Ant Colony System Algorithm

Tutorial

für die Vorlesung

Advanced Software Engineering

des Studiengangs Angewandte Informatik

an der Dualen Hochschule Baden-Württemberg Mosbach

von

5703004, 1716504

6. Mai 2022

1 Metapher

Der Ant-Colony-Optimization-Algorithmus (ACO) ist von dem sozialen Verhalten von Insekten und insbesondere von Ameisen inspiriert. Dabei ist ein Individuum einer Kolonie wenig komplex, durch das Zusammenspiel aller Individuen sind Ameisen allerdings in der Lage, gemeinsam komplexe Aufgaben zu bewältigen.

Der hier vorgestellte Algorithmus lässt sich von der Futtersuche von Ameisen ableiten. Das Verhalten wird in dem sog. *Double-Bridge-Experiment* analysiert. Dabei haben Ameisen genau zwei Wege, um von ihrem Nest zu einer Futterquelle zu gelangen. Einer der beiden Wege ist dabei länger als der andere. Zu Beginn schwärmen die Ameisen aus und nutzen beide Wege, um zu der Futterquelle zu gelangen. Nach einiger Zeit wird man allerdings beobachten, dass sich eine Ameisenstraße über den kürzeren der beiden Wege bildet. Die Ameisen nutzen also eine Strategie, um den schnellsten Weg von ihrem Nest zum Futter zu finden. Diese Strategie ist maßgeblich auf das Schwarmverhalten und ausgesetzte Pheromonspuren zurückzuführen. Finden Ameisen Futter, so kommunizieren sie den Weg ihren Artgenossen durch einen Duftstoff über den Weg, den sie gelaufen sind. Andere Ameisen folgend schließlich dieser Duftspur und gelangen ebenfalls an das Futter. Kürzer Wege werden so öfter frequentiert von Ameisen abgelaufen, wodurch die Pheromonspur auf diesem Weg stärker wird, während die Pheromonspur auf den anderen Wegen mit der Zeit verfliegt. So nehmen immer mehr Ameisen die kürzeste Strecke und es bilden sich die typischen Ameisenstraßen. Damit nutzen Ameisen ihr Schwarmverhalten, um den Weg von ihrem Nest zur Futterstelle zu optimieren. Eine Eigenschaft, die sich auch für das TSP zunutze gemacht werden kann.

2 Strategie

Die Strategie für diesen Algorithmus liegt darin, dass sowohl das kollektive Schwarmverhalten durch die Pheromonspur genutzt wird, um bekannte Wege zu exploitieren und den günstigsten Pfad zu finden, als auch ein individuelles Verhalten in die Wegfindung einzelner Ameisen mit einfliest, das durch eine einfache Heuristik simuliert wird.

Ameisen laufen verschiedene Pfade iterativ ab und teilen die Pheromoninformationen über das kollektive Gedächtnis in einer Pheromonmatrix, die jeder Ameise zur Verfügung steht. Auf diese Weise lässt sich effektiv ein Optimum für den gewünschten Suchraum finden.

3 Prozedur

Für das kollektive Gedächtnis wird eine Pheromonenmatrix angelegt, die die Pheromonenwerte aller Kanten eines Graphen beinhalten. Diese Werte fließen in die Entscheidung für einen Knotenpunkt bei jeder Iteration mit ein. Die Pheromonenmatrix ist dabei das kollektive Gedächtnis der Ameisenkolonie. Jede Ameise nutzt außerdem eine Metaheuristik, mit der sie selbstständig einen nächsten Knotenpunkt auswählt. Beide Faktoren können gewichtet werden, um ihren jeweiligen Einfluss auf die Entscheidung einer einzelnen Ameise zu steuern. Daraus lässt sich ein stochastisches Modell entwickeln, das sich mathematisch folgendermaßen ausdrücken lässt:

$$p_{i,j}^{k}(t) = \frac{\tau_{i,j}^{\alpha} * \eta_{i,j}^{\beta}}{\sum_{k=1}^{c} \tau_{i,k}^{\alpha} * \eta_{i,k}^{\beta}} \tag{1}$$

Dabei ist $p_{i,j}$ die Wahrscheinlichkeit, mit der die Kante des Graphen von Position i nach Position j von einer Ameise gelaufen wird, $\tau_{i,j}$ ist die Pheromoninformation für diese Kante und wird über das kollektive Gedächtnis mit allen Ameisen geteilt, $\eta_{i,j}$ ist die heuristische Information, die beispielsweise durch die Berechnung der Euklidischen Distanz ziwschen den einzelnen Punkten im Graph berechnet wird; α ist die Gewichtung, mit der die Pheromoninformationen, also das kollektive Gedächtnis, auf die Entscheidung der Ameise Einfluss nehmen und β ist entsprechend die Gewichtung für die Einflussnahme der Heuristik, also dem individuelle Verhalten, auf die Entscheidung. Um bessere Ergebnisse zu erzielen, werden die Ergebnisse außerdem ins Verhältnis zu den Ergebnissen aller Ameisen gesetzt.

Mit dieser grundlegenden Formel lassen sich verschiedene Pfade im gewünschten Suchraum ablaufen. Nach jeder Iteration werden die Pheromonenwerte in der Pheromonenmatrix aktualisiert. Wird die maximale Anzahl an Iterationen erreicht oder das Optimum ändert sich nicht oder nur geringfügig, ist das Optimum gefunden. Der Algorithmus liefert den Pfad, der nach der letzten Iteration als bester Pfad ausgewählt wurde. Das ist derjenige, auf dem die höchste Pheromonkonzentration zu finden ist.

4 Pseudocode

Der Hauptteil des ACO ist eine einfache `while` -Schleife, die solange durchlaufen wird, bist eine Abbruchbedingung erfüllt ist. Diese kann beispielsweise durch eine maximale Anzahl an Iterationen festgelegt werden oder durch einen Fitness-Wert, der die Veränderung zum vorherigen Ergebnis angibt. Fällt dieser Werte unter eine bestimmte Schranke, ist das Ergebnis nah genug am Optimum und der Algorithmus kann terminiert werden. Für den Algorithmus ergibt sich folgender Pseudocode:

```
while (Abbruchbedingung trifft nicht zu) do
    GenerierePfadeFuerAmeisen();
    AktualisierePheromone();
    AktualisiereMomentanesOptimum();
done
```

In jedem Durchlauf läuft jede Ameise mithilfe der in 3 beschriebenen Formel einen gesamten Pfad für das TSP ab. Anschließend wird die Pheromonmatrix für den nächsten Durchlauf aktualisiert. In diesem Schritt wird auch ein *evaporation*-Faktor (Verdampfung) angewendet, der angibt, wie viel vom ursprünglichen Pheromonenwert pro Kante für die nächste Iteration behalten wird. Damit werden weniger frequentierte Kanten schneller im kollektiven Gedächtnis als nichtzielführend markiert, wodurch der Algorithmus schneller ein globales Optimum finden kann. Außerdem lässt sich der Algorithmus auch bei dynamischen Problemen einsetzen. Würden sich Ausgangsparameter ändern, kann über die *evaporation* sichergestellt werden, dass ursprüngliche vielversprechende Lösungen, die durch die Parameteränderung nicht mehr zielführend sind, mit der Zeit wieder vergessen werden. Am Ende jedes Durchlaufs wird der beste Pfad ermittelt. Dieser stellt das momentane gefundene globale Optimum dar.

5 Exploration and Exploitation

Die Exploration, also das Erkunden des neuen Suchraums, wird maßgeblich durch die Gewichtung β der Metaheuristik beeinflusst. Die Erkundung geschieht dadurch, dass Ameisen an jedem Knotenpunkt im Graphen wahrscheinlichkeitsbasiert auswählen, welche Kante sie entlanggehen. Je höher β im Gegensatz zu α gewählt wird, desto mehr Suchraum wird exploriert, allerdings leidet darunter die Exploitation. Diese wiederum wird durch die Gewichtung α des Einflusses des kollektiven Gedächtnisses bei der Wegfindung über die Pheromonenmatrix beeinflusst. Ein ausgeglichenes Verhältnis zwischen α und β ist zu wählen, um die Suche nach einem Optimum möglichst effizient zu gestalten.

6 Entscheidungsregeln für Schwarmverhalten

Ameisen im ACO nutzen zum einen das kollektive Gedächtnis, um wahrscheinlichkeitsbasiert an jedem Knotenpunkt eines Graphen die nächste Kante auszuwählen (τ), zum anderen eine Metaheuristik (η), um ein individuelles Verhalten einer Ameise zu simulieren. Wie stark diese beiden Faktoren einen Einfluss auf die Wegfindung einer Ameise haben, hängt von den Gewichtunksfaktoren α und β ab. Wird α beispielsweise auf 0 gesetzt, so fließen die Informationen des kollektiven Gedächtnisses und damit die Pheromoninformationen nicht in die

Wegfindung mit ein. Eine Ameise wird also einen Weg wählen, der nur auf der aktuellen Distanz zu einem nächsten Knotenpunkt basiert. Damit gleicht das Suchen eines Optimums einer Suche mit Bruteforce und ist nicht effizient. Wird hingegen β auf 0 gesetzt, so berücksichtigt eine Ameise nur die Pheromoninformationen, wodurch keine neuen Wege mehr exploriert werden. Es besteht die Gefahr, dass das globale Optimum nicht gefunden wird, da jegliches individuelle Verhalten fehlt.

7 Parameterabhängigkeiten

Für den ACO können diverse Startparameter definiert werden. Neben den bereits ausführlich erläuterten Parametern α und β, kann mithilfe des Parameters *evaporation* Angegeben werden, wie viel Prozent des ursprünglichen Pheromonwerts nach einer Iteration behalten werden. Damit wird das Verfliegen des Duftstoffes simuliert. Ein höherer Wert lässt die Pheromonenwerte kleiner werden, dadurch kann angepasst werden, wie stark auch niedrigfrequentierte Pfade in der nächsten Iteration zur Suche nach einem Optimum miteinbezogen werden. Es sollten eher geringe Werte gewählt werden. Bei einer *evaporation* von 1.0 (100%) werden nach jeder Iteration die Pheromone zurückgesetzt, wodurch das kollektive Gedächtnis nach jeder Iteration gelöscht wird. Das hat zur Folge, dass der Algorithmus nicht das globale Optimum finden kann und Ameisen eine Bruteforce-Suche im gesamten Suchraum starten.

Mit einem weiteren Parameter q kann definiert werden, wie viele Pheromone von einer einzelnen Ameise auf einem Pfad liegengelassen wird. Ein höherer Wert führt zu einer höheren „Auflösung“in der Pheromonmatrix. Wird beispielsweise $q = 1$ gewählt und es laufen 10 Ameisen eine Kante entlang, beträgt der Pheromonwert dieser Kante nach der ersten Iteration zunächst 10. Durch die *evaporation*, die in diesem Beispiel auf 0.1 (10%) gesetzt werden soll, wird der Pheromonwert dieser Kante schließlich den Wert 9 annehmen. Die Differenz zwischen diesen beiden Werten ist 1. Wird hingegen $q = 10$ gewählt, so nimmt der Pheromonwert dieser Kante den Wert 100, bzw. nach der Verdampfung 90 an. Die Differenz der beiden Werte (vor und nach der Verdampfung) beträgt hier bereits 10. Das hat zur Folge, dass höhere Werte für q eine größere Unterscheidung von Pheromonwerten in der Matrix hat, wodurch die Intensität, die ein potentiell guter Pfad für das Optimum auf das kollektive Gedächtnis hat, verstärkt wird.

Mit dem *antFactor* lässt sich berechnen, wie viele Ameisen in Abhänigkeit von Knotenpunkten für den Algorithmus erzeugt werden. Hier sollte ein Wert zwischen 0.6 und 0.8 gewählt werden. Ein Wert von 0.6 würde bedeuten, dass für 60% von Knotenpunkten, Ameisen erstellt werden, bei einer Anzahl von 100 Knoten in einem Graphen also 60 Ameisen zur Verfügung stehen, um das Optimum für diesen Graphen zu finden. Mit diesem Parameter kann die Effizient des Algorithmus gesteigert werden, da nicht für jeden Knotenpunkt eine Ameise zur Verfügung

stehen muss, wodurch die Komplexität reduziert und die Geschwindigkeit bei der Ausfürhung des ACO erhöht wird.

Der *randomFactor* kann verwendet werden, um bei der Auswahl von nächsten Knotenpunkten einen kleinen Zufall einzubauen, wodurch die Exploration erhöht wird, da Ameisen nicht unbedingt der nach der in Abschnitt 3 vorgestellten Wahrscheinlichkeitsberechnung besten Weg nehmen müssen.

Zuletzt kann mit dem Parameter *maximumIterations* festegelgt werden, wie viele Iterationen durchlaufen werden sollen, bevor der Algorithmus terminiert wird. Dies verhindert, dass ein Algorithmus nicht terminiert, wenn keine Lösung gefunden wird, bzw. die gefundene Lösung noch nicht dem globalen Optimum entspricht, eine weitere Suche das Ergebnis aber nicht erheblich verbessern würde.

8 Zusammenspiel *Ant* und *AntColony*

Bei einer Konfiguration von $\alpha = 2$ und $\beta = 2$, werden τ und η gleichermaßen berücksichtigt. Damit wird sowohl das kollektive Gedächtnis (Pheromonenmatrix), als auch das individuelle Verhalten (Metaheuristik) in die Wegfindung mit einbezogen. Dies führt zu einer ausgewogenen Exploration und Exploitation des Suchraums. Mit einer Konfiguration, in der die *evaporation* auf 0.05 gesetzt wird, werden die Werte in der Pheromonenmatrix nach jeder Iteration um 5% gesenkt. Es werden also nur 95% der eigentlichen Pheromonenwerte für die nächste Iteration übernommen. Das hat zur Folge, dass Wege, die einmal als vielversprechend angesehen wurden, mit der Zeit auch wieder vergessen werden können, wenn ein anderer, besserer Weg gefunden wird. Das erhöht die Leistungsfähigkeit des ACO. Mit dem Wert *q* = 500 wird für jede Ameise, die eine Kante des Graphen entlanggeht, der Pheromonwert für diese Kante um 500 erhöht. Damit lässt sich eine gute Auflösung der Pheromonenmatrix erzielen. Der *antFactor* von 0.8 sorgt dafür, dass für 80% des Suchraums, Ameisen erzeugt werden, die den gesamten Suchraum erkunden. Im Falle des TSP würden bei 100 Städten also die Ameisenkolonie eine Größe von 80 Ameisen annehmen, wobei jede Ameise in jeder Iteration einen Pfad für das TSP sucht. Ein höherer Wert erhöht die Komplexität pro Iteration, ein niedrigerer Wert erhöht hingegen die Anzahl an Iterationen, dafür wird die Rechenaufwand pro Iteration minimiert. Ein *randomFactor* von 0.01 fügt der Auswahl bei der Wegfindung von Ameisen für den nächsten Knotenpunkt eine Variation hinzu. Mit einer Wahrscheinlichkeit von 1% wählt die Ameise also einen anderen Knoten, als den, den sie aufgrund der Berechnung gewählt hätte.