Ant Colony Optimization (ACO)

Daniel Blum
PG Meta-Heuristiken
Universität Dortmund

6. Mai 2003

Inhaltsverzeichnis

1 Biologischer Hintergrund **2**

2 Von der Natur zum Algorithmus **2**

3 Allgemeines Konzept für ACO Algorithmen **2**
3.1 Allgemeine Idee 2
3.2 Struktur eines ACO Algorithmus für die Optimierung 3
3.3 High-Level Beschreibung eines ACO Algorithmus 4
3.4 Parameter 5

4 Vor- und Nachteile **5**
4.1 Vorteile 5
4.2 Nachteile 5

5 Anwendungsbeispiele **5**
5.1 ANT-TSP 5
5.1.1 Ant-Cycle Variante von Ant-System 7
5.1.2 Algorithmus 8
5.1.3 Laufzeit 8
5.1.4 Verbesserungen 8
5.2 AntNet 9
5.3 Weitere Anwendungsbeispiele 9

6 Varianten **10**
6.1 Anti-Pheromone 10
6.2 Multi-Colony 10

1 Biologischer Hintergrund

ACO Algorithmen wurden inspiriert durch ein Vorbild aus der Natur, und zwar durch Ameisenkolonien. Obwohl die einzelnen Ameisen der Kolonie relativ simplen Verhaltensweisen folgen, schafft es die Kolonie als ganzes, komplexe Probleme zu bewältigen, wie z.B. das Finden von kurzen Pfaden vom Nest zu Futterquellen und zurück. Eine entscheidende Rolle spielen hierbei Pheromone, die die Ameisen als Orientierung benutzten und bei der Fortbewegung auch selber verteilen. Eine einzelne Ameise bewegt sich fast zufällig. Trifft sie allerdings auf Pheromonspuren, dann folgt sie ihnen mit einer Wahrscheinlichkeit, die mit der Stärke der Spur wächst. Da sie weiterhin selbst Pheromone verteilt, wird die Spur nun verstärkt. Je mehr Ameisen einer Spur folgen, um so attraktiver wird sie. Des weiteren zieht eine attraktivere Spur mehr Ameisen an, ihr zu folgen. Es handelt sich also um einen autokatalytischen Prozess, bzw. um positives Feedback. Begrenzt wird die Zunahme von Ameisen, die einen Pfad benutzen durch verschieden Faktoren, wie z.B. die Anzahl der Ameisen der Kolonie, konkurierende Pfade oder Zufall gesteuertes Verlassen des Pfades. [1]

2 Von der Natur zum Algorithmus

ACO Algorithmen orientieren sich an den Vorbildern der Natur, doch da nicht die Modellierung von Ameisenkolonien im Vordergrund steht, sondern das Übertragen von Ideen, um Probleme zu lösen, ist die biologische Plausibilität nicht immer gegeben. Die Ameisen im Algorithmus werden mit zusätzlichen Verhaltensweisen versehen, erhalten unter Umständen ein Gedächtnis und leben in diskreten Zeitintervallen. In der Natur gibt es keine kontrollierende Instanz, die einen Überblick über das komplette Geschehen hat, die komplette Struktur ist selbstorganisiert. Eine Ameise z.B. kann nicht entscheiden, ob ihr Weg sinnvoll im Vergleich zu anderen ist. Bei Algorithmen hingegen können auch globale Informationen über den Systemzustand ermittelt und verarbeitet werden. Diese können dann in das weitere Vorgehen einfließen.

3 Allgemeines Konzept für ACO Algorithmen

3.1 Allgemeine Idee

Bei ACO Algorithmen wird das Problem in einen Graphen überführt. Jede mögliche Lösung muß durch einen Weg im Graphen repräsentiert sein. Die Kosten einer jeweiligen Lösung ergeben sich durch die Kosten der Kanten des Weges im Graphen. Das Ziel ist es nun, einen Weg minimaler Kosten durch den Graph zu finden.

Die Strategie ist hier dem Vorbild aus der Natur ähnlich. Analog zu den Pheromonen in der Natur werden künstliche Pheromon-Verteilungen auf den Kanten des Graphen verwaltet. Die künstlichen Ameisen bewegen sich ähnlich ihren natürlichen Verwandten zufällig, aber durch die Pheromonkonzentration

beeinflußt, durch den Graph. Der Ablauf des Algorithmus verläuft oft zyklisch. Jede der Ameisen läuft pro Zyklus auf einem Pfad durch den Graph. Danach werden die Pheromonkonzentrationen entsprechend der Pfade verändert, die von Ameisen gewählt wurden. Hierbei gibt es verschiedene Ansätze, bei denen alle gewählten Pfade, eine Teilmenge oder nur der erfolgreichste Pfad der gesamten Ameisen in die Pheromonverteilung einfließen [Ant System] [9]. Positives Feedback spielt demnach eine entscheidende Rolle. Denn Kanten gewählter Pfade werden durch erhöhte Pheromonkonzentration noch attraktiver. Somit können die Konzentrationen auf vielversprechenden Pfaden im Graphen erhöht werden, und die weitere Suche der Ameisen wird sich mit höherer Wahrscheinlichkeit im Bereich dieser Lösungen konzentrieren.

Die Technik des positiven Feedbacks allein birgt jedoch das Risiko, das der Algorithmus frühzeitig konvergiert, und bei einer nicht optimalen Lösung stagniert. Um diesem Effekt entgegenzuwirken, ist auch negatives Feedback Teil der ACO Algorithmen. Die Verflüchtigung von Pheromon (Evaporation), bei dem nach jedem Zyklus die Pheromonkonzentration der Kanten reduziert wird, soll verhindern, dass gute, aber nicht sehr gute Lösungen durch Reinforcement eine weitere Suche behindern. Weiterhin begrenzt sie das sonst nach oben offene Ansteigen der Konzentration. Die Stärke der Evaporation darf weder zu stark noch zu schwach sein. Ist sie zu stark, wird der Effekt der kollektiven Suche zerstört. Informationen über bereits gefundene gute Teillösungen, die in der Pheromonverteilung vorhanden sind, werden zerstört. Ist sie nicht stark genug, erfüllt sie ihren Zweck der Verhinderung von nicht ausreichend guten Lösungen nicht mehr.

3.2 Struktur eines ACO Algorithmus für die Optimierung

Für ACO Algorithmen läßt sich eine Struktur beschreiben, der auf passende kombinatorische Optimierungprobleme angewandt werden kann. [1][Seite 67]

Als Voraussetzung zur Anwendung eines ACO Algorithmus müssen folgende Elemente definierbar sein:

- Eine Problemrepräsentation, die es den Ameisen ermöglicht, Lösungen zu konstruieren bzw. zu modifizieren. Hierbei werden zwei Informationen genutzt. Zum einen die Pheromonverteilung durch eine probabilistische Übergangsfunktion, also Informationen über die bereits durchgeführten Suchen, und desweiteren eine lokale Heuristik, die statisch über die Attrakivität der Kanten Auskunft gibt.
- Eine Heuristik, mit der eine Attraktivität η der Kanten im Graph ermittelt werden kann. Z.B. durch die Entfernung zwischen Städten im Beispiel Ant System [Ant System], siehe weiter unten.
- Eventuell eine Methode, die sicher stellt, dass keine Randbedingungen verletzt werden, so dass nur passende Lösungen konstruiert werden.
- Eine Pheromon-Update-Regel, die beschreibt, wie die Pheromonverteilung τ auf den Kanten des Graphen verändert wird.

In der Initialisierungsphase des Algorithmus wird zunächst die Pheromonverteilung τ initialisiert. Weiterhin werden die Ameisen plaziert. Nun durchläuft der Algorithmus t_{max} Iterationen. Jede Iteration besteht aus zwei Hauptprozeduren. Zum einen der Konstruktion von Lösungen durch die Ameisen. Hierbei wird für jede einzelne Ameise eine Lösung ermittelt unter Anwendung einer probabilistischen Übergangsfunktion p. Diese ermittelt in Abhängigkeit von der Position der Ameise, der Pheromonverteilung, der lokalen Heuristik und unter Umständen von Randbedingungen den nächsten Knoten, zu dem sich die Ameise bewegt. Üblicherweise wirkt sich eine höhere Pheromonverteilung auf einer Kante durch eine höhere Wahrscheinlichkeit der Ameise aus, diese zu wählen. Eine Ameise bewegt sich von einem Startknoten so lange durch den Graph, bis sie eine Lösung konstruiert hat. Durch Beachten von Randbedingungen in der Übergangsfunktion p ist es möglich, nur gültige Lösungen zu erstellen. Dies ist jedoch nicht zwingend erforderlich. Sind die Lösungen ermittelt, erfolgt der zweite Teil. Die Güte der Lösungen wird bestimmt, und die Pheromonverteilung entsprechend angepaßt. Die jeweils beste Lösung wird gespeichert. Nach t_{max} Iterationen wird die beste Lösung ausgegeben.

Die Pheromonverteilung des Graphen stellt das Gedächtnis über bisherige Ergebnisse dar. Sie wird ständig angepaßt und gibt Aufschluß darüber, in welchen Regionen des Lösungsraumes sich gute Lösungen befinden. Die beste gefundene Lösung läßt sich häufig nicht oder nicht direkt aus der Pheromonverteilung ablesen.

3.3 High-Level Beschreibung eines ACO Algorithmus

```
Initialisiere Pheromonverteilung τ

Platzierung der Ameisen

FOR t = 1 TO t_max DO

     Lasse Ameisen Lösungen konstruieren
     Dabei bewegen sie sich entsprechend der Übergangsfunktion p

     Berechne Kosten der Lösungen

     IF verbesserte Lösung gefunden THEN
            Update beste Lösung

     Aktualisieren der Pheromonverteilung durch Anwendung der Regel

END FOR

Gib die beste Lösung aus
```

3.4 Parameter

Einige wichtige Parameter, die das Verhalten von ACO-Algorithmen beeinflussen, sind unter anderem die Anzahl der Ameisen, Parameter zur Pheromonverteilungsänderung, Gewichtungen zwischen lokaler Heuristik und Beachtung der Pheromonverteilung in der Übergangsfunktion der Ameisen. Diese Parameter sind oft Problem-abhängig und müssen durch Probieren ermittelt werden.

4 Vor- und Nachteile

4.1 Vorteile

- Effektive Suche, da Teilergebnisse genutzt werden, aber trotzdem ein großer Suchraum abgedeckt wird.
- Die Algorithmen sind nicht rein Zufallsgesteuert. Sie sind in der Lage, zusätzliche allgemeine Informationen in Form lokaler Heuristiken aufzunehmen.
- Eine gewisse Anpassungsfähigkeit auch bei dynamischen Problemen ist gegeben. Allerdings können bereits existierende Pheromonverteilungen zu starkt sein, so daß Veränderungen nicht attraktiv genug sind, um sich durchzusetzen.

4.2 Nachteile

- ACO-Algorithmen sind auf spezielle Problemgruppen beschränkt. Die Probleme müssen als Graphen darstellbar sein, auf denen Lösungen konstruierbar sind.
- Viele Ergebnisse sind empirisch, ohne gesicherte theoretische Grundlage, zum Beispiel bezüglich Konvergenz gegen gute Lösungen. Es gibt allerdings Bestrebungen in diese Richtung, z.B. [4]

5 Anwendungsbeispiele

5.1 ANT-TSP

Der historisch erste ACO Algorithmus wurde 1991 von Marco Dorigo, Vittorio Maniezzo und Alberto Colorni vorgestellt [Ant System]. Er suchte Lösungen für das Traveling Salesman Problem. Der vorgestellte Algorithmus wird in der Literatur oft als Ant-System bezeichnet, bzw. in Kombination mit dem TSP als AS-TSP. Dieser Algorithmus wurde später noch verbessert, aber als erstes Anwendungsbeispiel scheint er gut geeignet, da das TSP-Problem gut bekannt ist, und die Problemstellung und Lösungsansatz nicht zu weit von der Analogie zur Natur entfernt sind.

Das TSP-Problem kann wie folgt beschrieben werden: Gegeben sind n Städte, und gesucht ist eine Tour minimaler Länge, die alle Städte genau einmal besucht. Sei nun d_{ij} die Distanz zwischen den Städten i und j. Eine Instanz des TSP-Problems kann als gewichteter Graph $G(V, E)$ beschrieben werden, wobei die Knoten V die Städte darstellen, und die Kanten E zwischen den Städten entsprechend der Distanz gewichtet sind.

Dieser Graph bildet die Grundlage, auf der sich die Ameisen bewegen werden.

Die Ameisen im Algorithmus sind nun simple Agenten, die sich durch den Graph bewegen, und dabei bestimmten Verhaltensmustern folgen:

- Sie bewegen sich zufällig, beginnend in ihrer Startstadt, von einer Stadt i in die nächste Stadt j, bis alle Städte besucht sind. Welche Stadt sie jeweils wählen, ist durch die Übergangswahrscheinlichkeit $p_{ij}(t)$ gegeben, die für die jeweilige Stadt ermittelt wird.
- Sie besuchen keine Stadt doppelt. Hierzu wird eine Tabuliste geführt.
- Der von ihnen gewählte Pfad wird mit entsprechenden Mengen Pheromon markiert.

Sichtbarkeit: Die Sichtbarkeit (visibility) η_{ij} ist definiert als $\frac{1}{d_{ij}}$ und somit ein Maß für die Nähe der Städte. Sie ermöglicht das Einfließen von allgemeinen lokalen Information, unabhängig von der Pheromonverteilung. In diesem Algorithmus stellt die Sichtbarkeit eine einfache Greedy-Heuristik dar.

Tabuliste: In der Tabuliste $tabu_k$ werden alle Städte aufgeführt, die von der Ameise k einschließlich ihrer Startstadt bereits besucht wurden. Durch die Tabuliste wird ausgeschlossen, dass die Ameise auf Grund der zufallsbeeinflußten Bewegung Städte doppelt besucht, und somit ungültige Wege produziert. Nach der Bewegung zu einer neuen Stadt wird die Liste aktualisiert. Die für eine Ameise noch möglichen Wahlmöglichkeiten *allowed* ergeben sich aus allen Städten abzüglich $tabu_k$.

Pheromonverteilung: Die Pheromonverteilung $\tau_{ij}(t)$ gibt die Konzentration von Pheromon zum Zeitpunkt t auf dem Weg zwischen den Städten i und j an. Die Pheromonverteilung wird beeinflußt von der bisherigen Wegewahl der Ameisen, und sie beeinflußt die zukünftige Wahl der Wege. Beim Starten des Algorithmus wird sie mit niedrigen gleichen Werten initialisiert. Es gibt verschiedene Regeln, nach denen die Pheromonverteilung geändert werden kann. Sie unterliegt ferner einem zeitlichen Verfall, der Evaporation.

Evaporation: Die Evaporation beschreibt die Verflüchtigung von Pheromon. Sie verhindert ein unbegrenztes Wachstum auf Kanten im Pfad und unterstützt die Exploration von anderen Lösungen.

Übergangswahrscheinlichkeit: Die Übergangswahrscheinlichkeit $p_{ij}(t)$ beschreibt die Wahrscheinlichkeit, mit der sich eine Ameise zum Zeitpunkt t, die sich in der Stadt i befindet, zur Stadt j bewegt. Dies hängt von der Pheromomverteilung τ, der Sichtbarkeit η und ihren Wahlmöglichkeiten *allowed* ab.

$$p_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{j \in allowed} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}} & \text{wenn } j \in \text{allowed} \\ 0 & \text{sonst} \end{cases}$$

Die Parameter α und β erlauben nun eine Gewichtung zwischen den durch die Pheromonverteilung bekannten Informationen über vorherige Zyklen und den lokalen Informationen, hier der Sichtbarkeit, wie sie eine Greedy Heuristik benutzen würde. Mit $\alpha = 0$ und $\beta = 1$ erhält man einen reinen Greedy-Algorithmus, mit $\alpha = 1$ und $\beta = 0$ einen rein zufällig gesteuerten.

Pheromon trail update rule

$\tau_{ij}(t)$ beschreibt die Pheromonintensität zum Zeitpunkt t auf der Kante (i, j). Nach jeder Iteration des Algorithmus wird die Pheromonintensität wie folgt verändert:

$\tau_{ij}(t + 1) = \rho \cdot (\tau_{ij}(t)) + \Delta\tau_{ij}(t, t + 1)$

ρ ist hierbei ein Koeffizient, der die Stärke der Evaporation beschreibt. ρ sollte < 1 sein, um unbegrenzte Anhäufung von Pheromon zu verhindern. $\Delta\tau_{ij}(t, t + 1)$ beschreibt das in dieser Iteration hinzugekommene Pheromon, welches auf den von den Ameisen gewählten Pfaden verteilt wird. Hierbei sind verschiedene Techniken möglich:

- Es können jeweils konstante Mengen Pheromon auf von den Ameise benutzten Kanten verteilt werden.
- Die Menge wird jeweils durch die Länge der Kanten geteilt.
- Die Menge wird durch die Länge der gesamten Tour geteilt.

In dem Paper [Ant System] werden drei verschieden Algorithmenvariationen genannt. Während bei den ersten beiden Varianten die Pheromonintesität nach der Tour jeder Ameise aktualisiert wird, erfolgt dies bei der dritten Variante (Ant-cycle) erst nach dem alle Ameisen ihre Tour durchgeführt haben. Diese dritte Variante lieferte in den Tests von Dorigo die besten Ergebnisse.

5.1.1 Ant-Cycle Variante von Ant-System

In der Ant-Cycle Variante von Ant-System beginnt jeder Zyklus mit dem Konstruieren einer Tour für jede Ameise k. Aus den jeweiligen Touren berechnen sich die Beiträge zur Pheromonverteilungsveränderung wie folgt:

$$\triangle\tau_{ij}^{k}(t, t{+}1) = \begin{cases} \frac{Q}{L_k} & \text{wenn Ameise } k \text{ Kante } (i, j) \text{ bei ihrer Tour benutzt hat} \\ 0 & \text{sonst} \end{cases}$$

Q ist hierbei eine Konstante und L_k die Länge der Tour der k-ten Ameise. Somit erzielen kürzere Touren höhere Pheromonbeiträge. Hier fließt die globale Information der Tourlänge mit ein.

Die Beiträge der einzelnen Ameisen werden summiert und es ergibt sich

$$\triangle\tau_{ij}(t,t+1) = \sum_{k=1}^{m} \triangle\tau_{ij}^{k}(t,t+1)$$

als gesamtes Pheromon, welches auf Kante (i,j) hinzugefügt wird.

5.1.2 Algorithmus

```
Initialisierung der Pheromonverteilung τ

Platzierung der Ameisen in Städten

FOR t = 1 TO t_max DO

   FOR k = 1 TO m DO

       Konstruktion einer Tour für Ameise k

       Berechnung der Tourkosten

       IF verbesserte Lösung gefunden THEN
            Update beste Lösung

       Ermittlung von △τ_k(t, t+1)

   END FOR

   Aktualisierung der Pheromonverteilung τ

END FOR

Ausgeben der besten Lösung
```

5.1.3 Laufzeit

Die Laufzeit läßt sich für n Städte, m Ameisen und z Zyklen als $O(z \cdot n^2 \cdot m)$ bestimmen. Dorigo, Maniezzo und Colorni stellten für die Ant-Cycle Variante von Ant-System einen linearen Zusammenhang zwischen der Anzahl der Städte und der optimalen Anzahl von Ameisen fest. Somit ergibt sich $O(z \cdot n^3)$.

5.1.4 Verbesserungen

Mit dem ACS, Ant Colony System, wurde dieser Ansatz verbessert. Zum Tragen kommen hierbei eine andere Übergangsregel, eine andere Pheromon trail update rule, ein lokales Update von Pheromon um die Exploration zu fördern, sowie eine Kandidatenliste mit nahegelegenen Städten, die bevorzugt werden.

5.2 AntNet

ACO Agorithmen sind nicht nur auf statische Probleme beschränkt, sondern durchaus in der Lage, passende dynamische Probleme zu optimieren. Für Routing in Netzwerken existiert unter anderem ein Ansatz von Di Caro und Dorigo [10].

Hier gilt es, in einem Netzwerk von N Knoten, die untereinander nicht alle direkt verbunden sind, Pakete jeweils von ihren Quellknoten q zu ihren Zielknoten z zu befördern. Für jeden Knoten wird eine Tabelle geführt, in der festgehalten wird, mit welcher Wahrscheinlichkeit Knoten zu bestimmten Nachbarknoten weitergeleitet werden, je nach dem, welchen Zielknoten sie besitzten. Ein Paket wird also an einem Knoten nicht immer zum selben Nachbarknoten geleitet, auch wenn es dasselbe Ziel hat. Die Wahrscheinlichkeiten der Weiterleitung zu den Nachbarknoten können nun optimiert werden, um gute Ergebnisse zu erzielen. Entscheidend ist, dass die Wahrscheinlichkeiten an temporäre oder lokale Veränderungen des Verkehrs im Netzwerk angepaßt werden können.

AntNet setzt dazu Ameisen ein, die sich neben den Paketen im Netzwerk von Quell- zu Zielknoten bewegen. Sie werden zufällig erzeugt und an Startknoten ins Netz gesetzt. Nun bewegen sie sich wie Pakete, bis sie ihr Ziel erreicht haben. Dies sind die „forward“-Ameisen. Sie protokollieren ihren Weg und die Zeit, die sie gebraucht haben. Sollten sie einen Kreis gelaufen haben, da sie durch die zufallsbeeinflußte Bewegung einen Knoten doppelt besuchen, wird die Information der Kreistour entfernt. Am Zielknoten erzeugen sie eine „backward“-Ameise, die den selben Weg zurück durchläuft, und werden aus dem Netzwerk entfernt. Die „backward“-Ameise paßt auf ihrem Weg die Wahrscheinlichkeiten in den Tabellen der Knoten an, die von der „forward“-Ameise durchlaufen wurden. Durch den kontinuierlichen Einsatz von Ameisen werden auch Veränderungen im Netzwerk implizit erkannt, und entsprechende Anpassungen werden vorgenommen.

5.3 Weitere Anwendungsbeispiele

ACO Algorithmen existieren für einige weitere Probleme. Es gibt unter anderem Ansätze für:

- Quadratic Assignment [7]
- Sequential Ordering [6]
- Vehicle Routing [8]
- Färben von Graphen [5]
- 2D HP Protein Folding [3]

6 Varianten

6.1 Anti-Pheromone

Weitere Ideen im Bereich ACO umfassen unter anderem einen neuen Pheromon-Typ, Anti-Pheromon, wie ihn James Montgomery und Marcus Randall [2] beschreiben. Die Idee hierbei ist, dass die Ameisen durch die Anhäufung von Pheromon gute Teillöstungen verstärken, aber die Informationen über schlechte Lösungen ebenso verwendet werden könnten. So sollen Explorationen in Bereichen schlechterer Teillösungen vermindert werden, so dass sich die Suche noch mehr auf bessere Regionen konzentriert, und geringere Anteile der Ameisen die Suche in schlechteren Regionen vorantreibt. Sie schlagen drei Varianten vor:

Substractive Anti-pheromone: Hier wird Pheromon auf den Pfaden der schlechtesten Tour einer Ameise abgezogen, um sie in folgenden Zyklen weniger attraktiv zu machen.

Preferential Anti-pheromone: Hier werden zwei Pheromon-Typen verwaltet, einer für gute Lösungen und einer für schlechte Lösungen. Die Idee ist, dass ein Teil der Ameisen sich mehr am ersten Pheromon orientiert, und so im Bereich der zur Zeit besten Lösung sucht, und ein anderer Teil durch den zweiten Typ in schlechteren Bereichen exploriert.

Explorer Ants: Explorer Ants sind eine kleine Teilmenge der Ameisen, die von Kanten mit wenig Pheromon angezogen werden, sich also konträr zu den normalen Ameisen verhalten. Sie sollen eine Suche fern den Hauptpfaden sicherstellen.

Montgomery und Randall testeten mit verschiedenen Problemen im Vergleich mit ACS. Es wurden keine signifikanten Verbesserungen nachgeweisen.

6.2 Multi-Colony

Ein weiterer Ansatz sind Multi-Colony Techniken. Hierbei werden verschiedene Kolonien seperat berechnet, die unter Umständen auch von unterschiedlichen Faktoren beeinflußt werden. In gewissen Abständen werden Pheromonkonzentrationen der einzelnen Kolonien ausgetauscht.

Literatur

[Ant System] Marco Dorigo, Vittorio Maniezzo und Alberto Colorni. Positive Feedback as a Search Strategy.

[1] Eric Bonabeau , Marco Dorigo und Guy Theraulaz. From Natural to Artificial Swarm Intelligence. New York, 1999, Oxford University Press.

[2] James Montgomery und Marcus Randall. Anti-pheromone as a Tool for Better Exploration of Search Space. Ant Algorithms, KNCS 2463, Seiten 100-110. Berlin, 2002, Springer Verlag.

[3] Alena Shmygelska, Rosalia Aguirre-Hernandez und Holger Hoos. An Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. Ant Algorithms, KNCS 2463, Seiten 40-52. Berlin, 2002, Springer Verlag.

[4] Mauro Birattari, Gianni Di Caro und Marco Dorigo. Toward the Formal Foundation of Ant Programming. Ant Algorithms, KNCS 2463, Seiten 188-201. Berlin, 2002, Springer Verlag.

[5] Costa D. and A. Hertz (1997). Ants Can Colour Graphs. Journal of the Operational Research Society, 48, 295-305.

[6] Gambardella L. M. and M. Dorigo (1997). HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem. Tech. Rep. No. IDSIA 97-11, IDSIA, Lugano, Switzerland.

[7] Maniezzo V., A. Colorni and M. Dorigo (1994). The Ant System Applied to the Quadratic Assignment Problem. Tech. Rep. IRIDIA/94-28, Universit Libre de Bruxelles, Belgium.

[8] Bullnheimer B. (1999). Ant Colony Optimization in Vehicle Routing. Doctoral thesis, University of Vienna, January 1999.

[9] Dorigo M. und L.M. Gambardella (1997). Ant Colonies for the Traveling Salesman Problem. BioSystems, 43:73-81. Also Tecnical Report TR/IRIDIA/1996-3, IRIDIA, Universit Libre de Bruxelles

[10] Di Caro G. und M. Dorigo (1997). AntNet: A Mobile Agents Approach to Adaptive Routing. Tech. Rep. IRIDIA/97-12, Universit Libre de Bruxelles, Belgium.

[11] **Homepage für ACO**:
http://iridia.ulb.ac.be/∼ mdorigo/ACO/ACO.html

Glossar

- **ACO** - Ant Colony Optimization, Oberbegriff für Ameisen inspirierte Herangehensweisen, Optimierungsprobleme zu behandeln.
- **ACS** - Ant Colony System, verbesserte Version von AS, Algorithmus für das TSP Problem.
- **AS** - Ant System, erster ACO Algorithmus, Algorithmus für das TSP Problem.
- **Evaporation** - Verdunstung oder zeitlicher Verfall von (Pheromon-) Konzentrationen.
- **Pheromon** - Duftsubstanz, die von Ameisen verwendet wird, um Pfade zu markieren. Findet als Analogon in ACO Anwendung. Manchmal auch als "trail" oder "trail-substance" bezeichnet.