# Inhaltsverzeichnis

# 1 Einleitung

Beim der Entwickelung von Algorithmen stellt sich oft die Frage nach der besten Lösung. Es gibt tausende von Möglichkeiten ein gegebenes Problem anzugehen. Eine der besten Inspirationsquellen ist oft die Natur selbst, mit ihren erprobten Methoden liefert sie oft Vorbilder. Eins dieser Vorbilder sind Ameisenkolonien, sie liefern Ideen für das Kommunizieren, für das konstruieren und eben auch für das Wegfinden.

Ant Colony Optimization nimmt ein Problem als Graphen und findet mögliche Lösungen zu einem Problem. Hierbei hat die Futtersuche der Ameisen als Inspiration gedient, so hinterlassen Ameisen auf der Suche nach Nahrung Pheromonen, denen dann wiederum anderen Ameisen folgen. Dadurch können durch relative Simple Abfolgen komplexe Ziele erreicht werden.

Aber warum jetzt Wegfindung in Aufbausimulationen? In Aufbausimulation müssen oft Wege zwischen verschiedenen Punkten gefunden werden. Daher bietet sich hier ACO an. Besonders oft werden Waren zwischen unterschiedlichen Produktionsstätten transportiert werden und dann ist der Träger nichts anderes als die Ameise, der Startpunkt der Ameisenbau und das Ziel die Nahrung. Aufgrund der Iterativen Natur ACO bleiben meine Routen nicht statisch, aber erprobte Wege bleiben erhalten.

# 2 Der Algorithmus

## 2.1 Die Geschichte

Der Ant Colony Optimization Algorithmus wurde erstmalig von Marco Dorigo in seiner Doktorarbeit 1992 vorgestellt. Diese Variante ist als das Ant System (AS) bekannt. Seitdem gab es einige Weiterentwicklungen des ursprünglichen Algorithmus, wie zum Beispiel das Ant Colony System oder das Elitist Ant System. Ursprünglich wurde das Ant System anhand des travelling salesman problem (TSP) (s. 2.4) vorgestellt.

## 2.2 Graphen

Als Graphen versteht man eine Anzahl an Knoten($V_x$) und Kanten($E_0$). Eine Kante zwischen zwei Knoten wird als ($V_a$,$V_b$) bezeichnet, wobei $V_a$ und $V_b$ die beiden verbundenen Knoten sind(Abbildung 1). Werden den Kanten eine Richtung bzw. beide zugewiesen so spricht man von einem gerichtetem Graph. Weiterhin kann den Kante oder den Knoten ein Gewicht, also den Kostenmultiplikator für die jeweilige Kante bzw. Knoten. Bei ersterem spricht man von einem kantengewichteten Graph, bei letzterem von einem knotengewichtetem Graph.

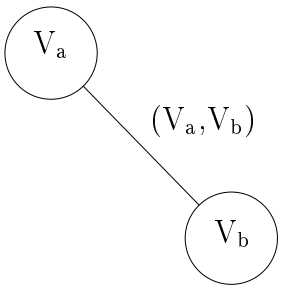


Abbildung 1: Einfacher Graph

Zwei Knoten gelten als benachbart wenn es ($V_a$,$V_b$) oder ($V_b$,$V_a$) gibt.

## 2.3 Die Kernidee

"*In ACO, a number of artificial ants build solutions to an optimization problem and exchange information on their quality via a communication scheme that is reminiscent of the one adopted by real ants.*"[4][1]. Die Ant Colony Optimization besteht grob aus zwei Bestandteilen, der Wahrscheinlichkeit für eine gegebene Kante und der Pheromonenupdateregel. Die hier vorgestellten Formel entsprechen der des Ant Systems (AS) (s. 2.1).

### 2.3.1 Wahrscheinlichkeit für eine Kante

Die Wahrscheinlichkeit für eine Kante gibt an wie hoch die Wahrscheinlichkeit (p) ist, dass eine gegeben Ameise ($k$) die Kante ($V_x$,$V_y$), wobei $V_x$ die momentane Position ist. Sie setzt sich aus der Menge an Pheromonen auf der Kante ($\tau$) und der Effektivität der Kante ($\eta$) zusammen $P_{xy}^k = \left(\tau_{xy}^\alpha\right) * \left(\eta_{xy}^\beta\right)$. Der Wert wird dann in einen prozentualen Anteil umgerechnet, indem $P_x^k y$ durch die Summen der Wahrscheinlichkeiten aller möglichen Kanten ($V_x$,$V_z$) mit $V_z$ als $V_z \in Nachbarn_x$.Daher ist:

$$p_{xy}^k = \frac{P}{\sum_{z \in Nachbern_x} P_{xz}^k}$$

### 2.3.2 Die Pheromonenupdateregel

Der zweite Teil ist die Pheromonenupdateregel. Sie ist für das Aktualisieren aller Kanten zuständig. Daher hat diese Regel einen großen Einfluss auf das Verhalten der Ameisen.Die einfachste Regel besagt:

$$tau_{xy} = (1 - \rho) * \tau_{xy} + \sum_{k}^{m} \Delta\tau_{xy}^k$$

Wobei $\rho$ angibt wie viel von dem Duftstoff verdampft, $m$ die Anzahl an Ameisen ist und $\Delta\tau_{xy}^k$ die Menge an Pheromonen ist, die die Ameise $k$ auf der Kante ($V_x$,$V_y$) hinterlassen hat. Dementsprechend ergibt sich $\Delta\tau_{xy}^k$ wie folgt wenn die Ameise $k$ über die Kante gegangen ist:

$$\Delta\tau_{xy}^k = Q/L_k$$

Wenn die Ameise nicht über die Kante gegangen ist, dann ist:

$$\Delta\tau_{xy}^k = 0$$

## 2.4 Das travelling salesman problem

Das travelling salesman problem (in deutsch: Reisehändlerproblem) ist ein relativ bekanntes Problem in der Computertechnik. Es stellt die Frage: "*Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?*"[8][2]. Der Ursprung dieser Fragestellung ist unbekannt.

Soll dieses Problem mit Hilfe von AS gelöst werden, so entspricht die Stadt einem Knoten und alle Knoten sind untereinander verbunden. Danach können die oben beschrieben Regeln angewandt werden (s. 2.3) Als Beispiel soll der Weg in einem Netzwerk mit sechs Städten gefunden werden, der Graph sähe dann wie in Abbildung 2 aus. An jeder Stadt wird eine Ameise gestartet. Der kürzeste Weg [6] sieht dann wie folgt aus 2 (rote Linien).

[1] DE: Bei dem Ant Colony Optimization Algorithmus, entwickeln künstliche Ameisen eine Lösung zu einem Optimierungsproblem und kommunizieren die Qualität ihrer Lösung via eines Verfahrens, welches von Ameisen adaptiert wurde.

[2] DE: Wenn eine List an Städten und deren Entfernungen zueinander bekannt ist, was ist dann der kürzeste Weg um alle Städte genau einmal zu besuchen und zum Anfang zurück zu kehren?

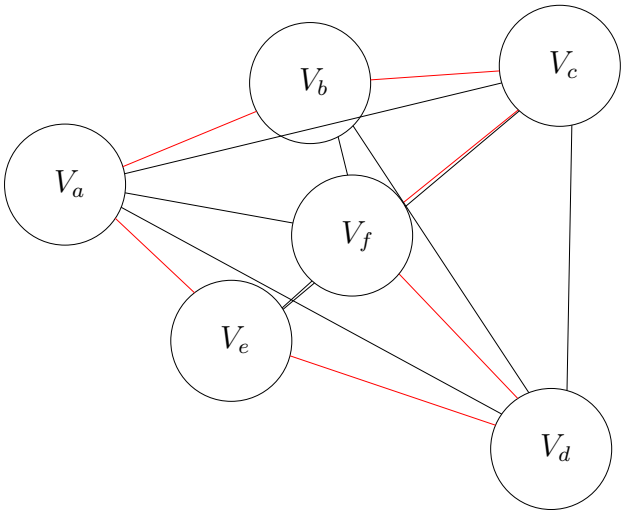


Abbildung 2: Netzwerk an Städten

### 2.5 Die Anwendung

Nachdem nun der theoretische Teile betrachtet wurde, sind natürlich auch die tatsächlichen Anwendungen für ACO interessant. Einer der offensichtlichen Bereiche ist, der der Wegfindung. So wird ACO unter anderem in der Netzwerktechnik genutzt um Wege für die Pakete zu finden. Aber auch in den Naturwissenschaften findet ACO Anwendung, als Beispiel in der Biologie zur Proteinfaltung oder in der Nanotechnik.

## 3 Anwendungen im Aufbauspiel

Nachdem nun der Algorithmus betrachtet wurde, folgt jetzt die Übertragung auf den Bereich der Aufbauspiele.

### 3.1 Das Problem

Ein Aufbauspiel soll aus zwei Teilen bestehen. Der erste sind die Gebäude die Waren verbrauchen und produzieren. Sie bestehen aus einer Liste an benötigten und produzierten Waren, einer Produktionsgeschwindigkeit, die angibt nach welcher Zeit die produzierten Waren, wenn die benötigten Waren vorhanden sind und im Inventar platz für die produzierten ist, in das Inventar hinzugefügt werden. Weiterhin bestehen Gebäude aus einer Inventurkapazität. Der zweite Teil sind die Waren selbst. Sie bestehen nur aus ihrem Volumen, daher wie viel Platz sie in dem Inventar einnehmen.

Als Ausgangslage dient das bauen von Produktionsketten. Da es unterschiedlichste Ketten mit variierender Komplexität gibt, soll Beispielhaft nur die der Bretterproduktion betrachtet werden.

Daher ergeben sich Zwei notwendige Gebäude. Ersteres ist der Holzfäller der in einem festen Zeitintervall von 2t[3] 1 Holzstamm[4] produziert, ohne dabei benötigte Ressourcen[5] zu haben. Die Lagerkapazitäten des Holzfällers sind 1L[6] Zweiteers Gebäude ist das Sägewerk das alle 1t[3] aus einem Holzstamm ein Brett[7] produziert. Das Sägewerk hat eine Lagerkapazität von 2L[6]. Als drittes Gebäude wird noch das Lagerhaus hinzugefügt, es dient nur dazu die produzierten Waren zu sammeln und die benötigten auf die Verbraucher zu verteilen. Weiterhin können die Lagerhäuser auch als Zwischenlager verwendet werden, um überschüssige Waren zu Lagern.

Die unterschiedlichen Gebäude werden durch Straßen miteinander verbunden.

[3] t: Beliebige Zeit
[4] Platz eingenommen: 2 VE
[5] Das fällen von Bäumen und deren Existenz werden zur Vereinfachung weggelassen
[6] L: Volumen des Lagers
[7] Platz eingenommen: 1 VE

### 3.2 Die Implementation

## 4 Fazit

# Literatur


[1] baeldung. *Ant Colony Optimization | Baeldung*. 21. Feb. 2021. URL: https://www.baeldung.com/java-ant-colony-optimization.

[2] Daniel Blum. "Ant Colony Optimization (ACO)". In: (). URL: https://ls11-www.cs.tu-dortmund.de/lehre/SoSe03/PG431/Ausarbeitungen/ACO.pdf.

[3] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo u. a. "Distributed optimization by ant colonies". In: *Proceedings of the first European conference on artificial life*. Bd. 142. Paris, France. 1991, S. 134–142.

[4] Marco Dorigo. "Ant colony optimization". In: *Scholarpedia* 2.3 (2007), S. 1461.

[5] Matthias Teschner. *Algorithmen und DatenstrukturenGraphen - Einführung*. 25. Feb. 2021. URL: https://cg.informatik.uni-freiburg.de/course_notes/info2_15_graph.pdf.

[6] thiagodnf. *ACO Simulator*. 28. Feb. 2021. URL: http://thiagodnf.github.io/aco-simulator/#.

[7] unkown. *Ant colony optimization algorithms - Wikipedia*. 28. Feb. 2021. URL: https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms.

[8] unkown. *Travelling salesman problem - Wikipedia*. 27. Feb. 2021. URL: https://en.wikipedia.org/wiki/Travelling_salesman_problem.