Ant Colony Optimierung
und
Machine Layout Probleme

Martin Kunkel

Seminar Optimierung, Sommersemester 2005
Universität Hamburg, Fachbereich Mathematik

betreut von
Prof. Dr. Matthias Gerdts
Prof. Dr. Michael Ulbrich

Inhaltsverzeichnis

1 Einleitung

Ant Colony Optimierung (ACO) ist ein heuristisches Verfahren zur Lösung kombinatorischer Optimierungsprobleme. Man geht hierbei von biologischen Untersuchungen von Ameisen aus und versucht ihre Schwarmintelligenz zu kopieren und in einem Algorithmus zu implementieren. In dem ersten Kapitel möchte ich auf diese Herleitung eingehen und einen ersten ACO Algorithmus zur Pfadfindung auf einem Graphen präsentieren.

Als direkte Anwendung der Ant Colony Optimierung kann man das Traveling Salesman Problem (TSP) bearbeiten. Dies ist ein gutes Training um das Thema zu vertiefen. Es soll an dieser Stelle aber darauf verzichtet werden und ich möchte auf das Buch von Dorigo und Stützle [1] verweisen, in dem das TSP und viele andere Probleme ausführlich behandelt werden.

Im zweiten Teil befasse ich mich mit so genannten Machine Layout Problemen. Diese sind Umordnungsprobleme in denen es darum geht Maschinen in einer Werkshalle optimal zu positionieren.

2 Ant Colony Optimierung

2.1 Künstliche Ameisen

In der Natur hinterlassen Ameisen auf ihrem Weg vom Nest zur Futterquelle Botenstoffe, so genannte Pheromone. Andere Ameisen folgen mit höherer Wahrscheinlichkeit Pfaden, auf denen bereits viele Pheromone liegen. Dadurch findet die Ameisenkolonie häufig den kürzesten Weg zum Futter. Um dies experimentell zu untersuchen kann man das Double Bridge Experiment durchführen (Abbildung 2.1). Man kann so nachweisen, dass fast alle Ameisen nach einiger Zeit auf dem kürzeren Weg laufen. Auch bei gleich langen Wegen konvergiert die Ameisenkolonie in 90 % der Fälle gegen eine Lösung.

Dorigo und Stützle [1] schlagen das nachfolgende Modell für die Ameisenkolonie und das Double Bridge Experiment vor. Das Problem wird als Graph $G = (V, E)$ geschrieben, wobei $V = \{1, 2\}$ die Menge der Knoten ist. Zwischen den Knoten gibt es die Kanten $E = \{l, s\}$ welche den kurzen (s) und den langen (l) Weg zwischen den Knoten bezeichnen. Das Gewicht der Kante s sei 1 und das der Kante l sei $r \in \mathbb{N}$. Die Zeit t wird diskret betrachtet ($t = 1, 2, 3, \ldots$). Es ergibt sich folgendes Modell für

$p_{ia}(t)$ die Wahrscheinlichkeit, dass eine Ameise am Knoten i den Weg a wählt,
$\varphi_{ia}(t)$ die Menge der Pheromone am Knoten i für den Weg a und
$m_i(t)$ die Anzahl der Ameisen im Knoten i.

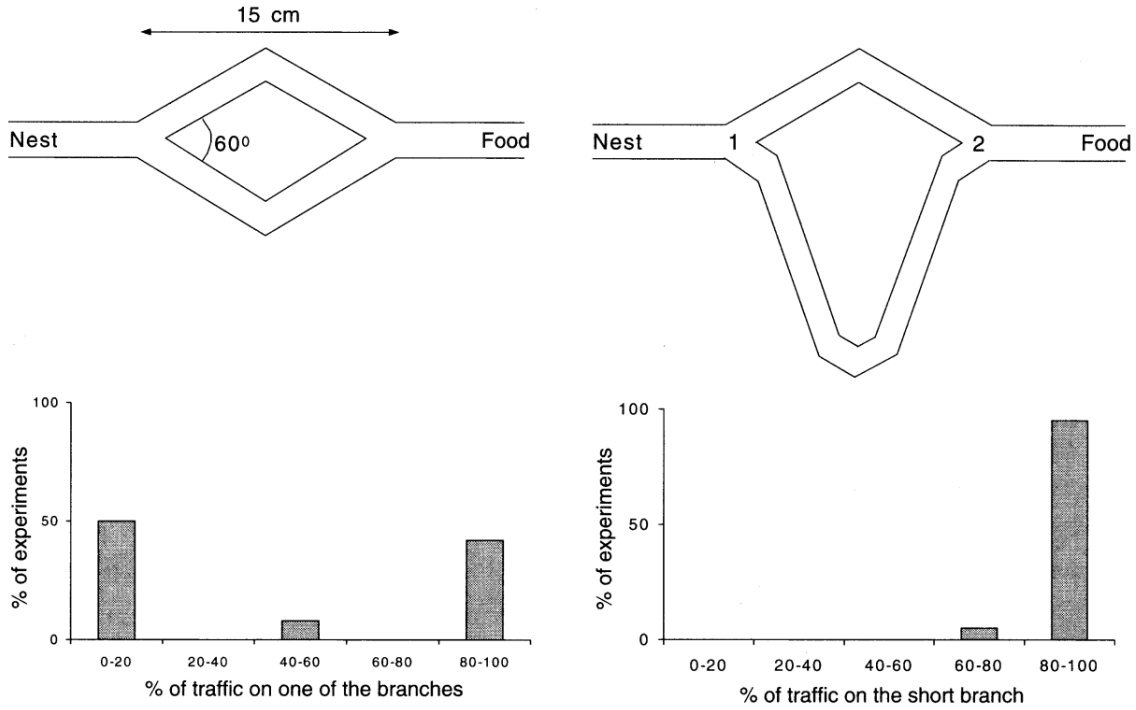

Abbildung 2.1: Double Bridge Experiment: Aufbau und Ergebnisse [1]

$$
\begin{aligned}
&\text{für } (i,j) = (1,2),(2,1): \\
&\qquad m_i(t) \;=\; p_{js}(t-1)m_j(t-1) + p_{jl}(t-r)m_j(t-r) \\
&\qquad \varphi_{is}(t) \;=\; \varphi_{is}(t-1) + p_{is}(t-1)m_i(t-1) + p_{js}(t-1)m_j(t-1) \\
&\qquad \varphi_{il}(t) \;=\; \varphi_{il}(t-1) + p_{il}(t-1)m_i(t-1) + p_{jl}(t-r)m_j(t-r) \\
&\text{für } i = 1,2 \text{ und } a = l,s: \\
&\qquad p_{ia}(t) \;=\; \frac{[\varphi_{ia}(t)]^\alpha}{[\varphi_{is}(t)]^\alpha + [\varphi_{il}(t)]^\alpha}
\end{aligned}
\tag{2.1}
$$

2.2 Kompliziertere Aufgaben

Um dem Fernziel der Lösung kombinatorischer Optimierungsprobleme näher zu kommen untersuchen wir nun beliebige ungerichtete Graphen $G = (V, E)$. Ein Knoten aus V soll hierbei der Startknoten (Nest) und einer das Ziel (Futter) sein. Dabei können neue Probleme auftreten die man sich am Beispiel der Extended Double Bridge (Abbildung 2.2) verdeutlichen kann.

Das erste Problem welches auftritt ist, dass die Ameisen Loops erzeugen können. Dies geschieht dadurch, dass eine Ameise einen Rundweg im Graphen läuft, aber nicht zum Ziel kommt. Auf ihrem Weg hinterlässt sie Pheromone und andere Ameisen folgen dem

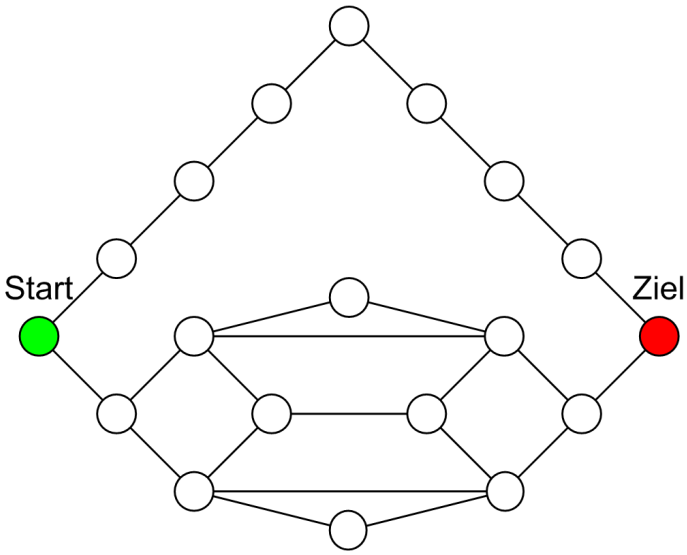

Abbildung 2.2: Extended Double Bridge [1]

Rundweg. Es kann passieren, dass am Ende fast alle Ameisen diesen Weg laufen und nie zum Ziel finden. Eine Lösungsmöglichkeit ist ein Gedächtnis, welches jede Ameise besitzt. Sie kann sich den gelaufenen Weg merken. Auf ihrem Weg zum Ziel (Vorwärts-Modus) hinterlässt die Ameise keine Pheromone. Ihr Weg hat also keinen Einfluss auf andere Ameisen. Am Ziel angekommen entfernt die Ameise alle Loops aus Ihrem Gedächtnis und läuft auf direktem Weg zurück zum Ziel (Rückwärts-Modus). Erst auf dem Rückweg legt die Ameise Pheromone ab. Auch in der Natur hat man festgestellt, dass manche Ameisenarten ein Gedächtnis besitzen [1].

Das zweite Problem ist die schnelle Stagnation auf nicht optimale Wege. Dies tritt beim Problem der Extended Double Bridge auf, da die Ameisen, die beim Start den oberen Knoten wählen, schnell beim Ziel sind und somit Pheromone ablegen können. Ameisen die einen unteren Weg wählen, finden einen besseren Weg, brauchen dafür aber unter Umständen länger. Eine Möglichkeit das Problem zu umgehen ist das Einbauen der Pheromone Evaporation, also dem Verflüchtigen der Botenstoffe. Dies tritt auch in der Natur auf, hat dort aber wenig Einfluss auf die Performance der Kolonie.

Insgesamt ergibt sich der folgende Algorithmus (Simple-ACO) welcher schematisch für jede Ameise in Abbildung (2.3) dargestellt ist.

Das Auswählen des nächsten Knotens erfolgt stochastisch. Die Ameise q befinde sich im Knoten i. Sie wählt den nächsten Knoten j mit der Wahrscheinlichkeit

$$p_{ij}^q = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{l \in \mathcal{N}_i^q} \tau_{il}^\alpha}, & \text{falls } j \in \mathcal{N}_i^q \\ 0, & \text{falls } j \notin \mathcal{N}_i^q \end{cases} \tag{2.2}$$

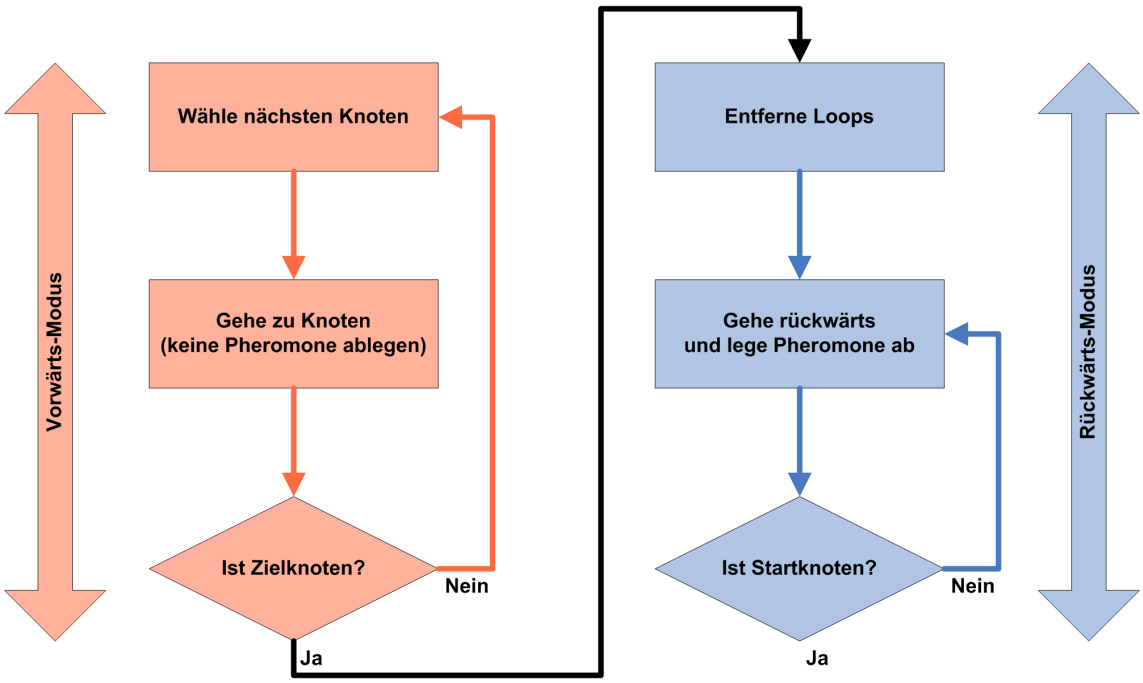

Abbildung 2.3: Simple-ACO für einzelne Ameise

aus, wobei $\mathcal{N}_i^q$ alle Knoten enthält die direkt mit i verbunden sind, aber nicht den zuletzt besuchten Knoten. τ_{ij} bezeichne die Anzahl der Pheromone auf der Kante (i, j). α ist eine zu wählende Konstante (häufig wird diese zu 1 gesetzt).

Für die Erhöhung der Pheromone auf dem Weg vom Ziel zum Start gibt es mehrere Möglichkeiten. Häufig hat sich folgende Methode als effektiv herausgestellt:

$$\tau_{ij} \leftarrow \tau_{ij} + 1/L^q \tag{2.3}$$

wobei L^q die Länge des Weges, den die Ameise gelaufen ist, bezeichnet.

Nachdem alle Ameisen einen Weg konstruiert haben wird die Pheromone Evaporation ausgeführt:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall (i, j) \in V \tag{2.4}$$

Die Konstante ρ muss je nach Problem gewählt werden und hat großen Einfluss auf die Konvergenz des Algorithmus wie in Abbildung (2.4) am Beispiel dargestellt ist.

2.3 ACO Metaheuristik

Ziel der Metaheuristik ist es, alle möglichen Ant Colony Algorithmen in eine allgemeine Form zu bringen. Das Metaheuristic Network [5] benutzt die Formulierung

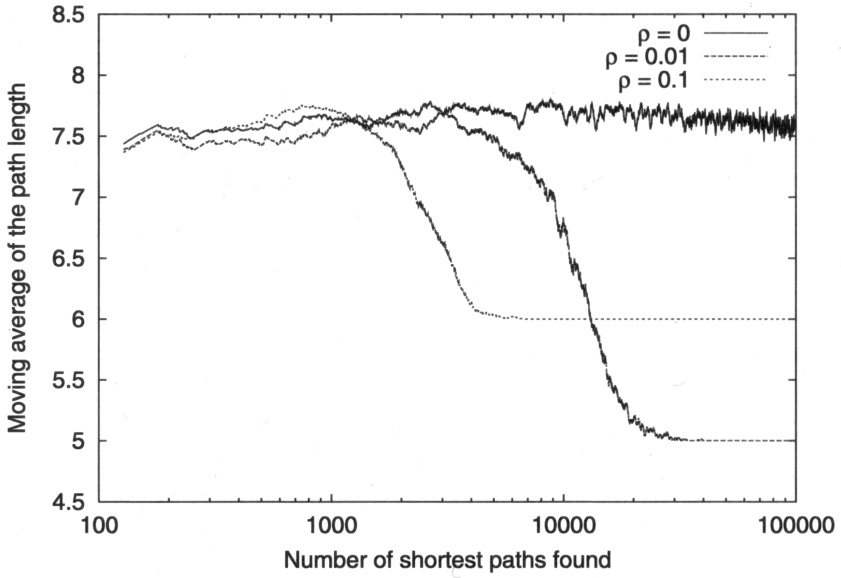

Abbildung 2.4: Verschiedene Parameter ρ am Beispiel der Extended Double Bridge [1]

```
procedure ACOMetaheuristic

    while (termination conditions not met)

        ScheduleActivities
            ConstructAntsSolutions()
            UpdatePheromones()
            DaemonActions()
        end ScheduleActivities

    end while

end procedure
```

`ScheduleActivites` steht für eine beliebige Reihenfolge der folgenden Operationen. `DaemonActions()` können beliebige Operationen sein, die nicht von einer Ameise alleine ausgeführt werden können (zum Beispiel die Pheromone Evaporation).

3 Machine Layout Probleme

Das nun zu untersuchende Problem ist das Robust Layout Machine Problem. In einer Werkshalle sind Maschinen aufgestellt zwischen denen Materialien transportiert werden. Ziel ist es ein neues Layout zu finden, so dass die Kosten minimiert werden. Kosten entstehen dabei durch den Transport der Materialen aber auch durch das Umstellen der Maschinen.

3.1 Modell

Wir beschränken uns auf Probleme die folgende Bedingungen erfüllen:

- Alle auftretenden Abmessungen und Positionen sind natürliche Zahlen und
- die Kosten für das Verschieben sind für jede Maschine konstant.
- Alle Maschinen sind rechteckig,
- es gibt zwei mögliche Ausrichtungen (horizontal und vertikal) für eine Maschine,
- der Abstand zweier Maschinen wird von den Mittelpunkten aus gemessen und
- der Rand der Werkshalle ist rechteckig.

Die im Folgenden verwendeten Bezeichnungen sind der Abbildung (3.1) zu entnehmen. In schwarz ist die aktuelle Position der Maschinen dargestellt, in rot die neue Position.

Das Problem kann als ganzzahliges Optimierungsproblem formuliert werden [2]. Wir benötigen aber nur die Zielfunktion:

$$\min\ C = \sum_{i,j=1}^{N} \epsilon_{ij} F_{ij} \left(|x_i - x_j| + |y_i - y_j| \right) + \sum_{i=1}^{N} A_i I_i \tag{3.1}$$

$$I_i = \begin{cases} 0, & \text{falls } x_i = a_i \text{ und } y_i = b_i \text{ und } Z_i = C_i \\ 1, & \text{sonst} \end{cases} \tag{3.2}$$

A_i sind die Kosten für das Umstellen der Maschine i, F_{ij} die Menge des Materials das von Maschine i zu Maschine j transportiert wird und ϵ_{ij} bezeichnet einen zusätzlichen, optionalen, Kostenfaktor.

Die Randbedingungen (kein Überlappen der Maschinen, kein Aufstellen außerhalb der Werkshalle) werden dadurch eingehalten, dass die Ameisen ein solches Layout nicht erstellen dürfen.

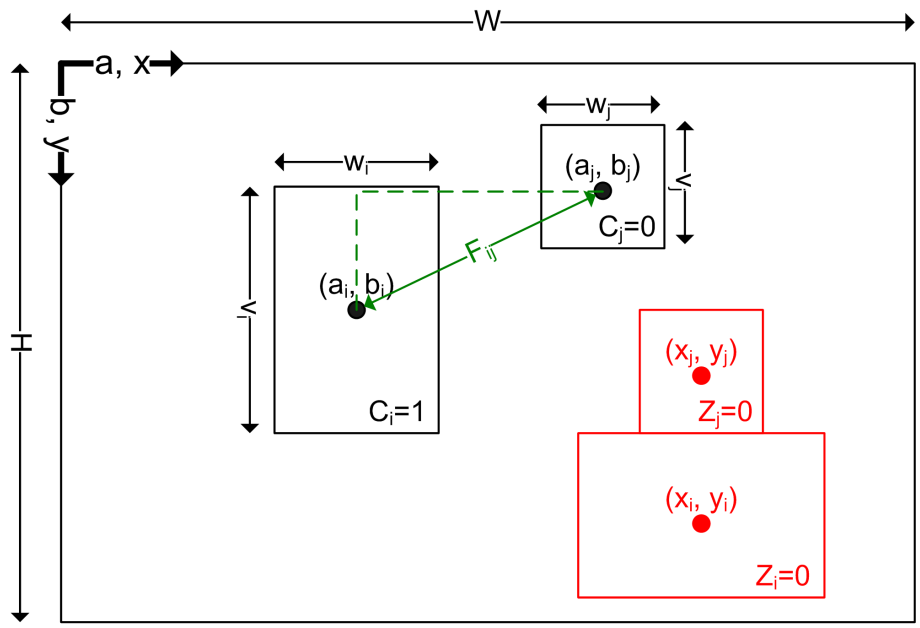

Abbildung 3.1: Bezeichnungen der Variablen des Machine Layout Problems

3.2 Anwenden der Ant Colony Optimierung

Um die Ant Colony Optimierung anzuwenden muss ein Graph erstellt werden, so dass es für jedes zulässige Layout mindestens einen Weg durch den Graphen gibt. Corry und Kozan [2] schlagen folgendes Modell vor (siehe auch Abbildung (3.2)) :

- Jede Maschine ist ein Knoten.
- Zu jeder Maschine gibt es $W \times H$ Gitterknoten, die die Werkshalle darstellen.
- Es gibt Kanten zwischen den Maschinenknoten und von jedem Maschinenknoten zu jedem Gitterknoten.

Insgesamt gibt es in dem Graphen also $N + NWH$ Knoten und $N^2 - N + NWH$ Kanten.

Die Ameisen starten auf einem zufällig ausgewählten Maschinenknoten. Eine Ameise wählt dann iterativ den nächsten Maschinenknoten (3.3) und eine Position für die Maschine (3.4). Auf dem Graphen läuft die Ameise zu der Maschine und anschließend zu jedem Gitterknoten der von der Maschine eingenommen wird. Nachdem alle Ameisen ein Layout erzeugt haben, werden die Pheromone abgelegt (3.5). Die gesamte Prozedur wird dann beliebig oft wiederholt.

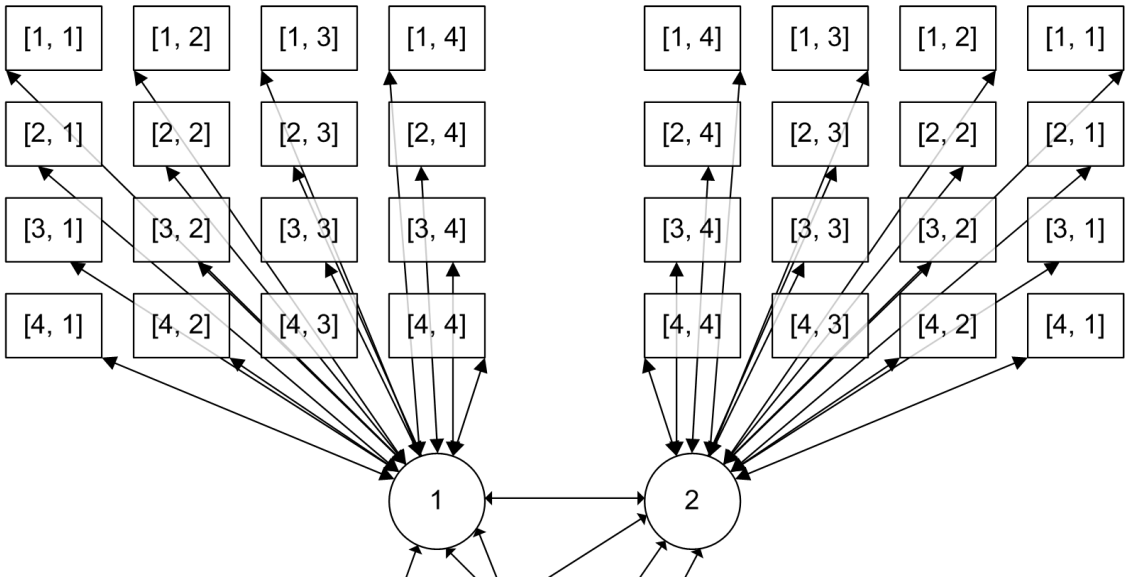

Abbildung 3.2: Teil des Graphen für $W = 4$ und $H = 4$

3.3 Wahl der nächsten Maschine

- Ameise q sei auf Maschinenknoten i.
- Fre_q sei die Menge aller Maschinenknoten die noch positioniert werden müssen.
- $\alpha \geq 0, \beta \geq 0, 0 \leq R_0 \leq 1$ sind Parameter.
- R sei eine Zufallszahl mit $R \sim U[0, 1]$.

Der nächste zu besuchende Maschinenknoten sei j. Dieser wird entweder deterministisch oder stochastisch ausgewählt:

$$j = \begin{cases} \arg\max_{k \in Fre_q} (trail_M(i,k))^\alpha H_M(i,k)^\beta, & \text{falls } R \leq R_0 \\ \\ J, & \text{sonst} \end{cases} \tag{3.3}$$

wobei $J \in Fre_q$ zufällig ausgewählt wird mit der Wahrscheinlichkeit

$$p_q^M(i,J) = \frac{(trail_M(i,J))^\alpha H_M(i,J)^\beta}{\sum_{k \in Fre_q} (trail_M(i,k))^\alpha H_M(i,k)^\beta} \tag{3.4}$$

$trail_M(i,j)$ sei die Anzahl der Pheromone die auf der Kante zwischen den Maschinen i und j liegen. H_M ist eine Heuristik, für die man

$$H_M(i,j) = \sum_{k \in L_q} \epsilon_{kj} F_{kj} \tag{3.5}$$

verwenden kann, wobei L_q die Menge der Maschinenknoten ist, die bereits positioniert sind. Man bevorzugt dadurch Maschinen, die einen großen Materialfluss zu den bereits positionierten Maschinen haben.

3.4 Wahl der Position

- Sei Vnt_q die Menge der freien Gitterknoten.
- Sei $M(i, Lcn)$ die Menge der Gitterknoten die die Maschine i an der Position Lcn einnehmen würde.

Eine Position Lcn sei das Tupel aus Position in der Ebene und Orientierung.

Die Position für Maschine i sei Lcn_i. Diese wird entweder deterministisch oder stochastisch ausgewählt:

$$Lcn_i = \begin{cases} \underset{\substack{Lcn \\ M(i,Lcn)\subseteq Vnt_q}}{\arg\max} \quad (trail_P^{avg}(i, Lcn))^\alpha H_P(i, Lcn)^\beta, & \text{falls } R \leq R_0 \\ \\ LCN, & \text{sonst} \end{cases} \tag{3.6}$$

wobei LCN mit $M(i, LCN) \subseteq Vnt_q$ zufällig ausgewählt wird mit der Wahrscheinlichkeit

$$p_q^P(i, LCN) = \frac{(trail_P^{avg}(i, LCN))^\alpha H_P(i, LCN)^\beta}{\sum_{M(i,Lcn)\subseteq Vnt_q}(trail_P^{avg}(i, Lcn))^\alpha H_P(i, Lcn)^\beta} \tag{3.7}$$

Die Pheromone $trail_P^{avg}(i, Lcn)$ für eine Position werden ermittelt aus der durchschnittlichen Anzahl der Pheromone aller Gitterknoten die von der Maschine i an der Position Lcn eingenommen werden:

$$trail_P^{avg}(i, Lcn) = \frac{\sum_{[x,y]\in M(i,Lcn)} trail_P(i, [x, y])}{v_i w_i} \tag{3.8}$$

Als Heuristikfunktion H_P kann man

$$H_P(i, Lcn) = \frac{1}{\sum_{k\in L_q} \epsilon_{ki} F_{ki}[|x_k - x_i| + |y_k - y_i|] + \sum_{k\in R_i} A_k} \tag{3.9}$$

mit

$$R_i = \{k \quad | \quad k \notin L_q; M(k, Lcn_{k0}) \subseteq Vcn_q; \quad M(k, Lcn_{k0}) \cap M(i, Lcn) \neq \emptyset \quad \} \tag{3.10}$$

verwenden. Es werden Positionen bevorzugt die die Zielfunktion minimieren. Der Term $\sum_{k\in R_i} A_k$ bestraft Positionen die andere, noch nicht positionierte, Maschinen zwingen würden umgestellt zu werden.

Die Bestimmung der freien Positionen Vnt_q ist aufwendig, kann aber effektiv gelöst werden, indem man das Gebiet in Blöcke unterteilt und nach dem Positionieren einer Maschine die Blöcke aktualisiert. Der genaue Algorithmus kann bei Corry und Kozan [2] nachgelesen werden.

3.5 Pheromone Ablage und Pheromone Evaporation

Nachdem alle Ameisen ein Layout erzeugt haben setze für alle Kanten (i, j):

$$trail_X(i,j) \rightarrow \min\{\tau_0, \underbrace{(1-\rho)trail_X(i,j)}_{\text{pheromone evaporation}} + \rho\Delta trail(i,j)\} \tag{3.11}$$

mit

$$\Delta trail(i,j) = \sum_{q \in K_{ij}} \frac{U}{C_q} + e_{ij}\frac{U}{C_{best}} \tag{3.12}$$

wobei

$$\begin{aligned} K_{ij} &= \{q|(i,j) \text{ ist Teil der Lösung von Ameise } q\} \\ e_{ij} &= \begin{cases} e, & \text{falls } (i,j) \text{ Teil der bisher besten Lösung ist} \\ 0, & \text{sonst} \end{cases} \end{aligned} \tag{3.13}$$

ρ, τ_0 und U sind zu wählende Parameter. C_q sind die Kosten für das Layout der Ameise q. C_{best} sind die Kosten für das bisher beste gefundene Layout.

3.6 Flexible Layout Machine Probleme

Als Erweiterung seien noch die Flexiblen Layout Machine Probleme erwähnt. Dabei werden Situationen betrachtet in denen man über mehrere Zeitintervalle mit unterschiedlichen Materialflüssen eine optimale Layoutstrategie sucht. Die Maschinen können jeweils am Anfang eines Zeitintervalls umgestellt werden. Die Anwendung der Ant Colony Optimierung kann ebenfalls bei Corry und Kozan [2] nachgelesen werden. Beispielprobleme für eigene Implementationen haben Yang und Peters [3] zusammengestellt und mit einer anderen Methode bearbeitet.

4 Fazit

Ein Nachteil der Ant Colony Optimierung ist die schwierige theoretische Untersuchung des Algorithmus. Aktuell sind Konvergenzbeweise nur für sehr spezielle ACO Algorithmen bekannt [1]. Diese können aber nur Aussagen treffen, dass der Algorithmus mit Wahrscheinlichkeit gegen 1 konvergiert, also dass

$$\lim_{\theta \to \infty} P^*(\theta) = 1 \tag{4.1}$$

ist. $P^*(\theta)$ sei dabei die Wahrscheinlichkeit, dass eine Ameise innerhalb der ersten θ Iterationen die globale Lösung konstruiert. Wichtig ist anzumerken, dass dieser Konvergenzbeweis keine Aussage über die Dauer bis zum Erreichen der Lösung liefert.

Schwierig ist auch das Wählen der verschiedenen Parameter. Sinnvoll ist sicherlich den Algorithmus mehrmals zu starten mit unterschiedlichen Parametern.

Die Performance der Ant Colony Optimierung ist gut für viele Probleme, insbesondere für das Traveling Salesman Problem [1]. Für die Machine Layout Probleme lieferte die Ant Colony Optimierung den bisher besten Algorithmus [2].

Literatur

[1] MARCO DORIGO, THOMAS STÜTZLE *Ant Colony Optimization*, Massachusetts Institute of Technology, 2004

[2] PAUL CORRY, ERHAN KOZAN *Ant Colony Optimisation for Machine Layout Problems*, Computational Optimization and Applications, 28, 287-310, 2004

[3] TAHO YANG, BRETT A. PETERS *Flexible machine layout design for dynamic und uncertain production environments*, European Journal of Operational Research, 108 (1998) 49-64

[4] E.A. SILVER, R. PETERSON *Decision Systems for Inventory Management and Production Planning*, Wiley, 1985

[5] *Metaheuristic Network project*, `http://www.metaheuristics.org`