

Ameisenalgorithmen und ihre Anwendung auf das RCPSP

Seminarausarbeitung

Sven Albrecht

29. November 2007

Abstract

Die vorliegende Seminararbeit wurde im Rahmen des Seminars „Complex Scheduling“ im Wintersemester 2007 / 2008 erstellt. Der Inhalt dieser Arbeit richtet sich an Leser, die bereits über Grundkenntnisse in diesem Bereich verfügen. Insbesondere Kenntnisse über das RCPSP werden vorausgesetzt. Um Kenntnisse über das RCPSP gegebenenfalls aufzufrischen wird an dieser Stelle auf Brucker und Knust [1] verwiesen.

Inhaltsverzeichnis

1	Einleitung	4
1.1	Biologischer Hintergrund	4
2	Ant Colony Metaheuristik	7
2.1	Formalisierung des Problems	7
2.2	Generelle Eigenschaften von ACO	8
3	Anwendung auf das RCPSP	10
3.1	Entscheidungen für Ameisen	11
3.2	Pheromonupdate	12
3.3	Heuristiken	13
3.4	Optionale Modifikationen	14
3.4.1	Kombination von direkter und summierter Auswertung	15
3.4.2	Parameteränderung während der Laufzeit	15
3.4.3	Entkommen aus lokalen Minima	16
3.4.4	Lokale Suche	16
3.4.5	Bidirektionale Planung	16
4	Ergebnisse und Diskussion	17
4.1	Ergebnisse	17
4.2	Diskussion	18
	Literatur	20

1 Einleitung

Als Ameisenalgorithmen bezeichnet man eine Metaheuristik zur Lösung komplexer Probleme, die sich sowohl mit ihrem Namen, als auch mit ihrer Lösungsstrategie an ihrem Vorbild aus der Natur orientieren. Sie gehen auf Marco Dorigo zurück, der 1992 in seiner Ph.D. Thesis „Optimization, Learning and Natural Algorithms“, als erster Strategien von Ameisen bei der Futterbeschaffung als Algorithmus formalisierte und auf andere Probleme übertrug.

Der folgende Abschnitt erläutert die biologische Motivation, die hinter den Prinzipien von Ameisenalgorithmen steht. Leser, die bereits über entsprechendes Hintergrundwissen verfügen oder lediglich an der Formalisierung von Ameisenalgorithmen interessiert sind, können diesen Teil überspringen und mit Kapitel 2 fortfahren.

1.1 Biologischer Hintergrund

Bestimmte Ameisenarten orientieren sich bei der Futtersuche anhand von chemischen Botenstoffen, den sogenannten *Pheromonen*. Pheromone werden von einzelnen Ameisen während ihrer Fortbewegung auf dem Boden abgesondert. Andere Ameisen können die hinterlegten Pheromone wahrnehmen und präferieren Wege, auf denen viele Pheromone abgelagert sind, gegenüber Wegen, die über weniger oder keine Pheromoninformationen verfügen. In einigen Experimenten [5] an echten Ameisen wurde der Nutzen dieser Art von Kommunikation demonstriert: Eine Ameisenkolonie wurde mit einer Futterquelle über verschiedene, unterschiedlich lange Wege verbunden. Werden zunächst alle Wege, die Kolonie und Futterquelle verbinden, von Ameisen frequentiert, lässt sich nach einer gewissen Explorationszeit eine deutliche Präferenz des kürzesten Weges erkennen. Dieses Verhalten wird durch die Pheromone gesteuert. Auf dem kürzesten Pfad können innerhalb des gleichen Zeitraums mehr Ameisen zur Futterquelle und zum Nest zurück laufen, als auf den längeren Pfaden. Somit können auf dem kürzeren Weg mehr Pheromone abgelagert werden, als auf den längeren Wegen. Die stärkeren Pheromonablagerungen sorgen schließlich dafür, dass weitere, von der Kolonie aufbrechende, Ameisen mit einer höheren Wahrscheinlichkeit diesen günstigsten Pfad wählen. Verstärkt wird dieser Effekt zusätzlich durch die Evaporation der Pheromone im Laufe der Zeit. Wird ein (ungünstiger) Pfad längere Zeit nicht mehr von Ameisen frequentiert, so weist er schließlich keine Pheromonablagerungen mehr auf. Dadurch hat er für die Ameisen auch keine zusätzliche Anziehungskraft mehr. In Abbildung 1, Seite 5 wird der Ablauf dieses Verhaltens veranschaulicht.

Zu beachten ist außerdem, dass echte Ameisen robust auf Störungen reagieren. Falls ein gefundener kürzester Pfad blockiert wird, gelingt es den Ameisen, nach kurzer Zeit, sich an die neue Situation anzupassen und eine möglichst gute Alternativroute zu finden. Diese Eigenschaft wird auch von Ameisenalgorithmen übernommen. Sowohl die Algorithmen, als auch das biologische Vorbild können keine optimale Lösung garantieren. In beiden Fällen wird jedoch meist ein gutes lokales Minimum erreicht.

Im nun folgenden Kapitel 2 wird die Übertragung des natürlichen Vorbilds auf eine Metaheuristik für Optimierungsprobleme beschrieben. Falls der Leser mit den Forma-

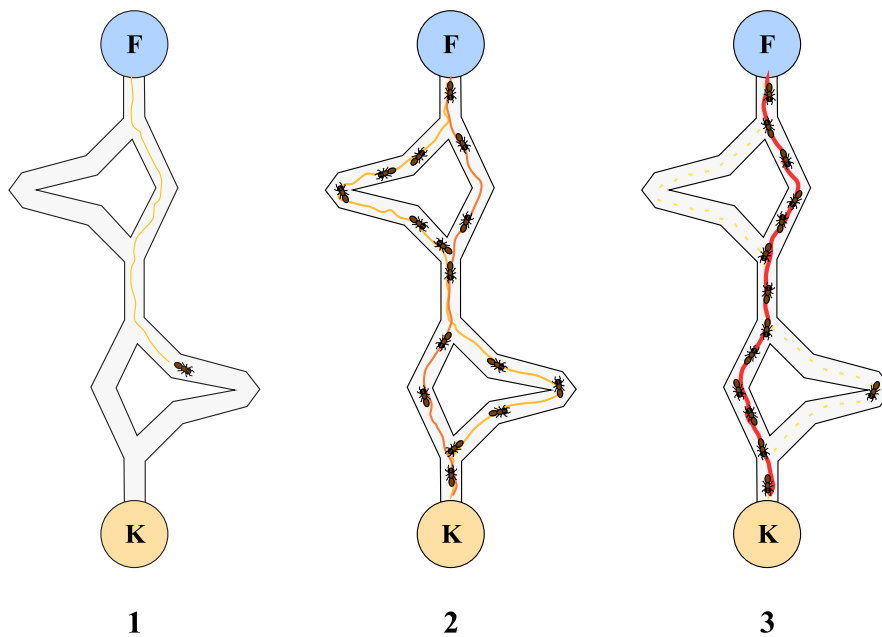


Abbildung 1: Veranschaulichung der Futtersuche von Ameisen. Die Futterquelle ist mit F markiert, die Kolonie mit K. Pheromone werden farblich auf den Wegen eingezeichnet. Je rötlicher und dicker die Pheromone dargestellt sind, desto stärker sind sie.

Phase 1 zeigt eine einzelne Ameise, die die Futterquelle entdeckt hat.

Phase 2 veranschaulicht die Explorationsphase, auf der alle möglichen Pfade frequentiert werden.

Phase 3 zeigt die Wege der Ameisen nach Konvergenz.

lismen und Notationen von Ameisenalgorithmen vertraut sein sollte, kann dieser Teil übersprungen werden und direkt zur Formulierung eines Ameisenalgorithmus für das RCPSP in Kapitel 3 übergegangen werden.

2 Ant Colony Metaheuristik

Die sogenannte *Ant Colony Metaheuristik*, im Folgenden als ACO bezeichnet, geht auf Marco Dorigo zurück. Die hier verwendeten Notationen sind an seine Notationen aus [4] angelehnt. Dort lässt sich auch eine ausführlichere Beschreibung der ACO finden, in der unter anderem eine Lösung des *Travelling Salesman Problem* (TSP) als Beispiel präsentiert wird.

2.1 Formalisierung des Problems

Um die generelle Idee der Ameisenalgorithmen auf ein Problem anwenden zu können, muss das Problem entsprechend formuliert werden. Im Folgenden werden einige grundsätzliche Notationen vereinbart, die so, oder in ähnlicher Form, in jeder Implementierung von ACO auftauchen. Die, von Dorigo verwendeten englischen Bezeichnungen werden jeweils in Klammern aufgeführt:

- Eine endliche Knotenmenge (oft auch bezeichnet als *components*)
 $C = \{c_1, c_2, \dots, c_{N_C}\}$
- Eine endliche Kantenmenge L (*transitions / connections*), auf den Elementen aus \tilde{C} . \tilde{C} sei als Teilmenge des Kartesischen Produkts $C \times C$ definiert.
 $L = \{l_{c_i c_j} \mid (c_i, c_j) \in \tilde{C}\}$
- Eine (lokale) Kostenfunktion $J_{c_i c_j}$ (*connection cost*) für jede Kante $l_{c_i c_j}$. Je nach Problemstellung kann die Zeit ein weiterer Parameter in der Kostenfunktion sein.
 $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$
- Eine endliche Menge an Nebenbedingungen Ω (*constraints*), geknüpft an die Elemente aus C und L .
 $\Omega \equiv \Omega(C, L, t)$
- Ein Zustand s (*state*) ist definiert, als eine Abfolge von Knoten aus C .
 $s = \langle c_i, c_j, \dots, c_k, \dots \rangle$
Sei S die Menge aller Zustände, so wird die Menge der gültigen Zustände (*feasible states*), unter Berücksichtigung von $\Omega(C, L, t)$, als \tilde{S} bezeichnet. Offensichtlich gilt $\tilde{S} \subseteq S$. Die Länge der Abfolge eines Zustandes wird durch $|s|$ angegeben.
- Als Nachbarschaft (*neighborhood structure*) zwischen zwei Zuständen $s_1 = \langle c_i, \dots, c_{s1} \rangle$ und $s_2 = \langle s_1, c_{s2} \rangle$, $s_1, s_2 \in S$ bezeichnet man den Fall, wenn s_2 eine Erweiterung von s_1 um einen Knoten $c_{s2} \in C$ ist und eine Kante $l_{c_{s1} c_{s2}} \in L$ zwischen dem letzten Knoten in s_1 und dem hinzugefügten Knoten besteht.
- Eine Lösung Ψ (*solution*) besteht aus einem gültigen Zustand $s \in \tilde{S}$ der alle Anforderungen des Problems erfüllt. Eine Lösung bezeichnet man als *mehrdimensional*, wenn sie über mehrere unterschiedliche Zustände realisiert werden kann.

- Es existieren Pfadkosten (*cost*) $J_\Psi(L, t)$ für jede Lösung Ψ . Hierbei ist $J_\Psi(L, t)$ eine Funktion über alle lokalen Kosten $J_{c_i c_j}$, der Kanten, die in der Lösung Ψ enthalten sind.

Ist ein Optimierungsproblem auf einen Graph $G = (C, L)$ abgebildet, so versucht die ACO Pfade in diesem Graph zu finden, die Lösungen Ψ entsprechen. Wann ein Pfad eine gültige Lösung darstellt, ist problemabhängig. Beispielsweise wäre eine gültige Lösung für das TSP ein Hamiltonkreis im Graph, also ein Pfad, bei dem jeder Knoten des Graphen genau einmal besucht wird. Bei der Pfadgenerierung wird versucht die Kosten $J_\Psi(L, t)$ zu minimieren, unter Berücksichtigung aller Nebenbedingungen Ω .

Pfade werden von künstlichen Ameisen iterativ generiert. Beginnend von einem Startknoten wird der Graph so lange traversiert, bis eine gültige Lösung erstellt ist. Einzelne Ameisen sind in der Lage gültige Lösungen zu erstellen und eine Lösung mittels der Pfadkosten zu bewerten. Um gezielt möglichst günstige Lösungen zu finden ist jedoch ein Informationsaustausch zwischen den Ameisen nötig. Der Informationsaustausch findet, ähnlich wie im natürlichen Vorbild, indirekt über eine Veränderung der Umgebung statt: In jedem Knoten $c_i \in C$ werden künstliche *Pheromone* τ_{ij} hinterlegt. Diese Pheromone τ_{ij} sind jeweils mit den ausgehenden Kanten $l_{c_i c_j}$, im Folgenden abgekürzt durch l_{ij} , des Knotens c_i assoziiert. Je höher der Pheromonwert τ_{ij} für eine Kante l_{ij} , desto attraktiver wird es für eine Ameise diese Kante als nächstes zu benutzen.

2.2 Generelle Eigenschaften von ACO

ACO und davon abgeleitete Algorithmen zeichnen sich durch einige generelle Eigenschaften aus. Die wichtigsten sind hier kurz skizziert:

- Einzelne Ameisen können inkrementell Lösungen generieren. Gute Lösungen entstehen im Allgemeinen jedoch erst durch Informationsaustausch zwischen Ameisen. Informationsaustausch wird über Pheromone realisiert.
- Ameisen greifen nur auf ihre eigenen und lokal, im gerade besuchten Knoten, abgelegte Informationen zurück.
- Ameisen selbst sind nicht adaptiv. Ihr Verhalten kann durch lokale Modifikationen der Pheromonwerte in einzelnen Knoten beeinflusst werden.

Einzelne Ameisen müssen ebenfalls über bestimmte Eigenschaften verfügen, damit die ACO verlässliche Ergebnisse liefert. Auch diese sind hier kurz aufgeführt:

- Ameisen suchen nach Lösungen mit minimalen Kosten $\hat{J}_\Psi = \min_\Psi J_\Psi(L, t)$
- Jede Ameise k besitzt ein Gedächtnis M^k . Dies wird benutzt um den aktuellen Zustand der Ameise festzuhalten. Über den Zustand kann der bisher zurückgelegte Pfad der Ameise ermittelt werden. Auf diese Weise kann stets auf Zulässigkeit geprüft werden, eine gefundene Lösung bewertet werden und, falls nötig, der Pfad zurückverfolgt werden.

- Eine Ameise k kann einen bestimmten Startzustand s_s^k besitzen und eine oder mehrere Endbedingungen.
- Eine Ameise k in Zustand $s_r = \langle s_{r-1}, i \rangle$ kann jeden Knoten in ihrer zulässigen Nachbarschaft N_i^k besuchen. Es gilt: $N_i^k = \{j \mid (j \in N_i) \wedge (\langle s_r, j \rangle \in \tilde{S})\}$ mit $N_i = \{c_j \mid l_{ij} \in L\}$
- Eine Ameise k in Knoten i kann sich zu jedem Knoten $j \in N_i^k$ bewegen. Die Wahl des nächsten Knotens wird probabilistisch getroffen.
- Die probabilistische Auswahl wird aufgrund einer lokal in Knoten i abgelegten Datenstruktur $\mathcal{A}_i = [a_{ij}]$ getroffen. In \mathcal{A}_i sind Pheromonwerte und heuristische Werte für jede ausgehende Kante l_{ij} abgelegt. Zusätzlich fließen dem Gedächtnis M^k und die Nebenbedingungen Ω in den Auswahlprozess ein.
- Wird von Knoten i als nächstes ein Nachbarknoten j besucht, so kann eine Ameise den Pheromonwert τ_{ij} aktualisieren.
- Ist eine Lösung erstellt worden, kann eine Ameise mit ihrem Gedächtnis M^k ihren zurückgelegten Pfad zurückverfolgen und dabei die Pheromonwerte aktualisieren.
- Von den zwei gerade vorgestellten Arten des Pheromonupdates sollte mindestens eine Art implementiert werden. Es können aber auch durchaus beide Methoden in einer Implementation zusammen verwendet werden.
- Sobald eine Ameise eine Lösung erstellt hat und evtl. ihren Pfad zurückverfolgt hat, stirbt sie und alle von ihr benutzten Ressourcen können freigegeben werden.

3 Anwendung auf das RCPSP

Um die im vorangegangenen Kapitel 2 vorgestellten Methoden auf eine Instanz des RCPSP anwenden zu können, muss das RCPSP zunächst in einer geeigneten Weise repräsentiert werden. Der von Merkle u.a. in [8] vorgestellte Ansatz nutzt die ACO Metaheuristik um Listen von Aktivitäten zu generieren, aus denen anschließend von Plangenerierungsschemata konkreten Pläne erstellt werden. Die in [8] verwendeten Schemata waren das PSGS und das SSGS. Informationen zu Plangenerierungsschemata sind in [1], Kapitel 3, Abschnitt 3.2 zu finden.

Um eine Liste von Aktivitäten für eine gegebene Instanz des RCPSP mittels ACO zu erstellen, muss die Instanz in einem für ACO geeigneten Graph abgebildet werden. Merkle u.a. [8] bilden dazu die Aktivitäten auf die Knotenmenge C ab. Innerhalb der Knotenmenge existiert ein zusätzlich generierter Startknoten c_0 und ein Endknoten c_{n+1} bei n gegebenen Aktivitäten. Von einem Knoten c_i gehen Kanten zu allen möglichen Nachfolgern aus. Mögliche Nachfolger eines Knotens c_i sind Knoten c_j , bei deren Aktivitäten eine Vorangsbeziehung der Art $i \rightarrow j$ besteht, und die Menge aller Knoten, die bei denen keine Vorangsbeziehung, auch keine indirekte, zu Aktivität i besteht. Von allen verfügbaren Kanten eines Knotens werden, wenn der Knoten von einer Ameise k besucht wird, in der Regel nicht alle zulässig sein. Über die Nebenbedingungen $\Omega(C, L, t)$ werden jene Kanten l_{ij} ausgeschlossen, bei denen für die Knoten c_j noch nicht alle Vorangsbeziehungen erfüllt sind. Über das Gedächtnis M^k der Ameise werden zudem alle Kanten ausgeschlossen, die zu einem Knoten c_j führen, der bereits besucht wurde. Die entsprechende Aktivität könnte ansonsten zweimal in die Liste der Aktivitäten eingeplant werden. Jeder von einer Ameise besuchter Knoten wird jeweils an das Ende der Aktivitätenliste eingefügt. Die momentane Liste entspricht also dem momentanen Zustand. Pheromonwerte werden, im Gegensatz zur eigentlichen ACO nicht in den einzelnen Knoten hinterlegt, sondern in einer sogenannten *Pheromonmatrix*. Die Pheromonmatrix ist eine $(n+2) \times (n+2)$ Matrix, falls es n Aktivitäten gibt. Pheromonwerte τ_{ij} geben an, wie günstig es zu sein scheint, Aktivität j an Stelle i in der Aktivitätenliste einzufügen. Als Abbruchkriterium des Algorithmus wird entweder eine feste Anzahl von Generationen vorgeschlagen oder wenn die durchschnittlichen Lösungen sich über eine bestimmte Anzahl von Generationen nicht mehr verändert haben.

Alle Ameisen starten bei dem Startknoten c_0 und durchlaufen den Graphen so lange von Knoten zu Knoten, bis sie zum Endknoten c_{n+1} gelangen. Unter den oben beschriebenen Voraussetzungen entsteht so eine zulässige Liste von Aktivitäten aus der mit Hilfe des SSGS oder des PSGS ein zulässiger Plan generiert werden kann. Hierbei ist zu beachten, dass nach der obigen Notation der Knoten c_{n+1} erst erreicht werden kann, wenn alle seine Vorangsbeziehungen erfüllt sind, das heißt, wenn alle anderen Aktivitäten bereits in die Liste eingefügt wurden.

Die Entscheidung welcher Knoten aus der Menge der zulässigen Knoten von einer Ameise als nächstes besucht wird, wird probabilistisch getroffen. Die Wahrscheinlichkeiten werden hier einerseits von den Pheromonwerten τ_{ij} der Pheromonmatrix als auch von heuristischen Werten η_{ij} beeinflusst.

Nachdem nun die generelle Idee der Anwendung von ACO zur Lösung einer RCPSP

Instanz beschrieben wurden folgen nun eine algorithmische Details. Abschnitt 3.1 beschreibt die Probabilistische Auswahl des nächsten Knotens. In Abschnitt 3.2 wird erläutert, wie das Pheromonupdate vorgenommen wird, während in 3.3 auf die Adaption bekannter Heuristiken für ACO eingegangen wird. In Abschnitt 3.4 werden schließlich noch einige optionale Modifikationen vorgestellt.

3.1 Entscheidungen für Ameisen

Eine Ameise, die sich in einem Knoten c_i ungleich dem Endknoten befindet, muss entscheiden welcher Knoten aus der Menge der zulässigen Knoten als nächster besucht werden soll. Diese Entscheidung wird probabilistisch getroffen und von zwei verschiedenen Größen beeinflusst. Die Menge an Pheromonen τ_{ij} in der Pheromonmatrix, gibt Auskunft darüber, wie häufig Aktivität j an Stelle i der Aktivitätenliste von vorangegangenen Ameisen platziert wurde.

für eine Kante l_{ij} von Knoten c_i zu einem Knoten c_j geben Auskunft darüber, wie häufig diese Kante schon von vorangegangenen Ameisen benutzt wurde. Je höher der Pheromonwerte τ_{ij} , desto wahrscheinlicher scheint Aktivität j an Position i zu einer vorteilhaften Lösung zu gehören. Details zur Verteilung von Pheromonwerten für die Kantenmenge L des Graphen werden in Abschnitt 3.2 behandelt. Die zweite Größe, welche die Auswahl des nächsten Knotens beeinflusst ist der heuristische Wert η_{ij} . In den heuristischen Werten wird versucht Vorwissen über die Problemstellung zu nutzen, um einen möglichst gute Position zu finden. Im Gegensatz zu den Pheromonwerten ändern sich die heuristischen Werte η_{ij} nicht während der Iterationen des Algorithmus. Verschiedene Heuristiken werden in Abschnitt 3.3 diskutiert. Merkle u.a. stellen in [8] zwei verschiedene Arten vor, um die Wahrscheinlichkeiten für den nächsten in die Aktivitätenliste einzufügenden Knoten zu ermitteln. Die Menge der momentan zulässigen Aktivitäten (und damit auch Knoten) sei mit \mathcal{E} bezeichnet. Die sogenannte direkte Auswertung setzt die Wahrscheinlichkeit p_{ij} den Knoten j an Stelle i einzufügen folgendermaßen fest:

$$p_{ij} = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h \in \mathcal{E}} [\tau_{ih}]^\alpha [\eta_{ih}]^\beta} \quad (1)$$

Über die Parameter $\alpha, \beta \in \mathbb{R}, \alpha, \beta \geq 0$ wird der relative Einfluss von von Pheromonwerten und heuristischen Werten bestimmt. In Formel (1) werden ausschließlich die direkt in der Pheromonmatrix und die heuristischen Werte benutzt, um die Wahrscheinlichkeiten für die Position i zu bestimmen. Von den auswählbaren Knoten bekommt der Knoten die größte Wahrscheinlichkeit, bei dem das Produkt aus relativem Pheromonwert und relativem heuristischem Wert am größten ist. Eine Alternative Möglichkeit zur Bestimmung der Wahrscheinlichkeiten ist die summierte Auswertung. Hier werden die Pheromonwerte, für das Einfügen eines Knotens j zu einem früheren Zeitpunkt miteinbezogen. So können Aktivitäten bevorzugt werden, bei denen die Pheromonwerte bereits zu für einen früheren Platz der Aktivitätenliste groß waren.

$$p_{ij} = \frac{(\sum_{k=1}^i [\gamma^{i-k} \cdot \tau_{ij}])^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{h \in \mathcal{E}} (\sum_{k=1}^i [\gamma^{i-k} \cdot \tau_{ih}])^\alpha \cdot [\eta_{ih}]^\beta} \quad (2)$$

Die Parameter α und β entsprechen den Parametern aus Formel (1). Der Parameter $\gamma \in \mathbb{R}, \gamma > 0$ bestimmt, wie Pheromonwerte früherer Entscheidungen den relativen Wert der Pheromone beeinflussen. Werte $\gamma < 1$ verleihen den Pheromonen weiter zurückliegender Entscheidungen weniger Gewicht, $\gamma > 1$ stärken den Einfluss der Pheromone früherer Entscheidungen. Für $\gamma = 1$ werden alle Pheromonwerte $\tau_{kj}, k \in \{1, 2, \dots, i\}$ gleich stark gewertet.

3.2 Pheromonupdate

Da die Pheromonwerte τ_{ij} eine große Rolle in der Auswahl des nächsten Knotens spielen (vgl. Formel (1) und (2)), beeinflusst die Wahl, wie Pheromone verteilt werden das Verhalten des Algorithmus entsprechend. Zu Beginn des Algorithmus wird jeder Pheromonwert τ_{ij} mit dem gleichen Startwert $\tau_0 > 0$ versehen. Diese Initialisierung ist notwendig, um zu Beginn des Algorithmus in der Lage zu sein, nach den gegebenen Auswertungsmethoden die Wahrscheinlichkeiten für eine Entscheidung zu bestimmen. Da $[\tau_{ij}]^\alpha$ als Faktor einfließt gilt sowohl nach (1) als auch nach (2) $p_{ij} = 0$ für $\tau_{ij} = 0$. Sind die Pheromone initialisiert, so werden alle Pheromone nach einer Generation gemäß folgender Formel aktualisiert:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} \quad (3)$$

Der Parameter ρ mit $\rho \in \mathbb{R}, 0 \leq \rho < 1$ bestimmt, wie schnell abgelegte Pheromone evaporieren. Dies wird vorgenommen, damit alte Pheromone keinen zu starken Einfluss entwickeln und so das Auffinden anderer Lösungen behindern. Anschließend werden zusätzliche Pheromone für die beste innerhalb dieser Generation gefundene Lösung Ψ^* verteilt. Die Lösung Ψ^* entspricht einem Zustand $\langle c_0, c_i, \dots, c_k, c_{n+1} \rangle$, in dem die Knoten genau in der Reihenfolge vorkommen, in der sie in der Aktivitätenliste aufgeführt sind. Die Pheromone, die zu dieser Lösung gehören sind die Pheromone τ_{ij} , bei denen der Index i der Position des Knoten j in der Aktivitätenliste entspricht. Alle τ_{ij} , die dieses Kriterium erfüllen, werden folgendermaßen modifiziert:

$$\tau_{ij} = \tau_{ij} + \rho \cdot \frac{1}{2T^*} \quad (4)$$

Der Parameter T^* entspricht dem Makespan für die Lösung Ψ^* . Je besser also die Lösung ist, desto mehr werden die Pheromonwerte, die an der Lösung beteiligt sind, erhöht. Es findet also eine nachträgliche Bewertung generierter Lösungen statt und kein iteratives Verteilen von Pheromonen, während der Lösungsgenerierung (vgl. 2.2) Auf gleiche Weise werden auch die Pheromonwerte der insgesamt besten gefundenen Lösung modifiziert. Dies führt dazu, dass künftige Ameisen mit höherer Wahrscheinlichkeit in der Nähe der bisher besten Lösung nach weiteren Lösungen suchen. Eine solche Strategie kann dazu führen, dass sich der Algorithmus schnell in ein lokales Minimum bewegt und aus diesem nicht wieder herausfindet, da stets Pheromone erhöht werden, die in Richtung des lokalen

Symbol	Bedeutung
LF_i	'latest finish time' von Aktivität i
LS_i	'latest start time' von Aktivität i
ES_i	'earliest start time' von Aktivität i
S_i^*	Menge aller Nachfolger von Aktivität i
S_i	Menge der direkten Nachfolger von Aktivität i
\mathcal{E}	Menge aller auswählbaren Aktivitäten
\mathcal{Q}	Menge der Ressourcen
R_l	Kapazität von Ressource l
p_i	Dauer von Aktivität i
r_{il}	Ressourcenanforderung von Aktivität i an Ressource l

Tabelle 1: Notationen für Heuristiken

Minimums führen. In Abschnitt 3.4.3 wird eine Methode vorgestellt, mit der man dieses Risiko vermindern kann.

3.3 Heuristiken

Die hier vorgestellten Heuristiken fließen gemäß Formel (1) und (2) in die Wahrscheinlichkeiten ein, dass Aktivität j an Position i in der Aktivitätenliste auftaucht. Die heuristischen Werte η_{ij} verändern sich im Gegensatz zu den Pheromonen nicht. Ihre Aufgabe ist, die Ameisen in den ersten Iterationen in vielversprechend erscheinende Regionen zu führen. Problematisch können Heuristiken werden, wenn sie im Verhältnis zu den Pheromonen zu stark gewichtet sind und somit zu viel Einfluss auf den Weg einer Ameise ausüben. Bevor die einzelnen Heuristiken kurz vorgestellt werden, sind einige Notationen in Tabelle 1 vereinbart.

Auf die einzelnen Heuristiken soll hier nur kurz eingegangen werden. Die Bedeutung der Heuristiken lässt sich schnell mit Hilfe der entsprechenden Formel und einer kurzen Erläuterung erschließen. Relevant werden die Heuristiken im Hinblick auf die erzielten, in Abschnitt 4.1 diskutierten Ergebnisse. Weitere Heuristiken, die sich auch prinzipiell für ACO adaptieren lassen, sind in [1], Kapitel 3, Abschnitt 3.3 zu finden. Alle hier vorgestellten Heuristiken haben gemeinsam, dass sie sich aus Eigenschaften der Aktivitäten im Vergleich mit den anderen, zu diesem Zeitpunkt zulässigen, Aktivitäten ergeben. Zudem ergeben sich die Werte für die einzelnen Heuristiken stets dadurch, dass sie mit den Werten anderer verfügbarer Aktivitäten ins Verhältnis gesetzt werden. Dies spielt bei der Anwendung für ACO eine Rolle, da es hier nicht nur auf eine heuristische Reihenfolge unter den Aktivitäten ankommt, sondern die heuristischen Werte direkt über Formel (1) und (2) in den Algorithmus einfließen. Zum Beispiel wachsen meist die spätestens Endzeitpunkte für Aktivitäten die erst spät zulässig werden (die also viele Vorbedingungen haben). Dadurch, dass sie mit allen anderen zulässigen Aktivitäten ins Verhältnis gesetzt werden, lassen sich für diese Aktivitäten eher ähnliche heuristische Werte erzielen, wie für früh zulässige Aktivitäten. Mit η_{ij} wird stets der heuristische Wert bezeichnet, mit dem vorgeschlagen wird Aktivität j an Position i in der Aktivitätenliste zu platzieren.

Da die Position i nicht direkt in die Heuristik einspielt, sondern nur indirekt über die Menge der bereits zulässigen Aktivitäten \mathcal{E} , findet sich i folglich nicht auf der rechten Seite der angegebenen Formeln.

Die „Latest Finish Time“ (LFT) Heuristik bevorzugt Aktivitäten, deren spätester Endzeitpunkt im Verhältnis zu den anderen zulässigen Aktivitäten früh ist:

$$\eta_{ij} = \max_{k \in \mathcal{E}} LF_k - LF_j + 1$$

„Latest Start Time“ (LST) bevorzugt Aktivitäten deren spätester Startzeitpunkt im Verhältnis klein ist, die also früh gestartet werden müssen:

$$\eta_{ij} = \max_{k \in \mathcal{E}} LS_k - LS_j + 1$$

„Minimum Slack Time“ (MSL) versucht Aktivitäten, die ein verhältnismäßig kleinen Spielraum bezüglich ihrer Anfangszeit haben, möglichst schnell einzuplanen:

$$\eta_{ij} = \max_{k \in \mathcal{E}} (LS_k - ES_k) - (LF_j - ES_j) + 1$$

„Most Total Succesors“ (MST) berücksichtigt die Anzahl aller Nachfolger einer Aktivität:

$$\eta_{ij} = |S_j^*| - \min_{k \in \mathcal{E}} |S_k^*| + 1$$

„Greatest Rank Positional Weight All“ (GRPWA) sortiert Aktivitäten nach ihrer Dauer, sowie der Summe der Dauer aller ihrer Nachfolger:

$$\eta_{ij} = p_j + \sum_{i \in S_j^*} p_i - \min_{k \in \mathcal{E}} \left(p_k + \sum_{i \in S_k^*} p_i \right) + 1$$

„Weighted Resource Utilization and Precedence“ (WRUP) betrachtet die Ressourcenanforderungen durch eine Aktivität und die Anzahl ihrer direkten Nachfolger:

$$\eta_{ij} = \omega |S_j| + (1 - \omega) \sum_{l \in \mathcal{Q}} \frac{r_{jl}}{R_l} - \min_{k \in \mathcal{E}} \left(\omega |S_k| + (1 - \omega) \sum_{l \in \mathcal{Q}} \frac{r_{kl}}{R_l} \right) + 1 \quad \omega \in [0, 1]$$

3.4 Optionale Modifikationen

Nachdem in den vorangegangenen Abschnitten die Übertragung der generellen Methoden der ACO auf das RCPSP vorgestellt wurden, werden in diesem Abschnitt einige optionale Modifikationen vorgestellt. Mit Hilfe dieser Modifikationen lassen sich teilweise bessere Ergebnisse erzielen, jedoch sind sie keine Voraussetzungen für ein korrektes Arbeiten des Algorithmus.

3.4.1 Kombination von direkter und summierter Auswertung

Problembedingt kann sich sowohl die direkte Auswertung, als auch die summierte Auswertung (siehe 3.1) als günstig erweisen. Merkle u.a. schlagen daher in [8] eine Kombination von direkter und summierter Auswertung vor. Um dies zu erreichen schlagen sie vor, für die Berechnung der Wahrscheinlichkeiten Formel (1) zu benutzen und innerhalb der Formel den Pheromonwert τ_{ij} durch den folgenden Ausdruck τ'_{ij} zu ersetzen:

$$\tau'_{ij} = c \cdot x_i \cdot \tau_{ij} + (1 - c) \cdot y_i \cdot \sum_{k=1}^i \gamma^{i-k} \tau_{kj}$$

Mit $x_i = \sum_{h \in \mathcal{E}} \sum_{k=1}^i \gamma^{i-k} \tau_{kh}$ und $y_i = \sum_{h \in \mathcal{E}} \tau_{ih}$. Der Parameter $c \in \mathbb{R}, 0 \leq c \leq 1$ bestimmt dabei die Gewichtung von direkter zu summierter Auswertung. Die Faktoren x_i und y_i sind eingefügt, um den relativen Einfluss von direkter und summierter Auswertung anzupassen. Merkle u.a. behaupten in [8], dass für $c = 1$ eine direkte Auswertung vorliegen würde und für $c = 0$ eine summierte Auswertung. Dies wäre der Meinung des Autors nach nur korrekt, wenn man auf die Faktoren x_i und y_i in obiger Formel verzichten würde.

3.4.2 Parameteränderung während der Laufzeit

Während die in den vorigen Abschnitten erwähnten Parameter (α, β, ρ) normalerweise einmal zu Beginn des Algorithmus fest gewählt werden, wird von Merkle u.a. vorgeschlagen, einige dieser Parameter während der Laufzeit zu verändern. Der Parameter β bestimmt den Einfluss der Heuristik auf den Entscheidungsprozess einer Ameise (vgl. Formel (1) und (2)). Zu Beginn des Algorithmus ist die Heuristik in der Regel hilfreich, um gute Lösungen zu generieren. Später können zu große heuristische Werte jedoch verhindern, dass Ameisen guten Pheromonen folgen und so die Erstellung guter Lösungen behindern. Deshalb scheint es sinnvoll, mit einem recht großen Startwert für β zu beginnen und β im Verlauf des Algorithmus bis auf 0 zu verkleinern, so dass später nur noch Pheromone die Wahrscheinlichkeiten für die Ameisen beeinflussen.

Mit ρ wird die Konvergenzgeschwindigkeit des Algorithmus gesteuert. Je höher ρ , desto schneller evaporieren alte Pheromonwerte (siehe Formel (3)) und desto mehr Pheromone werden auf die derzeit besten Lösungen verteilt (Formel (4)). Der Algorithmus wird also schneller konvergieren. Kleine Werte von ρ führen dazu, dass die Konvergenzgeschwindigkeit sinkt und der Algorithmus einen größeren Teil des Suchraums explorieren kann. Falls die Anzahl von Generationen für den Algorithmus beschränkt ist, schlagen Merkle u.a. deshalb zunächst einen kleinen Startwert für ρ vor, der während dem größten Teil der Laufzeit beibehalten werden sollte. Gegen Ende der Laufzeit sollte ρ dann erhöht werden, um das Konvergenzverhalten zu verbessern. Inwieweit man ρ verändern sollte, wenn die Anzahl der Generationen nicht begrenzt ist, wird nicht weiter beschrieben.

3.4.3 Entkommen aus lokalen Minima

Nach Abschnitt 3.2 wird stets die beste, bisher gefundene Lösung mit weiteren Pheromonen versehen. Das kann dazu führen, dass die Suche zu stark auf die Nachbarschaft dieser bisher besten Lösung konzentriert wird. Besonders nachteilig ist dies, wenn in der Nähe dieser Lösung keine weiteren guten Lösungen existieren. Um einem zu frühen Konvergenz in einem lokalen Minimum vorzubeugen kann der Algorithmus so modifiziert werden, dass die bisher beste Lösung nur für eine maximale Anzahl g_{max} von Generationen mit weiteren Pheromonen versehen wird, solange keine bessere Lösung gefunden wurde. Nach g_{max} Generationen wird dann die beste Lösung innerhalb der Generation für das Update der Pheromonwerte zur besten bisher gefundenen Lösung.

3.4.4 Lokale Suche

Manchmal lassen sich die von ACO gefundenen Lösungen durch lokale Suchverfahren in der Nähe der Lösungen weiter verbessern. Als Nachbarschaften für die lokale Suche werden in Merkle u.a. [8] Rechtsshifts von Aktivitäten in der Liste der Aktivitäten oder der Tausch der Positionen von zwei Aktivitäten angegeben. Weitere Details zu diesen und anderen Nachbarschaften finden sich in [1], Kapitel 3, Abschnitt 3.4.

3.4.5 Bidirektionale Planung

Statt das Problem nur mit einer Kolonie von Ameisen zu bearbeiten, kann man teilweise bessere Lösungen erzielen, wenn man eine zweite Kolonie hinzunimmt, die das RCPSP von 'hinten nach vorne' plant. Dazu müssen alle Vorangbeziehungen umgedreht werden und die Rolle der beiden Dummy-Knoten c_0 und c_{n+1} wird vertauscht. Anschließend wird auf den beiden Problemen eine unabhängige ACO gestartet. Nach einer vorher festgelegten Anzahl von Generationen werden die bisher erzielten Ergebnisse beider Kolonien verglichen. Die Kolonie, die im Durchschnitt in den letzten Generationen bessere Ergebnisse erzielt hat, wird weitergeführt, der Optimierungsprozess der anderen Kolonie wird abgebrochen.

Heuristik	Abweichung von LB in %	Standardabweichung
LST	35.43	0.046
LFT	35.64	0.043
GRPWA	35.79	0.049
MTS	36.38	0.004
WRUP ($\omega = 0$)	39.22	0.080
WRUP ($\omega = 0.5$)	38.32	0.082
WRUP ($\omega = 1$)	38.70	0.037

Tabelle 2: Auswirkung der verschiedenen Heuristiken auf die Performanz, bei ansonsten gleichen Parametern. In der mittleren Spalte wird jeweils die durchschnittliche prozentuale Abweichung der generierten Lösungen von einer unteren Schranke LB (basierend auf kritischem Pfad) angegeben. Die rechte Spalte beschreibt die Standardabweichung der genierten Lösungen.

4 Ergebnisse und Diskussion

In diesem Kapitel soll, aus Platzgründen nur sehr knapp, auf die Ergebnisse von Merkle u.a. [8] eingegangen werden. Anschließend erfolgt in Abschnitt 4.2 eine kurze Bewertung, welche die persönliche Sicht des Autors zum Ansatz von Merkle u.a. widerspiegelt.

4.1 Ergebnisse

Um ihren Algorithmus mit anderen Verfahren zu vergleichen, greifen Merkle u.a. auf PSPLIB [6; 7] zurück. Von den dort zur Verfügung gestellten Instanzen werden die `j120.sm` Instanzen als Benchmarkset ausgewählt.

Um vergleichbare Ergebnisse zu erzielen werden von allen verwendeten Algorithmen maximal 5000 Pläne generiert. Die von Merkle u.a. implementierte ACO Variante zeichnet sich durch folgende Eigenschaften aus: Pro Generation werden 5 Ameisen verwendet. Es werden maximal 850 Generationen durchlaufen, während der ersten 100 Generationen wird bidirektional geplant (vgl. 3.4.5). Nach 850 Generationen oder wenn die durchschnittliche Lösung pro Generation sich 10 Generationen lang nicht verändert hat, geht der Algorithmus zu einer lokalen Suche über, bis die Maximalanzahl von 5000 Plänen erreicht ist.

Folgende Parameterkonfiguration wurde benutzt: $\alpha = 1$, $c = 0.5$, $\gamma = 1.0$. Parameter β bekommt als Startwert 2.0 und wird linear so verringert, dass er nach 425 Generationen auf 0 fällt. ρ wird zu Beginn auf 0.025 gesetzt und für die letzten 200 Generationen auf 0.075 erhöht. Die Anzahl von Generationen, bevor die beste Lösung nicht mehr mit zusätzlichen Pheromonen versehen wird ist $g_{max} = 10$. Es wurden die verschiedenen Heuristiken getestet, von denen LST die besten Ergebnisse lieferte. Tabelle 2 liefert eine Übersicht, über das Verhalten der verschiedenen Heuristiken.

Im Vergleich zu den anderen Algorithmen lieferte die in Merkle u.a. [8] vorgestellte ACO bei 5000 generierten Plänen die besten Ergebnisse. Eine kurze Übersicht über die Ergebnisse des Vergleichs sind in Tabelle 3 hinterlegt. Für 278 der 600 Benchmarkpro-

Algorithmus	Durchschnittliche Abweichung von LB in %
ACO für RCPSP	35.43
Selbstadaptierender Genetischer Algorithmus	35.60
Genetischer Algorithmus	36.74
Zeitorientiertes Branch&Bound	37.10
Simulated Annealing	37.68

Tabelle 3: Vergleich mit einigen anderen Algorithmen. Je kleiner die prozentuale Abweichung in der rechten Spalte, desto besser. Diese Tabelle stellt nur einen kleinen Ausschnitt der Ergebnisse dar, für genauere Informationen siehe [8].

bleme wurde die bis dato beste Lösung eingestellt. Ohne eine Beschränkung auf 5000 erstellte Pläne konnte der Algorithmus für 130 Instanzen neue beste Lösungen finden.

Für eine bei weitem umfangreichere Beschreibung der Ergebnisse sei der interessierte Leser an dieser Stelle direkt auf [8] verwiesen.

4.2 Diskussion

Die in Merkle u.a. [8] gezeigten Ergebnisse wirken zunächst einmal beeindruckend. Jedoch stellt sich bei der Einordnung des vorgestellten Algorithmus mit anderen Algorithmen die Frage der Vergleichbarkeit. Die Autoren von [8] sind augenscheinlich bemüht die Algorithmen unter gleichen Bedingungen (5000 generierte Pläne) antreten zu lassen. Hierbei müsste für eine wirklich gerechte Bewertung allerdings auch noch der Aufwand berücksichtigt werden, den die einzelnen Verfahren für die Erstellung der 5000 Pläne betreiben. Zumindest im Vergleich mit einigen der anderen Algorithmen betreibt die vorgestellte ACO einen deutlich höheren Aufwand. Verfahren, bei denen Pläne mit weniger Aufwand generiert werden können, und dafür in der Regel mehr Pläne zum Erzielen guter Ergebnisse brauchen, sind bei einer solchen Bewertungsmethode im Nachteil. Allerdings ist es zugegebenermaßen extrem schwierig, wirklich geeignete und faire Kriterien zur Beurteilung der Performanz von Algorithmen zur Lösung von Optimierungsproblemen zu finden.

Ein weiterer Kritikpunkt schließt sich an den vorigen Punkt an: Bei dem vorgestellten Verfahren werden die eigentlichen Pläne durch das PSGS oder das SSGS generiert. Der Ameisenalgorithmus liefert lediglich eine Liste von Aktivitäten, welche von den Plangenerierungsschemata als Eingabe benutzt wird. Für die Generierung dieser Listen wurde mit den Ameisenalgorithmen ein sehr hoher Aufwand getrieben. Es bleibt zu hinterfragen, ob man die Erstellung von Aktivitätenlisten nicht auf andere Art und Weise effizienter lösen könnte.

Nach der persönlichen Meinung des Autors können bei einem statischen Problem, wie der Listenerstellung, Ameisenalgorithmen ihre große Stärke nicht ausspielen. Zwar bleibt der Lösungsansatz elegant, da er eine einfach nachvollziehbare und relativ leicht implementierbare Idee umsetzt, aber der eigentliche Vorteil von Ameisenalgorithmen besteht in ihrer Adaptivität im Bezug auf Änderungen in ihrer Umgebung. Beispielsweise beim Routing von Datenströmen in realen Netzwerken, bei denen schwankende Auslas-

tung des Netzwerkes und Ausfall einzelner Netzknoten berücksichtigt werden müssen, können diese Stärken viel eher angewandt werden. Dem interessierten Leser sei deshalb die Lektüre von beispielsweise [2; 3] nahegelegt.

Literatur

- [1] BRUCKER, Peter ; KNUST, Sigrid: *Complex Scheduling (GOR-Publications)*. Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006. – ISBN 3540295453
- [2] DI CARO, Gianni ; DORIGO, Marco: AntNet: Distributed Stigmergetic Control for Communications Networks. In: *Journal of Artificial Intelligence Research* 9 (1998), S. 317–365
- [3] DI CARO, Gianni ; DORIGO, Marco. *Two ant colony algorithms for best-effort routing in datagram networks*. 1998
- [4] DORIGO, Marco ; DI CARO, Gianni: The Ant Colony Optimization Meta-Heuristic. In: CORNE, David (Hrsg.) ; DORIGO, Marco (Hrsg.) ; GLOVER, Fred (Hrsg.): *New Ideas in Optimization*. London : McGraw-Hill, 1999, S. 11–32
- [5] GOSS, S. ; ARON, S. ; DENEUBOURG, J. ; PASTEELS, J.: Self-organized shortcuts in the Argentine ant. In: *Naturwissenschaften* 76 (1989), December, Nr. 12, S. 579–581
- [6] KOLISCH, R. ; SCHWINDT, C. ; SPRECHER, A. *Benchmark Instances for Project Scheduling Problems*. 1998
- [7] KOLISCH, R. ; SPRECHER, A. *PSPLIB — a project scheduling problem library*. 1997
- [8] MERKLE, Daniel ; MIDDENDORF, Martin ; SCHMECK, Hartmut: Ant Colony Optimization for Resource-Constrained Project Scheduling. In: WHITLEY, Darrell (Hrsg.) ; GOLDBERG, David (Hrsg.) ; CANTU-PAZ, Erick (Hrsg.) ; SPECTOR, Lee (Hrsg.) ; PARMEE, Ian (Hrsg.) ; BEYER, Hans-Georg (Hrsg.): *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Las Vegas, Nevada, USA : Morgan Kaufmann, 10-12 2000. – ISBN 1-55860-708-0, S. 893–900