h_da HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Prof. Dr. Alexander del Pino
Fachbereich Informatik

Genetische Algorithmen

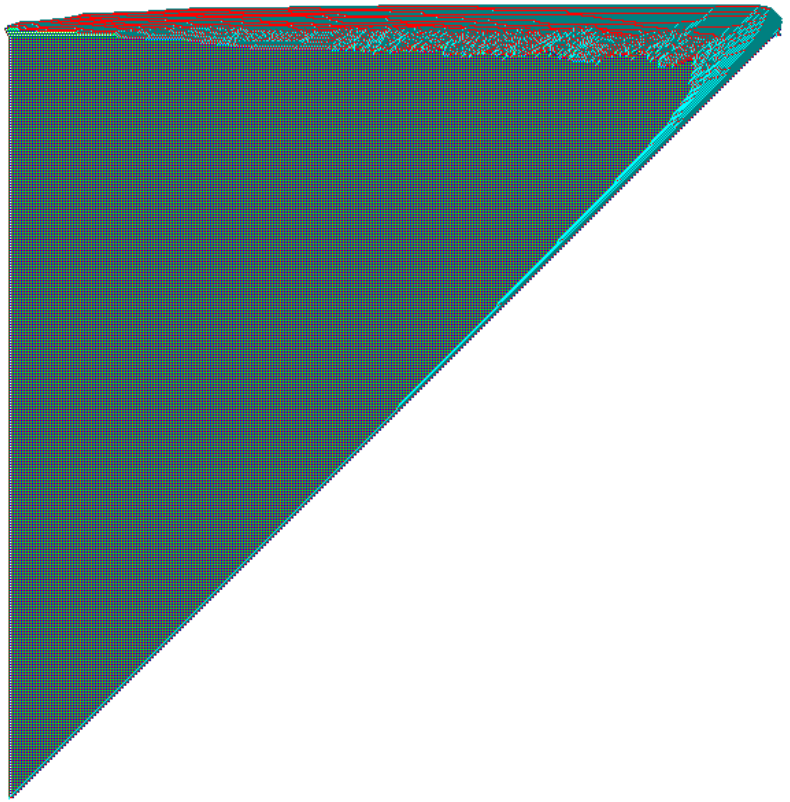


Image: 031692e615d131df58e39a4c4f4b9a54. © Alexander del Pino

11. Teil

Ameisenkolonie- und Partikelschwarm-Optimierung

Ameisenkolonie-Optimierung

Genetische Algorithmen vs. Ameisenkolonie-Optimierung

Der Begriff Ameisenkolonie-Optimierung *(ant colony optimization, ACO)* steht für eine Reihe von Verfahren, die Anfang der 1990er Jahre ursprünglich von *Marco Dorigo* für die Lösung von kombinatorischen Optimierungsproblemen entwickelt wurden.

Ameisenkolonie-Optimierung

Genetische Algorithmen vs. Ameisenkolonie-Optimierung

Der Begriff Ameisenkolonie-Optimierung *(ant colony optimization, ACO)* steht für eine Reihe von Verfahren, die Anfang der 1990er Jahre ursprünglich von *Marco Dorigo* für die Lösung von kombinatorischen Optimierungsproblemen entwickelt wurden.

Ein Vergleich mit evolutionären, z.B. genetischen Algorithmen ergibt:

- Beide sind sogenannte Metaheuristiken, also algorithmische Rahmenwerke, welche mit nur geringen Änderungen zum Finden von Näherungslösungen für eine Vielzahl von Problemen verwendet werden können.

Ameisenkolonie-Optimierung

Genetische Algorithmen vs. Ameisenkolonie-Optimierung

Der Begriff Ameisenkolonie-Optimierung *(ant colony optimization, ACO)* steht für eine Reihe von Verfahren, die Anfang der 1990er Jahre ursprünglich von *Marco Dorigo* für die Lösung von kombinatorischen Optimierungsproblemen entwickelt wurden.

Ein Vergleich mit evolutionären, z.B. genetischen Algorithmen ergibt:

- Beide sind sogenannte Metaheuristiken, also algorithmische Rahmenwerke, welche mit nur geringen Änderungen zum Finden von Näherungslösungen für eine Vielzahl von Problemen verwendet werden können.
- Beide Verfahren sind durch biologische Prozesse inspiriert: Genetische Algorithmen durch die Evolution, ACO durch die Futtersuche bei Ameisen.

Ameisenkolonie-Optimierung

Genetische Algorithmen vs. Ameisenkolonie-Optimierung

Der Begriff Ameisenkolonie-Optimierung *(ant colony optimization, ACO)* steht für eine Reihe von Verfahren, die Anfang der 1990er Jahre ursprünglich von *Marco Dorigo* für die Lösung von kombinatorischen Optimierungsproblemen entwickelt wurden.

Ein Vergleich mit evolutionären, z.B. genetischen Algorithmen ergibt:

- Beide sind sogenannte Metaheuristiken, also algorithmische Rahmenwerke, welche mit nur geringen Änderungen zum Finden von Näherungslösungen für eine Vielzahl von Problemen verwendet werden können.
- Beide Verfahren sind durch biologische Prozesse inspiriert: Genetische Algorithmen durch die Evolution, ACO durch die Futtersuche bei Ameisen.
- Beide Verfahren sind inhärent parallel und entwickeln nicht nur eine Lösung, sondern eine Menge von Lösungen.

Ameisenkolonie-Optimierung

Nahrungssuche bei Termiten

Unter Stigmergie *(stigmergy)* bezeichnet man eine indirekte Kommunikation durch Modifikation der Umgebung.

Ameisenkolonie-Optimierung

Nahrungssuche bei Termiten

Unter Stigmergie *(stigmergy)* bezeichnet man eine indirekte Kommunikation durch Modifikation der Umgebung.

- Bei der Nahrungssuche verwenden Ameisen Stigmergie, um anderen Ameisen den Weg zu einer Nahrungsquelle aufzuzeigen. Sie hinterlassen dazu Spuren von Pheromonen *(pheromone)*. Andere Ameisen werden dadurch angeregt, diesen Spuren zu folgen.

Ameisenkolonie-Optimierung

Nahrungssuche bei Termiten

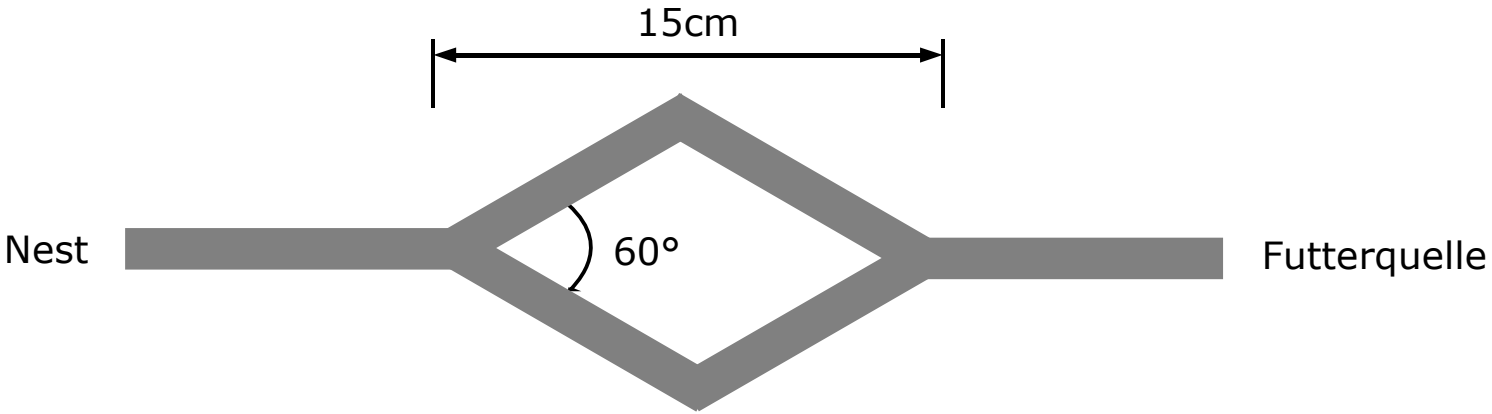
Unter Stigmergie *(stigmergy)* bezeichnet man eine indirekte Kommunikation durch Modifikation der Umgebung.

- Bei der Nahrungssuche verwenden Ameisen Stigmergie, um anderen Ameisen den Weg zu einer Nahrungsquelle aufzuzeigen. Sie hinterlassen dazu Spuren von Pheromonen *(pheromone)*. Andere Ameisen werden dadurch angeregt, diesen Spuren zu folgen.
- *Deneubourg* et al. (1990) untersuchten das Verhalten von Argentinischen Ameisen *(Iridomyrmex humilis)* bei dem dem Anlegen und Verfolgen von Pheromonspuren durch sogenannte Doppelbrückenexperimente. Diese Ameisen sind fast blind.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

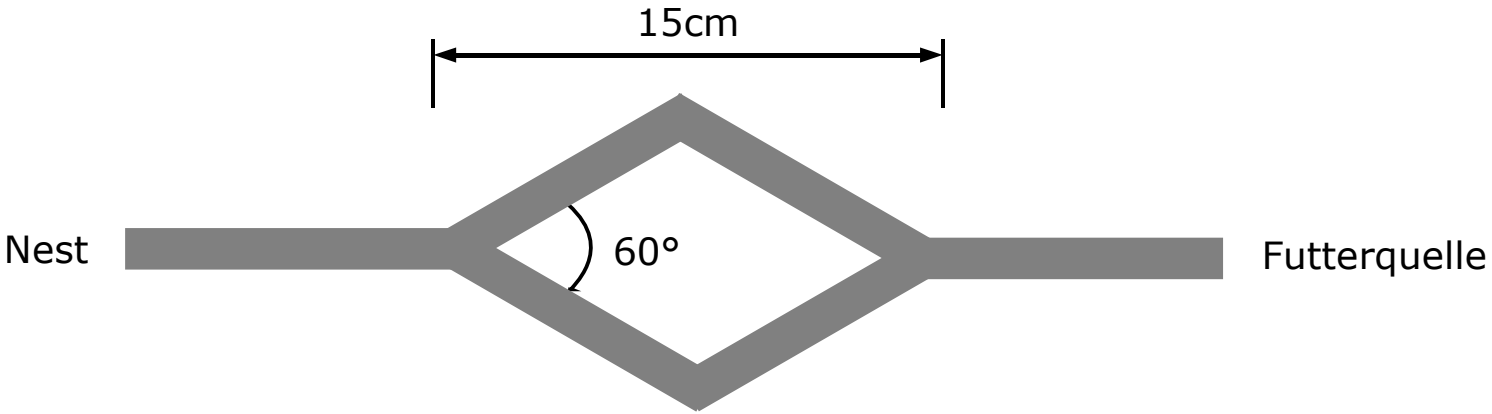
In dem Doppelbrückenexperiment *(double bridge experiment)* war das Nest mit der Nahrungsquelle durch eine Doppelbrücke mit zwei gleich langen Pfaden verbunden.



Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

In dem Doppelbrückenexperiment *(double bridge experiment)* war das Nest mit der Nahrungsquelle durch eine Doppelbrücke mit zwei gleich langen Pfaden verbunden.



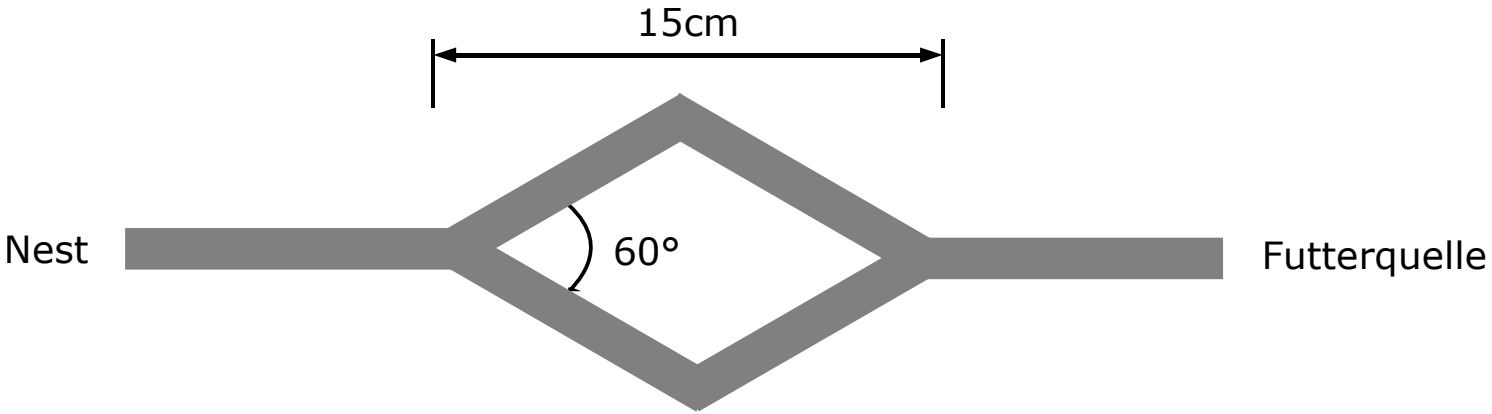
Ergebnisse:

- Anfangs werden beide Pfade mit gleicher Wahrscheinlichkeit durchlaufen.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

In dem Doppelbrückenexperiment *(double bridge experiment)* war das Nest mit der Nahrungsquelle durch eine Doppelbrücke mit zwei gleich langen Pfaden verbunden.



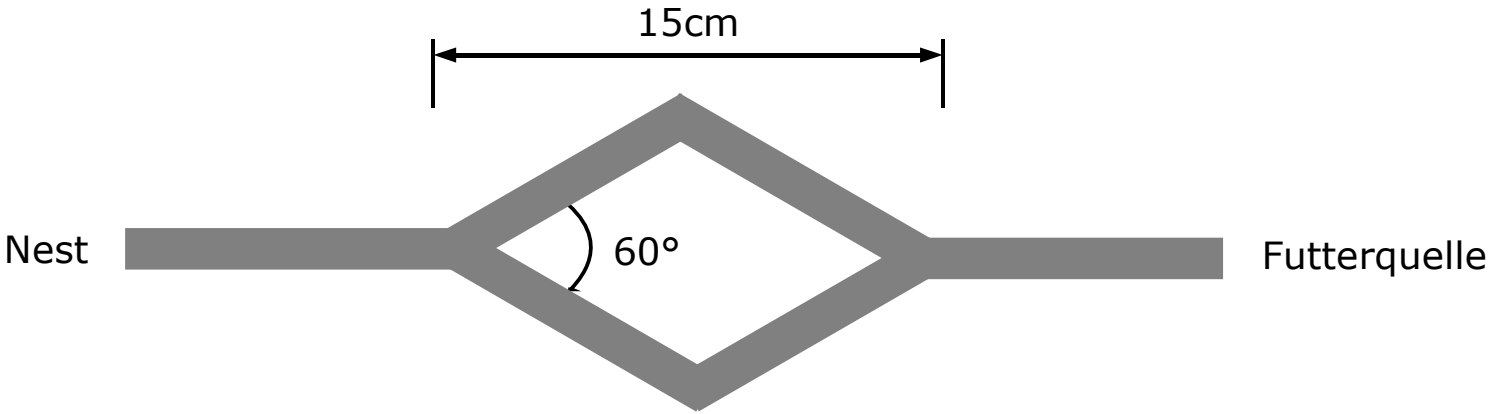
Ergebnisse:

- Anfangs werden beide Pfade mit gleicher Wahrscheinlichkeit durchlaufen.
- Durch die Pheromon-Markierungen wird aber mittels eines positiven Feedback-Effekts *(autocatalysis)* nach kurzer Zeit einer der beiden Pfade bevorzugt.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

In dem Doppelbrückenexperiment *(double bridge experiment)* war das Nest mit der Nahrungsquelle durch eine Doppelbrücke mit zwei gleich langen Pfaden verbunden.



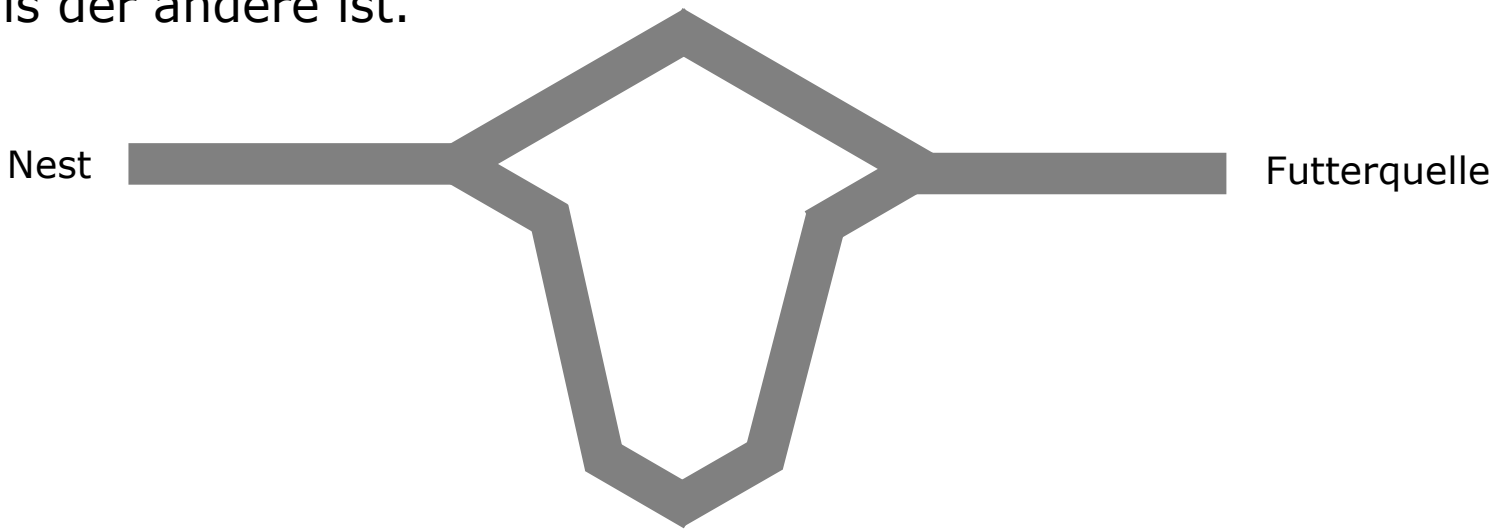
Ergebnisse:

- Anfangs werden beide Pfade mit gleicher Wahrscheinlichkeit durchlaufen.
- Durch die Pheromon-Markierungen wird aber mittels eines positiven Feedback-Effekts *(autocatalysis)* nach kurzer Zeit einer der beiden Pfade bevorzugt.
- Später laufen quasi alle Ameisen nur noch über den bevorzugten Pfad.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

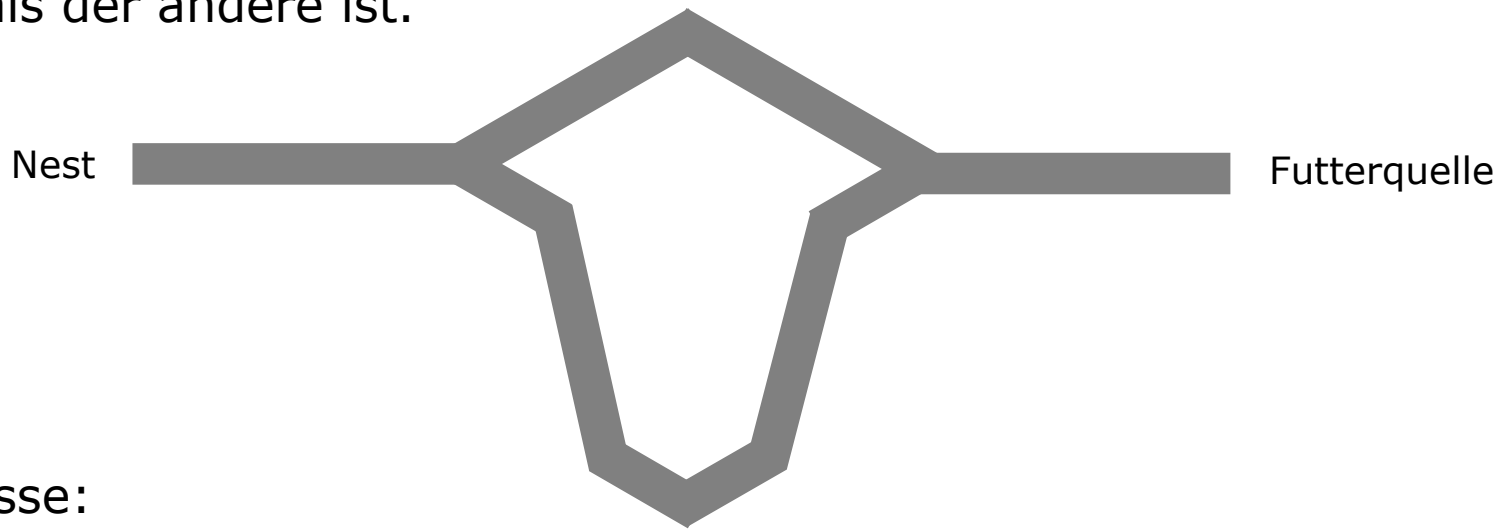
Goss et al. (1989) modifierten die Doppelbrücke so, dass ein Pfad wesentlich länger als der andere ist.



Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

Goss et al. (1989) modifierten die Doppelbrücke so, dass ein Pfad wesentlich länger als der andere ist.



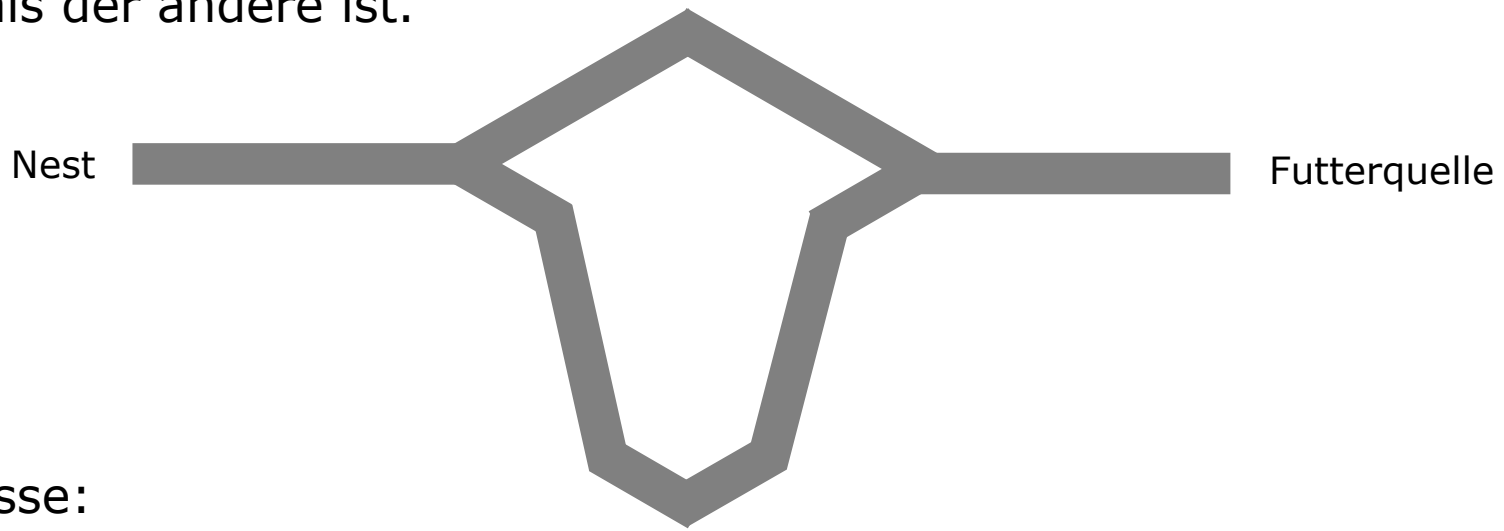
Ergebnisse:

- Ameisen, die den kürzeren Pfad gewählt haben, kommen auch wieder schneller wieder im Nest an. Dadurch werden auf dem kürzeren Pfad schneller Pheromone hinterlassen als auf dem längeren Pfad.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

Goss et al. (1989) modifierten die Doppelbrücke so, dass ein Pfad wesentlich länger als der andere ist.



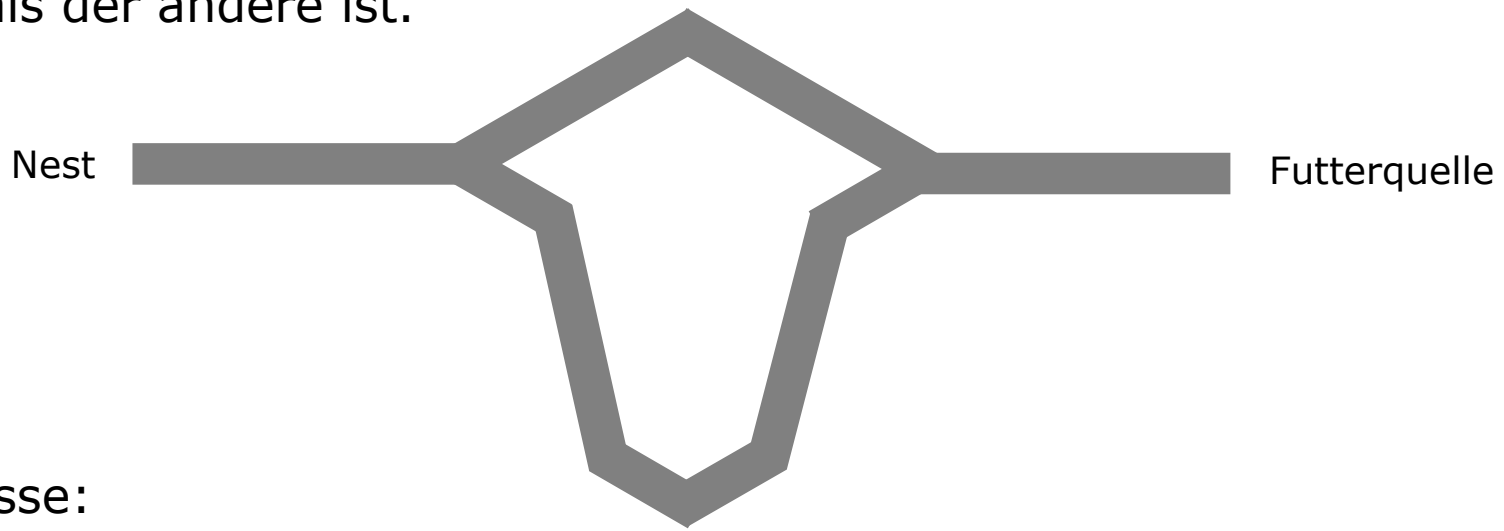
Ergebnisse:

- Ameisen, die den kürzeren Pfad gewählt haben, kommen auch wieder schneller wieder im Nest an. Dadurch werden auf dem kürzeren Pfad schneller Pheromone hinterlassen als auf dem längeren Pfad.
- Die nächste Ameise wählt deshalb den kürzeren Pfad mit einer etwas höheren Wahrscheinlichkeit aus.

Ameisenkolonie-Optimierung

Das Doppelbrückenexperiment

Goss et al. (1989) modifierten die Doppelbrücke so, dass ein Pfad wesentlich länger als der andere ist.



Ergebnisse:

- Ameisen, die den kürzeren Pfad gewählt haben, kommen auch wieder schneller wieder im Nest an. Dadurch werden auf dem kürzeren Pfad schneller Pheromone hinterlassen als auf dem längeren Pfad.
- Die nächste Ameise wählt deshalb den kürzeren Pfad mit einer etwas höheren Wahrscheinlichkeit aus.
- Nach einiger Zeit laufen die meisten Ameisen über den kürzeren Pfad.

Ameisenkolonie-Optimierung

Welchen Pfad soll ich auswählen ? Ein Modell

Deneubourg et al. modellierten das Verhalten der Ameisen wie folgt. Sei A_i bzw. B_i die Menge an Pheromon auf Pfad A bzw. auf Pfad B. Dann beträgt die Wahrscheinlichkeit $prob_A$ dass eine Ameise Pfad A auswählt:

$$prob_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n}$$

$$prob_B = 1 - prob_A$$

Ameisenkolonie-Optimierung

Welchen Pfad soll ich auswählen ? Ein Modell

Deneubourg et al. modellierten das Verhalten der Ameisen wie folgt. Sei A_i bzw. B_i die Menge an Pheromon auf Pfad *A* bzw. auf Pfad *B*. Dann beträgt die Wahrscheinlichkeit $prob_A$ dass eine Ameise Pfad *A* auswählt:

$$prob_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n}$$

$$prob_B = 1 - prob_A$$

- Der Parameter *n* steuert dabei die Nichtlinearität des Verhaltens. Je größer *n* ist, umso eher wird der Pfad, der mit nur geringfügig mehr Pheromon als der andere Pfad markiert ist, ausgewählt.

Ameisenkolonie-Optimierung

Welchen Pfad soll ich auswählen ? Ein Modell

Deneubourg et al. modellierten das Verhalten der Ameisen wie folgt. Sei A_i bzw. B_i die Menge an Pheromon auf Pfad A bzw. auf Pfad B. Dann beträgt die Wahrscheinlichkeit $prob_A$ dass eine Ameise Pfad A auswählt:

$$prob_A = \frac{(k + A_i)^n}{(k + A_i)^n + (k + B_i)^n}$$

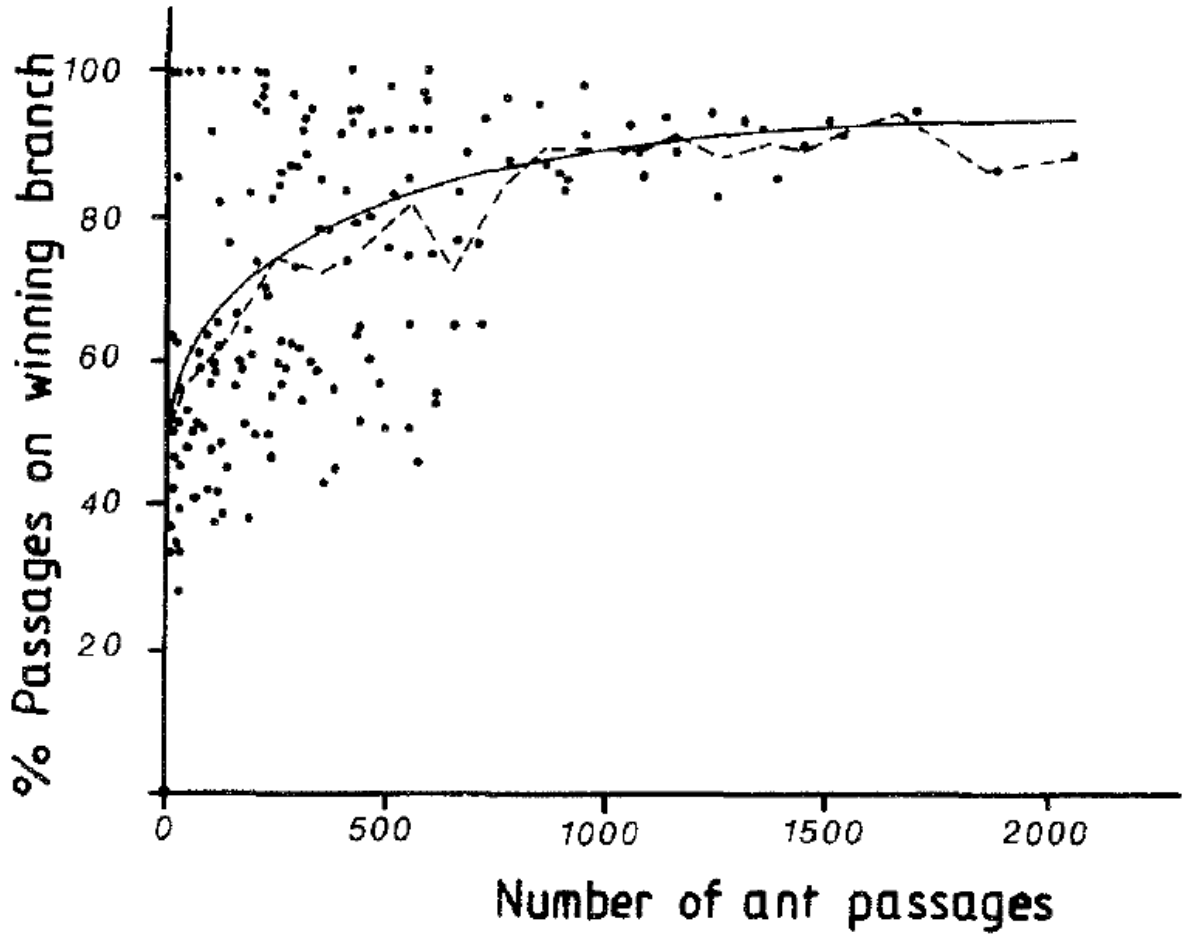
$$prob_B = 1 - prob_A$$

- Der Parameter n steuert dabei die Nichtlinearität des Verhaltens. Je größer n ist, umso eher wird der Pfad, der mit nur geringfügig mehr Pheromon als der andere Pfad markiert ist, ausgewählt.
- Der Parameter k steuert die Attraktivität eines nicht markierten Pfades. Je größer k ist, umso mehr muss ein Pfad mit Pheromon markiert sein, damit die Entscheidung für einen bestimmten Pfad über den reinen Zufall hinaus geht.

Ameisenkolonie-Optimierung

Modell vs. Messungen

Mit diesem Modell konnte man das tatsächlich beobachtete Verhalten gut erklären. Die Kurve basiert auf der vorigen Formel mit $n = 2$ und $k = 20$.



Bildquelle: J.-L. Deneubourg, S. Aron, S. Goss, J. M. Pasteels. *The Self-Organizing Exploratory Pattern of the Argentine Ant*. Journal of Insect Behavior, Vol. 3, No. 2, 1990

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Das von *Deneubourg* et al. beschriebene Verhalten verwendete *M. Dorigo* um ein allgemeines kombinatorisches Optimierungsverfahren – ACO – zu entwickeln.

Bevor man ein Problem mit ACO angeht, muss es gegebenfalls so umgeformt werden, dass die Suche nach einer Lösung als die Konstruktion eines kürzesten Pfades in einem Graph *(construction graph)* aufgefasst werden kann.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Das von *Deneubourg* et al. beschriebene Verhalten verwendete *M. Dorigo* um ein allgemeines kombinatorisches Optimierungsverfahren – ACO – zu entwickeln.

Bevor man ein Problem mit ACO angeht, muss es gegebenfalls so umgeformt werden, dass die Suche nach einer Lösung als die Konstruktion eines kürzesten Pfades in einem Graph *(construction graph)* aufgefasst werden kann.

- Jede Kante in diesem Graph ist ein Baustein *(solution component)* zu einer Lösung.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Das von *Deneubourg* et al. beschriebene Verhalten verwendete *M. Dorigo* um ein allgemeines kombinatorisches Optimierungsverfahren – ACO – zu entwickeln.

Bevor man ein Problem mit ACO angeht, muss es gegebenfalls so umgeformt werden, dass die Suche nach einer Lösung als die Konstruktion eines kürzesten Pfades in einem Graph *(construction graph)* aufgefasst werden kann.

- Jede Kante in diesem Graph ist ein Baustein *(solution component)* zu einer Lösung.
- Ein Pfad durch mehrere Kanten ist eine Teillösung *(partial solution)*, welche gewissen, problemspezifischen Randbedingungen genügen muss.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Das von *Deneubourg* et al. beschriebene Verhalten verwendete *M. Dorigo* um ein allgemeines kombinatorisches Optimierungsverfahren – ACO – zu entwickeln.

Bevor man ein Problem mit ACO angeht, muss es gegebenfalls so umgeformt werden, dass die Suche nach einer Lösung als die Konstruktion eines kürzesten Pfades in einem Graph *(construction graph)* aufgefasst werden kann.

- Jede Kante in diesem Graph ist ein Baustein *(solution component)* zu einer Lösung.
- Ein Pfad durch mehrere Kanten ist eine Teillösung *(partial solution)*, welche gewissen, problemspezifischen Randbedingungen genügen muss.
- Dieser Graph wird von virtuellen Ameisen durchlaufen. Der durchlaufene Pfad entspricht der (Teil-)lösung, die von dieser Ameise gefunden wird.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Das von *Deneubourg* et al. beschriebene Verhalten verwendete *M. Dorigo* um ein allgemeines kombinatorisches Optimierungsverfahren – ACO – zu entwickeln.

Bevor man ein Problem mit ACO angeht, muss es gegebenfalls so umgeformt werden, dass die Suche nach einer Lösung als die Konstruktion eines kürzesten Pfades in einem Graph *(construction graph)* aufgefasst werden kann.

- Jede Kante in diesem Graph ist ein Baustein *(solution component)* zu einer Lösung.
- Ein Pfad durch mehrere Kanten ist eine Teillösung *(partial solution)*, welche gewissen, problemspezifischen Randbedingungen genügen muss.
- Dieser Graph wird von virtuellen Ameisen durchlaufen. Der durchlaufene Pfad entspricht der (Teil-)lösung, die von dieser Ameise gefunden wird.
- Die Ameisen bringen an den Kanten des Graphen unterschiedlich viel Pheromon an, um andere Ameisen auf die Qualität ihrer gefundenen Lösung hinzuweisen.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Befindet sich eine Ameise an einem Knoten *i*, so durchläuft sie die Kante hin zu einem Nachbarknoten *j* mit einer Wahrscheinlichkeit, die u. a. von der Menge des dort abgelegten Pheromons abhängig ist.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Befindet sich eine Ameise an einem Knoten *i*, so durchläuft sie die Kante hin zu einem Nachbarknoten *j* mit einer Wahrscheinlichkeit, die u. a. von der Menge des dort abgelegten Pheromons abhängig ist.

- Pheromon kann im Laufe der Zeit auch wieder verdunsten. Dadurch erreicht man, dass der Einfluß von mittelmäßigen Lösungen im Laufe der Zeit immer mehr abnimmt.

Ameisenkolonie-Optimierung

ACO – Die Grundidee

Befindet sich eine Ameise an einem Knoten *i*, so durchläuft sie die Kante hin zu einem Nachbarknoten *j* mit einer Wahrscheinlichkeit, die u. a. von der Menge des dort abgelegten Pheromons abhängig ist.

- Pheromon kann im Laufe der Zeit auch wieder verdunsten. Dadurch erreicht man, dass der Einfluß von mittelmäßigen Lösungen im Laufe der Zeit immer mehr abnimmt.
- Um zu verhindern, dass sich Zyklen – die sich ja auch sehr rasch verstärken würden – ausbilden, wird ein Pfad erst dann mit Pheromon markiert, nachdem eine komplette Lösung gefunden wurde, und alle Zyklen aus diesem Pfad entfernt wurden.

Wie kann man effizient Zyklen in einem solchen Pfad entdecken ?

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- ConstructAntsSolutions ist für das Management einer Kolonie von virtuellen Ameisen verantwortlich die den Graphen wie eben beschrieben durchlaufen und sukzessive eine Lösung aufbauen.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- ConstructAntsSolutions ist für das Management einer Kolonie von virtuellen Ameisen verantwortlich die den Graphen wie eben beschrieben durchlaufen und sukzessive eine Lösung aufbauen.
- Die Nachbarschaft zu dem Knoten an dem sich eine Ameise gerade befindet, besteht aus denjenigen Knoten, die alle Randbedingungen des zu lösenden Problems erfüllen.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- UpdatePheromones kann sowohl während eine Lösung aufgebaut wird, als auch nachdem eine fertige Lösung vorliegt, aufgerufen werden, um die Menge des Pheromons an den betrachteten Kanten zu verändern.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- UpdatePheromones kann sowohl während eine Lösung aufgebaut wird, als auch nachdem eine fertige Lösung vorliegt, aufgerufen werden, um die Menge des Pheromons an den betrachteten Kanten zu verändern.
- UpdatePheromones berechnet ebenfalls die Verdunstung von Pheromon.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- DaemonActions ist ein Platzhalter für etwaige globale Aktionen, die nicht von einzelnen Ameisen durchgeführt werden können.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- ScheduleActivities ist dafür verantwortlich, diese drei Prozesse ConstructAntsSolutions, UpdatePheromones und DaemonActions zu koordinieren.

Ameisenkolonie-Optimierung

Die ACO Metaheuristik

ACO lässt sich wie folgt beschreiben.

```
procedure ACOMetaheuristic
   ScheduleActivities
      ConstructAntsSolutions
      UpdatePheromones
      DaemonActions    % optional
   end-ScheduleActivities
end-procedure
```

Quelle: M. Dorigo, T. Stützle. *Ant Colony Optimization*. MIT Press, 2004, S. 38

- ScheduleActivities ist dafür verantwortlich, diese drei Prozesse ConstructAntsSolutions, UpdatePheromones und DaemonActions zu koordinieren.
- Es ist der jeweiligen Implementation überlassen, inwieweit ScheduleActivities diese Prozesse synchronisiert und/oder parallelisiert.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph *G(N, K)* bestehend aus einer Menge *N* von Städten und einer Menge *K* von Kanten zwischen diesen Städten.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph *G(N, K)* bestehend aus einer Menge *N* von Städten und einer Menge *K* von Kanten zwischen diesen Städten.

- Für jede Kante $(i,j) \in K$ ist die Distanz d_{ij} bekannt. Die d_{ij} werden in einer Distanzmatrix gespeichert.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph *G(N, K)* bestehend aus einer Menge *N* von Städten und einer Menge *K* von Kanten zwischen diesen Städten.

- Für jede Kante $(i,j) \in K$ ist die Distanz d_{ij} bekannt. Die d_{ij} werden in einer Distanzmatrix gespeichert.
- Sollte der Graph unvollständig sein, können weitere Kanten hinzugefügt werden, deren Distanz so hoch ist, dass eine gute Lösung niemals diese Kante beinhaltet.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph *G(N, K)* bestehend aus einer Menge *N* von Städten und einer Menge *K* von Kanten zwischen diesen Städten.

- Für jede Kante $(i,j) \in K$ ist die Distanz d_{ij} bekannt. Die d_{ij} werden in einer Distanzmatrix gespeichert.
- Sollte der Graph unvollständig sein, können weitere Kanten hinzugefügt werden, deren Distanz so hoch ist, dass eine gute Lösung niemals diese Kante beinhaltet.
- Eine Variante dieses Problem ist asymmetrisch. Dann gibt es mindestens eine Kante für die $d_{ij} \neq d_{ji}$ gilt.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph *G(N, K)* bestehend aus einer Menge *N* von Städten und einer Menge *K* von Kanten zwischen diesen Städten.

- Für jede Kante $(i,j) \in K$ ist die Distanz d_{ij} bekannt. Die d_{ij} werden in einer Distanzmatrix gespeichert.
- Sollte der Graph unvollständig sein, können weitere Kanten hinzugefügt werden, deren Distanz so hoch ist, dass eine gute Lösung niemals diese Kante beinhaltet.
- Eine Variante dieses Problem ist asymmetrisch. Dann gibt es mindestens eine Kante für die $d_{ij} \neq d_{ji}$ gilt.
- Gesucht ist der kürzeste geschlossene Pfad, der alle Städte in *N* enthält.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

Gegeben ist ein Graph $G(N, K)$ bestehend aus einer Menge N von Städten und einer Menge K von Kanten zwischen diesen Städten.

- Für jede Kante $(i,j) \in K$ ist die Distanz d_{ij} bekannt. Die d_{ij} werden in einer Distanzmatrix gespeichert.
- Sollte der Graph unvollständig sein, können weitere Kanten hinzugefügt werden, deren Distanz so hoch ist, dass eine gute Lösung niemals diese Kante beinhaltet.
- Eine Variante dieses Problem ist asymmetrisch. Dann gibt es mindestens eine Kante für die $d_{ij} \neq d_{ji}$ gilt.
- Gesucht ist der kürzeste geschlossene Pfad, der alle Städte in N enthält.
- Der vollständige Graph im Problem des Handlungsreisenden kann als ein Konstruktionsgraph aufgefasst werden. Das Problem des Handlungsreisenden ist daher unmittelbar mit ACO behandelbar.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

- Jede Kante dieses Graphen wird mit einem Wert ϕ_{ij} versehen, der die Menge an Pheromon für diese Kante angibt. Diese Werte werden in einer Pheromon-Matrix gespeichert.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

- Jede Kante dieses Graphen wird mit einem Wert ϕ_{ij} versehen, der die Menge an Pheromon für diese Kante angibt. Diese Werte werden in einer Pheromon-Matrix gespeichert.
- Alle Werte in der Pheromon-Matrix werden mit einem Wert ϕ_0 initialisiert, um zu vermeiden, dass der Algorithmus zu stark von den ersten Suchergebnissen beeinflusst wird.

$$\forall (i, j): \phi_{ij} = \phi_0 = \frac{m}{C^{nn}}$$

Wobei *m* die Anzahl der Ameisen und C^{nn} beispielsweise die Länge der Rundreise mit der *Nearest-Neighbor Heuristik* ist.

Ameisenkolonie-Optimierung

Problem des Handlungsreisenden

- Jede Kante dieses Graphen wird mit einem Wert ϕ_{ij} versehen, der die Menge an Pheromon für diese Kante angibt. Diese Werte werden in einer Pheromon-Matrix gespeichert.
- Alle Werte in der Pheromon-Matrix werden mit einem Wert ϕ_0 initialisiert, um zu vermeiden, dass der Algorithmus zu stark von den ersten Suchergebnissen beeinflusst wird.

$$\forall (i, j): \phi_{ij} = \phi_0 = \frac{m}{C^{nn}}$$

Wobei m die Anzahl der Ameisen und C^{nn} beispielsweise die Länge der Rundreise mit der *Nearest-Neighbor Heuristik* ist.

- Es wird eine Heuristik verwendet um auszudrücken, wie wünschenswert es ist, nach der Stadt i die Stadt j zu besuchen. Diese ist invers proportional zu der Entfernung zwischen diesen beiden Städten, also $w_{ij} = 1 / d_{ij}$.

Ameisenkolonie-Optimierung

Konstruktion einer Lösung

Eine Ameise k welche sich in Stadt i aufhält, besucht anschließend Stadt j mit folgender Wahrscheinlichkeit:

$$p_{ij} = \frac{\phi_{ij}^{\alpha} * w_{ij}^{\beta}}{\sum_{l \in N_i^k} \phi_{il}^{\alpha} * w_{il}^{\beta}}$$

wobei N_i^k die mögliche Nachbarschaft von Ameise k ist, wenn sie sich in Stadt i befindet. Diese Nachbarschaft enthält nur die noch nicht besuchten Städte.

Ameisenkolonie-Optimierung

Konstruktion einer Lösung

Eine Ameise k welche sich in Stadt i aufhält, besucht anschließend Stadt j mit folgender Wahrscheinlichkeit:

$$p_{ij} = \frac{\phi_{ij}^{\alpha} * w_{ij}^{\beta}}{\sum_{l \in N_i^k} \phi_{il}^{\alpha} * w_{il}^{\beta}}$$

wobei N_i^k die mögliche Nachbarschaft von Ameise k ist, wenn sie sich in Stadt i befindet. Diese Nachbarschaft enthält nur die noch nicht besuchten Städte.

Die Parameter α und β legen den Einfluss des Pheromons und der Heurisik fest.

- Wenn $\alpha = 0$ ist, dann werden die am nächsten benachbarten Städte bevorzugt ausgewählt. Dies entspricht dem Verhalten eines gierigen Algorithmus *(greedy algorithm)*.

Ameisenkolonie-Optimierung

Konstruktion einer Lösung

Eine Ameise *k* welche sich in Stadt *i* aufhält, besucht anschließend Stadt *j* mit folgender Wahrscheinlichkeit:

$$p_{ij} = \frac{\phi_{ij}^{\alpha} * w_{ij}^{\beta}}{\sum_{l \in N_i^k} \phi_{il}^{\alpha} * w_{il}^{\beta}}$$

wobei N_i^k die mögliche Nachbarschaft von Ameise *k* ist, wenn sie sich in Stadt *i* befindet. Diese Nachbarschaft enthält nur die noch nicht besuchten Städte.

Die Parameter α und β legen den Einfluss des Pheromons und der Heurisik fest.

- Wenn $\alpha = 0$ ist, dann werden die am nächsten benachbarten Städte bevorzugt ausgewählt. Dies entspricht dem Verhalten eines gierigen Algorithmus *(greedy algorithm)*.
- Wenn $\beta = 0$ ist, dann werden benachbarte Städte nur nach der Pheromon-Markierung ausgewählt.

Ameisenkolonie-Optimierung

Berechnung des Pheromons

Nachdem alle Ameisen eine Rundreise berechnet haben, werden die Pheromon-Markierungen neu berechnet. Zuerst wird die Verdunstung des Pheromons mit Hilfe einer Verdunstungsrate ρ ermittelt.

$$\phi_{ij} = (1 - \rho) * \phi_{ij}$$

Ameisenkolonie-Optimierung

Berechnung des Pheromons

Nachdem alle Ameisen eine Rundreise berechnet haben, werden die Pheromon-Markierungen neu berechnet. Zuerst wird die Verdunstung des Pheromons mit Hilfe einer Verdunstungsrate ρ ermittelt.

$$\phi_{ij} = (1-\rho) * \phi_{ij}$$

Anschließend werden die Pheromon-Markierungen aller *m* Ameisen addiert:

$$\phi_{ij} = \phi_{ij} + \sum_{k=1}^{m} \Delta \phi_{ij}^{k}$$

Ameisenkolonie-Optimierung

Berechnung des Pheromons

Nachdem alle Ameisen eine Rundreise berechnet haben, werden die Pheromon-Markierungen neu berechnet. Zuerst wird die Verdunstung des Pheromons mit Hilfe einer Verdunstungsrate ρ ermittelt.

$$\phi_{ij} = (1-\rho) * \phi_{ij}$$

Anschließend werden die Pheromon-Markierungen aller *m* Ameisen addiert:

$$\phi_{ij} = \phi_{ij} + \sum_{k=1}^{m} \Delta \phi_{ij}^{k}$$

Wenn die Ameise *k* nach der Stadt *i* die Stadt *j* besucht, so ist ihr Teilbeitrag:

$$\Delta \phi_{ij}^{k} = 1 / L^{k}$$

wobei L^k die Länge der Rundreise von Ameise *k* ist. Andernfalls ist ihr Teilbeitrag natürlich 0. Je kürzer die Rundreise, desto mehr Pheromon wird also zur Markierung der Kante (i,j) beigesteuert.

Ameisenkolonie-Optimierung

Verbesserungen - Elitismus

Eine mögliche Verbesserung ist die Verwendung von Elitismus. Dabei wird bei der Berechnung des Pheromons zusätzlich noch Pheromon für die derzeit beste bekannte Rundreise R^{bb} hinzugefügt. Dies ist auch ein Beispiel für eine DaemonAction.

$$\phi_{ij} = \phi_{ij} + \sum_{k=1}^{m} \Delta \phi_{ij}^{k} + e \, \Delta \phi_{ij}^{bb}$$

Ameisenkolonie-Optimierung

Verbesserungen - Elitismus

Eine mögliche Verbesserung ist die Verwendung von Elitismus. Dabei wird bei der Berechnung des Pheromons zusätzlich noch Pheromon für die derzeit beste bekannte Rundreise R^{bb} hinzugefügt. Dies ist auch ein Beispiel für eine DaemonAction.

$$\phi_{ij} = \phi_{ij} + \sum_{k=1}^{m} \Delta \phi_{ij}^{k} + e \, \Delta \phi_{ij}^{bb}$$

Hierbei steuert das Gewicht e, wie stark R^{bb} in die Berechnung der Pheromon-Markierung eingeht. Es gilt

$$\Delta \phi_{ij}^{bb} = 1 / L^{bb}$$

wenn die derzeit beste bekannte Rundreise von Stadt i nach Stadt j führt, andernfalls ist dieser Wert gleich null. L^{bb} ist die Länge von R^{bb}.

Ameisenkolonie-Optimierung

Weitere Verbesserungen

- Rangbasierte Verfahren. Die Ameisen werden nach der Güte ihrer Lösung sortiert. Nur die besten k Ameisen markieren die Kanten mit Pheromon. Dabei kann der Suchraum stark eingeschränkt werden. Abhilfe kann durch das Einführen einer oberen Grenze *(min max ant system)* für die Menge des Pheromons geschaffen werden.

Ameisenkolonie-Optimierung

Weitere Verbesserungen

- Rangbasierte Verfahren. Die Ameisen werden nach der Güte ihrer Lösung sortiert. Nur die besten k Ameisen markieren die Kanten mit Pheromon. Dabei kann der Suchraum stark eingeschränkt werden. Abhilfe kann durch das Einführen einer oberen Grenze *(min max ant system)* für die Menge des Pheromons geschaffen werden.
- Verbesserung der Lösungen durch lokale Suche, bevor die erzeugte Rundreise zu Berechnung der Pheromonmenge herangezogen wird. Siehe auch die Übung von Teil 9.

Partikelschwarm-Optimierung

Flocks, Herds, and Schools: A Distributed Behavioral Model

C. W. Reynolds (1987).

The aggregate motion of a flock of birds, a herd of land animals, or a school of fish is a beautiful and familiar part of the natural world. But this type of complex motion is rarely seen in computer animation. This paper explores an approach based on simulation as an alternative to scripting the paths of each bird individually. The simulated flock is an elaboration of a particle system, with the simulated birds being the particles.

Quelle: Craig W. Reynolds. *Flocks, Herds, and Schools: A Distributed Behavioral Model*. Computer Graphics, Vol. 21, No. 4, 1987

Partikelschwarm-Optimierung

Partikelschwärme

J. Kennedy und *R. C. Eberhard* entwickelten in der Mitte der 1990er Jahre ein Optimierungsverfahren, die Partikelschwarmoptimierung *(particle swarm optimization, PSO)*, welches auf dem Konzept der sozialen Interaktion zwischen einzelnen Lebewesen innerhalb einer Gruppe *(particle swarm)* basiert.

Partikelschwarm-Optimierung

Partikelschwärme

J. Kennedy und *R. C. Eberhard* entwickelten in der Mitte der 1990er Jahre ein Optimierungsverfahren, die Partikelschwarmoptimierung *(particle swarm optimization, PSO)*, welches auf dem Konzept der sozialen Interaktion zwischen einzelnen Lebewesen innerhalb einer Gruppe *(particle swarm)* basiert.

> So why, after all, did we call our paradigm a „particle swarm?“ Well, to tell the truth, our very first programs were intended to model the coordinated movements of bird flocks and schools of fish. As the programs evolved from modeling social behavior to doing optimization, at some point the two-dimensional plots we used to watch the algorithms perform ceased to look much like bird flocks or fish schools and started looking more like swarms of mosquitos. The name came as simply as that.
>
> Quelle: James Kennedy, Russell C. Eberhard, Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001

Partikelschwarm-Optimierung

Publikationen und Referenzen zum Thema PSO

Year	Google Scholar Hits	IEEE Xplore Publications
1995	(28)	(2)
1996	(14)	(0)
1997	(24)	(3)
1998	(49)	(4)
1999	(62)	(7)
2000	(78)	(8)
2001	(118)	(11)
2002	(256)	(46)
2003	(373)	(82)
2004	(850)	(181)
2005	(1210)	(279)
2006	(1230)	(462)

Bildquelle: R. Poli, J. Kennedy, T. Blackwell. *Particle Swarm Optimization*. Swarm Intelligence, Vol.1, No. 1, June 2007

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm“) beträgt oftmals zwischen 20 und 50 Partikel.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm“) beträgt oftmals zwischen 20 und 50 Partikel.
- Die Simulation erfolgt in diskreten Zeitschritten *(iterations)*.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm") beträgt oftmals zwischen 20 und 50 Partikel.
- Die Simulation erfolgt in diskreten Zeitschritten *(iterations)*.
- Jedes Partikel besitzt zum Zeitpunkt t eine Position $p(t) = \langle p_1, p_2, \ldots, p_n \rangle$, sowie eine momentane Geschwindigkeit $v(t) = \langle v_1, v_2, \ldots, v_n \rangle$.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm“) beträgt oftmals zwischen 20 und 50 Partikel.
- Die Simulation erfolgt in diskreten Zeitschritten *(iterations)*.
- Jedes Partikel besitzt zum Zeitpunkt t eine Position $p(t) = \langle p_1, p_2, \ldots, p_n \rangle$, sowie eine momentane Geschwindigkeit $v(t) = \langle v_1, v_2, \ldots, v_n \rangle$.
- Ebenso kennt jedes Partikel seine beste bisher erreichte Fitness und die zugehörige Position p_{best}.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm") beträgt oftmals zwischen 20 und 50 Partikel.
- Die Simulation erfolgt in diskreten Zeitschritten *(iterations)*.
- Jedes Partikel besitzt zum Zeitpunkt t eine Position $p(t) = \langle p_1, p_2, \ldots, p_n \rangle$, sowie eine momentane Geschwindigkeit $v(t) = \langle v_1, v_2, \ldots, v_n \rangle$.
- Ebenso kennt jedes Partikel seine beste bisher erreichte Fitness und die zugehörige Position p_{best}.
- Bei jedem Zeitschritt wird die Fitness jedes Partikels berechnet. Ebenso wird seine Position und Geschwindigkeit neu berechnet.

Partikelschwarm-Optimierung

PSO - Die Grundidee

Ein PSO-Algorithms sucht nach einem Optimum (Maximum oder Minimum) in einem *n*-dimensionalen Raum.

- Lösungskandidaten werden durch einzelne Partikel repräsentiert. PSO ist also ebenso wie GA und ACO ein populationsbasiertes Verfahren. Die Größe der Population („Schwarm") beträgt oftmals zwischen 20 und 50 Partikel.
- Die Simulation erfolgt in diskreten Zeitschritten *(iterations)*.
- Jedes Partikel besitzt zum Zeitpunkt t eine Position $p(t) = \langle p_1, p_2, \ldots, p_n \rangle$, sowie eine momentane Geschwindigkeit $v(t) = \langle v_1, v_2, \ldots, v_n \rangle$.
- Ebenso kennt jedes Partikel seine beste bisher erreichte Fitness und die zugehörige Position p_{best}.
- Bei jedem Zeitschritt wird die Fitness jedes Partikels berechnet. Ebenso wird seine Position und Geschwindigkeit neu berechnet.
- Die Simulation endet, nachdem ein ausreichend guter Lösungskandidat gefunden wurde oder nach einer maximalen Anzahl von Zeitschritten.

Partikelschwarm-Optimierung

Nachbarschaften

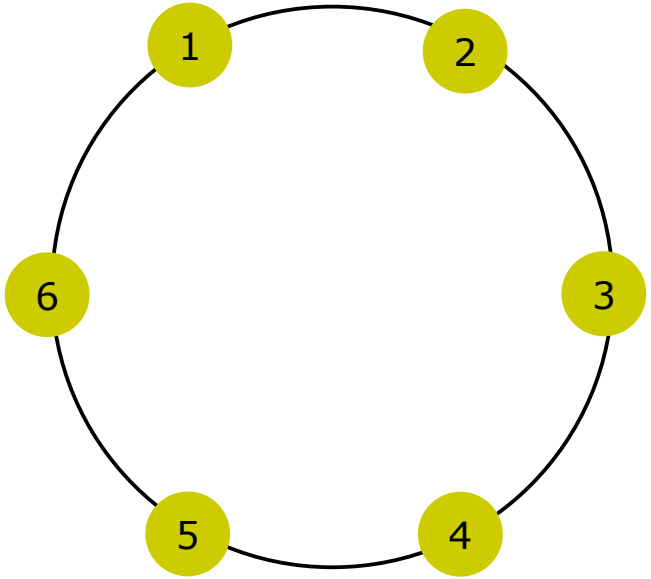
Die soziale Umgebung eines Partikels wird durch eine Nachbarschaft definiert.

Partikelschwarm-Optimierung

Nachbarschaften

Die soziale Umgebung eines Partikels wird durch eine Nachbarschaft definiert.

- Bei einem Ring *(ring topology)* kennt jedes Partikel genau zwei Nachbarn.

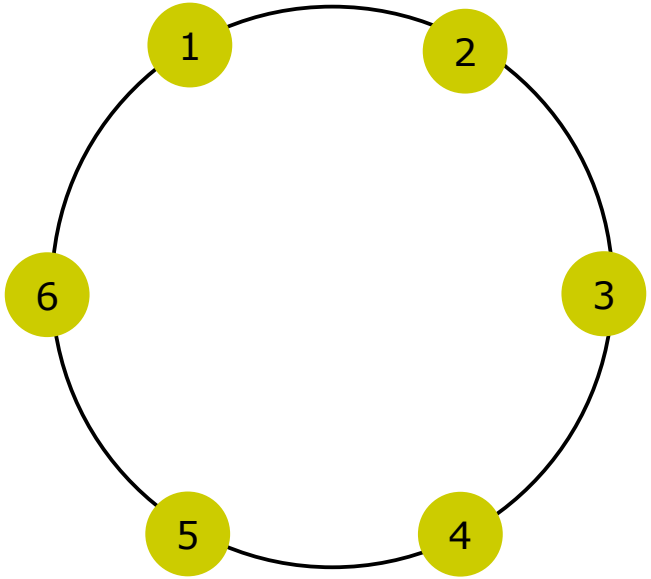


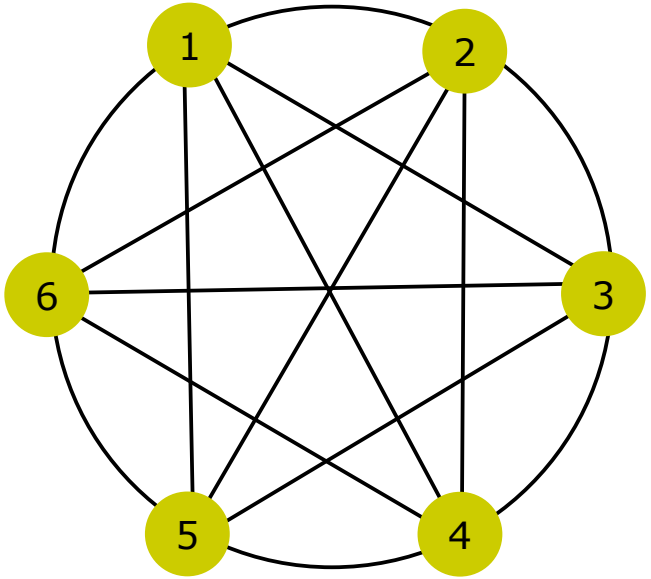
Partikelschwarm-Optimierung

Nachbarschaften

Die soziale Umgebung eines Partikels wird durch eine Nachbarschaft definiert.

- Bei einem Ring *(ring topology)* kennt jedes Partikel genau zwei Nachbarn.
- Bei einem Stern *(star topology)* gehören alle Partikel einer einzigen grossen Nachbarschaft an.





Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0,\phi_1) * (p_{besti}(t) - p_i(t)) + U(0,\phi_2) * (p_{global}(t) - p_i(t))$$

Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0, \phi_1) * (p_{besti}(t) - p_i(t)) + U(0, \phi_2) * (p_{global}(t) - p_i(t))$$

Dabei ist

- *U(a, b)* ein Zufallsvektor, dessen einzelne Elemente u_i gleichmäßig zwischen [*a, b*] verteilt sind.

Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0, \phi_1) * (p_{besti}(t) - p_i(t)) + U(0, \phi_2) * (p_{global}(t) - p_i(t))$$

Dabei ist

- *U(a, b)* ein Zufallsvektor, dessen einzelne Elemente u_i gleichmäßig zwischen [*a, b*] verteilt sind.
- $p_{besti}(t)$ die Position zu der bisher besten erreichten Fitness dieses Partikels.

Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0, \phi_1) * (p_{besti}(t) - p_i(t)) + U(0, \phi_2) * (p_{global}(t) - p_i(t))$$

Dabei ist

- *U(a, b)* ein Zufallsvektor, dessen einzelne Elemente u_i gleichmäßig zwischen [*a, b*] verteilt sind.
- $p_{besti}(t)$ die Position zu der bisher besten erreichten Fitness dieses Partikels.
- $p_{global}(t)$ ist die Position eines Partikels *k* aus der Nachbarschaft von Partikel *i*, mit der bisher höchsten bekannten Fitness in der gesamten Nachbarschaft.

Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0, \phi_1) * (p_{besti}(t) - p_i(t)) + U(0, \phi_2) * (p_{global}(t) - p_i(t))$$

Dabei ist

- *U(a, b)* ein Zufallsvektor, dessen einzelne Elemente u_i gleichmäßig zwischen [*a, b*] verteilt sind.
- $p_{besti}(t)$ die Position zu der bisher besten erreichten Fitness dieses Partikels.
- $p_{global}(t)$ ist die Position eines Partikels *k* aus der Nachbarschaft von Partikel *i*, mit der bisher höchsten bekannten Fitness in der gesamten Nachbarschaft.
- Die Parameter Φ_1 und Φ_2 steuern stochastisch, inwieweit sich ein Partikel auf seine eigene Erfahrung stützt, bzw. inwieweit es aus den Erfahrungen der Partikel in seiner Nachbarschaft lernt.

Partikelschwarm-Optimierung

Berechnung der Partikelgeschwindigkeit

Die neue Geschwindigkeit eines Partikels *i* wird wie folgt berechnet:

$$v_i(t+1) = \omega * v(t) + U(0, \phi_1) * (p_{besti}(t) - p_i(t)) + U(0, \phi_2) * (p_{global}(t) - p_i(t))$$

Dabei ist

- *U(a, b)* ein Zufallsvektor, dessen einzelne Elemente u_i gleichmäßig zwischen [*a, b*] verteilt sind.
- $p_{besti}(t)$ die Position zu der bisher besten erreichten Fitness dieses Partikels.
- $p_{global}(t)$ ist die Position eines Partikels *k* aus der Nachbarschaft von Partikel *i*, mit der bisher höchsten bekannten Fitness in der gesamten Nachbarschaft.
- Die Parameter Φ_1 und Φ_2 steuern stochastisch, inwieweit sich ein Partikel auf seine eigene Erfahrung stützt, bzw. inwieweit es aus den Erfahrungen der Partikel in seiner Nachbarschaft lernt.
- Der Parameter ω steuert die Viskosität des Mediums in dem die Partikel sich bewegen, und sollte im Laufe der Simulation abnehmen.

Partikelschwarm-Optimierung

Berechnung der Partikelposition

Die neue Position eines Partikels i ergibt sich wie folgt:

$$p_i(t+1) = p_i(t) + v_i(t)$$

Partikelschwarm-Optimierung

PSO Pseudocode

```
for (alle Partikel i ) {
    p_i = zufallsposition();
    v_i = <0, 0, ..., 0>;
}
while (Abbruchkriterium noch nicht erreicht)
    for (alle Partikel i)
        f = fitness (p_i)
        if f > f(p_besti) then p_besti = p_i;
        if f > f(p_global) then p_global = p_i;
    }
    for (alle Partikel i)
        v_i = neue Geschwindigkeit nach Folie 11.75
        p_i = neue Position nach Folie 11.76
    }
}
```

Partikelschwarm-Optimierung

Literatur

- R. Poli, J. Kennedy, T. Blackwell. *Particle Swarm Optimization*. Swarm Intelligence, Vol.1, No. 1, June 2007
- J. Kennedy, R. C. Eberhard, Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001