größer ist, wirken sich die Pheromonspuren stärker aus, wenn β größer ist, werden den heuristischen Informationen mehr Bedeutung beigemessen. Auch dies kann an Hand des Beispiels nachvollzogen werden. Die Parameter α und β seien 1. Angenommen die erste Ant startet nach der Initialisierung in der Stadt A. Die Nachbarschaft von A, also Städte die als nächstes besucht werden können, sind D,B oder C. Da bislang noch keine Informationen verteilt wurden, sind die Pheromonspuren $\tau_{AB} = \tau_{AC} = \tau_{AD} = 0,198$. Die heuristischen Informationen η können 3.1 entnommen werden. Die Wahrscheinlichkeit, dass die Ant als nächstes die Stadt B besucht wird berechnet sich dann zu:

$$p_{AB} = \frac{[\tau_{AB}]^1 [\eta_{AB}]^1}{\sum_{l \in N_i^k} [\tau_{Al}]^1 [\eta_{Al}]^1} = \frac{0,198 \cdot 0,149}{0,198 \cdot 0,149 + 0,198 \cdot 0,075 + 0,198 \cdot 0,125} \approx 0,427 \tag{3.3}$$

analog gilt

$$p_{AC} = \frac{0,198 \cdot 0,075}{0,198 \cdot 0,149 + 0,198 \cdot 0,075 + 0,198 \cdot 0,125} \approx 0,215 \quad (3.4)$$

$$p_{AD} = \frac{0,198 \cdot 0,125}{0,198 \cdot 0,149 + 0,198 \cdot 0,075 + 0,198 \cdot 0,125} \approx 0,358 \quad (3.5)$$

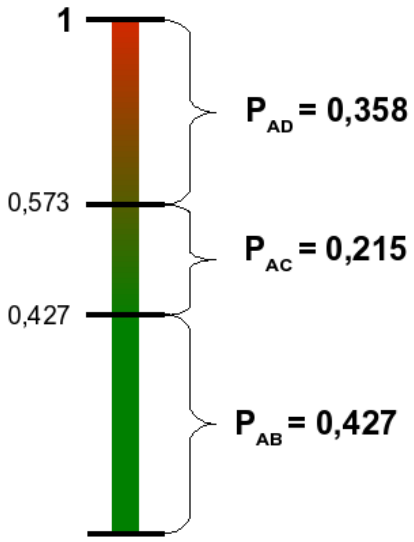


Abbildung 3.5.: Auswahl nach Wahrscheinlichkeiten

Diese Wahrscheinlichkeiten haben Einfluss darauf, welche Stadt als nächstes besucht wird. Abbildung 3.5 soll dies demonstrieren. Auf Grund der Formel 3.2 ergibt sich für jeden möglichen nächsten Schritt der Ant eine Wahrscheinlichkeit zwischen 0 und 1. Die Summe der Wahrscheinlichkeiten aller möglichen Schritte addiert sich dabei zu 1. Wenn die einzelnen Wahrscheinlichkeiten wie in Abbildung 3.5 dargestellt auf einem Zahlenstrahl abgebildet werden, so kann die Auswahl einer bestimmten Möglichkeit erfolgen, indem eine gleichverteilte Zufallszahl x zwischen 0 und 1 erzeugt wird. Mit $0 \leq x \leq 0,427$ wird Stadt B als Nächstes gewählt, mit $0.427 < x \leq 0,573$ C und mit $0,573 < x \leq 1$ D. Damit beträgt die Wahrscheinlichkeit 42,7% dass die Ant Stadt B wählt, 35,8% dass D gewählt wird und 21,5% dass C die nächste Stadt ist.

Nachdem alle Ants ihre Touren komplettiert haben, werden zunächst die Pheromone auf allen Kanten vermindert. Dies entspricht dem Verdampfen von Pheromonen. Die Pheromonverdampfung folgt folgender Formel

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall (i,j) \in L \quad (3.6)$$

Das heißt auf allen Kanten wird die Menge der Pheromone um einen bestimmten Prozentsatz vermindert. Über die Verdampfungsrate ρ kann geregelt werden, wieviel

Pheromone verdampfen sollen. Sei $\rho = 0,5$ dann würden nach der ersten Iteration Ants die Pheromone auf allen Kanten auf

$$\tau_{ij} = (1 - 0,5) \cdot 0,198 = 0,099 \tag{3.7}$$

verringert werden. Nach der Verdampfung werden die Pheromone der einzelnen Ants verteilt. Für eine Ant werden auf allen Kanten, die sie passiert hat Pheromone verteilt:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \quad \forall (i,j) \in L \tag{3.8}$$

Δ_{ij}^{k} ist die Menge an Pheromonen, die die Ant k deponiert:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^k} & \text{, wenn die Kante(i, j) zu } \mathsf{T}^{\mathsf{k}} \text{ gehört} \\ 0 & \text{, ansonsten} \end{cases} \tag{3.9}$$

C^k ist die Länge der Strecke T^k, die von der Ant k gefunden wurde. Je kürzer die Strecke umso mehr Pheromone werden auf den zu dieser Strecke gehörenden Kanten verteilt. Angenommen eine Ant aus dem Beispiel hat nach der ersten Iteration die Strecke A-D-B-C konstruiert, dann platziert sie Pheromone auf die Kanten τ_{AD} , τ_{DB} und τ_{BC}. Die Strecke hat eine Gesamtlänge von $C = 8,0 + 3,1 + 7,2 = 18,3$. Damit folgt

$$\Delta\tau = \frac{1}{C} = \frac{1}{18,3} \approx 0,0546 \tag{3.10}$$

Damit erhöht sich die den Kanten AD, BD und BC zugeordnete Pheromonmenge auf die nach der Verdampfung vorhandene Pheromonmenge + $\Delta\tau$:

$$\tau_{AD} = \tau_{BD} = \tau_{BC} = 0,099 + 0,0546 = 0,1536 \tag{3.11}$$

Natürlich kann sich die Pheromonmenge noch weiter ändern, wenn andere Ants dieser Iteration Lösungen konstruiert haben, die dieselben Kanten mit benutzen und daher ebenfalls weitere Pheromone auf diese Kanten deponieren.

Es hat sich gezeigt, dass die Performance des AS im Vergleich zu anderen Metaheuristiken nicht ideal ist, insbesondere bei größeren Problemen[11].

3.3.2. Elitist Ant System

Eine erste Erweiterung des Ant Systems war das Elitist Ant System (EAS)[10]. EAS funktioniert im wesentlichen wie AS. Was modifiziert wurde, ist die Pheromonverteilung. Es werden extra Pheromone auf die Kanten der Strecke verteilt, die die beste Lösung (best-so-far Tour) T^{bs} darstellt, die bislang gefunden wurde. Zu den Kanten der besten Lösung wird eine Pheromonmenge von $\frac{e}{C^{bs}}$ hinzugefügt. Hierbei ist C^{bs} die Länge der besten Tour und der Parameter e bestimmt, wieviel Gewicht der best-so-far

Tour beigemessen wird. Damit sieht die modifizierte Formel zur Pheromonverteilung wie folgt aus:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + e\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in L \tag{3.12}$$

$\Delta\tau_{ij}^{k}$ ist identisch zu Formel 3.9. Für $\Delta\tau_{ij}^{bs}$ gilt analog

$$\Delta\tau_{ij}^{bs} = \begin{cases} \frac{1}{C^{bs}} & \text{, wenn die Kante(i, j) zu } T^{bs} \text{ gehört} \\ 0 & \text{, ansonsten} \end{cases} \tag{3.13}$$

Das bedeutet alle Kanten die zur best-so-far Tour T^{bs} gehören, bekommen eine zusätzliche Pheromonmenge von $\frac{e}{C^{bs}}$. Mit $e = 4$ würde auf den Kanten der Strecke A-D-B-C folgender Pheromonbetrag verteilt werden (wenn eine Ant die Strecke durchlaufen hat, vgl. Formel 3.10):

$$\Delta\tau = \frac{1}{C} + e \cdot \frac{1}{C^{bs}} = \frac{1}{18,3} + \frac{4}{18,3} \approx 0,273 \tag{3.14}$$

Wichtig ist, dass solange A-D-B-C die best-so-far Tour bleibt, in allen weiteren Pheromonverteilungszyklen auf alle zugehörigen Kanten mindestens die Menge $\frac{e}{C^{bs}} = \frac{4}{18,3}$ Pheromone deponiert wird, unabhängig davon ob in dieser Iteration die Kante von einer Ant genutzt wurde.

Es hat sich gezeigt, dass das EAS in der Lage ist nach weniger Iterationen bessere Ergebnisse als AS zu finden.

3.3.3. Rank Based Ant System

Eine weitere AS Erweiterung ist das Rank Based Ant System (AS_{rank})[4]. Es handelt sich ebenfalls um eine Modifikation der Pheromonverteilung. Der prinzipielle Ablauf gleicht dem AS. Nachdem die Ants ihre Touren konstruiert haben, werden diese nach Länge geordnet. Nur die Ants, die die besten Lösungen produziert haben dürfen Pheromone verteilen. Je besser die Lösung umso mehr Pheromone werden verteilt. Es werden lediglich die jeweils besten Lösungen einer Iteration betrachtet. Zusätzlich werden auf der best-so-far Lösung immer Pheromone verteilt, auch wenn in einer Iteration keine neue best-so-far Lösung entstanden ist, sondern diese noch aus einer früheren Iteration stammt.

Damit folgt für die Formel zur Pheromonverteilung:

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{r=1}^{w-1} (w-r)\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in L \tag{3.15}$$

Mit dem Parameter w wird eingestellt wieviele Ants Pheromone verteilen. Die $w-1$ besten sowie die best-so-far Ant sind die einzigen Ants die Pheromone verteilen. Die Definition von $\Delta\tau_{ij}^{r}$ für die Pheromonmenge, die auf den Kanten der r-besten Lösung hinzugefügt wird ist analog zu 3.10 und 3.13:

$$\Delta\tau_{ij}^{r} = \begin{cases} \frac{1}{C^r} & \text{, wenn die Kante(i, j) zu } \mathsf{T}^{\mathsf{r}} \text{ gehört} \\ 0 & \text{, ansonsten} \end{cases}$$

Da die Formel vom Aufbau den entsprechenden Formeln des AS und EAS entspricht, wird hier kein Beispiel berechnet. Wie in den anderen Varianten ist die Pheromonmenge $\Delta\tau$ umgekehrt proportional zur entsprechenden Streckenlänge C. AS_{rank} weist ebenfalls eine deutlicheVerbesserung gegenüber AS auf. Nach [4] ist die Performance sogar noch ein wenig besser als beim EAS.

3.3.4. MAX-MIN Ant System

Das MAX-MIN Ant System (MMAS)[23] ändert die AS an 4 entscheidenden Punkten:

1. Die besten Touren werden ganz besonders gewichtet: Nur die beste Ant der aktuellen Iteration oder die best-so-far Ant dürfen Pheromone deponieren.

2. Auf Grund von (1) kann es dazu kommen, dass die Suche stagniert und die Ants immer derselben Tour folgen. Um ein unbegrenztes Wachstum der Pheromonkonzentration zu vermeiden, wird der Wertebereich der Pheromonkonzentrationen auf einer Kante auf das Intervall $[\tau_{min}, \tau_{max}]$ begrenzt.

3. Zu Beginn der ACO werden alle Pheromonspuren auf das obere Limit τ_{max} gesetzt. Da zudem die Verdampfungsrate der Pheromone gering ist, werden zu Beginn viele Touren erforscht ohne dass die Ants sich zu schnell auf die zuerst gefundenen konzentrieren.

4. Das System wird beobachtet. Tritt es in eine Stagnationsphase ein bzw. wurde für eine bestimmte Anzahl an Zyklen keine neue verbesserte Tour generiert, dann werden die Pheromonspuren gemäß (3) neu initialisiert.

Damit ist die Formel für das Pheromonupdate sehr einfach:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}, \quad \forall (i,j) \in L \tag{3.16}$$

Wie in den anderen Varianten gilt $\Delta\tau_{ij}^{best} = \frac{1}{C^{best}}$. Also für die Strecke A-D-B-C $\frac{1}{18{,}3}$ wie in den Beispielen zu AS und EAS. Es darf entweder nur die best-so-far oder die iteration-best Ant Pheromone verteilen. Wann welche Ant zum Zuge kommt ist implementierungsabhängig. Wird bevorzugt die best-so-far Ant genommen, so wird die Suche schnell um die Lösung T^{bs} fokussiert, wird mehr Gewicht auf die iteration-best Ant gelegt, so ist die Suche weniger gerichtet.

3.3.5. Ant Colony System

Das Ant Colony System (ACS)[9] unterscheidet sich stärker vom AS als die bisher gezeigten Varianten. Die Entscheidungsregel, welche Stadt als nächstes zu besuchen ist, wurde modifiziert. Das Aktualisieren und Verdampfen von Pheromonen wird nur auf den Kanten, die zur best-so-far Tour gehören, durchgeführt. Jede Ant die sich von Stadt i zu j bewegt, sorgt dafür, dass eine gewisse Menge Pheromone von der Kante (i, j) entfernt wird.

Wenn eine Ant k sich in der Stadt i befindet, wird die nächste Stadt nach folgender Formel gewählt

$$j = \begin{cases} argmax_{l \in N_i^k} \left\{ \tau_{il} \left[\eta_{il}\right]^\beta \right\}, & wenn\ q \leq q_0 \\ J, & ansonsten \end{cases} \tag{3.17}$$

Diese Formel wird von Dorigo als „*pseudorandom proportional rule*“[11] bezeichnet. Hierbei ist q eine gleichverteilte Variable im Wertebereich $[0, 1]$, q_0 ist ein Parameter, wobei gilt $0 \leq q_o \leq 1$. J wird entsprechend der AS Wahrscheinlichkeitsverteilung (Formel 3.2) ermittelt, wobei der Parameter $\alpha = 1$ gesetzt wird:

$$p_{ij}^k = \frac{\tau_{ij} \left[\eta_{ij}\right]^\beta}{\sum_{l \in N_i^k} \tau_{il} \left[\eta_{il}\right]^\beta} \tag{3.18}$$

Zusammenfassend bedeutet das für das Verhalten der Ants: Mit einer Wahrscheinlichkeit von q_0 wählt die Ant den bestmöglichen Zug basierend auf den vorliegenden Pheromonspuren und den heuristischen Informationen (1. Fall von 3.17). Hierbei spielt der Zufall bei der Wahl des nächsten Zuges keine Rolle, sondern nur die bislang hinterlegten Pheromonspuren und die vorhandenen heuristischen Informationen. Es handelt sich also eher um eine lokale Suche, die Exploration neuer Wege findet nicht statt. Würde im Beispiel in der ersten Iteration eine Ant in Stadt B stehen, so würde sie auf Grund der Regel die Stadt D als nächstes wählen. Die Nachbarschaft, d.h. die Städte die als nächstes besucht werden könne besteht für eine aus A kommende Ant aus den Städten C und D. In der ersten Iteration ist $\tau_{il} = \tau_0$ und damit für alle Kanten gleich. Die heuristischen Informationen sind für die Stadt D $\eta_{BD} \approx 0,323$ maximal, während sie für C $\eta_{BC} \approx 0,139$ geringer sind. Das Maximum für $\tau_{Bl} \left[\eta_{Bl}\right]^\beta$ wird also für $l = D$ erreicht.

Mit einer Wahrscheinlichkeit von $(1 - q_0)$ findet eine explorative Suche auf Basis der Wahrscheinlichkeiten gemäß der Formel 3.18 statt, ähnlich wie bei dem AS. Diese Formel entspricht 3.2 mit $\alpha = 1$, so dass das dortige Beispiel hier ebenfalls zutrifft. Mit dem Parameter q_o lässt sich also einstellen, ob sich das System bei der Suche auf Lösungen auf den Bereich der best-so-far Tour konzentrieren soll, oder ob verstärkt neue Touren gesucht werden sollen.

Bei dem ACS werden Pheromonspuren nur für die best-so-far Lösung gelegt. Auch die Verdampfung findet ausschließlich auf dem Weg der best-so-far Ant statt. Damit ergibt sich die kombinierte Formel für Pheromonplatzierung und Verdampfung:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in T^{bs} \tag{3.19}$$

Die zu deponierende Pheromonmenge wird um den Faktor ρ gesenkt, damit ergibt sich die neue Pheromonmenge auf der Kante als der gewichtete Mittelwert zwischen der vorher vorhandenen und der neu deponierten Menge. Die Definition von $\Delta\tau_{ij}^{bs}$ entspricht 3.13. Mit $\rho = 0,2$, $\tau_{AB} = 0,054$ und der best-so-far Tour A-D-B-C ergibt sich die aktualisierte Pheromonmenge auf der Kante AB zu

$$\tau_{AB} = (1-0,2) \cdot 0,04 + 0,2 \cdot \frac{1}{18,3} \approx 0,0429 \tag{3.20}$$

Zusätzlich zum Update der Pheromonspuren der best-so-far Tour, findet ein lokales Pheromonupdate statt, wenn eine Ant eine Kante (i,j) passiert hat:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0 \tag{3.21}$$

Durch die Pheromonreduktion soll verhindert werden, dass der Algorithmus zu schnell stagniert weil alle Ants sich auf die bislang gefundenen Touren konzentrieren. Hierbei sind ξ und τ_0 Parameter. τ_0 ist die Menge an Pheromone die zur Initialisierungsphase auf allen Kanten deponiert wurde. Dorigo[9] schlägt $\tau_0 = \frac{1}{nC^{nm}}$ vor, wobei n der Anzahl der Städte entspricht und C^{nm} die Länge einer heuristisch konstruierten Tour ist (vergleiche AS). Im Beispiel wäre $\tau_0 = \frac{1}{4*20,2} \approx 0,0124$. Für ξ gilt $0 \leq \xi \leq 1$. Ein geeigneter Wert für ξ ist je nach Problem experimentell zu ermitteln[9]. Mit $\xi = 0,1$ würde sich die Pheromonmenge aus 3.20 wie folgt verändern wenn eine Ant die Kante AB benutzt:

$$\tau_{AB} = (1-0,1) \cdot 0,0429 + 0,1 \cdot 0,0124 \approx 0,0399$$

3.4. ACO zur Energieeffizienzoptimierung

In dieser Arbeit wurde die ACO ausgewählt um die Energieverteilung im Netz zu optimieren. Auf Grund der technischen Randbedingungen erschien keine andere der z.B in Abschnitt 2.6 vorgestellten Metaheuristiken geeignet. Da das Netzwerk aus verteilten Systemen besteht, existiert nirgendwo globales Wissen, oder ein globales Abbild des Gesamtsystems. Da die einzelnen Peers technisch aus Kosten- und Energieverbrauchsgründen möglichst einfach sein sollen, haben sie eine geringe Rechenleistung und vor allem wenig Speicher. Auch wenn es theoretisch möglich wäre, dass ein „SuperPeer“ globales Wissen über das Netz hat, d.h. die lokalen Informationen aller anderen Teilnehmer sammelt, und sie in ein komplettes Abbild des Netzes integriert, so macht dies technisch keinen Sinn: Die Informationen müssten bei jeder Änderung im Netz übertragen werden, was den Netzwerkverkehr und damit die benötigte Bandbreite erhöht. Außerdem wäre ein SuperPeer ein „Single Point Of Failure“ (SPOF), dass heißt fällt dieser aus, käme es zu einem Denial of Service. Daher ergibt sich folgende Situation im Netz: Jeder Peer verfügt über lokales Wissen, d.h. er kennt sehr genau

seinen aktuellen Energiestatus und seinen Energiebedarf. Die Existenz anderer Peers ist ebenfalls bekannt, jedoch nichts über deren inneren Zustand.

In dieser Situation sind alle Metaheuristiken, die mit ganzen Lösungsinstanzen arbeiten nicht anwendbar: Simulated Annealing oder LocalSearch *modifizieren* eine gegebene Lösung. Dies ist aber nur sinnvoll möglich, wenn die Entität, die die Lösung modifiziert genügend Informationen über den inneren Zustand der einzelnen Peers hat. Wenn die Modifikation also vorsieht, dass ein Peer x mehr Energie abgeben soll, dann sollte ja vorher bekannt sein, ob dieser Peer die Energie überhaupt bereitstellen könnte, d.h. es muss etwas über den inneren Zustand des Peers x bekannt sein. Noch problematischer sind Evolutionäre Algorithmen. Für sie gilt das oben gesagte ebenfalls, zudem setzt der Ansatz voraus, dass *mehrere* potentielle Lösungen (die Population) im Speicher gehalten und miteinander verknüpft werden müssen. Dies vervielfacht daher den Speicherverbrauch in dem Peer, der die Optimierung vornehmen soll.

Die ACO dagegen ist fähig eine Optimierung mit rein lokalem Wissen vorzunehmen. Die Ants wandern von Peer zu Peer wobei jeder nur seinen Teil zur Lösung beiträgt, d.h. nur auf Grund seines lokalen Wissens entscheidet ein Peer welchen Beitrag er zur Lösung geben kann. Es ist nicht relevant, was die vorherigen Peers zur Lösung beigetragen haben oder was die nachfolgenden Peers anbieten können. Es ist auch nicht nötig, dass ein Peer verschiedene Lösungen gegeneinander abwägt und den Algorithmus in eine bestimmte Richtung lenkt. Dies passiert über die im Netz verteilten Pheromonspuren, welche die Ants auf Grund der Daten der gefundenen Lösung auf ihrem Rückweg durch das Netz verteilen können. Neue Ants lassen sich auf ihrem Weg durch das Netz von diesem Gedächtnis vergangener Lösungen leiten.

Dennoch zeigt sich, dass die bekannten ACO Varianten meist ein gewisses globales Wissen voraussetzen: Wenn die ACO auf einem Rechner abläuft, welcher den kompletten Problemgraphen simuliert, hat man natürlich zu jedem beliebigen Zeitpunkt Zugriff auf sämtliche Pheromonspuren und den Zustand sämtlicher Ants und Komponenten. Damit sind einige der oben dargestellten Optimierungen, wie das vorab Verteilen von Pheromonen auf alle Kanten des Graphen, oder das Ranking aller bislang gefunden Lösungen, überhaupt erst möglich. Daher musste für diese Arbeit eine angepasste ACO Variante entwickelt werden, welche im folgenden Kapitel detailliert vorgestellt wird.

4. Asynchronous Ant System

In diesem Kapitel soll die in dieser Arbeit implementierte ACO Variante vorgestellt werden. Zuerst wird die Problemstellung und deren Modellierung beschrieben. Dazu wird noch einmal auf die allgemeine ACO Modellierung in 3.2 und 3.2.1 zurückgegriffen.

Die hier vorgestellte ACO Variante, das Asynchronous Ant System (AAS), ist in der Lage das Problem unter Berücksichtigung der technischen Randbedingungen zu lösen. Die technischen Gegebenheiten lassen sich auf 2 Hauptpunkte reduzieren:

- Geringe Speicherkapazität und Rechenleistung der einzelnen Teilnehmer im Netz, da sie preiswert sein sollen und vor allem selber nur wenig Energie verbrauchen sollen
- Geringe Bandbreite der Netzwerkverbindung, da schmalbandige Funkverbindungen energieeffizienter sind

Der implementierte Algorithmus folgt in wesentlichen Punkten dem ursprünglichen Ant System (AS) von Dorigo. Die Unterschiede sind in der hohen Asynchronität des Gesamtsystems begründet. Im Gegensatz zur Simulation des Problemgraphen auf einem Rechner oder einem Rechnercluster ist der Problemgraph hier auf das physikalisch existierende Netzwerk abgebildet. Da alle Nodes komplett autonome Einheiten sind, arbeiten sie asynchron, so dass das schrittweise Vorgehen der bekannten ACO Varianten der Art

- als Erstes bewege alle Ants
- als Zweites aktualisiere die Pheromone

nicht realisiert werden kann. Vielmehr können auf Grund verschieden schneller Netzwerkverbindungen und Peers manche Ants sich schneller und andere langsamer durch den Problemgraphen/das Netzwerk bewegen. Da jeder Node nur über lokales Wissen verfügt, ist auch kein globales Wissen über den Status der ACO Optimierung vorhanden. Auf Grund der genannten technischen Randbedingungen ist es auch nicht sinnvoll, dieses Wissen irgendwo zu akkumulieren. Eine Akkumulation des Wissens würde zu Lasten der Bandbreite gehen und der entsprechende Peer, der alle Informationen zusammenführt, müsste entsprechend mit Rechenleistung und Speicher ausgestattet sein. Damit sind alle Arten von Aktionen, die einen globalen Zugriff und eine globale Sicht auf das Problem voraussetzen, nicht realisierbar. Aus diesem Grund sind allerdings auch viele erweiterte ACO Varianten, die z.B. extra Pheromone auf Grund der

Rangfolge der Ergebnisse verteilen (EAS, MAX-MIN, AS_{rank}), nicht anwendbar, da dies ein globales Wissen aller zur Zeit gefundenen Lösungen voraussetzt.

Daher wurde bei der Entwicklung einer angepassten ACO Variante von der einfachsten ACO Variante, dem Ant System, ausgegangen. Die hier vorgestellte Variante bezeichnen wir als „Asynchronous Ant System“ (AAS).

4.1. Das Energieproblem

In einem verteiltem Netzwerk aus Energieverbrauchern soll sichergestellt werden, dass ein Gerät nur dann aktiv werden kann, wenn genügend Energie zur Verfügung steht. Das System soll zu jedem Zeitpunkt wissen, wieviel Energie momentan verbraucht wird. Es wird davon ausgegangen, dass jeder Verbraucher ein Node im Netzwerk ist.

Jeder Node kann eine gewisse Leistung P ein- oder abschalten. Jedem Verbraucher ist ein Comfort Impact C_i zugeordnet, der ein Maß dafür ist, wie ungern der Node den Verbraucher abschalten würde. Einem Node wird eine Energiemenge E zugeordnet, diese teilt sich in den aktiven E_a und passiven Energieanteil E_p auf. Aktive Energie ist einem Node zugeordnet und wird derzeit benutzt: Ein Node der ein 25 Watt Leuchtmittel eingeschaltet hält hat einen aktiven Energieanteil von $E_a = 25W$. Schaltet dieser Node jetzt die Lampe aus, so sinkt der aktive Energieanteil dieses Nodes um 25 W während der passive Anteil um diesen Betrag steigt: $E_a = 0W$ und $E_p \leftarrow E_p + 25W$. Der Node wird seinen passiven Energieanteil anderen Nodes die Energie benötigen in Form von Energiezertifikaten gegen eine Comfort Impact von $C_i = 0$ anbieten. Ein Energiezertifikat beschreibt eine Menge von Energie, die von einem Peer des Netzwerkes an einen Anderen transferiert werden kann. Dabei sinkt die dem Sender des Zertifikates zugeordnete Energiemenge $(E_a + E_p)$ um die im Zertifikat angegebene Menge, während der Empfänger des Zertifikates die enthaltene Energiemenge seiner eigenen zuschlagen kann.

Da Nodes einmal arbitrierte, nicht mehr benötigte Energie als passiven Anteil behalten, wird sich die freie Energie über das Netz verteilen, anstatt an bestimmte Quellnodes (Steckdosen o.ä.) gebunden zu sein. Dies sollte die Effizienz des ACO Algorithmus erhöhen und gleichzeitig die Verwaltung vereinfachen, da nicht mehr benötigte Energie nicht an eine zentrale Stelle zurück gegeben wird, was dem dezentralen Gedanken widersprechen würde.

Wenn ein Node N_i das Netz veranlassen möchte die Energiemenge X abzugeben (z.B. weil seinerseits ein größeres Kontingent benötigt wird), schickt er seine Ants los. Beim Start wird in der Ant die einzuholende Energiemenge X gespeichert und der Comfort Impact C_i auf 0 gesetzt. Am Node N_{i+1} angekommen bietet der Peer eine Lösung an, z.B. Abgabe einer Energiemenge von x_i bei einem Comfort Impact von c_i. Die Ant vermerkt, dass der jetzt noch einzusparende Energieanteil $X = X - x_i$ beträgt und der Gesamt Comfort Impact $C = C + C_i$ ist. Dann wandert die Ant (beeinflusst von eventuell schon vorhandenen Pheromonspuren) zum nächsten Node. Fällt bei einem Node X auf 0 so wurde eine Lösung gefunden. Die Ant geht in den Backward Mode und verteilt auf dem Pfad um so mehr Pheromone je geringer C

ist. Auf diese Weise entstehen gute Lösungen, die die angefragte Energiemenge bei gleichzeitig geringem Comfort Impact bereitstellen.

Das dargestellte Verfahren kommt nicht nur zum Zuge wenn das Energiekontingent zu 100% ausgelastet ist und ein neuer Verbraucher eingeschaltet werden soll (dies wird relativ selten der Fall sein), sondern jedes mal wenn ein Verbraucher eingeschaltet wird. Wird ein Verbraucher ausgeschaltet, so behält der zugehörige Node das Energiekontingent solange als passiven Energieanteil, bis es von einem anderen Teilnehmer abgefordert wird. Wird also eine 75W Glühlampe ausgeschaltet, so bleiben die 75W dem Peer weiter zugeordnet, er wird sie aber vorbeikommenden Ants mit einem Comfort Impact von 0 (da die Energie von diesem Peer nicht mehr gebraucht wird) anbieten.

4.2. Modellierung

Aussagen aus 3.2 oder 3.2.1, welche für die ACO allgemein gültig sind und auch hier gelten, wie z.B. „Die Menge der potentiellen Lösungen S ist ein Subset von χ: $S \subseteq \chi$“ werden hier nicht noch einmal aufgeführt.

- S ist die Menge möglicher Lösungen, also eine Menge einer Menge Nodes, die in ihrer Gesamtheit den Energiebedarf einer Ant decken
- Die Bewertungsfunktion $f(s,t)$ weist jeder Lösung $s \in S$ einen Kostenwert zu, der von der Höhe des Gesamt Comfort Impacts einer Lösung abhängig ist.
- Die Menge der Randbedingungen $\Omega(t)$: Ein Node kann nicht mehr Energie anbieten, als ihm zur Verfügung steht.

Damit enthält die Darstellung des Problems folgende Elemente:

- Eine endliche Menge $C = \{c_1, c_2, ..., c_{N_C}\}$ von Komponenten. Hierbei handelt es sich um die physikalischen Nodes im Netzwerk. Jeder Embedded Node an einem Verbraucher ist aus Sicht der ACO eine Komponente.
- Die möglichen States des Problems sind definiert als Sequenzen endlicher Länge aus den Elementen von C: $x =< c_i, c_j, ...c_h >$. Die Menge aller möglichen States sei χ. Es ist wichtig, dass hier nur Folgen enthalten sein können, die auf Grund der Netzwerktopologie auch in dieser Reihenfolge erreicht werden können. Hier kann es Einschränkungen geben, da die Netzwerkverbindungen zwischen den Nodes u.U. keinen voll transitiven Graphen ergeben.

4.2.1. Verhalten der Ants

Jede Ant hat folgende Eigenschaften:

- Sie wandern durch den Konstruktionsgraph $G_C(C, L)$ auf der Suche nach optimalen Lösungen $s^* \in S^*$. Da in diesem Fall der Konstruktionsgraph das reale Netzwerk ist, hängt die Struktur des Graphen von der Netzwerktopologie ab. Die Bewegungsfreiheit der Ants ist eingeschränkt durch die Tatsache, dass die Netzwerkstruktur nicht unbedingt voll transitiv ist. Daraus folgt, dass die Ants umso flexibler agieren können je enger vermascht sich das Netzwerk darstellt.

- Eine Ameise hat ein Gedächtnis M^k, das benutzt werden kann um Informationen über den Weg zu speichern, den die Ameise bis jetzt verfolgt hat. Hier speichern die Ants die Adresse jeden Nodes der bislang besucht wurde. Zudem ist für jeden Node hinterlegt, wieviel Energie er freigeben kann und welchen Comfort Impact das nach sich zöge.

- Eine Ameise hat einen Startzustand x_s^k sowie eine oder mehrere Abbruchbedingungen e^k. Im Startzustand ist eine Ant mit ihrer Problembeschreibung (Menge der zu akquirierenden Energie, Priorität der Anfrage) ausgestattet. Im Gedächtnis ist die Energie enthalten, über die der Urpsrungspeer selber verfügt. Es gibt folgende Abbruchbedingungen:

  - das Problem ist gelöst, die Ant hat genügend Energieangebote gesammelt
  - Der TTL Wert der Ant erreicht 0. (Das TTL Feld einer Ant wird nach jedem Schritt um 1 dekrementiert, damit eine Ant nicht für unbestimmte Zeit vergeblich nach einer Lösung sucht)
  - das Problem ist nicht gelöst und es gibt keine Nachbar Nodes die besucht werden können

- Eine Ant wählt ihren nächsten Schritt mit Hilfe einer probabilistischen Entscheidungsregel. Diese Regel ist ist eine Funktion die von den lokal verfügbaren Pheromonspuren abhängt. Heuristische Informationen werden nicht herangezogen.

Das AAS nutzt derzeit keine heuristischen Informationen. Schon bei der allgemeinen formalen Beschreibung wurde erwähnt, dass die Ants gleichzeitig und unabhängig voneinander arbeiten. Dies gilt in ganz besonderem Maße für die hier vorgestellte Implementierung, da die Ants von Knoten zu Knoten wandern und dementsprechend die Berechnungen des Algorithmus zu jedem Zeitpunkt gleichzeitig auf verschiedenen Rechnersystemen statt finden. Im Extremfall kann es sich hier um verschiedene Hard- und Softwareplattformen handeln, die gemeinsam eine ACO durchführen. Dadurch bleibt es nicht aus, dass sich auf Grund unterschiedlicher Leistung und Auslastung der Nodes die Ants mit unterschiedlichen Geschwindigkeiten durch das Netz bewegen. Dadurch entsteht eine extrem hohe Asynchronität, die es so in den klassischen Implementierungen des ACO, die meist auf einem einzelnen Rechner ausgeführt werden, nicht gibt. Auf die Konsequenzen dieser Asynchronität wird im Kapitel Implementierung näher eingegangen.

4.3. Auswahl des nächsten Peers

Wenn eine Ant sich an einem Peer i befindet, muss eine Entscheidung getroffen werden, welcher Peer j als nächstes besucht wird. Dies entspricht der Auswahl der nächsten Stadt beim TSP. Die Wahrscheinlichkeit, dass eine Ant k nach dem Peer i den Peer j wählt errechnet sich wie folgt:

$$p_{ij}^k = \frac{\alpha + \tau_{ij}}{\sum_{l \in N_i^k} \tau_{il} + \left|N_i^k\right| \cdot \alpha} \tag{4.1}$$

Die Formel ist von der AS Formel 3.2 zur Auswahl der nächsten Komponente abgeleitet. Heuristische Informationen gibt es in dem hier vorgestellten System nicht, da ein Peer nichts über den Zustand der anderen Peers weiß; jeder Peer verfügt nur über lokales Wissen. Wie bei 3.2 muss $j \in N_i^k$ sein. N_i^k ist die Nachbarschaft von i, d.h. eine Menge von Peers, die die Ant als nächstes besuchen kann. $\left|N_i^k\right|$ ist die Anzahl der Elemente in N_i^k. N_i^k ist damit abhängig von der Netzwerktopologie, bzw. hier von den Informationen, die vom L3 Discovery bislang über das Netzwerk gesammelt wurden. Bei α handelt es sich um einen Parameter, der jedem Nachbarpeer eine gewisse Grundwahrscheinlichkeit gibt. Dies ist nötig, da es auf Grund des verteilten technischen Systems ja nicht möglich ist die Pheromonspuren vorab zu initialisieren. Damit sind zu Beginn alle $\tau_{ij} = 0$ und damit auch die Wahrscheinlichkeit, dass der entsprechende Peer j gewählt wird. Das Verhältnis zwischen α und der Menge der Pheromone, die die einzelnen Ants verteilen, bestimmt ob die Suche eher explorativ verläuft (α im Verhältnis groß), oder ob sie sich eher auf die bislang gefundenen Touren konzentriert (α im Verhältnis klein). Der Peer, der als nächstes besucht werden soll kann mittels eines Zufallsgenerators aus den Wahrscheinlichkeiten ermittelt werden wie beim Ant System in Abschnitt 3.3.1 dargestellt.

4.4. Pheromonverteilung

Im Gegensatz zum AS werden beim AAS nicht die Pheromone für alle Ants auf allen Kanten gleichzeitig aktualisiert, da der Algorithmus im verteilten System nicht sequenziell abläuft, und es somit keinen definierten Zeitpunkt zum Aktualisieren der Pheromonspuren gibt. Die Pheromonspuren werden von jeder Ant die sich im Backward Mode befindet, einzeln aktualisiert. Da nur Ants, die eine mögliche Lösung konstruiert haben, in den Backward Mode übergehen, verteilen alle Backward Ants Pheromone.

Eine Backward Ant die an Peer i ankommt, hinterlegt auf diesem Peer eine Pheromonspur, die anzeigt, wie ertragreich es ist, von diesem Peer aus als nächstes Peer j aufzusuchen. Dabei ist j der Peer, den diese Ant im Forward Mode nach Peer i besucht hat. Als erstes wird die hinzuzufügende Pheromonmenge berechnet

$$\Delta\tau_{ij} = E_{pj} + E_{aj} - C_j + \beta \cdot Q \tag{4.2}$$

Wie bei dem in 3.3 vorgestellten ACO Varianten, soll die Formel für die Pheromonverteilung eine Wertung vornehmen, d.h. mehr Pheromone auf den Kanten guter Lösungen verteilt. Beim TSP ist das einfach, da nur die Streckenlänge relevant ist: Kürzere Strecken sind besser. Bei der Energieverteilung gibt es mehrere Faktoren, die eine gute bzw. schlechte Lösung ausmachen, welche alle einbezogen werden sollten. Der Term bezieht zum einen die lokalen Informationen ein, wie gut der nächste Peer j ist, d.h. wieviel Energie er angeboten hat und welchen Comfort Impact das nach sich zöge. Zudem wird die Gesamtqualität Q der Lösung betrachtet, von der der Peer j ja nur ein Teil ist. Über den Parameter β kann beeinflusst werden, ob die Pheromonverteilung sich eher nach der Gesamtqualität der Lösung richtet, oder ob der Beitrag des Peers, auf den diese Spur zeigt stärker gewichtet werden soll. Es kann sein, dass $\Delta\tau_{ij}$ sehr klein oder sogar unter 0 ist. Wenn $\Delta\tau_{ij}$ unter dem Schwellwert γ liegt, dann wird die entsprechende Pheromonspur τ_{ij} um den Wert γ inkrementiert, ansonsten um $\Delta\tau_{ij}$

$$\tau_{ij} \leftarrow \begin{cases} \tau_{ij} + \Delta\tau_{ij} & , \; wenn \; \Delta\tau_{ij} \geq \gamma \\ \tau_{ij} + \gamma & , \; wenn \; \Delta\tau_{ij} < \gamma \end{cases} \tag{4.3}$$

Die Regelung mit dem Schwellwert γ ist nötig, damit jede Kante, die auf dem Weg einer brauchbaren Lösung liegt, auch Pheromone abbekommt. τ_{ij} wird immer dann zu klein, wenn der Comfort Impact einer Komponente im Verhältnis zur abgegebenen Energie sehr hoch ist. Dennoch handelt es sich um eine gültige Lösung, die über die Pheromonspuren von anderen Ants erkennbar sein sollte.

4.5. Pheromon Verdampfung

Auch beim AAS werden die Pheromonspuren mit der Zeit wieder abgebaut, so dass der Algorithmus mit der Zeit weniger optimale Lösungen wieder vergessen kann. Während dies bei den in Abschnitt 3.3 vorgestellten klassischen ACO Varianten meist passiert nachdem alle Ants ihre Touren komplettiert haben, ist dies im verteilten Netz nicht möglich. Da die Peers nur über lokales Wissen verfügen, ist dem einzelnen Peer der Zeitpunkt zu dem alle Ants bewegt worden sind, unbekannt. Daher greift AAS wieder auf das natürliche Vorbild zurück: Die Pheromonspuren verdampfen zeitabhängig, d.h. jeder Peer ist dafür verantwortlich nach einer festgelegten Zeit seine gespeicherten Pheromonspuren nach einer vorher definierten Formel zu verringern. Damit ist die Pheromonverdampfung zwischen den Peers zwar asynchron, d.h. es kann nicht garantiert werden, dass alle Peers zum exakt gleiche Zeitpunkt oder in den exakt gleichen Intervallen ihre Pheromone verdampfen, aber durch Definition des Intervalls und der Verdampfungsformel sollte im gesamten Netz ein näherungsweise gleichförmiger Pheromonabbau stattfinden.

Da auf den Embedded Peers zeitaufwändige Floating Point Berechnungen vermieden werden sollten (viele Mikrocontroller haben keine FPU), ist eine Formel wie 3.6 nicht ideal. Statt dessen wird das Verhalten einer derartigen Formel diskret mit einem einfachen Stufenverfahren angenähert:

$$\tau_{ij} \leftarrow \begin{cases} \tau_{ij} - \Delta\tau_1 & , wenn\ \tau_{ij} \geq T_1 \\ \tau_{ij} - \Delta\tau_2 & , wenn\ T_1 > \tau_{ij} \geq T_2 \\ \tau_{ij} - \Delta\tau_3 & , wenn\ T_2 > \tau_{ij} \geq \Delta\tau_3 \\ 0 & , ansonsten \end{cases} \tag{4.4}$$

Der Wert einer Pheromonspur wird um einen konstanten Wert $\Delta\tau$ verringert. Je größer der Wert von τ_{ij} umso größer $\Delta\tau$. Der Wert einer Pheromonspur kann nicht negativ werden, das Minimum sind keine Pheromone $\tau_{ij} = 0$. Über die Parameter T_x (Threshold) kann festgelegt werden, ab welcher Pheromonmenge τ_{ij}, wieviel Pheromone $\Delta\tau$ abgezogen werden. Es muss gelten $\Delta\tau_1 > \Delta\tau_2 > \Delta\tau_3$ und $T_1 > T_2 > \Delta\tau_3$. Bei Bedarf kann dass Verfahren natürlich mit weiteren Abstufungen realisiert werden.

4.6. Beispiel

In diesem Abschnitt soll ein kurzes Beispiel zum Ablauf der AAS gegeben werden. In Abbildung 4.1 ist der Zustand des Netzwerkes zu verschiedenen Zeitpunkten t abgebildet. Zum Zeitpunkt $t = 0$ befindet sich das System in einem statischen Zustand. Es gibt 3 Verbraucher A,B und C von denen lediglich der Verbraucher C aktiv ist und eine Energie von 20 W konsumiert.

Zum Zeitpunkt $t = 1$ ist die Anforderung vom Benutzer gekommen, den Peer A einzuschalten. Das Beispiel geht davon aus, dass Peer A eine Energie von 100 W benötigt. Peer A verfügt selber nur über 10 W Energie. Diesen Anteil bucht A von seinem Konto passiver Energie auf sein Konto aktiver Energie um. Es fehlen noch 90 W, daher startet Peer A eine ACO. Die Ant vermerkt die Menge der zu akquirierenden Energie, in diesem Fall 90 W, sowie die Priorität dieser Anfrage. Sie entspricht hier der Gerätepriorität von A, Priority=100. Der erste Teil der Lösung ist die von A selber zur Verfügung gestellte Energie. Die Ant muss jetzt einen der Nachbarpeers von A aufsuchen. Im Beispiel kommen nur die Peers B oder C in Frage. Die Wahrscheinlichkeit welcher Peer genommen wird, errechnet sich aus Formel 4.1. Mit $\alpha = 2$ ergeben sich die Wahrscheinlichkeiten Peer B oder C aufzusuchen wie folgt:

$$p_{AB} = p_{AC} = \frac{\alpha + \tau_{ij}}{\sum_{l \in N_i^k} \tau_{il} + |N_i^k| \cdot \alpha} = \frac{2 + 0}{0 + 0 + 2 \cdot 2} = 0,5 \tag{4.5}$$

Da zu Beginn der Optimierung die Pheromonspuren für beide Peers $\tau = 0$ sind, ist die Wahrscheinlichkeit für jeden Peer exakt gleich. Die reale Implementierung führt allerdings für den ersten Schritt der Ants keine Wahrscheinlichkeitsbewertung durch, sondern schickt statt dessen eine Ant an jeden Nachbarn. Erst bei den weiteren Schritten kommen die Wahrscheinlichkeiten zum Zuge.

Wir gehen davon aus, dass die Ant zu Peer B weiter geschickt wurde. Dieser Peer ist nicht aktiv, ihm ist aber noch passive Energie zugeordnet. Diese kann er zum Zeitpunkt $t = 2$ an die Ant abgeben. Da er nur über 80W verfügt, genügt das nicht den Energiebedarf der Ant zu decken. Der Anteil von 80 W wird der Ant angeboten. Der Comfort Impact hierfür ist 0, da sich der Peer B ja nicht abschalten muss um

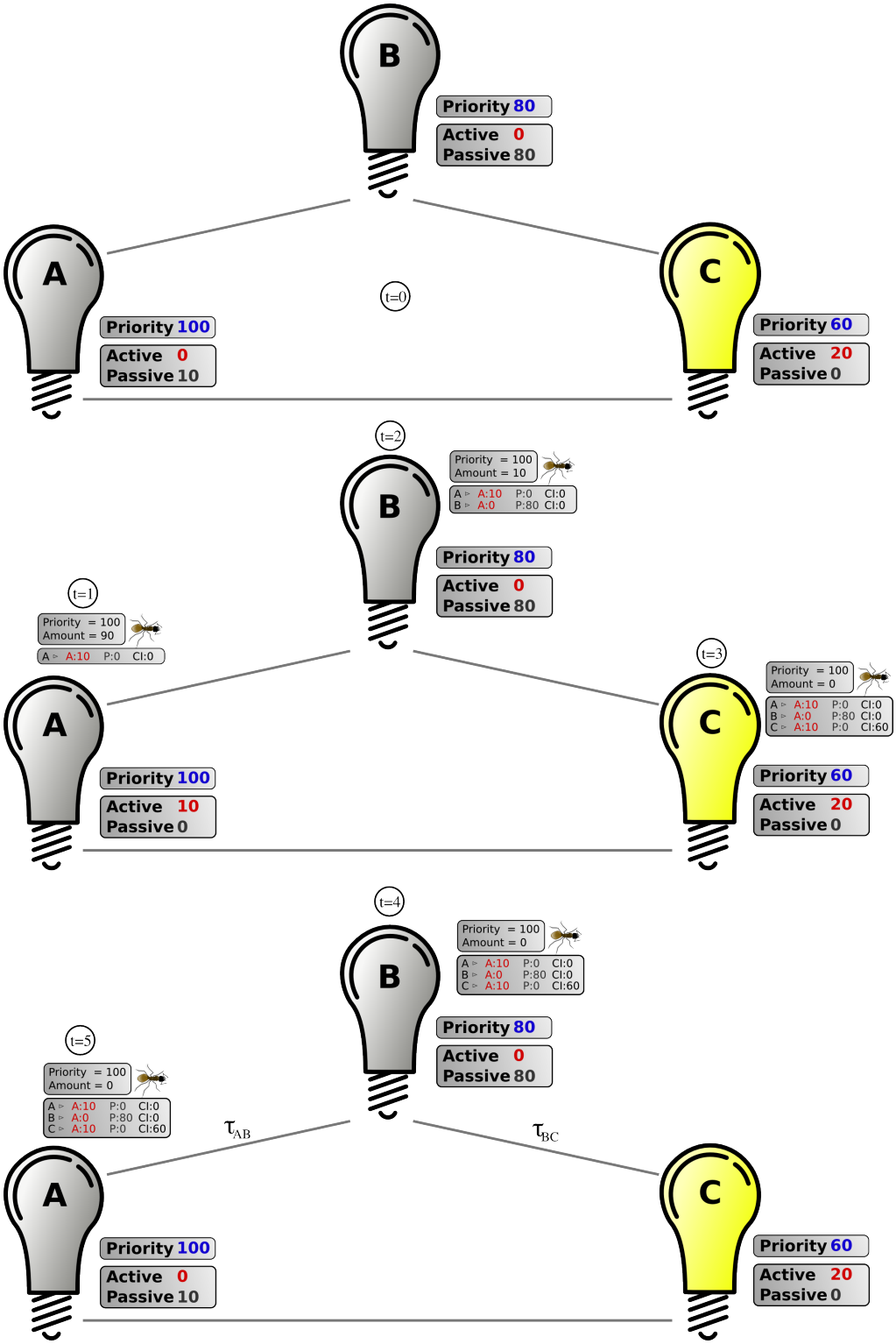


Abbildung 4.1.: AAS Beispielszenario

die Energie zur Verfügung zu stellen. Es fehlen immer noch 10 W, also muss die Ant weiter wandern. Es würden jetzt wieder die Wahrscheinlichkeiten ausgewertet, um das nächste Ziel für die Ant zu finden. In diesem Fall ist der Peer C die einzige Möglichkeit, da A und B ja bereits besucht wurden.

Zum Zeitpunkt $t = 3$ ist die Ant an Peer C angekommen. Dieser Peer hat keine passive Energie abzugeben. Er hat aber einen aktiven Energieanteil von 20 W, was ausreicht da die Ant nur noch 10 W benötigt. Peer C bietet 10 Watt an, da seine Priorität, geringer ist, als die der Ant. Wäre die Priorität von C größer 100 würde er keine Energie anbieten. Weil sich der Peer C abschalten müsste, wenn er 10 Watt abgäbe, bekommt dieser Teil der Lösung einen Comfort Impact größer 0. Peer C setzt $C = 60$, was seiner Priorität entspricht.

Weil damit eine Lösung gefunden ist (Amount=0), wird die Ant von Peer C in den Backward Mode umgeschaltet. Vorher wird die Gesamtqualität der Lösung berechnet. Wie dies geschieht, ist von der Implementierung des konkreten Problems abhängig. In der Implementierung berechnet sich die Qualität zu $Q = 255 - C_{Gesamt} = 255 - 60 = 195$. Dabei ist C_{Gesamt} die Summe der Comfort Impacts aller Teillösungen.

Die Ant verfolgt ihren Weg zurück, d.h. als erstes wandert sie zu Peer B. Auf Peer B wird die Pheromonspur τ_{BC} für die Kante BC hinterlegt. Diese Spur sagt etwas darüber aus, wie ergiebig es ist, Peer C aufzusuchen. Nach Formel 4.2 ergibt sich mit $\beta = 0,1$ ein Pheromondelta von

$$\Delta\tau_{BC} = E_{pC} + E_{aC} - C_C + \beta \cdot Q = 0 + 10 - 60 + 19,5 = -30,5 \tag{4.6}$$

Da $\Delta\tau_{BC} < 0$ würde das zu einem Abbau von Pheromonen führen, was nicht sinnvoll ist, da es sich bei Peer C um einen Bestandteil einer gültigen Lösung handelt. Jetzt greift Fall 2 aus Formel 4.3, d.h. die Pheromone τ_{BC} werden um γ inkrementiert. Sei $\gamma = 10$, so ist die Menge der Pheromone

$$\tau_{BC} \quad = \quad 0 + 10 = 10 \tag{4.7}$$

Von Peer B wandert die Ant zurück zu Peer A. Auf Peer A wird die Pheromonspur τ_{AB} für die Kante AB hinterlegt. Diese Spur sagt etwas darüber aus, wie ergiebig es ist, Peer A aufzusuchen. Analog zu τ_{BC} berechnet sich τ_{AB}:

$$\Delta\tau_{AB} = E_{pB} + E_{aB} - C_B + \beta \cdot Q = 80 + 0 - 0 + 19,5 = 99,5 \tag{4.8}$$

Da $\Delta\tau_{AB} > 0$ ergibt sich τ_{AB} direkt zu.

$$\tau_{AB} = 0 + 99,5 = 99,5 \tag{4.9}$$

Peer A hat jetzt eine Lösung bekommen, und kann entscheiden, was zu tun ist. Ist Peer A bereits eine bessere Lösung bekannt (Q größer), dann würde die Lösung dieser Ant verworfen werden. Handelt es sich um die bislang beste Lösung, würde sie gespeichert und später unter Umständen umgesetzt, d.h. die Energieanteile würden transferiert werden. Da im Beispiel der Beginn er Optimierung gezeigt wurde, wird

Peer A die Lösung speichern und noch weitere Generationen von Ants aussenden um eventuell eine bessere Lösung zu finden.

4.7. Erweiterungen & Performance

Das hier vorgestellte AAS wurde in dieser Arbeit nicht mit anderen ACO Varianten verglichen. Um diese Messungen durchführen zu können, müsste AAS auf das TSP Problem angepasst werden, oder es müssten andere ACO Varianten für das Energieproblem adaptiert werden. Dann wäre der Vergleich mit klassischen ACO Varianten aber nur per Simulation möglich, da wie in diesem Kapitel gezeigt, die technischen Randbedingungen dafür sorgen, dass die herkömmlichen ACO Varianten auf dem System nicht implementiert werden können und damit ungeeignet sind das Problem zu lösen. Im Abschnitt 3.3 wird gezeigt, dass AS die ACO Variante mit der relativ schlechtesten Performance ist. Da das AAS auf dem AS basiert und auf Grund der technischen Randbedingungen noch einige weitere Kompromisse eingehen muss, ist zu vermuten das AAS, wenn es unter gleichen Voraussetzungen auf das TSP Problem angewendet würde, eine Performance erreichen würde die noch unter der von AS liegt. Es geht aber auch nicht darum mit AAS eine neue hochperformante ACO Variante einzuführen, sondern eine im technischen Kontext überhaupt applizierbare Variante. Dies war mit keiner der klassischen ACO Varianten möglich. Das hier vorgestellte AAS ist sowohl bei der Simulation kleinerer Netzwerke ($<$10 Peers) als auch in der Implementierung auf den Embedded Controllern in der Lage sehr gute Lösungen zu finden.

Für die Zukunft sind einige AAS Erweiterungen denkbar, die die Performance erhöhen könnten: Es kann untersucht werden, ob es Sinn macht, wenn die Ants im Forward Mode wie bei der ACS beim Übergang von einem Peer zum anderen Pheromone abbauen. Auch könnten die Ergebnisse mehrerer vorangegangener Optimierungen als heuristische Informationen herangezogen werden, wo in der Vergangenheit gute Lösungen lagen. Ob, und in welchem Maße solche Optimierungen die Performance verbessern können müsste in umfangreicheren Simulationen untersucht werden.

5. Implementierung

In diesem Abschnitt soll die Implementierung des AAS vorgestellt werden. Es wird die Java Simulation sowie die Implementierung auf den Embedded Peers vorgestellt.

Zunächst wird die Umgebung beschrieben, in der die Implementierung getestet wurde. Der Abschnitt Datenstrukturen beschreibt, wie für das AAS relevante Datenstrukturen wie die Ants oder die Pheromone abgebildet werden. Danach wird die entwickelte Simulation vorgestellt. Die letzten beiden Kapitel befassen sich mit der konkreten Implementierung auf dem Embedded System: Die Funktionsweise und Protokollformate der L3 Services werden erklärt. Im Abschnitt Ant Verarbeitung wird gezeigt, wie das AAS konkret umgesetzt wurde.

5.1. Testumgebung

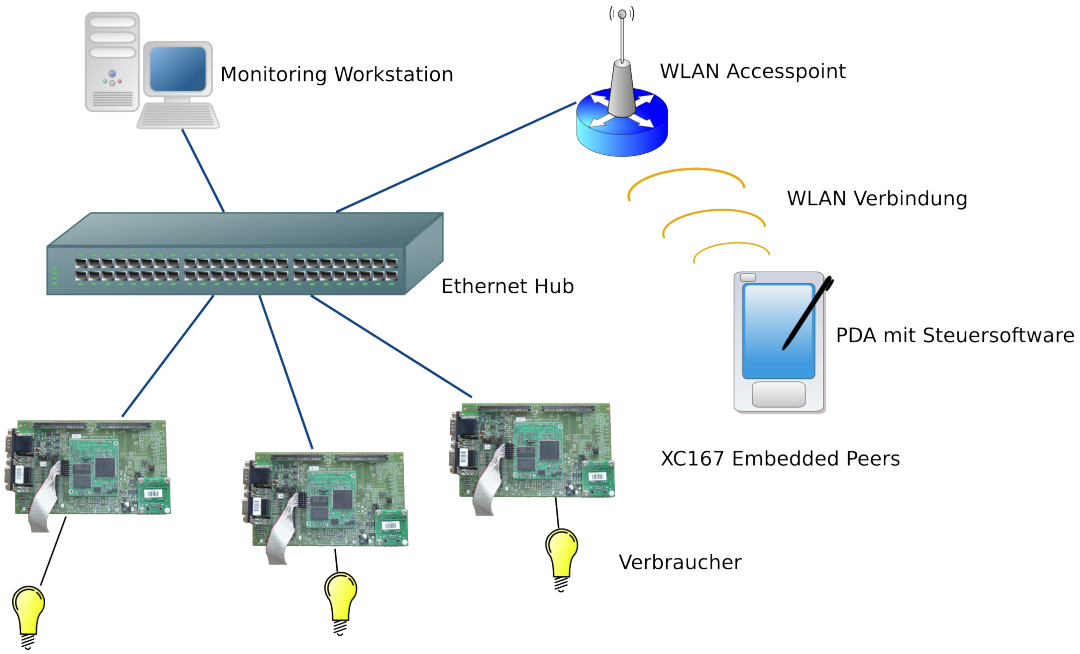


Abbildung 5.1.: Systemaufbau

Abbildung 5.1 zeigt den Aufbau des Gesamtsystems mit allen Komponenten. Jeder der XC167 Embedded Peers kontrolliert einen elektrischen Verbraucher. In der Zeichnung ist dies durch eine Glühbirne symbolisiert. Die XC167 Boards sind unter-

einander über einen Hub per Ethernet vernetzt. Theoretisch kann auch ein Switch benutzt werden, aber ein Hub hat den Vorteil, dass die Monitoring Workstation, welche ebenfalls an den Hub angeschlossen ist, den gesamten Netzwerkverkehr zwischen den XC167ern empfangen und analysieren kann. Es handelt sich bei der Monitoring Workstation um einen normalen PC auf dem der Netzwerk Analyser WireShark (siehe 2.2.3) installiert ist. Damit können in der Entwicklungsphase Implementierungsfehler aufgedeckt und das Verhalten des Gesamtsystems beobachtet werden. Der Accesspoint sorgt für einen drahtlosen Zugangspunkt zum Ethernet Netzwerk. Der PDA verbindet sich per WLAN mit dem Accesspoint und hat damit ebenfalls Zugriff auf das Netz der XC167 Peers. Vom PDA aus können die Embedded Peers angewiesen werden ihre Verbraucher ein- oder auszuschalten. Die PDA Applikation visualisiert hierbei die Energiezustände der einzelnen Embedded Peers.

Die Simulationssoftware ist auf der Übersicht nicht zu sehen, da sie unabhängig vom Netz arbeitet und nur zur Entwicklung des Algorithmus eingesetzt wird.

5.2. Datenstrukturen

Bei der Modellierung der Datenstrukturen wurde darauf geachtet, diese möglichst generisch zu halten. Insbesondere sind ACO generische Informationen (Richtung der ANT, Pheromonspuren) streng von den problemspezifischen Daten getrennt (konkrete Beschreibung des Problems und der Lösung). Die Beschreibung der Datenstrukturen in diesem Kapitel orientiert sich an der Embedded C Version. Der Algorithmus und die Datenstrukturen sind in der Java Simulation jedoch weitestgehend übereinstimmend.

Zwischen den Peers müssen zur Durchführung der Optimierung immer gewisse Informationen ausgetauscht werden. Die Ants selber enthalten generische Informationen, wie z.B. die Qualität der gefundenen Lösung oder ob es sich um einen Backward oder Forward Ant handelt. Diese Informationen werden für jede ACO Optimierung gebraucht und sind immer gleich. Dann gibt es noch Informationen, die problemspezifisch sind: Das ist zum einen die Beschreibung des Problems selber und zum anderen die (Teil)lösungen für dieses Problem.

Auf den Peers selber müssen die Pheromonspuren verwaltet werden, welche bei einem Optimierungslauf von den Ants erzeugt werden.

Folgende Datentypen werden in den in diesem Abschnitt beschriebenen Datenstrukturen verwendet:

Datentyp	
boolean	Eine boolsche Variable. Kann die Werte TRUE (1) oder FALSE (0) annehmen.
uint8	Vorzeichenloser 8 Bit Wert
uint16	Vorzeichenloser 16 Bit Wert
l3_uuid	Eine L3 Netzwerkadresse, 64 Bit Länge

5.2.1. ANT

Es handelt sich hier um eine Datenstruktur für eine generische Ant. In der Implementierung sind die Ants Datenpakete, die über das L3 Netz übertragen werden. Diese Rahmendatenstruktur ist nicht problemspezifisch, sondern spezifiziert nur die generischen Teile die für jede Art von ACO Optimierung benötigt werden. Die Problembeschreibung sowie die Teillösung werden hinter die Ant Struktur angeordnet, so dass verschiedene ACO Optimierungen hier sehr flexibel eigene Daten einhängen können.

```
ANT {
    boolean direction;
    uint16 service;
    uint16 aco_id;
    l3_uuid originator;
    uint16 solution_quality;
    uint8  solutionPosition
    uint8  solutionLength;
    uint8  ttl;
}
```

Feld	Bedeutung
direction	Zeigt an, ob eine ANT sich im Forward Modus (1) oder Backward Modus (2) befindet
service	Dient dazu, verschiedene ACO Optimierungen zu benutzen. Jede Applikation die ACO Optimierungen durchführt, muss einen anderen Service Wert benutzen. Unterschiedliche Service Nummern implizieren unterschiedliche Datenstrukturen für die Problembeschreibung und die SolutionElements. Der in dieser Arbeit implementierte Service zu Energieoptimierung hat die Service Nr. 0xEE
aco_id	Das Feld aco_id ermöglicht es, dass die Pheromonspuren mehrere Instanzen desselben ACO Services auseinander gehalten werden können wenn sie gleichzeitig ablaufen (s.u.)
originator	Die L3 UUID des Peers, der diese Optimierung angestoßen hat und an den Lösungen interessiert ist.
solutionQuality	Qualität der Lösung, die diese Ant erzeugt hat. 255 entspricht der besten Lösung, 0 der schlechtest möglichen. Der Peer, dessen SolutionElement die Lösung komplettiert berechnet die solutionQuality unter Einbeziehung aller SolutionElemente bevor er die Ant in den Backward Modus schaltet

Feld	Bedeutung
solutionPosition	Zeigt bei der Backward Ant an, an welcher Position die Ant sich auf Ihrer Rückreise gerade befindet. Dies ist hauptsächlich aus Performance Gründen wichtig, damit Peers die eine Backward empfangen nicht erstmal aufwendig sich selber in den SolutionElements finden müssen, um das nächste Ziel für die Ant zu finden.
ttl	TimeToLive Zähler für diese Ant. Wird im Forward Modus von jedem Peer dekrementiert. Erreicht eine Ant ttl=0 ohne eine Lösung gefunden zu haben, stirbt sie

Damit eine Applikation mehrere Optimierungen zeitgleich vornehmen kann (also wenn z.B zwei Verbraucher gleichzeitig eingeschaltet werden sollen), muss für jeden gestarteten ACO Vorgang eine möglichst eindeutige *aco_id* Nummer gewählt werden. Der ACO Service verwaltet Pheromone für verschiedene *aco_id* Werte getrennt, so dass z.B. mehrere Peers gleichzeitig eine Energieoptimierung durchführen können, ohne dass die Algorithmen durch unpassende Pheromonspuren unerwünscht beeinflusst werden.

Der konkrete Wert der *solutionQuality* ist zwar insoweit problemspezifisch, als dass nur problemspezifischer Code in der Lage ist eine Lösung zu bewerten, aber der Vergleich verschiedener Lösungen eines Problems anhand eines normalisierten Qualitätsindikators kann wieder generisch erfolgen.

5.2.2. Problem Repräsentation

Eine Ant muss immer eine Beschreibung des Problems sowie die gerade konstruierte Lösung enthalten. Da diese Informationen anwendungsspezifisch sind, sind sie nicht integraler Bestandteil der Ant Datenstruktur. Die Problembeschreibung für die Energieakquise ist sehr einfach:

```
ENERGY_PROBLEM {
    uint16 amount;
    uint8 priority;
    boolean eatActive;
}
```

Feld	Bedeutung
amount	Menge der Energie die akquiriert werden soll in Watt
priority	Priorität des Peers, der die Energie benötigt. Dieses Feld ist wichtig um zu entscheiden, ob ein Peer abgeschaltet werden kann um Energie für einen anderen Peer bereit zu stellen

Feld	Bedeutung
eatActive	Nur wenn dieses Bit gesetzt ist, werden auch andere Peers abgeschaltet um die nötige Energie zu akquirieren. Alle Ants starten immer mit eatActive=0. Unter bestimmten Bedingungen wird das Bit auf 1 gesetzt (s.u.)

Das *priority* Feld ist sehr wichtig um die Funktionsweise und die Entscheidungen, die die Optimierung trifft zu beeinflussen. Das *priority* Feld sollte zum einen in einer sinnvollen Weise vorkonfiguriert sein, so dass z.B. die Heizung immer eine höhere Priorität als das Radio hat, zum anderen ist das *priority* Feld aber auch eine Möglichkeit für den Benutzer, das System nach seinen Wünschen anzupassen, indem z.B. von Benutzerseite aus die Priorität für das Radio über die von der Deckenbeleuchtung gesetzt werden kann.

Zum Beispiel kann die Vorgabe sein, dass die Heizung die Priorität $p_{Heizung} = 100$ hat, die Beleuchtung $p_{Beleuchtug} = 20$ und das Radio $p_{Radio} = 15$. Das System sollte es dem Benutzer ermöglichen p_{Radio} auf 25 zu erhöhen, um seine Präferenz für das Radio gegenüber der Beleuchtung auszudrücken, aber es sollte nicht möglich sein $p_{Radio} > p_{heizung}$ zu setzen. In der vorliegenden Implementierung werden die Prioritäten fest vorgegeben und dem Benutzer die Möglichkeit eingeräumt alle Prioritäten um +/- 10 Punkte zu variieren.

Das *eatActive* Feld sorgt dafür, dass jede Ant zu Beginn (*eatActive*=false) versucht nur passive, d.h. nicht genutzte Energie einzusammeln. Es wird also versucht die Energie aus dem System zu arbitrieren ohne einen Verbraucher abschalten zu müssen. Es gibt zwei Situationen in denen eatActive von einem Peer auf true gesetzt wird:

- ttl=2 und es wurde noch keine Lösung gefunden. Es wird davon ausgegangen, dass eine Ant die nach hinreichend vielen Schritten noch keine Lösung aus rein passiven Anteilen gefunden hat, dies auch nicht mehr schaffen wird
- Wenn von dem aktuellen Peer kein Peer mehr erreicht werden kann, den die Ant bislang noch nicht besucht hat, und noch keine vollständige Lösung konstruiert wurde. In diesem Fall wird eatActive=true gesetzt und bereits besuchte Peers (von denen bislang nur passive Energieanteile gesammelt wurden) können noch einmal besucht werden

Wenn eatActive gesetzt ist, wird aktive Energie nur von denjenigen Peers angeboten deren Priorität niedriger als die im *priority* Feld der Ant angegebene ist. Niederpriore Peers können also keine höher prioren Verbraucher abschalten.

5.2.3. Lösungen

Die (Teil)lösung einer Ant wird als ein Array von SolutionElement Objekten hinter die Ant Struktur (5.2.1) und der Problembeschreibung (5.2.2) an das zu übertragende Datenpaket gehängt. Jeder Peer, der eine Ant empfängt, fügt ein SolutionElement von folgendem Format hinzu:

```
ENERGY_SOLUTION_ELEMENT {
    l3_uuid peer;
    uint16 passive;
    uint16 active;
    uint8 wasAlreadyHere;
}
```

Feld	Bedeutung
l3_uuid	UUID des Peers, der diesem Teil der Lösung angeboten hat
passive	Anteil passiver Energie in Watt, den dieser Peer an den Anfrager abgeben kann
active	Anteil aktiver Energie in Watt, den dieser Peer an den Anfrager abgeben kann
wasAlreadyHere	Wenn die ANT den Peer mit der UUID l3_uuid bereits einmal mit eatActive=true besucht hat, wird wasAlreadyHere auf einen Wert $\neq 0$ gesetzt (1 für das SolutionElement des ersten Besuches, 2 für das Zweite) um sicher zu stellen, dass dieser Peer kein weiteres Mal von dieser Ant besucht wird.

Das *wasAlreadyHere* Element ist wichtig, damit eine Ant einen Peer nicht öfter als sinnvoll besucht. Bei der Optimierung besucht eine Ant einen Peer maximal 2 mal: Beim ersten mal mit *Problem.eatActive*=false, wird nur passive Energie angefragt. In einem späteren Optimierungsschritt, wenn *Problem.eatActive*=true ist, kann der Peer noch einmal besucht werden um u.U auch aktive Energieanteile zu erhalten. Sobald ein Peer aktive Energieanteile angeboten hat, werden die *wasAlreadyHere* Werte in seinen SolutionElements auf einen Wert $\neq 0$ gesetzt, da ein weiterer Besuch dieses Peers sinnlos wäre, wenn er bereits seine aktive Energie angeboten hat.

5.2.4. Pheromon Storage

Jeder Peer muss Pheromonspuren, die von den Ants hinterlegt werden, speichern. Dazu verfügt jeder Peer über ein Array von Pheromon Objekten:

```
PHEROMONE {
    uint16  aco_id;
    uint8   amount;
    l3_uuid uuid;
}
```

Feld	Bedeutung
aco_id	aco_id der Optimierung, zu der diese Pheromonspur gehört. Ermöglicht das zeitgleiche Ablaufen verschiedener Optimierungen. Siehe auch Beschreibung der Ant in 5.2.1
amount	Menge der hinterlegten Pheromone. Je höher der Wert um so wahrscheinlicher ist es für Ants den Peer, der im Feld uuid angegeben ist zu besuchen
uuid	UUID des Peers, auf den sich diese Spur bezieht

Das Pheromon Array wird als Ringpuffer verwaltet, d.h. die neuesten Pheromoneinträge von Ants überschreiben die jeweils ältesten. Wenn eine Ant für einen bestimmten Peer eine Pheromonspur hinterlegen will, und es befindet sich bereits eine Spur für denselben Peer mit derselben *aco_id* im Puffer, so wird der entsprechende Eintrag aktualisiert.

5.3. Verhalten bei Fehlern

Es ist absolut notwendig, dass das Gesamtsystem im Gebäude sehr fehlertolerant ist. Das bedeutet insbesondere, dass sich Fehler einzelner Peers nicht auf das gesamte Netz auswirken dürfen. Fehlerhafte Peers führen sicher zu einer Degradation der Servicequalität, sie dürfen jedoch nicht zu einer Service Verweigerung führen.

Fehlertoleranz gegenüber nicht korrekt arbeitenden Teilnehmern ist eine besondere Herausforderung, da das L3 Netz in besonderem Maße die Zusammenarbeit der Peers betont, um komplexe Aufgaben überhaupt erst bewältigen zu können.

Als Beispiel kann ein Beleuchtungskörper im Treppenhaus herangezogen werden. Dieser erkennt auf Grund seines lokalen Wissens und seiner lokalen Regeln, dass er ab einem gewissen Dämmerungsgrad leuchten muss. Dazu wird mit einer hochprioren Energie Akquisitions Anfrage versucht, das Energiebudget zu bekommen. Wenn auf Grund einer HF Störung keine Verbindung mit anderen Peers möglich ist, und folglich der ACO keine Ergebnisse liefert, sollte die Leistung dennoch erbracht werden, lediglich das Energie Accounting ist nicht mehr korrekt (Service degradation) da kein Energiebudget arbitriert werden konnte.

Ein anderes Beispiel: Mit Hilfe einer ACO Optimierung wurde eine Lösung gefunden, welche es erfordert, dass ein Peer Energiebudgets von 3 Peers akquiriert um seinen Bedarf zu decken. Angenommen die Akquirierung von Peer 1 und 2 gelingt, aber von Peer 3 nicht (z.B. weil er inzwischen ausgefallen ist). In diesem Fall hat der erste Peer ein Energiedefizit. Es gibt generell 2 Möglichkeiten auf einen derartigen Fehler zu reagieren[2]

1. Rollback: Die klassische Variante. Es wird wieder ein alter korrekter Zustand hergestellt. Im Beispiel hieße das, Peer 1 und 2 ihre Kontingente zurück zu geben und einen neuen ACO Lauf starten.

2. Rollforward: Beim Rollforward wird der Systemstatus so wie er ist hingenommen und „repariert“ so dass das System wieder in einen zulässigen Zustand übergeht. Im Beispiel bedeutet das, dass der Peer für die von Peer 3 nicht erhaltene Energiemenge einfach einen neuen Optimierungslauf startet

Im in dieser Arbeit skizzierten System ist in der Regel ein Rollforward zu favorisieren. Dies entspricht einem „Reparieren“ des Systems. Dies ist der bessere Weg, da nicht garantiert werden kann, dass ein Rollback überhaupt möglich ist. Auf Grund der hochdynamische Natur des Netzes wäre es schwierig atomare Transaktionen über mehrere Peers zu realisieren, was für ein ordnungsgemäßes Rollback nötig wäre. Wenn ein Rollback ebenfalls fehl schlägt, hat ein System, dass bei der Fehlertoleranz auf eben diesen Mechanismus setzt, ein Problem. Beim Rollforward tritt dieses Problem nicht auf, zudem hat es den Vorteil, dass ein Rollforward schneller zu realisieren ist, als ein Rollback mit anschließender Wiederholung der Transaktion.

5.4. Repast Java Simulation

Bevor der ACO auf den Embedded Systems implementiert wurde, wurde eine Simulation des ACO Algorithmus erstellt. In einer Simulation lässt sich das Verhalten des Algorithmus besser beobachten und es ist einfacher verschieden Netzwerktopologien mit einer beliebigen Anzahl an Peers zu simulieren. Die Java Simulation wurde mit Hilfe des Java Frameworks Repast (2.2.1). erstellt

5.4.1. Benutzeroberfläche

Dieser Abschnitt zeigt die Benutzeroberfläche der AAS Simulation unter Repast. Auf die allgemeine Bedienung des Repast GUI wird hier nicht eingegangen. Eine Einführung hierzu findet sich auf der Repast Website[1] oder in [3].

Nachdem die AAS Simulation in Repast geladen wurde, müssen im *ACO Simulation Settings* Dialog (Abbildung 5.2) die Modellparameter gesetzt werden. Es ist auf jeden Fall nötig ein XML Konfigurationsfile anzugeben aus dem die Netzwerktopologie (5.4.2) geladen werden soll. Der Pfad zu diesem Konfigurationsfile kann in das *Load-From* Textfeld eingetragen werden. Mit dem *Browse* Button kann das File alternativ über einen Dateirequester ausgewählt werden. Die beiden Parameter *WorldXSize* und *WorldYSize* haben nur einen Einfluss auf die graphische Darstellung. Sie legen fest, wie groß das Fenster ist in dem die Peers und Netzwerkverbindungen graphisch dargestellt werden. Diese Werte sollten nur erhöht werden, wenn im Konfigurationsfile sehr viele Peers angegeben sind.

Wenn das Modell initialisiert wird, öffnen sich einige Fenster. Neben der Repast Konsole (Abbildung 5.7), die Ausgaben des Modells anzeigt, wird eine grafische Darstellung der Netzwerktopologie geöffnet. Jeder Peer wird als blaues Rechteck dargestellt (Abbildung 5.3). Der Name des Peers wird in gelber Schrift angezeigt. Oben

[1] `http://repast.sf.net/`

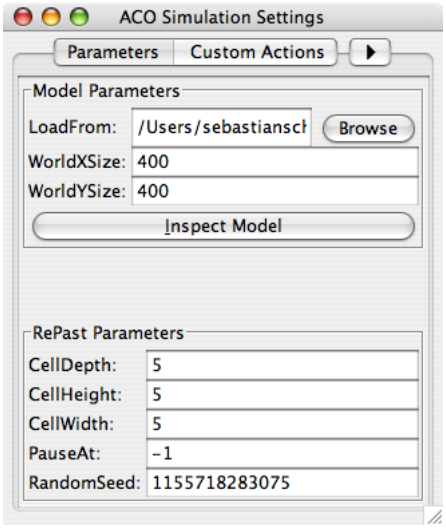


Abbildung 5.2.: Simulationsparameter

links befindet sich ein rechteckiger Bereich in dem kurze Statusnachrichten ausgegeben werden, wenn der Benutzer eine Optimierung anstößt. Die Peers können vom Benutzer mittels Drag&Drop an andere Positionen gezogen werden, wobei die Verbindungen zwischen den Peers erhalten bleiben und nachgezogen werden.

Jeder der angezeigten Peers hat bestimmte, für die Optimierung relevante Eigenschaften. Mit einen einfachen Linksklick auf den Peer öffnet sich der *Peer Inspector*. Hier ist es möglich gewisse Parameter eines Peers zu ändern. Es kann angegeben werden über welche Menge aktiver und passiver Energie dieser Peer verfügt. Mit *Priority* wird die Priorität dieses Peers festgelegt. Dieser Peer wird aktive Energie nur abgeben, wenn die anfragende Ant eine höhere Priorität besitzt. Der Werte unter *Needed Amount* gibt an, wieviel Energie der Peer zu akquirieren versucht und sollte hier normalerweise nicht von Hand geändert werden. Die Änderung des Peernamens ist nur eine kosmetische Korrektur. Allerdings sollte der Peername nicht verändert werden, während eine Optimierung durchgeführt wird. Das Feld *State* zeigt an, in welchen Zustand sich die interne ACO Logik dieses Peers befindet. Dieses Feld ist nur für Debugging Zwecke nützlich und sollte normalerweise nicht von Hand verändert werden.

Um die Simulation anzuweisen eine Optimierung durchzuführen, muss in den *Custom Actions* Reiter (Abbildung 5.5) des *ACO Simulation Settings* Dialogs (Abbildung 5.2) gewechselt werden. Der Button *Setup ACO Run* öffnet einen Dialog, der es erlaubt die anstehende Optimierung zu parametrieren (Abbildung 5.6).

In diesem Fenster kann eine Optimierung ausgehend von einem bestimmten Peer angestoßen werden. Mit der *Select Peer* Combobox kann angegeben werden, welcher Peer Energie akquirieren soll. Der *Select Amount* Slider konfiguriert die zu akquirierende Menge. Der hier angegebene Wert ist dann im Inspector des entsprechenden Peers unter *NeededAmount* wieder zu finden. Mit *Priority* kann die Priorität der An-

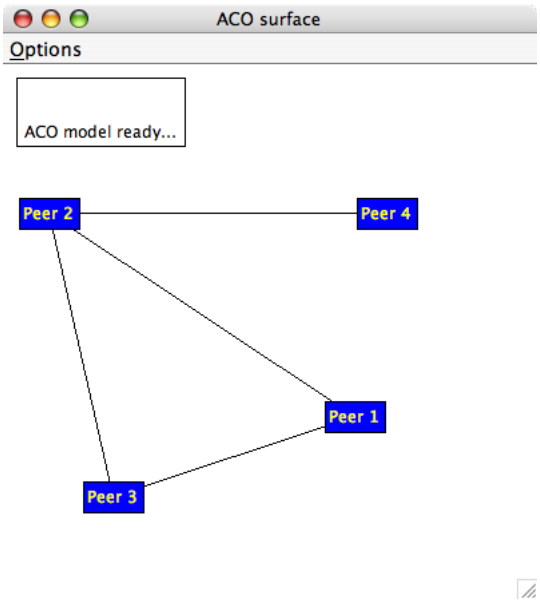


Abbildung 5.3.: Netzwerktopologie Visualisierung

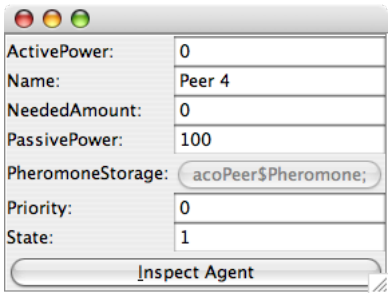


Abbildung 5.4.: Peer Inspector

frage konfiguriert werden. Diese Priorität muss nichts mit der Priorität des Initiators zu tun haben, aber andere Peers im Netz werden ihre aktive Energie nur dann abgeben, wenn die hier angegebene Priorität höher ist als ihr *Priority* Feld im Inspector. Mit einem Klick auf *Execute* werden die Änderungen wirksam.

Wenn man das Model jetzt über die Repast GUI startet so beginnt die Optimierung. Im Verlauf der Optimierung erscheinen Statusmeldungen in der Repast Konsole (Abbildung 5.7). Sobald die Optimierung abgeschlossen ist, d.h. wenn entweder eine Lösung umgesetzt wurde oder die Optimierung erfolglos beendet wurde, geht das Repast Model automatisch in den Pause Zustand über.

Während mit einem Model gearbeitet wird, wird ständig eine Grafik über die verfügbare Energie im Gesamtnetzwerk aktualisiert (Abbildung 5.8). Die Grafik ist aufgeteilt in aktive und passive Energie. Es finden Änderungen statt, wenn der Benutzer manuell über einen Peer Inspector die Energiewerte eines Peers ändert oder wenn auf

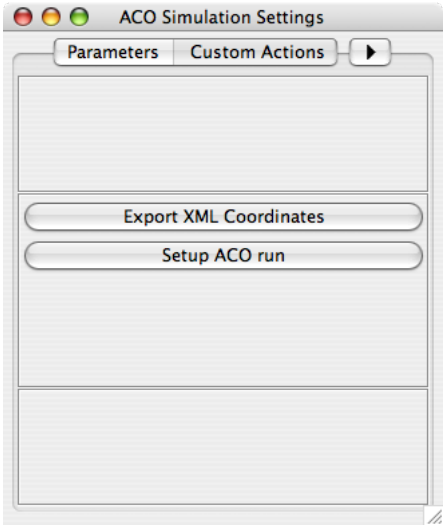


Abbildung 5.5.: AAS Custom Actions

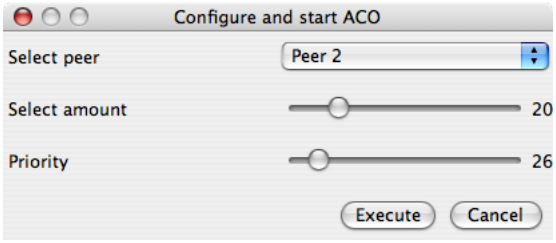


Abbildung 5.6.: ACO Optimierung initialisieren

Grund einer Optimierung Energiezertifikate zwischen zwei Peers ausgetauscht werden, so dass passive Energie in aktive übergeht. Der Graph bietet eine gewisse Kontrolle, dass das Programm korrekt funktioniert: Solange nicht manuell an den Energiewerten der einzelnen Peers etwas verstellt wird, muss die Gesamtenergiemenge im Netzwerk immer konstant sein, d.h. wenn sich die aktive Energie um einen bestimmten Betrag erhöht muss gleichzeitig die passive Energie um den gleichen Betrag sinken und andersherum.

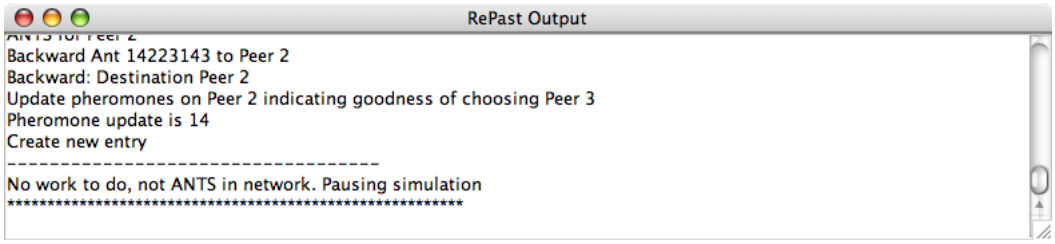


Abbildung 5.7.: Repast Konsole während der Optimierung

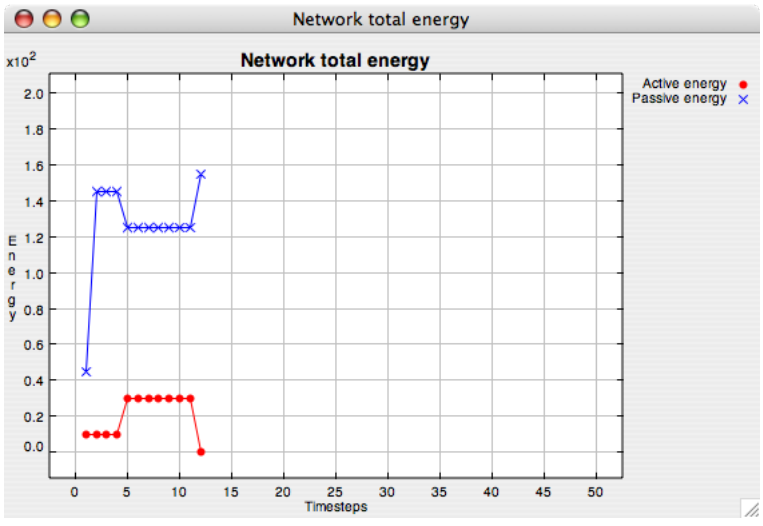


Abbildung 5.8.: Gesamtenergie Grafik

5.4.2. Topologiekonfiguration

Der große Vorteil an einer Simulation ist, dass unterschiedliche Szenarien und Topologien simuliert werden können. In der AAS werden die Beschreibungen von einem externen XML File geladen. In diesem File können verschiedene Peers und die ihnen zugeordneten Energiemengen definiert werden. Welcher Peer zu welchem anderen Peer Verbindung aufnehmen kann wird ebenfalls definiert. Hierbei sind alle Verbindungen bidirektional, d.h. wenn eine Verbindung von Peer 1 zu Peer 2 definiert ist, so kann die Kommunikation zwischen diesen beiden Peers in beide Richtungen erfolgen.

Für das Konfigurationsfile existiert ein XML Schema (2.2.2) Beschreibung. Das ant.xds Schema (A.1) definiert den Aufbau einer Toplogiekonfiguration formal. Damit kann der verwendete Schema fähige XML Parser Xerces (2.2.2) schon beim Parsen des Files feststellen, ob das XML File eine gültige Topologiebeschreibung darstellt. Dadurch ist weniger Überprüfungs- und Fehlerbehandlungscode in der eigentlichen Verarbeitung der Daten nötig.

Das Konfigurationsfile für die in Abbildung 5.3 gezeigte Topologie sieht wie folgt aus:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : testnetwork.xml
    Created on : 2. April 2006, 18:58
    Author     : Sebastian Schildt
    Description:
        Test network for AAS simulation.
-->

<ACO-NET xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='acoNetwork.xsd'>
    <!-- xsi:noNamespaceSchemaLocation='file:/acoNetwork.xsd'> -->

    <peer name="Peer 1" >
        <position posX="241" posY="252" />
        <energy active="0" passive="13" />
    </peer>

    <peer name="Peer 2" >
        <position posX="12" posY="100" />
        <energy active="10" passive="20" />
    </peer>

    <peer name="Peer 3" priority="3" >
        <position posX="60" posY="312" />
        <energy active="0" passive="12" />
    </peer>

    <peer name="Peer 4" >
```

```
        <position posX="268" posY="102" />
        <energy active="0" passive="0" />
    </peer>

    <edge >
        <from>Peer 2</from>
        <to>Peer 1</to>
    </edge>

    <edge >
        <from>Peer 2</from>
        <to>Peer 3</to>
    </edge>

    <edge >
        <from>Peer 1</from>
        <to>Peer 3</to>
    </edge>

    <edge >
        <from>Peer 2</from>
        <to>Peer 4</to>
    </edge>

</ACO-NET>
```

In den Zeilen 11 und 12 wird auf die Schema Beschreibung *acoNetwork.xsd* verwiesen, welche der Parser benutzt um die Korrektheit des Dokumentes zu verifizieren. Zuerst werden in den Zeilen 10-32 4 Peers definiert. Den Peers müssen Namen gegeben werden. Die Angabe der aktiven und passiven Energieanteile ist optional. Werden sie weggelassen, wird den Attributen der Wert 0 zugewiesen. Die Angabe im `position` Element sagt, an welchen Koordinaten der Peer in der GUI gezeichnet wird. In den Zeilen 36 bis 54 werden die Netzwerkverbindungen zwischen den Peers definiert. Jeder Verbindung, d.h. jedem *`edge`* Element muss ein *`from`* und ein *`to`* Peer zugewiesen werden. Da alle Verbindungen bidirektional sind, ist es egal welcher Peer als *`from`* und welcher als *`to`* angegeben wird.

Folgende Constraints sind im XML Schema definiert und können bereits vom XML Parser überprüft werden:

- Jedes Dokument *muss* mindestens ein `peer` und ein `edge` Element enthalten
- Jedes `peer` Element *muss* ein String Attribut `name` besitzen
- Das Attribut `name` eines `peer` Elementes *muss eindeutig* sein
- Jedes `peer` Element *kann* das numerische Attribut `priority` besitzen

- Jedes `peer` Element *kann* optional *genau ein* `position` Subelement besitzen
- Hat ein `peer` Element ein `position` Subelement, so *muss* dieses die numerischen Attribute `posX` und `posY` enthalten
- Jedes `peer` Element *kann* optional *genau ein* `energy` Subelement besitzen
- Hat ein `peer` Element ein `energy` Subelement so *muss* dieses die numerischen Attribute `active` und `passive` enthalten
- Jedes `edge` Element *muss* die Subelemente `from` und `to` enthalten
- Der Inhalt der Subelemente `from` und `to` eines `edge` Elementes *muss* mit dem Inhalt des Attributs `name` eines der `peer` Elemente übereinstimmen

Beispiele für weitere in der Simulation untersuchte Topologien finden sich in Abschnitt A.2.

5.4.3. ACO Datenstrukturen

Die zur Optimierung gehörenden Datenstrukturen der Java Simulation sind im wesentlichen denen aus der C Implementierung nachempfunden (siehe Abschnitt 5.2) um eine Vergleichbarkeit zu gewährleisten. Die entsprechenden Java Klassen sind in Abbildung 5.9 dargestellt.

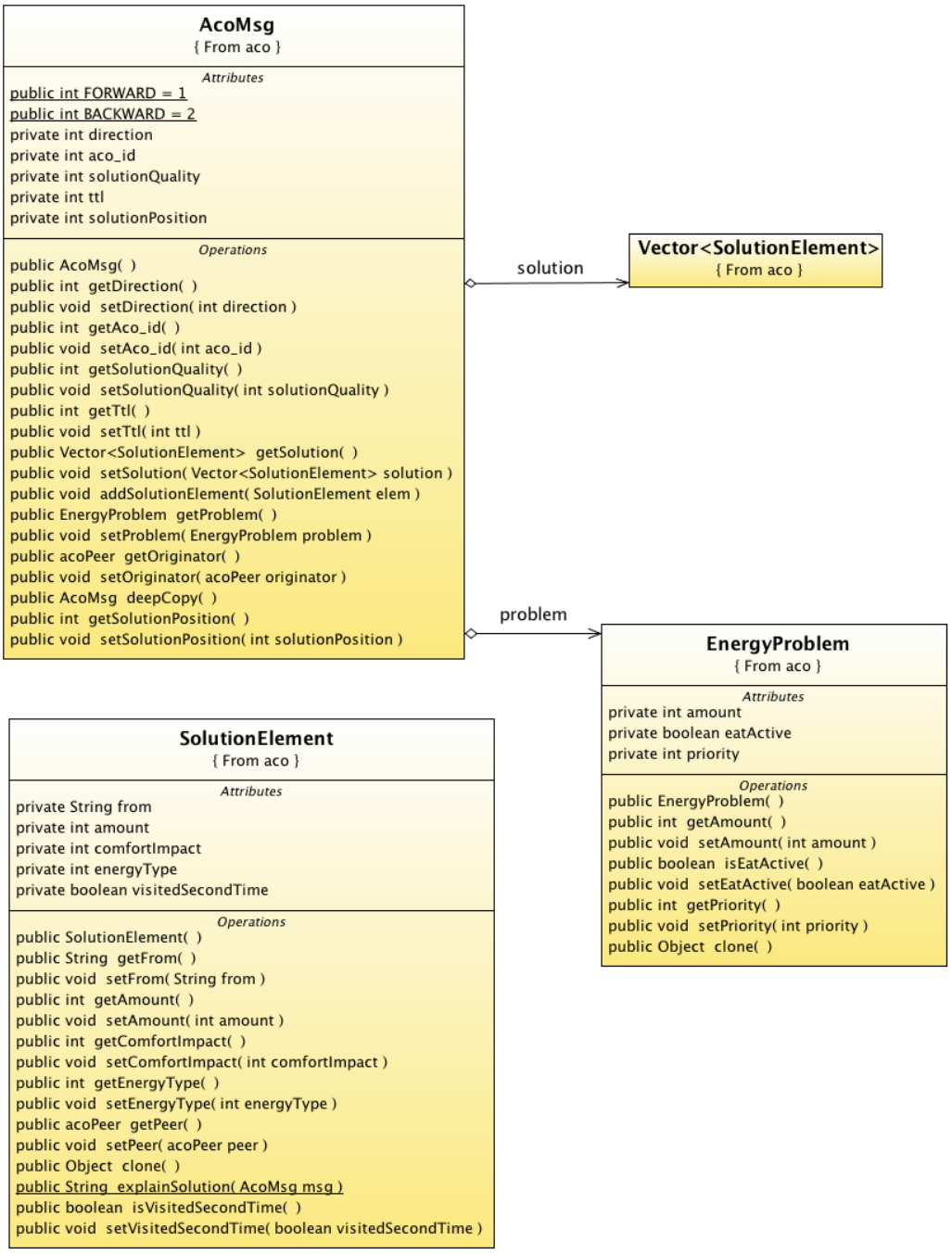


Abbildung 5.9.: Datenstrukturen der Simulation - Klassendiagramm

5.5. L3 Services

Dieser Abschnitt behandelt die L3 Services, die in ihrer Gesamtheit den ACO Algorithmus für die Energieverteilung im verteilten System realisieren. Folgende Services sind Bestandteil der Implementierung:

Service	Funktion
ACO Service	Ein L3 Service mit der Service ID 0xAC. Alle Peers die ACO fähig sind müssen diesen Services implementieren. Ants werden zwischen verschiedenen Instanzen des ACO Service hin und her geschickt
Interface Service	Problemspezifischer Service mit der L3 Service ID 0xAB. Ermöglicht die Abfrage des aktuellen Geräte- und Energiestatus. Empfängt Ein- und Ausschaltbefehle
Trade Service	Problemspezifischer Service mit der L3 Service ID 0xAA. Ermöglicht es Energiekontingente zwischen den verschiedenen Peers auszutauschen

5.5.1. ACO Service

Das ACO Framework verwendet die L3 Service ID 0x0AC um Ants zu transferieren. Der ACO Service entscheidet in welche Richtung die Ant sich weiter bewegen soll und er verteilt die Pheromone. Die Konstruktion und Evaluierung von Lösungen ist problemabhängig und wird von der Applikation vorgenommen, die das ACO Framework nutzt.

Die Nachrichten, die der ACO Service verschickt bestehen aus einer Ant Datenstruktur (5.2.1) gefolgt von der Problembeschreibung (5.2.2) und einer oder mehreren SolutionElements (5.2.3). Der ACO Service nutzt ein TTL (TimeToLive) Feld (5.2.1) um zu verhindern, dass Ants auf der Suche nach einer Lösung endlos im Netzwerk kreisen. Das TTL Feld wird von jedem Peer, der eine Forward Ant empfängt dekrementiert. Das bedeutet eine Ant die mit TTL x gestartet wird, kann maximal Lösungen errechnen, die aus x SolutionElements bestehen. Wenn eine Ant Generation n keine Lösung gefunden hat, die die Applikation akzeptiert (Abbruchbedingung nicht erfüllt), so wird die Ant Generation n+1 mit einer höheren TTL versehen. Dies wird solange wiederholt bis eine Lösung akzeptiert wird, oder die maximale Anzahl an Ant Generationen gestartet wurde.

Wenn die Applikation entscheidet, dass eine Ant eine Lösung gefunden hat, wird diese Ant in den Backward Mode versetzt. Der ACO Service sorgt dann dafür, dass die ANT sich rückwärts zum Initiator der Optimierung bewegt, d.h. dass alle Peers von der Backward Ant in der umgekehrten Reihenfolge wie von der Forward Ant durchlaufen werden. Eine Backward Ant kann nicht auf direktem Wege zum Initiator zurück geschickt werden, da sie auf dem Rückweg der Lösungsqualität entsprechende Pheromonspuren verteilen muss. Diese müssen auf allen von der Forward Ant besuchten Peers hinterlegt werden.

5.5.2. Interface Service

Der Interface Service hat die L3 Service ID 0xAB. Mit dem Interface Service kann der Gerätestatus sowie der aktuelle Energiezustand eines Gerätes abgefragt werden. Ferner kann über den Interface Service ein Ein- oder Abschalten des Gerätes initiiert werden.

Folgende Kommandos können an den Interface Service gesendet werden

Kommando	
0x20	Eine Energiestatusanfrage. Wenn dieses Kommando empfangen wird, wird der aktuelle Energiestatus zurück gesendet
0x01	Der Einschaltbefehl. Wird eine 0x01 empfangen, wird das Gerät eingeschaltet sofern genügend Energie lokal zur Verfügung steht. Ansonsten wird ein ACO Optimierungslauf initiiert um genügend Energie zu akquirieren
0x00	Der Ausschaltbefehl. Wird eine 0x00 empfangen, wird das Geräte ausgeschaltet. Der aktive Energieanteil wird zum passiven übertragen

Wenn eine Energiestatusanfrage empfangen wird, oder wenn sich der Status z.B. durch Ein- oder Ausschalten verändert wird eine Energiestatusmeldung mit folgendem Format versendet:

Byte	Inhalt
1	0x20
2	Aktueller Gerätestatus: 0x00=Off, 0x01=On, 0x02 Transitional
3-4	Aktive Energie, die dieser Peer derzeit nutzt
5-6	Passive Energie, die diesem Peer zur Verfügung steht

Der Gerätestatus „Transitional“ zeigt an, dass ein Gerät auf Nutzeranforderung eingeschaltet werden soll, aber diesem Gerät derzeit nicht genügend aktive Energie zur Verfügung steht. Während die ACO Optimierung läuft bleibt das Gerät im Zustand „Transitional“. Kann die Optimierung genügend Energie akquirieren, wechselt das Gerät selbstständig in den Zustand „On“ und sendet eine entsprechende Energiestatusmeldung. Schlägt die Optimierung fehl, wechselt das System auf die gleiche Weise in den Zustand „Off“.

5.5.3. Trade Service

Der Trade Service ermöglicht es Energiezertifikate zwischen verschiedenen Geräten zu verschieben. Wenn der ACO eine Lösung gefunden hat, wir der Trade Service aktiv um diese Lösung umzusetzen. Der Trade Service ermöglicht es, Energiezertifikate zwischen den Peers zu transferieren. Dabei wird mit einem einfachen Acknowledge gearbeitet, um die Transaktion abzusichern. Im ersten Schritt sendet der ein Peer eine Anfrage an einen anderen Peer um die benötigte Energie anzufordern:

Byte	Inhalt
1	0x1 (ACO_TRADE_GIVE). Bedeutet, dass dies eine Energieanfrage ist
2-3	Menge der passiven Energie die angefordert wird
4-5	Menge der aktiven Energie die angefordert wird

Der Peer, der die Nachricht empfängt überprüft, ob er die angefragte Energie zur Verfügung stellen kann. Es wird davon ausgegangen, dass der Trade Service nur auf ehrlich Art und Weise genutzt wird, d.h. bei der Anfrage nach aktiver Energie wird nicht noch einmal überprüft ob der Anfrager von der Priorität her überhaupt berechtigt ist die Energie anzufordern. Diese Überprüfung wurde ja während des ACO Durchlaufes aus dem dieser Lösungsbestandteil kommt bereits vorgenommen.

Wenn der angefragte Peer Energie zur Verfügung stellen kann, sendet er folgendes Energiezertifikat als Antwort:

Byte	Inhalt
1	0x2 (ACO_TRADE_ACK). Bedeutet dass in diesem Paket Energie zur Verfügung gestellt wird
2-3	Menge der passiven Energie, die zur Verfügung gestellt wird
4-5	Menge der aktiven Energie, die zur Verfügung gestellt wird

Bevor ein Peer die ACO_TRADE_ACK Nachricht schickt, dekrementiert er seine eigenen Energiewerte um den abzugebenden Energieanteil. Kann ein Peer nur einen Teil der in der ACO_TRADE_GIVE angefragten Energiemenge zur Verfügung stellen, sendet er ebenfalls ein ACO_TRADE_ACK mit entsprechend geringeren Energiebeträgen. Der Peer, der die ACO_TRADE_ACK Nachricht empfängt, kann die im Energiezertifikat enthaltenen Anteile für passive & aktive Energie sofort zu seinem Budget hinzufügen.

Kann ein angefragter Peer überhaupt keine Energie zur Verfügung stellen, sendet er ebenfalls eine entsprechende Nachricht:

Byte	Inhalt
1	0x3 (ACO_TRADE_NACK). Bedeutet, dass dieser Peer keine Energie zur Verfügung stellen kann

5.6. PDA Anwendung

Um die Verbraucher ein- oder auszuschalten und den Energiestatus zu visualisieren entstand im Rahmen einer Projektarbeit eine PDA Applikation als Interface zwischen dem Benutzer und den Embedded Komponenten im Netz. Die Anwendung baut über das im PDA integrierte WLAN Modul eine Verbindung zu den Embedded Peers auf und interagiert mit dem L3 Interface Service (siehe 5.5.2). Das bedeutet, die Applikation kann einzelne Verbraucher ein- und ausschalten sowie den aktuellen Energiestatus abfragen. Das GUI zeigt alle gefundenen Peers in einer tabellarischen Übersicht an (Abbildung 5.10). Für jeden Peer werden folgende Informationen angezeigt:

Abbildung 5.10.: EePDA Anwendung

Spalte	Bedeutung
UUID	Die L3 UUID des Peers
Name	Der L3 Name des Peers
Act	Menge der aktiven Energie, die diesem Peer zugeordnet ist
Pas	Menge der passiven Energie, die diesem Peer zugeordnet ist
State	Aktueller Status des Peers. Grün bedeutet der Verbraucher ist eingeschaltet; Rot bedeutet ausgeschaltet. Ein gelber Indikator bedeutet, der Verbraucher soll eingeschaltet werden und ist dabei die nötige Energie zu akquirieren (Zustand Transitional, siehe 5.5.2)

Mit einem Klick auf den Indikator ganz rechts kann der Status jedes Peers umgeschaltet werden. Ist der Peer ausgeschaltet, wird ein Einschaltbefehl gesendet und andersherum. Befindet sich der Peer im Transitional Zustand (gelber Indikator) bewirkt ein Klick auf den Indikator nichts. Wenn ein Peer seinen Energiezustand ändert, zum Beispiel durch Ein- oder Ausschalten oder das Abgeben von Energie an einen anderen Teilnehmer so aktualisiert sich die Anzeige für Pas, Act und State automatisch. Zusätzlich kann über den *GetPeerState* Button auch manuell der Energiestatus eines beliebigen Peers abgefragt werden.

5.7. Ant Verarbeitung

Die Art und Weise wie ein Peer empfangene Ants verarbeitet, ist maßgeblich für das Verhalten des Gesamtsystems. In diesem Abschnitt soll im Detail dargestellt werden, wie ein Peer sich verhält, wenn er eine Ant empfängt.

5.7.1. Forward Ants

Das Prinzip der Ant Bewegung ist ACO typisch: Ein Peer, der zusätzliche Energie benötigt startet Ants im Forward Mode. Diese Forward Ants werden von den Nachbar Peers empfangen, welche dann jeweils ihren Teil zu der Lösung beitragen und die Ants gegebenenfalls weiter schicken. Die Abbildung 5.11 zeigt im Detail wie Forward Ants verarbeitet werden.

Als erstes wird bei einer empfangenen Forward Ant der TTL Zähler dekrementiert. Alle Ants beginnen ihr Leben mit der Einstellung eatActive=false, d.h. Peers werden dieser Ant vorerst nur passive Energieanteile anbieten. Sinn der Sache ist es, eine Lösung zu finden, bei der zur Befriedigung der Energieanforderung keine anderen Verbraucher abgeschaltet werden müssen. Unterschreitet das TTL Feld jedoch einen gewissen kritischen Wert (GoActiveThreshold), dann wird eine Ant in den „aggressiven“ Modus umgeschaltet, d.h. fortan werden andere Peers dieser Ant auch ihre aktiven Energieanteile anbieten sofern die Priorität der Ant höher ist als jene des Peers. Dem Umschalten einer Ant in den aggressiven Modus bei einer bestimmten TTL Schwelle liegt der Gedanke zu Grunde, dass die Wahrscheinlichkeit hoch ist, dass die Ant es nicht schaffen wird, eine Lösung nur aus passiven Energieanteilen zu generieren, wenn es ihr in den vorherigen Schritten nicht gelungen ist.

Danach wird überprüft, ob die Ant diesen Peer zum ersten oder bereits zum zweiten Mal besucht. Wenn dieser Peer zum Ersten mal besucht wird heißt das, er hat dieser Ant noch nicht seine eventuell verfügbaren passiven Energieanteile angeboten. Wenn genügend passive Energie vorhanden ist, um den gesamten Bedarf der Ant zu befriedigen, wird die entsprechende Menge Energie der Ant angeboten. Steht weniger passive Energie zur Verfügung, als die Ant insgesamt benötigt so wird die gesamte passive Energie dieses Peers der Ant angeboten. Da passive Energie verfügbare, ungenutzte Energie ist, erfolgt ihre Abgabe unabhängig von der Priorität der Ant oder des Peers.

Nachdem der Ant eventuell vorhandene passive Energie angeboten wurde, oder wenn die Ant diesen Peer bereits das zweite Mal besucht, wird entscheiden, ob dieser Ant auch aktive Energieanteile angeboten werden können. Dazu müssen folgende Bedingungen erfüllt sein

- Der Energiebedarf der Ant muss größer 0 sein (es kann sein, dass der Energiebedarf bereits vorher von der passiven Energie gedeckt wurde)
- Die Ant muss sich im aggressiven Modus befinden
- Die Priorität der Ant muss höher sein als die Priorität des Peers

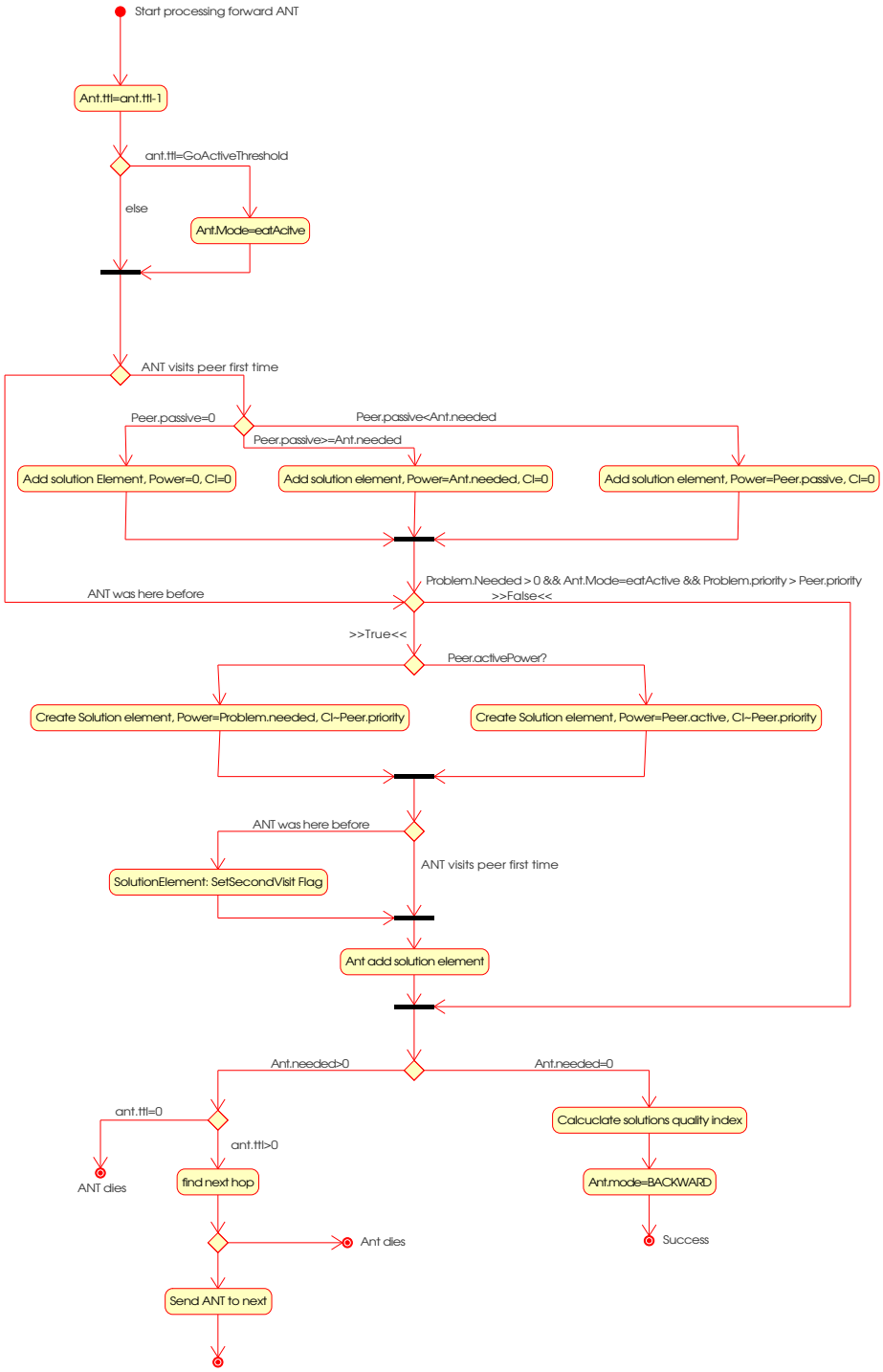


Abbildung 5.11.: ANT Forward Aktivitätsdiagramm

Wenn alle Bedingungen erfüllt sind, wird die aktive Energie auf die gleiche Weise wie die passive Energie zugeteilt: Ist genügend aktive Energie vorhanden um den kompletten Energiebedarf der Ant zu decken, wird die entsprechende Energiemenge der Ant angeboten. Genügt die vorhanden aktive Energie nicht, wird der Ant die gesamt verfügbare aktive Energie angeboten. Das Anbieten von aktiver Energie hat immer einen Comfort Impact zur Folge. Dieser ist abhängig von der Priorität des Peers.

Aus den angebotenen passiven und aktiven Energieanteilen wird ein neues SolutionElement erzeugt. Sofern dies der zweite Besuch dieser Ant auf dem Peer war, wird das entsprechende Flag im SolutionElement gesetzt. Der Energiebedarf in der Problembeschreibung der Ant wird um die Summe aus angebotener passiver & aktiver Energie dekrementiert.

Als letztes wird überprüft, ob durch die aktiven und passiven Energieanteile dieses Peers der Energiebedarf der Ant komplett gedeckt werden kann, d.h. ob eine Lösung gefunden wurde. Wenn eine Lösung gefunden wurde, berechnet der Peer einen Qualitätsindex dieser Lösung. Dazu werden die Comfort Impact Werte aller SolutionElements dieser Ant herangezogen. Die Ant wird in den Backward Mode versetzt, so dass sie zum Urheber der Optimierung zurück geschickt werden kann.

Sollte dieser Peer keine Lösung gefunden haben, wird die TTL der Ant überprüft. Ist sie bereits 0, stirbt die Ant. Ansonsten wird versucht, die Ant an einen weiteren Peer weiter zu leiten. Der nächste Peer muss ein Nachbarpeer dieses Peers sein. Peers, die von dieser Ant noch nicht besucht wurden, werden bevorzugt. Die Pheromonspuren auf dem aktuellen Peer beeinflussen die Wahrscheinlichkeiten bei der Auswahl des nächsten Peers (4.3). Die Ant wird an den ausgewählten Peer geschickt. Sollte es nicht möglich sein, einen nächsten Peer zu bestimmen, z.B. weil alle verfügbaren Nachbarpeers bereits zweimal besucht wurden, stirbt die Ant.

5.7.2. Backward Ants

Wenn eine Ant eine Lösung gefunden hat, muss sie sich zum Urheber der Optimierung zurück bewegen, um dort die potentielle Lösung abzuliefern. Dabei verfolgt eine Ant genau den Weg zurück, den sie auf dem Hinweg bereits genommen hat. Die Ant kann nicht direkt zum Urheber zurück geschickt werden. Da je nach Netzwerktopologie dass Netz nicht transitiv ist, ist nicht sicher gestellt, dass überhaupt ein direkter Weg vom letzten Peer zum Urheber der Optimierung existiert. Außerdem muss die Ant auf ihrem Rückweg auf allen besuchten Peers Pheromone entsprechend der Qualität der Lösung verteilen um spätere Ants zu führen. Abbildung 5.12 zeigt im Detail, wie eine solche Backward Ant von einem Peer verarbeitet wird.

Zuerst wird überprüft, ob der aktuelle Peer der Urheber der Optimierung war, d.h. ob die Ant bereits ihr Ziel erreicht hat. Ist dies nicht der Fall wird geschaut von welchem Peer die Backward Ant geschickt wurde, d.h. der Peer den die Ant im Forward Mode nach dem aktuellen Peer besucht hat. Für diesen Peer wird dann im aktuellen Peer eine Pheromonspur hinterlegt. Die Menge der deponierten Pheromone ist abhängig von der Menge der Energie, die der entsprechende Peer angeboten hat und

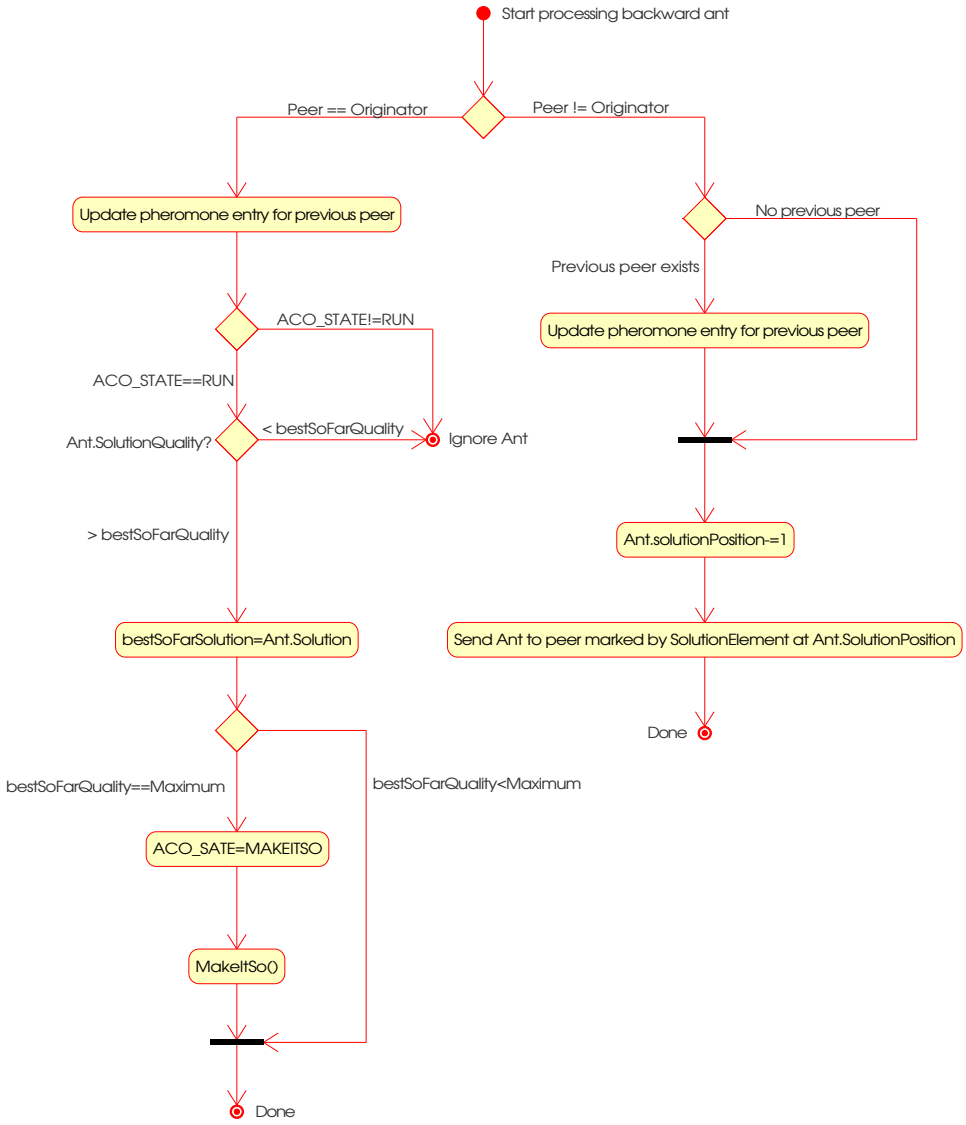


Abbildung 5.12.: ANT Backward Aktivitätsdiagramm

dem resultierenden Comfort Impact. Die Qualität der Gesamtlösung fließt ebenfalls in die Pheromonberechnung ein (4.4). Die Pheromone geben also an, wie empfehlenswert es für eine Forward Ant ist, nach diesem Peer denselben Nachfolger auszuwählen wie die aktuelle bearbeitete Backward Ant es im Forward Mode getan hat. Der Peer bei dem eine Backward gestartet ist, ist ein Sonderfall: Auf dem Ausgangspunkt einer Backward Ant, also dem Peer, auf dem eine Ant vom Forward in den Backward Mode geschaltet wurde können keine Pheromone hinterlegt werden, da es ja keinen nächsten Peer gibt.

Nach der Pheromonverteilung wird das solutionPosition Feld der Ant um 1 dekrementiert. Dieses Feld zeigt auf ein SolutionElement der Ant und wird dazu benutzt zu verfolgen, wo sich die Ant auf ihrem Rückweg zum Urheber der Optimierung gerade befindet. Als letztes wird die Ant zum nächsten Peer gesendet. Das ist der Peer, der von dem SolutionElement an der Position solutionPosition beschrieben wird.

Wird zu Beginn der Verarbeitung festgestellt, dass die Backward an ihrem Ziel, d.h. dem Urheber der Optimierung angekommen ist, wird die Ant anders verarbeitet. Als Erstes wird wie im Normalfall auch auf dem aktuellen Peer eine Pheromonspur deponiert. Dann wird überprüft, ob sich der Peer noch im Staus ACO_STATE_RUN befindet, d.h. ob eine ACO durchgeführt wird. Ist dies nicht der Fall, wird die Ant ignoriert. Dieser Fall kann z.B. eintreten wenn eine frühere Ant eine perfekte Lösung gefunden hat, welche bereits zur Realisierung kam, so dass sich der Peer im Realisierungs- oder Idlestate befindet. Die aktuelle Ant wäre dann ein Nachzügler, der für die Optimierung nicht mehr relevant ist. Wenn die ACO Optimierung noch nicht abgeschlossen ist, wird überprüft ob die Lösung der Ant besser ist, als die bislang beste Lösung in diesem Optimierungslauf. Ist dies nicht der Fall, wird die Ant ignoriert (da ja bereits eine bessere Lösung bekannt ist). Wenn diese Ant die bislang beste Lösung enthält, wird sie als beste Lösung auf diesem Peer gespeichert. Ist die Lösung der Ant perfekt, d.h. hat sie einen Comfort Impact von 0, kann sie direkt umgesetzt werden, da keine Ant eine bessere Lösung erzeugen kann. Dazu wird der Status des Peers auf MAKEITSO gesetzt, was bedeutet, dass dieser Peer sich jetzt in der Realisierungsphase befindet. Die Umsetzungsroutine (5.5.3) wird gestartet und die Bearbeitung der Backward Ant ist abgeschlossen.

6. Zusammenfassung und Ausblick

Es war Ziel dieser Arbeit ein hochdynamisches Netz verteilter autonomer Peers in die Lage zu versetzen im Verbund die dem Netz zur Verfügung stehende Energie möglichst ideal im Sinne des Benutzers einzusetzen. Die einzelnen Netzteilnehmer arbeiten hierbei komplett asynchron und besitzen nur eine begrenzte lokale Rechenleistung und Speichermenge. Es wurde gezeigt, dass es sich bei dem Problem um ein kombinatorisches Optimierungsproblem handelt, welches sich für entsprechend umfangreiche Probleminstanzen unabhängig von der zur Verfügung stehenden Rechenleistung nur mit einem nicht deterministischen Algorithmus in annehmbarer Zeit lösen lässt. Eine Sichtung bekannter Metaheuristiken (2.6) führte zu der Idee, dass die Ant Colony Optimization in der Lage sein sollte eine Optimierung auf Basis eines komplett asynchronen Netzes nur mit Hilfe lokaler Informationen durchzuführen. Im Verlaufe der Entwicklung hat sich gezeigt, dass die ACO dazu nur in der Lage ist, wenn bekannte ACO Varianten modifiziert werden, um die Besonderheiten des Systems widerzuspiegeln. Diese Arbeit schlägt daher das Asynchronous Ant System AAS (4) als algorithmischen Ansatz vor.

Mit dem AAS war es möglich, das Problem der Energieverteilung im verteilten Netz zu lösen. Sowohl die Simulation als auch die Applikation im Embedded Umfeld demonstrieren die prinzipielle Eignung des Algorithmus. In einem nächsten Schritt sollte das AAS mit deutlich größeren Probleminstanzen getestet werden, da in einem kompletten Gebäude sicher einige 100 Peers zu erwarten sind. An Hand umfangreicher Simulationen kann empirisch festgestellt werden, welchen Einfluss die einzelnen Parameter des AAS auf die Performance des Algorithmus haben. Ebenfalls kann getestet werden, ob es noch Optimierungsmöglichkeiten im Algorithmus selber gibt. Einige Vorschläge hierzu macht bereits Abschnitt 4.7.

Das Konzept der Energiezertifikate ist ausbaubar. In dieser Arbeit bestehen die Energiezertifikate lediglich aus einer bestimmten Menge nicht näher spezifizierter Energie. Die Beschreibung dieser Zertifikate kann ausgedehnt werden, so dass der verteilten Energie Attribute zugewiesen werden. Energie kann nach Erzeuger und Art unterschieden werden, wie z.B. hauseigene Photovoltaik, Atom Strom aus dem Netz, oder „grüner“ Strom aus dem Netz. Dadurch wird der Faktor der Energie mehrdimensional, was weitere Optimierungs- und Auswertungsmöglichkeiten eröffnet. Anstatt irgendwelche Energie zu verwenden, kann das Netz sich darauf konzentrieren möglichst regenerativ erzeugte Energie einzusetzen. Solange die zur Verfügung stehende Kapazität regenerativer Energie nicht voll ausgeschöpft ist, können dem Benutzer alle Wünsche erfüllt werden. Wird unerwünschte Energie aus dem Netz zur Deckung aller Verbräuche benötigt, so muss es für den Verbraucher „teurer“ werden, Energie zu verbrauchen. Auch kann auf diese Weise der ökologische Footprint der einzelnen Benutzer

genauer ermittelt werden. Die Attribuierung einer Quelle für die in einem Zertifikat enthaltene Energie würde eine genauere, besser an die aktuelle Situation angepasste Steuerung des Energieumsatzes ermöglichen. Zertifikate, die von der Photovoltaik Anlage auf dem Dach stammen, können von dieser wieder eingezogen werden, wenn der Lichteinfall nachlässt. Das Netz muss dann mit Mechanismen wie dem AAS ein neues Energiegleichgewicht finden.

Wenn man das Gesamtsystem betrachtet, fallen weitere Herausforderungen ins Auge: Wie in dieser Arbeit gezeigt, ist das Verhalten des AAS in hohem Maße von den Prioritäten der einzelnen Peers abhängig. Diese Prioritäten werden in dieser Arbeit als gegeben hingenommen. Was sie jedoch widerspiegeln, sind Regeln sowie die Bedürfnisse der Menschen, nach denen sich das System richten sollte. Die Prioritäten sind maßgeblich entscheidend für die Lösungen die das AAS als gut anerkennt und die umgesetzt werden. Damit sind sie die direkte Schnittstelle zum Menschen, der die Entscheidungen des Systems akzeptieren muss. Damit der Mensch die Entscheidungen akzeptiert, müssen sie für ihn nachvollziehbar sein und nicht willkürlich erscheinen. Es ist daher sehr interessant zu untersuchen, auf welche Weise und in welchem Umfang der Mensch in der Lage sein sollte die Prioritäten zu beeinflussen.

Die Vorgabe der Prioritäten ist eine Möglichkeit des Menschen dem System Vorgaben zu machen. Damit die Entscheidungen des Gesamtsystem nachvollziehbar werden, ist der umgekehrte Weg, Erkenntnisse des Systems an den Menschen weiter zu geben ebenso wichtig. Es ist ein wesentliches Ziel des Projektes, dem Menschen seinen Energieverbrauch bewusst zu machen. Der Mensch soll mit der Zeit *lernen* welche seiner Aktionen welche Menge an Energieumsatz nach sich ziehen. Dazu ist es nötig, dass das System jeden Schaltvorgang und jeden Energieverbrauch einer Person oder Entität zuordnen kann, und diese Informationen irgendwo hinterlegt. Ein erster Schritt wurde mit der PDA Anwendung bereits gemacht. Die Anwendung ist in der Lage auf Wunsch jede Änderung des Energiezustandes in eine Datenbank zu protokollieren. Die eigentliche Herausforderung liegt hier in der sinnhaften Auswertung der Daten. Hierbei sollte auch der Datenschutz nicht vergessen werden, denn die akkumulierten Energieverbrauchsinformationen einer Person ergeben nicht nur ein Profil seiner Energienutzung sondern lassen auch Aussagen über seine Bewegungen über die Zeit zu, da die meisten Verbraucher ortsfest sind.

Mit dem Mensch im Mittelpunkt gibt es also einen Informationsfluss vom Menschen zum System und zurück. Hierbei stellt sich die Frage, wie der Mensch mit dem System interagieren soll, und wie ihm die Informationen repräsentiert werden sollen. Ein PDA, wie in dieser Arbeit verwendet, ist sicher eine aus IT Sicht einfache Lösung, welche jedoch kaum alltagstauglich ist. Eine Lösung zur Identifizierung der einzelnen Benutzer sind RFID Chips, die jeder Mensch bei sich tragen kann um eine Personalisierung der Energieumsätze möglich zu machen. Ein entsprechend ausgerüstetes Gebäude wird jedoch komplexere Interaktionen zulassen als das einfache Ein/Aus wie es z.B. ein Lichtschalter darstellt. Vielmehr sollte der Mensch dem System seine Vorlieben mitteilen und auch einfache Regeln aufstellen können, wie beispielsweise „Immer wenn das TV Gerät eingeschaltet wird, bitte alle Lampen außer Lampe A ausschalten“. Eine triviale Möglichkeit dieses Problem zu lösen wäre eine lokal fest

installierte Menüführung, d.h. es befänden sich Touchpanels in den einzelnen Räumen ohne dass der Mensch selber seinen PDA dabei haben muss. Eine andere interessante Lösung wäre ein akustisches Interface, so dass Anforderungen per Sprache möglich sind. Mit einem beschränkten Vokabular ist dies heute schon ohne Training auf bestimmte Personen möglich.

Insgesamt würden diese Arbeitsgebiete darauf abzielen den Menschen besser in das technische System zu integrieren. Das liefert dann bessere Parameter für die darunter liegenden Techniken wie dem AAS, welche damit ein besseres Verhalten des Systems in Bezug auf die Bedürfnisse des Menschen erreichen können.

A. Anhang

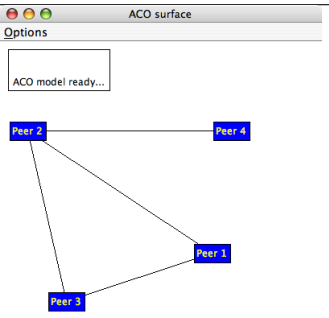
A.1. XML Schema für Toplogiebeschreibungen

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : acoNetwork.xsd
    Created on : 2. April 2006, 12:46
    Author     : sebastian
    Description:
        Define ACO Networks vor AAS simulation
-->

    <!-- <xsd:schema targetNamespace="http://www.example.com/ACO"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        xmlns:aco="http://www.example.com/ACO" > -->

 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="urn:mytest" xmlns:aco="urn:mytest" targetNamespace="urn:mytest"
elementFormDefault="qualified"> -->

     <!-- Describe toplevel element -->
     <xsd:element name="ACO-NET">
         <xsd:complexType>
             <xsd:sequence>
                 <xsd:element name="peer" type="peerType" minOccurs="1" maxOccurs="
                     unbounded"/>
                 <xsd:element name="edge" type="edgeType" minOccurs="1" maxOccurs="
                     unbounded">

                 </xsd:element>
             </xsd:sequence>

         </xsd:complexType>

         <!-- references from edge to peer -->

         <xsd:unique name="test">
             <xsd:selector xpath="peer"/>
             <xsd:field xpath="@name"/>
         </xsd:unique>

         <xsd:key name="fromKey">
             <xsd:selector xpath="peer"/>
             <xsd:field xpath="@name"/>
         </xsd:key>

         <xsd:keyref name="fromRef" refer="fromKey">
             <xsd:selector xpath="edge"/>
             <xsd:field xpath="from"/>
         </xsd:keyref>

                 <xsd:keyref name="toRef" refer="fromKey">
             <xsd:selector xpath="edge"/>
             <xsd:field xpath="to"/>
         </xsd:keyref>


     </xsd:element>


     <!-- Describe types -->
     <xsd:complexType name="peerType">
         <xsd:sequence>
             <!-- position is optional, if not give a random value is generated -->
             <xsd:element name="position" minOccurs="0" maxOccurs="1">
                 <xsd:complexType>
```

```
                    <xsd:attribute name="posX" use="required" type="xsd:int"/>
                    <xsd:attribute name="posY" use="required" type="xsd:int"/>
                </xsd:complexType>
            </xsd:element>
        <!-- The active & passive power a peer has. Optional, if not given 0 is assumed -->
            <xsd:element name="energy" minOccurs="0" maxOccurs="1">
                <xsd:complexType>
                    <xsd:attribute name="active" use="required" type="xsd:int"/>
                    <xsd:attribute name="passive" use="required" type="xsd:int"/>
                </xsd:complexType>
            </xsd:element>

        </xsd:sequence>
        <xsd:attribute name="name" use="required" type="xsd:string"/>
        <xsd:attribute name="priority" use="optional" type="xsd:int"/>


        <!-- <xsd:attribute ref="peerName" />-->

    </xsd:complexType>


    <xsd:complexType name="edgeType">
                <xsd:sequence>
                <xsd:element name="from" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="to" type="xsd:string" minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>

                <!--<xsd:attribute name="from" use="required"/>
        <xsd:attribute name="to" use="required"/>-->

    </xsd:complexType>


</xsd:schema>
```

A.2. Topolgiedaten

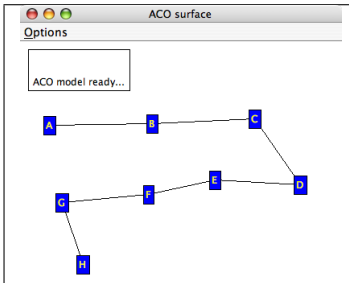
A.2.1. TestNetwork.xml

Abbildung	Beschreibung
	Einfacher Testcase aus 4 Peers

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : testnetwork.xml
    Created on : 2. April 2006, 18:58
    Author     : Sebastian Schildt
    Description:
        Test network for AAS simulation.
```

```
-->

<ACO-NET xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='acoNetwork.xsd'>
    <!-- xsi:noNamespaceSchemaLocation='file:/acoNetwork.xsd'> -->

    <peer name="Peer 1" >
        <position posX="241" posY="252" />
        <energy active="0" passive="13" />
    </peer>

    <peer name="Peer 2" >
        <position posX="12" posY="100" />
        <energy active="10" passive="20" />
    </peer>

    <peer name="Peer 3" priority="3" >
        <position posX="60" posY="312" />
        <energy active="0" passive="12" />
    </peer>

    <peer name="Peer 4" >
        <position posX="268" posY="102" />
        <energy active="0" passive="0" />
    </peer>


    <edge  >
        <from>Peer 2</from>
        <to>Peer 1</to>
    </edge>

    <edge  >
        <from>Peer 2</from>
        <to>Peer 3</to>
    </edge>

    <edge  >
        <from>Peer 1</from>
        <to>Peer 3</to>
    </edge>

    <edge  >
        <from>Peer 2</from>
        <to>Peer 4</to>
    </edge>


</ACO-NET>
```

A.2.2. LongLine.xml

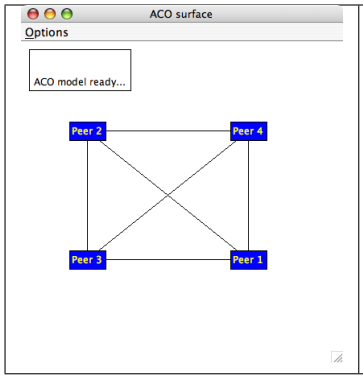
Abbildung	Beschreibung
	Beispiel für eine „schlechte“ Topologie. Vernetzung der Peers untereinander mangelhaft

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : longLine.xml
    Created on : 12. Juni 2006, 18:58
    Author     : Sebastian Schildt
    Description:
        Test case for ACO, purely artificial topology: One long
            line
-->

<ACO-NET xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='acoNetwork.xsd'>


    <peer name="D" >
        <position posX="340" posY="167" />
        <energy active="0" passive="0" />
    </peer>

    <peer name="A" >
        <position posX="29" posY="93" />
        <energy active="0" passive="0" />
    </peer>

    <peer name="F" priority="10" >
        <position posX="153" posY="179" />
        <energy active="0" passive="0" />
    </peer>

    <peer name="H" >
        <position posX="70" posY="266" />
```

```
        <energy active="0" passive="4" />
    </peer>

    <peer name="C" >
        <position posX="284" posY="85" />
        <energy active="0" passive="0" />
    </peer>

    <peer name="B" >
        <position posX="157" posY="91" />
        <energy active="0" passive="0" />
    </peer>

    <peer name="G" >
        <position posX="44" posY="189" />
        <energy active="10" passive="3" />
    </peer>

    <peer name="E" >
        <position posX="235" posY="161" />
        <energy active="0" passive="0" />
    </peer>



    <edge  >
        <from>A</from>
        <to>B</to>
    </edge>

    <edge  >
        <from>B</from>
        <to>C</to>
    </edge>

    <edge  >
        <from>C</from>
        <to>D</to>
    </edge>

    <edge  >
        <from>D</from>
        <to>E</to>
    </edge>

    <edge  >
        <from>E</from>
        <to>F</to>
    </edge>
```

```
    <edge  >
        <from>F</from>
        <to>G</to>
    </edge>

    <edge  >
        <from>G</from>
        <to>H</to>
    </edge>

</ACO-NET>
```

A.2.3. FullyConnected.xml

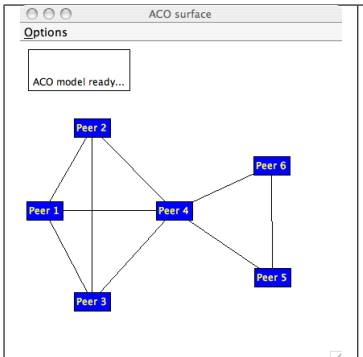
Abbildung	Beschreibung
	Maximal vermaschtes Netzwerk

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : fullyConnected.xml
    Created on : 2. April 2006, 18:58
    Author     : Sebastian Schildt
    Description:
        Test network for AAS simulation.
        All nodes connected, fully transitive
-->

<ACO-NET xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='acoNetwork.xsd'>
    <!-- xsi:noNamespaceSchemaLocation='file:/acoNetwork.xsd'> -->

    <peer name="Peer 1" >
        <position posX="260" posY="260" />
        <energy active="0" passive="13" />
    </peer>

```

```
    <peer name="Peer 2" >
        <position posX="60" posY="100" />
        <energy active="10" passive="20" />
    </peer>

    <peer name="Peer 3" >
        <position posX="60" posY="260" />
        <energy active="0" passive="12" />
    </peer>

    <peer name="Peer 4" >
        <position posX="260" posY="100" />
        <energy active="0" passive="0" />
    </peer>



    <edge  >
        <from>Peer 1</from>
        <to>Peer 2</to>
    </edge>

    <edge  >
        <from>Peer 1</from>
        <to>Peer 3</to>
    </edge>

    <edge  >
        <from>Peer 1</from>
        <to>Peer 4</to>
    </edge>

    <edge  >
        <from>Peer 2</from>
        <to>Peer 3</to>
    </edge>

    <edge  >
        <from>Peer 2</from>
        <to>Peer 4</to>
    </edge>

    <edge  >
        <from>Peer 3</from>
        <to>Peer 4</to>
    </edge>
```

```
</ACO-NET>
```

A.2.4. Bridge.xml

Abbildung	Beschreibung
	2 Netzsegmente die über einen zentralen Peer verbunden sind.

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
    Document   : bridge.xml
    Created on : 2. April 2006, 18:58
    Author     : Sebastian Schildt
    Description:
        Test network for AAS simulation.
        To seperate segments connected via Peer 4
-->

<ACO-NET xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xsi:noNamespaceSchemaLocation='acoNetwork.xsd'>
    <!-- xsi:noNamespaceSchemaLocation='file:/acoNetwork.xsd'> -->


    <peer name="Peer 1" >
        <position posX="8" posY="199" />
        <energy active="0" passive="13" />
    </peer>

    <peer name="Peer 2" >
        <position posX="67" posY="96" />
        <energy active="100" passive="0" />
    </peer>

    <peer name="Peer 3" >
        <position posX="67" posY="312" />
        <energy active="0" passive="0" />
```

```
30      </peer>

        <peer name="Peer 4" >
            <position posX="169" posY="199" />
            <energy active="0" passive="0" />
35      </peer>

        <peer name="Peer 5" >
            <position posX="291" posY="282" />
            <energy active="0" passive="100" />
40      </peer>

        <peer name="Peer 6" >
            <position posX="290" posY="143" />
            <energy active="0" passive="100" />
45      </peer>




50      <edge  >
            <from>Peer 1</from>
            <to>Peer 2</to>
        </edge>

55      <edge  >
            <from>Peer 1</from>
            <to>Peer 2</to>
        </edge>

60      <edge  >
            <from>Peer 1</from>
            <to>Peer 3</to>
        </edge>

65      <edge  >
            <from>Peer 1</from>
            <to>Peer 4</to>
        </edge>

70      <edge  >
            <from>Peer 2</from>
            <to>Peer 3</to>
        </edge>

75      <edge  >
            <from>Peer 2</from>
            <to>Peer 4</to>
        </edge>
```

```
    <edge >
        <from>Peer 3</from>
        <to>Peer 4</to>
    </edge>

    <edge >
        <from>Peer 4</from>
        <to>Peer 5</to>
    </edge>

    <edge >
        <from>Peer 4</from>
        <to>Peer 6</to>
    </edge>

    <edge >
        <from>Peer 5</from>
        <to>Peer 6</to>
    </edge>


</ACO-NET>
```

A.3. Erklärung zur Diplomarbeit

Name : Schildt
Vorname : Sebastian
Matr.-Nr. : 1155317
Studiengang : Ingenieur-Informatik

An den Prüfungsausschuss
des Fachbereichs Automatisierungstechnik
der Universität Lüneburg
Volgershall 1
21339 Lüneburg

Erklärung zur Diplomarbeit
Ich versichere, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Lüneburg, den

Abbildungsverzeichnis

2.1. phyCore XC167 5
2.2. Relais auf einem Phytec Expansionboard 6
2.3. WireShark Analyser mit L3 Plugin 8
2.4. iPaq HX2790 9
2.5. L3 Nachrichtenformat 11
2.6. L3 Architektur Übersicht[21] 11

3.1. Ameisen auf Nahrungssuche 18
3.2. Minimal Beispiel 20
3.3. Auswahl der nächsten Stadt 24
3.4. TSP Instanz 25
3.5. Auswahl nach Wahrscheinlichkeiten 27

4.1. AAS Beispielszenario 42

5.1. Systemaufbau 45
5.2. Simulationsparameter 53
5.3. Netzwerktopologie Visualisierung 54
5.4. Peer Inspector 54
5.5. AAS Custom Actions 55
5.6. ACO Optimierung initialisieren 55
5.7. Repast Konsole während der Optimierung 56
5.8. Gesamtenergie Grafik 56
5.9. Datenstrukturen der Simulation - Klassendiagramm 60
5.10. EePDA Anwendung 64
5.11. ANT Forward Aktivitätsdiagramm 66
5.12. ANT Backward Aktivitätsdiagramm 68

Literaturverzeichnis

[1] Inernational Energy Agency. Key World Energy Statistics, 2005.

[2] F. Ingrand M.O. Killijian D. Powell B. Lussier, R. Chatila. On fault tolerance and robustness in autonomous systems. In Proceedings of the 3rd IARP-IEEE/RAS-EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments, Manchester (GB), 7-9 September 2004, 2004.

[3] Yvonne Bernard. *Implementierung einer Testumgebung mit Emergenzeigenschaften anhand eines agentenbasierten Simulators*. Universität Hannover, 2005.

[4] Bernd Bullnheimer and Christine Strauss Richard F. Hartl. A new rank based version of the ant system - a computational study, 1997.

[5] Scott Camazine, Jean-Louis Deneubourg, Nigel R. Franks, James Sneyd, Guy Theraulaz, and Eric Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2003.

[6] Marco Dorigo, Christian Blum, and Andrea Roli. HC-ACO The Hyper-Cube Framework for Ant Colony Optimization, 2001.

[7] Marco Dorigo and Gianni Di Caro. Ant Colony Optimization: A New Meta-Heuristic. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1470–1477, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

[8] Marco Dorigo, Gianni Di Caro, and Luca Maria Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5:137–172, 1999.

[9] Marco Dorigo and Luca Maria Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[10] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy, 1999.

[11] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

[12] World Commission On Environment and Development. *Our Common Future*. Oxford University Press, 1987.

[13] Pierre-Paul Grassé. La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie: Essai d'interprétation du comportement des Termites constructeurs. *Insectes Sociaux*, 6:41–83, 1959.

[14] John Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[15] IEEE. *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003.

[16] Infineon Technologies. *XC16x Family - Migration from C161/C164/C167 Microcontrollers to XC161/XC164/ XC167*, 12 2003.

[17] ISO/IEC. Iso/iec 7498-1: Information technology - open systems interconnection - basic reference model: The basic model, 1994.

[18] Vittorio Maniezzo. Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem.

[19] Steven F. Railsback, Steven L. Lytinen, and Stephen K. Jackson. Agent-based Simulation Platforms: Review and Development Recommendations.

[20] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann Verlag, 1973.

[21] Gideon Zenz Sebastian Schildt. *L3 Chat - Eine Beispielapplikation für das L3 Netzwerk.* Universität Lüneburg, 2005.

[22] Gideon Zenz Sebastian Schildt. *L3-Net Peer2Peer Protokoll - Protokollspezifikation Version 0.1.6a.* Universität Lüneburg, 2005.

[23] Thomas Stützle and Holger H. Hoos. MAX-MIN Ant System. *Journal of Future Generation Computer Systems*, 16:889–914, 2000.

[24] Eric van der Vlist. *XML Schema.* O'Reilly, 2002.

[25] Ralph Welge. Sensor networking with L3-NET - A SELF-X Middleware based on standard TCP/IP protocols. Embeded World Conference 2006.

[26] Ralph Welge. *SDL.RT-basierter Entwurf und Implementierung eingebetteter zeit- und sicherheitskritischer Systeme.* Shaker Verlag, 2001.

[27] ZigBee Alliance. *ZigBee Specification*, 2005.