TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

Fakultät für Mathematik und Informatik
Institut für Informatik
Lehrstuhl für Künstliche Intelligenz und Datenbanken

Bakkalaureatsarbeit

Eine Studie zur kombinatorischen Optimierung mit Ameisenalgorithmen

Samuel Dressel

Angewandte Informatik
Vertiefung: Künstliche Intelligenz

Matrikel: 59 963

05. November 2018

Betreuer/1. Korrektor:
Prof. Dr. H. Jasper

2. Korrektor:
M. Sc. V. Göhler

Inhaltsverzeichnis

1 Einleitung

2 Grundlagen

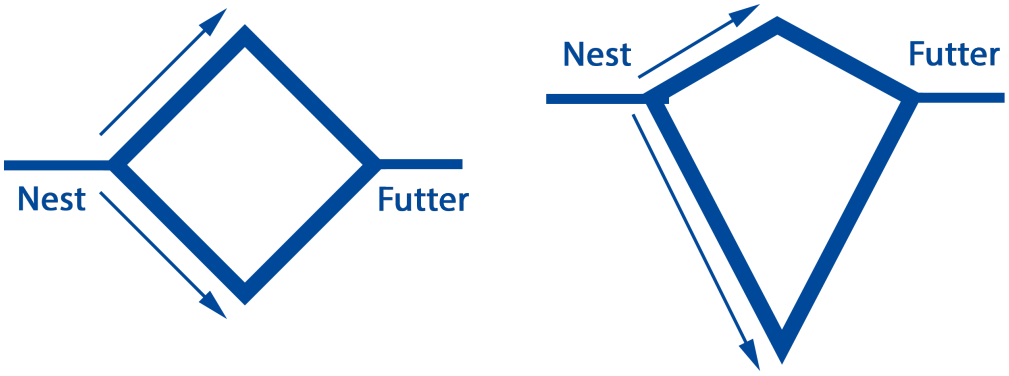
2.1 Der Ameisenalgorithmus (Ant Colony Optimization)

2.1.1 Biologische Grundlagen

Grundlage der Erörterung über die Funktionsweise des Ameisenalgorithmus bildet sicherlich ein Blick auf die biologischen Grundlagen. Dabei spielt die Kommunikation der Ameisen untereinander die zentrale Rolle. Ein Tierstaat wie er bei den Ameisen zu finden ist funktioniert nur mit einer effektiven und sinnvollen Kommunikation. Methoden zur Verständigung wie das Kommunizieren über Vibrationen und Berührungen sind eher die Ausnahme und kommen nur in speziellen Situationen zum Tragen ([1]). Dagegen wird zum größten Teil der Informationsaustausch über Duftstoffe (sog. Pheromone) bevorzugt. Diese werden durch verschiedene Drüsen erzeugt und wiederum in unterschiedlicher Kombination und Konzentration abgegeben. Diese Pheromone werden zum einen benutzt, um Nestgenossen zu erkennen oder um bei Gefahren Kampf- und Abwehrverhalten auszulösen. Hauptsächlich jedoch nutzen Ameisen die Pheromone um eine Duftspur über ihren Hinterleib abzugeben. Diese dienen ihnen und ihren Nestgefährten als Orientierungshilfe. Zum einen werden damit Straßen zu anderen Kolonien gebildet - zum anderen dient es dazu, anderen Ameisen den Weg zu einer Nahrungsquelle zu zeigen. Die Tatsache, dass ein Weg mit einer höheren Pheromonkonzentration bevorzugt wird, ist Grundlagen des Ameisenalgorithmus.

2.1.2 Der Ameisenalgorithmus

Der historische Ursprung des Algorithmus findet sich in den Versuchen von Jean-Louis Deneubourg und seine Kollegen ([2]). Das sogenannte "Double-Bridge-Experimentßeigte, dass Ameisen den kürzesten Weg aufgrund der Pheromonmarkierung finden. In dem Experiment ist eine Kolonie von Argentinischen Ameisen durch zwei Brücken mit einer Nahrungsquelle verbunden ([3]). Dabei können die Ameisen die Futterquelle nur über diese zwei Brücken erreichen. Im ersten Teil des Versuchs sind diese beiden Brücken jeweils gleich lang (Abbildung 1a). Zu Beginn erkunden die Ameisen die Umgebung der Kolonie bis sie eine Entscheidung über die Auswahl der Brücke treffen müssen. Lässt sich aufgrund einer noch nicht stattgefundenen Begehung der Brücken keine Pheromonspur feststellen, so entscheiden die Ameisen rein zufällig welche Brücke sie wählen. Die Wahrscheinlichkeit für beide Wege liegt bei gleichen Bedingungen bei ca. 50 Prozent. Wird der Versuch über längere Zeit durchgeführt, so wird durch Zufall die Pheromonkonzentration der einen Brücke höher sein als die andere. Diese wird dadurch attraktiver für die Ameisen und wird somit letztenendes der favorisierte Weg zur Nahrungsquelle. Im zweiten durchgeführten Versuch sind die beiden Brücken unterschiedlich lang (Abbildung 1b). Auch hier liegt die Wahrscheinlichkeit für beide Brücken zu Anfang bei ca. 50 Prozent. Da die Ameisen, die sich für die kürzere Brücke entscheiden, schneller wieder zuirück am Nest sind, steigt das Pheromonlevel auf diesem Weg deutlich schneller an. Nach einigen Iterationen kristalliert sich die kürzere Brücke als optimale Route heraus, während der längere Weg durch Pheromonreduktion immer unattraktiver wird. Im Vergleich zum ersten Versuch geschieht dieser Prozess wesentlich schneller und effektiver.



(a) Skizze zu Versuch 1; beide Wege sind gleich lang

(b) Skizze zu Versuch 2; die Wege sind unterschiedlich lang

Abb. 1: Double-Bridge-Experiment

2.2 Das Travelling-Salesman-Problem

2.2.1 Das Problem im Allgemeinen

Das Travelling Salesman Problem (im Folgenden mit TSP abgekürzt) ist eines der bekanntesten und meist untersuchten Optimierungsprobleme ([4]). Die Aufgabe, mit der sich das Problem beschäftigt, ist folgende: Ein Handlungsreisender soll in einer Rundreise n verschiedene Städte besuchen. Der Reisende startet dabei (zufällig) in einer dieser Städte und am Ende seiner Reise kehrt er auch wieder in diese Stadt zurück. Dabei sollen alle Städte nur einmal besucht werden und dabei die Weglängen bzw. die Kosten minimiert werden.
Formal lässt sich dieses Problem mathematisch so definieren: $G = (V, E)$ sei Graph und F die Menge aller Hamiltonkreise in G. Dabei beschreibt $V = \{1, ..., n\}$ die Menge der Knoten, E die Menge der Kanten. Für jede Kante $e \in E$ existierten ein Wert c_e, der die Gewichtung/die Kosten für diese Kante beschreibt. Diese Kosten als Kostenmatrix $C = (c_{ij})_{n \times n}$ dargestellt werden, dabei entspricht jeder Eintrag c_{ij} den Kosten der Kante, die den Knoten i mit dem Knoten j verbindet. Ziel ist das Finden des Hamiltonkreises $f \in F$, für den die Summe der einzelnen Kantenkosten minimal wird ([5]).
Seinen geschichtlichen Ursprung hat das TSP im Jahr 1832, als in Deutschland ein Buch mit dem Titel "Der Handlungsreisende, wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolges in seinen Geschäften gewiss zu seinërschien. Dieses Buch und dessen Inhalt definierte die Grundlagen des Problems. Die erste Benutzung des Ausdrucks Traveling Salesman Problemïst nicht ganz geklärt; in mathematischen Kreisen war dies in den Jahren 1931 - 1932 der Fall. ([6]). Dies geschah als mehrere amerikanische Mathematiker sich des Problems annahmen - wichtige Vertreter waren dabei Merrill FLood und Hassler Whitney, die dort die Überlegungen des österreichischen Mathematikers Karl Menger als Grundlage nahmen. Nach und nach wurde das Problem durch zahlreiche Veröffentlichungen immer präsenter und bis heute ist das TSP eines der schwierigsten und prominentesten Probleme in der Mathematik.
Unter der bislang unbewiesenen Annahme, dass die Komplexitätsklassen P und NP verschieden sind,gehört das TSP zur Klasse der NP-vollständigen Probleme, was bedeutet dass es sich nicht mit einem deterministischen Algorithmus in Polynomialzeit lösen lässt ([7]). Al-

le ansatzweise effektiven Möglichkeiten und Algorithmen zur Lösung des TSP sind deshalb nichtdeterministisch.

2.2.2 Ansätze und Algorithmen zur Lösung des Problems

Da das Problem wie oben schon erwähnt zu den NP-vollständigen Problemen gehört, gibt es keinen effizienten Algorithmus der bei einer hohen Anzahl von Städten eine optimale Tour findet. ([6, p. 150]). Entweder ist ein Algorithmus zur Lösung des TSP schnell oder er findet eine optimale Tour - aber er wird nicht beide Eigenschaften besitzen. Deswegen unterscheidet man wie auch allgemein bei anderen kombinatorischen Optimierungsproblemen zwischen exakten und heuristischen Algorithmen. Der einfachste Algorithmus zur exakten Lösung des TSP ist die sogenannte *"brute-force"* oder naive Methode ([4]). Dieser Algorithmus betrachtet nacheinander alle möglich Touren und deren Länge und ermittelt durch den Vergleich derselben die optimale und kürzeste Tour. Ist der Graph G, der dieses Problem modelliert, ein ungerichteter Graph, so muss man mit diesem Algorithmus $\frac{1}{2} \cdot (n-1)!$ verschiedene Rundreisen betrachten. Bei neun verschiedenen Städten ergeben sich daraus 20160 verschiedene Touren, bei 16 Städten dagegen schon 653 Milliarden Routen, was diesen Algorithmus für die meisten Optimierungsprobleme völlig unbrauchbar macht. Auch andere exakte Methoden haben dennoch oft einen hohen Rechen- und Zeitaufwand. Man entscheidet sich deswegen häufig dafür effiziente heuristische Algorithmen zu konstruieren, die zwar keine optimale Tour finden, aber zumindest eine annäherungsweise optimale Tour. Es existieren eine Vielzahl von solchen heuristischen Verfahren - eines der intuitivsten ist dabei der *Nearest-Neighbor-Algorithmus* ([8]). Hierbei wird ein zufälliger Knoten v_1 als Startknoten asugewählt. Danach wird iterativ immer der Knoten v_i ausgewählt, der dem zuletzt ausgewählten Knoten am nächsten liegt und noch nicht in der schon besuchten Knotenmenge V' enthalten ist. Der Algorithmus ist beendet, wenn alle Knoten besucht wurden. Das Problem hierbei ist, dass die letzte Kante, die den letzten Knoten mit dem Startknoten verbindet und den Hamiltonkreis vervollständigt, eine mehr oder weniger beliebige Länge haben kann und somit die Optimalität der gefundenen Lösung wesentlich einschränkt. Formal lässt sich der Algorithmus wie folgt darstellen: Ein

Algorithm 1 Nearest-Neighbor-Algorithm

Eingabe: $G = (V, E, c)$
Ausgabe: Tour V_C, die alle Knoten $v \in V$ besucht

1: $z_1 := v \in V$
2: $V' := V \setminus \{v_1\}$
3: $i := 2$
4: **while** $V' \neq \emptyset$ **do**
5: $\quad z_1 := min_{v \in V'}\, c(\{v_{i-1}, v\})$
6: $\quad V' := V' \setminus \{v_i\}$
7: $\quad i := i + 1$
8: $V_C := (v_1, ..., v_n, v_1)$

weiterer bekannter heuristischer Algorithmus ist die *Minimal-Spaaning-Tree-Heuristik, kurz MST*. Dabei wird zunächst ein minimaler Spannbaum B für den Graphen G ermittelt ([9]). Zur Ermittlung dieses minimalen Spannbaumes stehen mehrere Algorithmen zur Verfügung (Algorithmus von Prim, Algorithmus von Kruskal, ...) ([10]). Danach wird jede Kante innerhalb des Spannbaumes verdoppelt; es entsteht eulersche Graph G'. Nun wählt man einen beliebigen Startknoten und folgt den Kanten im Sinne einer Eulertour. Bereits besuchte Kanten werden dabei gestrichen und stattdessen die direkte Verbindung zwischen den jeweils

verbleibenden Knoten gewählt. Das *Verfahren von Christofides* baut auf diesem Algorithmus und erweitert diesen dadurch, dass der minimale Spannbaum nicht verdoppelt wird, sondern das minimale Matching von den Knoten in B sucht, die einen ungeraden Grad haben. Nach der Dreiecksungleich gilt, dass die Summe der Längen zweier Seiten a und b stets mindestens so groß ist wie die Länge der dritten Seite (formal: $c \leq a+b$). Die MST-Heuristik ist deswegen höchstens doppelt so lang, die Christofides-Heuristik höchstens höchstens 1,5-mal so lang wie die optimale Lösung([9]).

2.2.3 TSPLIB als Quelle für bekannte Probleme

Die in dieser Arbeit untersuchten Probleme stammen aus der TSPLIB der Universität Heidelberg. Bekannte Probleme wurden dort in ein einheitliches Datenformat gebracht und eignen sich deswegen gut zur Auswertung von Implementierungen des TSP. Vor allem die Existenz der Daten als XML-Datei (Extensible Markup-Language-File) vereinfacht die Anwendung der Lösungsalgorithmik auf viele verschiedene Probleme sehr einfach ([11]).

2.2.4 Möglichkeiten der Distanzberechnung

Um das TSP lösen zu können benötigt man Informationen über die Distanzen bzw. die Wegkosten zwischen den einzelnen Knoten. In der TSPLIB (2.2.3) sind dabei die Kantenwerte schon berechnet worden. Dieser Berechnung hängt vom Format der Positionsinformationen der einzelnen Knoten ab. Die zwei wichtigsten Distanzarten, die auch den Daten in dieser Arbeit zu grade liegen, sind die *Euklidische Distanz* und die *Geographische Distanz*. Die Euklidische Distanz (auch euklidischer Abstand) ist der triviale Abstand zwischen zwei Punkten den man auch durch Messung mit einem einfachem Längenmessgerät wie einem Lineal ermittelt ([12]). Die Distanz d_{xy} der Punkte x und y in einer Ebene mit den Koordinaten $x = (x_1, x_2)$ und $y = (y_1, y_2)$ ergibt sich dabei aus folgender Formel:

$$d_{xy} = \|x - y\|_2 = \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2} \tag{1}$$

Die Ermittlung der geographischen Distanz ist dagegen etwas komplizierter. Dies hat die Ursache, dass die Koordinaten im Fall der Erde auf einer Kugel liegen. Betrachtet man die Erde als ideale Kugel mit einem Radius $r = 6378,388\,km$ und liegen die Koordinaten in der Dezimalschreibweise vor, ergibt sich für die für zwei Städte i und j folgende Berechnung:

Algorithm 2 Geographische Distanz

1: $r = 6378,388$
2: $latitude_i = latitude_i \cdot \pi/180$; $latitude_j = latitude_j \cdot \pi/180$
3: $longitude_i = longitude_i \cdot \pi/180$; $longitude_j = longitude_j \cdot \pi/180$
4: $q_1 = cos(longitude_i - longitude_j)$
5: $q_2 = cos(latitude_i - latitude_j)$
6: $q_3 = cos(latitude_i + latitude_j)$
7: $d_{ij} = r * arccos(0,5 \cdot ((1 + q_1) \cdot q_2 - (1 - q_1) \cdot q_3)) + 1)$

Literatur

[1] Christian Dietrich and Erich Steiner. Das Leben unserer Ameisen - ein Überblick. *Biologiezentrum Linz/Austria.*

[2] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of lnsect Behavior, Vol. 3, No. 2, 1990*, 1989.

[3] M. Dorigo. Ant colony optimization. 2007.

[4] Berthold Vöcking, Helmut Alt, Martin Dietzfelbinger, Rüdiger Reischuk, Christian Scheideler, Heribert Vollmer, and Dorothea Wagner. *Taschenbuch der Algorithmen*, chapter Das Travelling Salesman Problem, pages 413–422. Springer-Verlag Berlin Heidelberg, 2008.

[5] G. Gutin and A. P. Punnen. *The Traveling Salesman Problem and Its Variations.* Kluwer Academic Publishers, 2002.

[6] E.L. Lawler, J.K.Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization.* John Wiley and Sons Ltd., 1985.

[7] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The traveling salesman problem: a computational study.* Princeton University Press, 2007.

[8] S. Lotz. Lösung und graphische Darstellung des Traveling Salesman Problems in einer Webapplikation. 2014.

[9] M. Grötschel. Schnelle Rundreisen: Das Travelling Salesman-Problem. *ZIB-Report 05-57*, 2005.

[10] Chapman & Hall/CRC. *Spanning Trees and Optimization Problems.* Bang Ye Wu, Kun-Mao Chao, 2004.

[11] Gerhard Reinelt. TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 1991.

[12] Elena Deza and Michel Marie Deza. *Encyclopedia of Distances.* Springer-Verlag Berlin Heidelberg, 2009.