Ant Colony Optimization
Ausarbeitung zum Vortrag im Seminar "Intelligente Algorithmen"
WS 2003/2004

Veranstalter: Stefan Kopp, Nadine Leßmann, Alfred Kranstedt

Jan Lümkemann

18.12.2003

Zusammenfassung

Diese Ausarbeitung stellt das Optimierungsverfahren Ant Colony Optimization (ACO) vor. Dabei wird das natürliche Verhalten von Ameisen bei der Wegsuche modelliert, um Optimierungsprobleme in der Informatik zu lösen. Nach einer Einführung in das biologische Modell, werden am Beispiel des Travelling Salesman Problems zwei solche Ameisenalgorithmen erläutert. Als Hauptquelle für diese Ausarbeitung diente das Buch "Swarm Intelligence. From Natural to Artificial Systems." von E. Bonabeau et al. [1].

Inhaltsverzeichnis

1 Motivation und Einleitung **3**

2 Verhalten natürlicher Ameisen **3**

3 Allgemeiner Ansatz **4**

4 Ameisenalgorithmen zur Lösung des TSP **5**

4.1 Modellierung künstlicher Ameisen 5

4.2 Ant System (AS) 6

4.3 Ant Colony System (ACS) 7

5 Konvergenzverhalten **8**

6 Weitere Anwendungen **9**

1 Motivation und Einleitung

Insekten, die in Schwärmen leben, wie z.B. Ameisen oder Bienen, lösen in der Natur komplexe Aufgaben. So sind zum Beispiel zum Nestbau oder zur effizienten Futtersuche Fähigkeiten erforderlich, die die Fähigkeiten eines einzelnen Tieres um ein Vielfaches übersteigen.
Daher hat man versucht das Verhalten von Insektenschwärmen mit Hilfe der Theorie der Selbstorganisation zu beschreiben. Das Modell der Selbstorganisation beschreibt das Entstehen von makroskopischen Mustern aus wechselwirkenden mikroskopischen Teilchen. Es hat sich gezeigt, dass dieses ursprünglich in der Physik und Chemie genutzte Modell auch das Verhalten von Insektenschwärmen beschreiben kann.
Viele der Probleme die von Insektenschwärmen gelöst werden, lassen sich auf Probleme in der Informatik übertragen (z.B. die Futtersuche als Pfadsuche auf Graphen). Diese Ausarbeitung beschäftigt sich mit Algorithmen, die das Travelling Salesman Problem (TSP) nach dem Modell der Pfadsuche von Ameisen lösen. Nach einer Beschreibung des Verhaltens von natürlichen Ameisen folgt die Modellierung von künstlichen Ameisen, die zur Problemlösung eingesetzt werden. Im Anschluss werden zwei Algorithmen vorgestellt, die das TSP mit Hilfe des Ameisenmodells lösen. Den Abschluss bildet eine Betrachtung des Konvergenzverhaltens von Ameisenalgorithmen und ein Überblick über weitere Anwendungen von Ameisenalgorithmen.

2 Verhalten natürlicher Ameisen

Das in diesem Kaptitel vorgestellte Verhalten von Ameisen bei der Pfadsuche (z.B. zwischen Nestern oder zwischen Nest und Futterstelle) basiert auf Experimenten von Deneubourg et al. [3].
Ameisen markieren einen zurückgelegten Weg mit Pheromonen. Nachfolgende Ameisen wählen bevorzugt Wege mit hoher Pheromonkonzentration. In einem Experiment von Deneubourg sind die Ameisen durch eine Brücke mit zwei Pfaden von einer Futterstelle getrennt (vgl. Abbildung 1).
Es zeigt sich, dass die Wahrscheinlichkeit, dass eine Ameise einen bestimmten Pfad wählt,

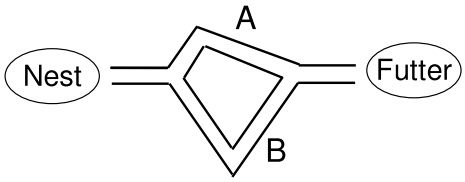

Abbildung 1: Versuchsaufbau Brückenexperiment

von der Anzahl der Ameisen abhängt, die den Pfad bereits genutzt haben. Sei A_i bzw. B_i die Anzahl der Ameisen, die Pfad A bzw. Pfad B gewählt haben, wobei i die Gesamtzahl

der Ameisen ist, die die Brücke passiert haben. Dann ergibt sich experimentell für die Wahrscheinlichkeit P_A bzw. P_B, dass die nachfolgende (i+1)te Ameise Pfad A bzw. Pfad B wählt, folgender Zusammenhang:

$$P_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n} = 1 - P_B \tag{1}$$

Der Exponent n bestimmt die Nichtlinearität in der Entscheidungsfunktion. Wenn n groß ist, führen schon kleine Unterschiede in der Pheromonkonzentration zu großen Unterschieden in den Wahrscheinlichkeiten. Der Parameter k beschreibt die Attraktivität eines noch nicht genutzten pheromonfreien Pfades. Experimente haben gezeigt, dass die Entscheidungfunktion (1) mit den Parameterwerten n≈2 und k≈20 das Ameisenverhalten besonders gut approximiert.
Wenn die beiden Pfade im Versuch eine unterschiedliche Länge haben, zeigt sich, dass die Ameisen nach einer zufälligen Phase, in der beide Wege gleich stark genutzt werden, den kürzeren Pfad bevorzugen. Ameisen, die den kürzeren Weg wählen, legen den Weg zur Futterstelle und zurück schneller zurück, d.h. hier steigt die Pheromonkonzentration schneller und der kürzere Pfad wird in der Folge stärker genutzt (vgl. auch (1)).
Deneubourg et al. [3] beobachteten die argentinischen Ameisenart Linepithema humile und zeigten, dass diese Art den minimalen aufspannenden Baum in einem Netz von Nestern finden kann. Diese Ergebnisse haben Dorigo 1992 [4] dazu bewegt, das oben beschriebene Verhalten von Ameisen erstmals zur Lösung des Travelling Salesman Problems (TSP) zu nutzen.

3 Allgemeiner Ansatz

Bevor in den folgenden Kapiteln die konkrete Anwendung von Ameisenalgorithmen auf das TSP vorgestellt wird, wird in diesem Kapitel allgemein beschrieben, welche Arten von Problemen sich mit Ameisenalgorithmen lösen lassen und wie das Verhalten der natürlichen Ameisen zur Lösung dieser Probleme genutzt wird.
Wie oben angedeutet modellieren Ameisenalgorithmen das Verhalten von natürlichen Ameisen bei der Wegsuche. Diese Wegsuche kann in der Informatik als Suche auf einem Graphen beschrieben werden. Daher können mit Ameisenalgorithmen Probleme gelöst werden, die sich als Wegsuche auf Graphen formulieren lassen. Solche Probleme werden durch die folgenden Parameter beschrieben:

1. Eine Menge von Randbedingungen, die die Eigenschaften des Problems beschreiben (z.B. Alle Knoten des Graphen müssen besucht werden.).
2. Eine endliche Menge von Knoten, die Teilprobleme darstellen.
3. Nachbarschaftsbeziehungen zwischen den Knoten die mögliche Übergänge zwischen den Knoten beschreiben.
4. Eine Lösung, die durch einen Weg durch den Graphen beschrieben wird, der alle Randbedingungen des Problems erfüllt.

5. Eine Kostenfunktion, die jeder Lösung ihre Kosten zuweist.

Ameisenalgorithmen lassen sich als Multiagentensysteme auffassen, bei denen künstliche Ameisen mit Hilfe von lokalen Heuristiken und Kommunikation über Pheromone die kostengünstigste Lösung als Weg auf dem Problemgraphen suchen. Ameisenalgorithmen sind lokale Suchverfahren (eine zufällige Lösung wird iterativ optimiert), die durch sog. Metaheuristiken unterstützt werden. Die Metaheuristik wird hier aus dem natürlichen Ameisenverhalten gewonnen, dessen technische Modellierung im folgenden Kapitel vorgestellt wird.

4 Ameisenalgorithmen zur Lösung des TSP

Dieses Kapitel beschreibt die grundlegenden Prinzipien von Ameisenalgorithmen zur Lösung des TSP. Zunächst wird ein Modell für künstliche Ameisen vorgestellt, das die Basis für die vorgestellten Algorithmen liefert. Der einfachste Ameisenalgorithmus zur Lösung des TSP ist das von Dorigo entworfene Ant System (AS) [4]. Neben diesem wird im folgenden auch das Ant Colony System (ACS) [5] vorgestellt, das das Konvergenzverhalten des AS für größere TSPs optimiert.

4.1 Modellierung künstlicher Ameisen

In Ameisenalgorithmen erzeugen virtuelle Ameisen Lösungen des TSP, indem sie von einer Stadt zur nächsten wandern. In jeder Iteration t des Algorithmus erzeugt jede Ameise eine vollständige Lösung des TSP. Während einer Iteration nutzen die Ameisen folgende Kriterien zur Auswahl der nächsten Stadt:

1. Eine *Tabu-Liste*: Das TSP schreibt vor, dass jede Stadt nur einmal besucht werden darf. Daher wird für jede Ameise k eine Liste (Tabu-Liste) der von k bereits besuchten Städte gespeichert. Diese Liste wird benutzt, um diejenigen Städte J_i^k zu ermitteln, die eine Ameise k, die sich gerade in der Stadt i befindet, noch besuchen muss.

2. Eine *lokale Heuristik*: Die Ameisen nutzen den Kehrwert $\eta_{ij} = 1/d_{ij}$ der Distanz d_{ij} zwischen den Städten i und j als lokales Gütemaß. Dieser wir auch Sichtbarkeit genannt und veranlasst die Ameisen, naheliegende Städte zu bevorzugen. Diese Heuristik wird auch bei der gierigen Suche eingesetzt.

3. Eine *globale Heuristik*: Nach jeder Iteration wird die virtuelle Pheromonkonzentration entlang der von den Ameisen gefundenen Lösungen aktualisiert. Dabei bezeichnet $\tau_{ij}(t)$ die Pheromonkonzentration auf der Kante von Stadt i zu Stadt j in der Iteration t. Die Pheromonkonzentration wird ebenfalls bei der Auswahl der nächsten Stadt berücksichtigt. Dabei bevorzugen die Ameisen Kanten mit hoher Pheromonkonzentration. Diese, im Gegensatz zur Sichtbarkeit, globale Information spiegelt den Lernprozess der Ameisen während der Iterationen wieder.

4.2 Ant System (AS)

Das AS ist der erste Ameisenalgorithmus, der zur Lösung des TSP genutzt wurde. Er wurde von Dorigo im Jahr 1992 entwickelt [4].
Das AS nutzt eine statistische Entscheidungsfunktion nach dem natürlichen Beispiel (1) für die Auswahl der nächsten Stadt. Die Wahrscheinlichkeit p, dass eine Ameise k in Iteration t von Stadt i zu Stadt j wandert, ist gegeben durch:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} & f\ddot{u}r\ j \in J_i^k \\ 0 & f\ddot{u}r\ j \notin J_i^k \end{cases} \tag{2}$$

Die Parameter α und β steuern die Relevanz der Pheromonkonzentration bzw. der Sichtbarkeit für die Berechnung der Wahrscheinlichkeiten. Wenn $\alpha = 0$ ist, wird die nächste Stadt allein aufgrund der Sichtbarkeit ausgewählt. Dies entspricht der klassischen gierigen Suche. Wenn $\beta = 0$ ist, konvergieren alle Lösungen sehr schnell zu einer möglicherweise nicht optimalen Lösung, die zufällig von vielen Ameisen gewählt wurde. Die Exploration neuer Wege wird unterdrückt.
Nach jeder Iteration t markiert jede Ameise k die Kanten (ij) ihrer Lösung $T^k(t)$ mit Pheromon. Die Pheromonmenge $\Delta\tau_{ij}^k(t)$ ist von der Güte der jeweils gefundenen Lösung $T^k(t)$ abhängig:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & f\ddot{u}r\ (i,j) \in T^k(t); \\ 0 & f\ddot{u}r\ (i,j) \notin T^k(t). \end{cases} \tag{3}$$

Hier ist $L^k(t)$ die Länge der gefundenen Tour. Q ist ein Parameter, der zwar nur geringen Einfluss auf das Suchergebnis hat, aber in etwa die Größe der optimalen Tour haben sollte. Die Verstärkung guter Lösungen durch die Erhöhung der Pheromonkonzentration allein garantiert noch nicht, dass auch global gute oder sogar optimale Lösungen gefunden werden. Um eine Stagnation in lokalen Optima zu vermeiden wird ein Zerfallskoeffizient ρ, $0 \leq \rho < 1$, eingeführt. Insgesamt ergibt sich nun die Pheromonkonzentration $\tau_{ij}(t)$ einer Kante (ij) in der Iteration t zu:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{4}$$

wobei $\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t)$ und m die Anzahl der Ameisen ist.
Zu Beginn des Algorithmus sind alle Kanten mit einer Pheromonmenge τ_0 initialisiert. Die Anzahl der Ameisen beeinflusst die Qualität der gefundenen Lösung sehr stark. Während zu viele Ameisen sehr schnell nicht optimale Lösungen favorisieren, wird der Verstärkungsmechnismus bei zu wenigen Ameisen direkt durch den Zerfall aufgezehrt. Es hat sich gezeigt, dass es sinnvoll ist, gleich viele Ameisen wie Städte einzusetzen [1].
Die Zeitkomplexität des AS beträgt $O(t \cdot n^3)$, wenn gleich viele Ameisen wie Städte verwendet werden. Ein Vergleich des AS mit anderen allgemeinen heuristischen Suchverfahren zeigt, dass AS für kleine TSPs (30-70 Städte) vergleichbare oder bessere Lösungen liefert (vgl. Tabelle 1), allerdings für größere TSPs erst nach sehr vielen Iterationen die optimale Lösung findet.

	Beste Lösung	Mittelwert	Standardabweichung
Ant System	420	420,5	1,3
Tabu Suche	420	420,6	1,5
Simulated Annealing	422	459,8	25,1

Tabelle 1: Die Tabelle (nach [1]) zeigt eine statistische Auswertung von Lösungen des TSP Oliver30. Jeder Algorithmus hatte 1 Stunde Rechenzeit auf einem IBM-kompatiblen PC mit einem 80386 Prozessor. Die Mittelwerte ergeben sich jeweils aus einer Serie von 10 Durchläufen der Algorithmen.

Dieses Verhalten liegt darin begründet, dass AS nicht sehr zielgerichtet auf die optimale Lösung zustrebt. Beispielsweise hat sich gezeigt, dass die mittlere Verzweigungszahl (Anzahl der genutzten Kanten je Stadt) auch nach 3000 Iterationen bei einem TSP mit 30 Städten noch größer als 5 ist. Das führt dazu, dass lokale Optima vermieden werden und eine große Toleranz gegenüber dynamischen Verhältnissen (Lage der Städte ist variabel) besteht.
Das Problem der langsamen Konvergenz kann durch eine lokale Optimierung umgangen werden. Der nächste Abschnitt stellt das Ant Colony System vor, in dem Dorigo und Gambardella versuchen, die Performanz des Ant System zu verbessern [5].

4.3 Ant Colony System (ACS)

Das Ant Colony System (ACS) verbessert den Algorithmus des AS, um direkter auf die optimale Lösung zuzusteuern. Dazu werden die folgenden vier Änderungen vorgenommen.

Übergangsfunktion. Die Übergangsfunktion erlaubt es nun, die Exploration neuer Lösungen besser einzustellen. Eine Ameise k in der Stadt i wählt als nächstes die Stadt j nach folgender Regel:

$$j = \begin{cases} argmax_{u \in J_i^k}\{[\tau_{iu}(t)] \cdot [\eta_{iu}(t)]^\beta\} & für\ q \leq q_0; \\ J & für\ q > q_0, \end{cases} \tag{5}$$

wobei q eine gleichverteilte Zufallsvariable auf dem Intervall [0,1] und q_0 ein Parameter ist. $J \in J_i^k$ ist eine Stadt, die zufällig gemäß folgender statistischen Auswahlfunktion ermittelt wird:

$$p_{iJ}^k(t) = \frac{[\tau_{iJ}(t)] \cdot [\eta_{iJ}]^\beta}{\sum_{l \in J_i^k}[\tau_{il}(t)] \cdot [\eta_{il}]^\beta} \tag{6}$$

Diese neue Übergangsfunktion (5) ist der Übergangsfunktion des AS (2) sehr ähnlich (für $\alpha = 1$). Wenn $q > q_0$ entspricht Funktion (5) Funktion (2). Wenn $q \leq q_0$, wird eine lokale Optimierung auf Grundlage der Sichtbarkeit und der bis zu dieser Iteration gefundenen Lösungen durchgeführt.
Die Einstellung des Parameters q_0 entspricht dem Einstellen der Temperatur beim Simulated Annealing Algorithmus. Je näher q_0 an 1 ist, desto lokaler läuft die Suche ab, mit der Gefahr

der Stagnation in einem lokalen Optimum. Ist q_0 nahe an 0 werden viele parallele Lösungen exploriert, wobei das globale Optimum erst sehr spät gefunden wird (siehe AS).

Globale Aktualisierung der Pheromonspuren. Im AS tragen alle Ameisen zur Erhöhung der Pheromonkonzentration auf den genutzten Kanten bei. Im ACS erhöht nur diejenige Ameise die Pheromonkonzentration auf den Kanten ihrer Lösung, die die beste bisher erreichte Lösung gefunden hat. Dadurch wird die Suche zielgerichteter. Die Aktualisierungsregel sieht nun wie folgt aus:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t), \tag{7}$$

wobei die (ij) die Kanten der besten Lösung T^+ sind und ρ wieder den Zerfall des Pheromons wiederspiegelt. $\Delta\tau_{ij}(t)$ ergibt sich aus der Länge L^+ der besten Lösung T^+:

$$\Delta\tau_{ij}(t) = \frac{1}{L^+}. \tag{8}$$

Lokale Aktualisierung der Pheromonspuren. Jede Ameise reduziert während einer Iteration die Pheromonkonzentration jeder genutzten Kante (ij). Dadurch werden oft genutzte Kanten für nachfolgende Ameisen unattraktiver, was die Suche in neue Richtungen treibt und eine Stagnation in lokalen Optima vermeidet. Die Ameisen nutzen die folgende Formel für die lokale Aktualisierung der Pheromonspuren:

$$\tau_{ij}(t) \leftarrow (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0. \tag{9}$$

Kandidatenliste. Für jede Stadt wird eine Kandidatenliste erzeugt, die die c nächsten Nachbarn der Stadt speichert. Eine Ameise wählt nun die nächste Stadt gemäß (5) aus der Kandidatenliste. Wurden bereits alle Städte aus der Kandidatenliste besucht, wandert die Ameise als nächstes in die nächste noch nicht besuchte Stadt.

Ein Vergleich (vgl. Tabelle 2) mit anderen Suchverfahren zeigt nun, dass das ACS auch für größere TSPs (bis zu 100 Städte) in annehmbarer Zeit gute Lösungen findet. Weitere Verbesserungen (z.B. lokale Optimierungen mit dem 3-OPT Verfahren) machen Ameisenalgorithmen auch für TSPs mit bis zu 1000 Städten anwendbar [1].

5 Konvergenzverhalten

Stützle und Dorigo haben die Konvergenz für eine bestimmte Klasse von Ameisenalgorithmen gezeigt [2]. Bei diesen Algorithmen erhöht nur die Ameise, die die global beste Lösung gefunden hat die Pheromonkonzentration und die Pheromonkonzentration ist nach unten begrenzt (ist z.B. bei ACS erfüllt).
Für die Wahrscheinlichkeit P^*, dass ein solcher Algorithmus nach t Iterationen mindestens eine optimale Lösung findet, gilt:

$$\forall\varepsilon\ \exists t : P^*(t) \geq 1-\varepsilon,\ \varepsilon > 0. \tag{10}$$

	ACS beste Lsg.	ACS Iter.	GA beste Lsg	GA Iter.	EP beste Lsg	EP Iter.	SA beste Lsg	SA Iter.
Eil50	425	1830	428	25000	426	100000	443	68512
Eil75	535	3480	545	80000	542	325000	580	173250
KroA100	21282	4820	21761	103000	N/A	N/A	N/A	N/A

Tabelle 2: Die Tabelle (nach [1]) zeigt einen statistischen Vergleich des ACS mit anderen von der Natur inspirierten Verfahren (Genetische Algorithmen, Evolutionäres Programmieren, Simulated Annealing) zur Lösung des TSP. Beim ACS wurden 20 Ameisen und 1250 Iterationen verwendet, damit die Ergebnisse der einzelnen Verfahren vergleichbar sind. Die Tabelle enthält Mittelwerte über jeweils 15 Durchläufe der Algorithmen. Die erste Spalte enthält jeweils die beste gefundene Lösung und die zweite Spalte die Anzahl der Lösungen, die vor der besten Lösung erzeugt wurden. Die verwendeten TSPs bestehen aus 50, 75 bzw. 100 Städten und sind in der TSPLIB [6] verfügbar.

D.h. in Grenzfall für $\varepsilon \to 0$ ($t \to \infty$) wird mit Sicherheit eine optimale Lösung gefunden.

6 Weitere Anwendungen

Diese Ausarbeitung konzentriert sich auf die Anwendung von Ameisenalgorithmen zur Lösung des TSP. In diesem Abschnitt werden nun einige weitere Anwendungen vorgestellt.
Ameisenalgorithmen werden zur Lösung verschiedener kombinatorischer ($\mathcal{NP}$-harter) Optimierungsprobleme eingesetzt. Da sich mit Ameisenalgorithmen Suchprobleme auf Graphen lösen lassen (s. Kap. 3), ähneln die folgenden Beispiele dem TSP sehr:

1. Das quadratische Zuordnungsproblem (Quadratic Assignment Problem QAP): Das QAP stellt die Aufgabe, n Objekte auf n Orte zu verteilen. Dabei ist jede Zuordnung mit Kosten belegt, die jeweils von der Entfernung zwischen den Objekten abhängen.
2. Das Vehicle Routing Problem (VRP): Eine Anzahl von Fahrzeugen muss eine Anzahl von Kunden von einem zentralen Depot aus versorgen. Dies ist mit möglichst geringen Kosten (Wegstrecke, Ladung der Fahrzeuge) zu erreichen.

Praktische Anwendung finden Ameisenalgorithmen außerdem in Routingverfahren für Telefonnetze (AntNet).

Literatur

[1] E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence. From Natural to Artificial Systems. Oxford University Press, 1999.

[2] O. Cordón, F. Herrera, T. Stützle. A Review on the Ant Colony Optimization Metaheuristic: Basis, Models and New Trends. In Mathware & Soft Computing 9 (2002).

[3] J.-L. Deneubourg, S. Aron, S. Goss, J.-M. Pasteels. The Self-Organizing Exploratory Pattern of the Argentine Ant. In Insect Behavior 3 (1990): 159-168

[4] M. Dorigo. Optimization, Learning and Natural Algorithms (auf italienisch). Dissertation, Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT, 1992.

[5] L. M. Gambardella, M. Dorigo. Solving Symmetric and Asymmteric TSPs by Ant Colonies. In Proceeding of IEE Conference on Evolutionary Computation, ICEC96 (1996): 622-627

[6] Bibliothek von Referenz-TSP: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/ (04.01.2004)