Ameisenkolonialgorithmen und andere schwarmbasierte Optimierungsverfahren

Christian Borgelt

Institut für Wissens- und Sprachverarbeitung
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2, D-39106 Magdeburg

borgelt@iws.cs.uni-magdeburg.de
http://fuzzy.cs.uni-magdeburg.de/∼borgelt/

Überblick

- **Optimierungsverfahren**
  - Allgemeine Problemstellung und Voraussetzungen
  - Einige einfache/klassische Optimierungsverfahren
- **Das Problem lokaler Optima**
  - Beispiele: reelle Funktion, Problem des Handlungsreisenden
  - Einige klassische Ansätze zum Problem lokaler Optima
- **Schwarm- und populationsbasierte Optimierung**
  - Grundlegendes Prinzip: Informationsaustausch zwischen Individuen
  - Genetische/Evolutionäre Algorithmen
  - Teilchenschwarmoptimierung (particle swarm optimization)
  - Ameisenkolonieoptimierung (ant colony optimization)
- **Zusammenfassung**

Optimierung: Problemstellung

- **Allgemeine Problemstellung**
  Gegeben sei eine Funktion $f : S \to \mathbb{R}$ $\quad (S : \text{Suchraum})$.
  Finde ein Element $s \in S$, das f optimiert (d.h. maximiert oder minimiert).

  O.B.d.A.: Finde ein Element $s \in S$, das f maximiert.
  (Ist f zu minimieren, kann man stattdessen $f' \equiv -f$ betrachten.)

- **Voraussetzung**
  Für ähnliche Elemente $s_1, s_2 \in S$ unterscheiden sich die Funktionswerte $f(s_1)$ und $f(s_2)$ nicht zu sehr (keine großen Sprünge in den Funktionswerten).

- **Mögliche Suchräume**
  Der n-dimensionale reelle Raum $\mathbb{R}^n$, die Permutationen von n Objekten, die Menge aller Partitionen einer gegebenen Menge etc.

- **Nebenbedingungen**
  Definition des Suchraums und Berücksichtigung in der Suche.

- **Mehrkriterienoptimierung**

Einige klassische Optimierungsverfahren

- **Vollständige/zufällige Enumeration** (brute force)
  Probiere alle/zufällig gewählte Elemente des Suchraums durch.
- **Analytische Lösung**
  Z.B. Ausnutzen der notwendigen Bedingung für ein Optimum (Pierre Fermat): Der Gradient der zu optimierenden Funktion (Vektor der partiellen Ableitungen der Funktion) verschwindet an einem Optimum.
- **Gradientenverfahren** (hill climbing)
  Ausnutzen lokaler Steigungsinformation und bessere Lösungen zu konstruieren; ggf. lokale Näherung durch eine einfache Funktion (z.B. Parabel).
- **Gierige Algorithmen** (greedy algorithms)
  Die Wahl lokal optimaler Alternativen kann das globale Optimum liefern.
- **Lokale Verbesserung / Zufallsaufstieg**
  Man kann (ggf. stochastisch) nach lokalen Verbesserungen suchen.
  Benötigt Definition einer Umgebung oder Nachbarschaft.

Gradientenverfahren

Voraussetzungen: $S \subseteq \mathbb{R}^n$, $f : S \to \mathbb{R}$ ist differenzierbar.

Idee: Mache von einem zufälligen Startpunkt aus kleine Schritte in Richtung des stärksten Anstiegs der Funktion, bis ein (lokales) Maximum erreicht ist.

1. Wähle (zufälligen) Startpunkt $\vec{x}^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right)$.

2. Bestimme den Gradienten am aktuellen Punkt $\vec{x}^{(i)}$:

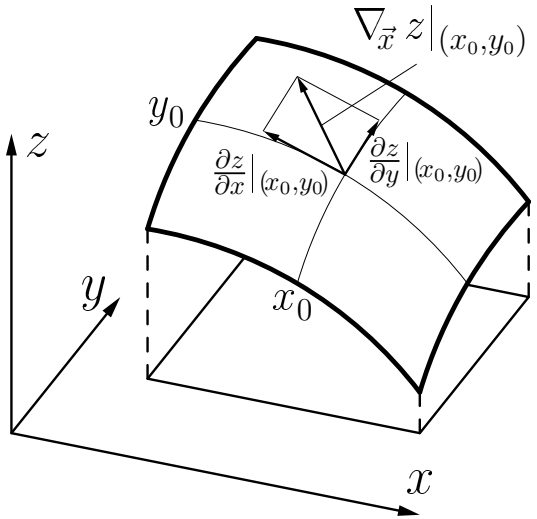
$$\nabla_{\vec{x}} f\left(\vec{x}^{(i)}\right) = \left(\frac{\partial}{\partial x_1} f\left(\vec{x}^{(i)}\right), \ldots, \frac{\partial}{\partial x_n} f\left(\vec{x}^{(i)}\right)\right)$$

3. Gehe ein kleines Stück in Richtung des Gradienten:

$$\vec{x}^{(i+1)} = \vec{x}^{(i)} + \eta \, \nabla_{\vec{x}} f\left(\vec{x}^{(i)}\right).$$

   η ist ein Schrittweitenparameter („Lernrate“ in neuronalen Netzen)

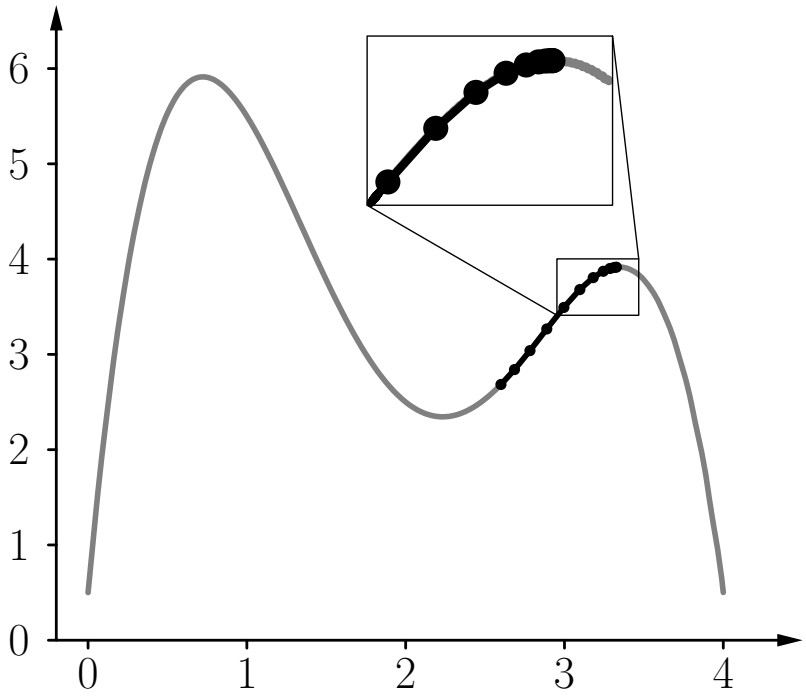
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.
   (Vorgegebene Anzahl Schritte ausgeführt, aktueller Gradient sehr klein)



Beispiel: Optimierung einer reellen Funktion

Beispielfunktion: $$f(x) = -\frac{5}{6}x^4 + 7x^3 - \frac{115}{6}x^2 + 18x + \frac{1}{2},$$

i	x_i	$f(x_i)$	$f'(x_i)$	Δx_i
0	2.600	2.684	1.707	0.085
1	2.685	2.840	1.947	0.097
2	2.783	3.039	2.116	0.106
3	2.888	3.267	2.153	0.108
4	2.996	3.492	2.009	0.100
5	3.097	3.680	1.688	0.084
6	3.181	3.805	1.263	0.063
7	3.244	3.872	0.845	0.042
8	3.286	3.901	0.515	0.026
9	3.312	3.911	0.293	0.015
10	3.327	3.915		



Gradientenaufstieg mit Startwert $x_0 = 2.6$ und Schrittweite $\eta = 0.05$.

Lokale Verbesserung / Zufallsaufstieg

Idee: Wenn die Funktion f nicht differenzierbar ist, kann man versuchen, die/eine Richtung, in der die Funktion f ansteigt, durch Auswerten aller/zufälliger Punkte in einer Umgebung des aktuellen Punktes zu bestimmen.

1. Wähle einen (zufälligen) Startpunkt $s_0 \in S$.
2. Wähle einen Punkt $s' \in S$ „in der Nähe“ des aktuellen Punktes s_i.
   (z.B. durch zufällige, aber nicht zu große Veränderung von s_i)
3. Setze
$$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i), \\ s_i, & \text{sonst.} \end{cases}$$
4. Wiederhole Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

Alternativ kann man in Schritt 2 mehrere/alle Punkte s' auf einmal erzeugen und dann zum besten übergehen, vorausgesetzt, er ist besser als der aktuelle Punkt.

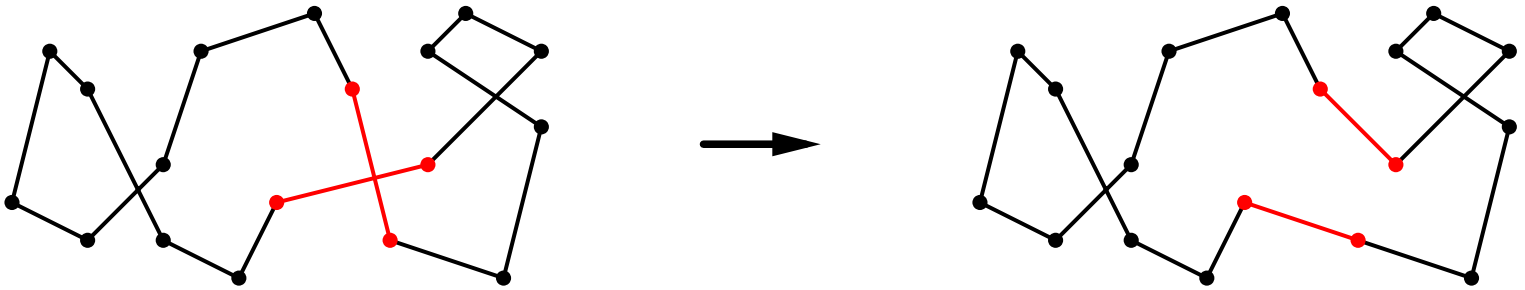
Beispiel: Problem des Handlungsreisenden

- **Gegeben:**
  - Eine Menge von n Städten (als Punkte in einer Ebene)
  - Abstände/Kosten der Wege zwischen den Städten (müssen nicht dem euklidischen Abstand entsprechen, können eventuell sogar asymmetrisch sein)
- **Gesucht:**
  - Rundreise minimaler Länge/Kosten durch alle n Städte, auf der keine Stadt mehr als einmal besucht wird
- **Mathematisch:** Suche eines Hamiltonkreises (enthält jeden Knoten genau einmal) mit minimalem Gewicht in einem Graphen mit gewichteten Kanten.
- **Bekannt:** Dieses Problem ist NP-vollständig, d.h., man kennt keinen Algorithmus, der dieses Problem in polynomialer Zeit löst.
- **Daher:** Für großes n ist in annehmbarer Zeit nur eine Näherungslösung berechenbar (die beste Lösung kann gefunden werden, dies ist aber nicht garantiert).

Beispiel: Problem des Handlungsreisenden

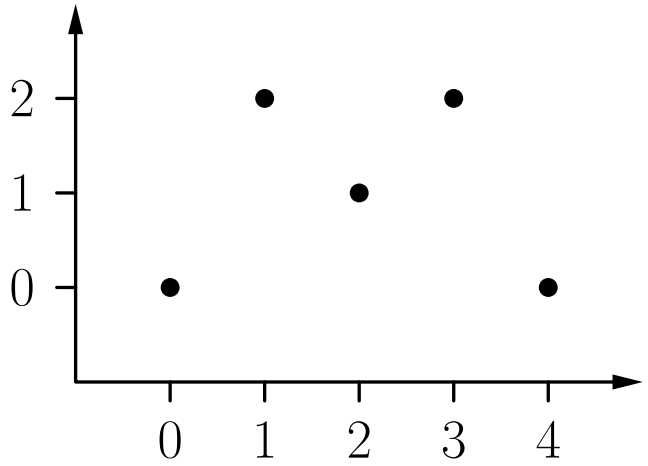
Betrachte Ansatz mit lokaler Verbesserung/Zufallsaufstieg

1. Bringe die Städte in eine zufällige Reihenfolge (zufällige Rundreise).
2. Wähle zufällig zweimal zwei Städte, die in der aktuellen Rundreise aufeinander folgen (alle vier Städte verschieden). Trenne die Rundreise zwischen den Städten jedes Paares auf und drehe den dazwischenliegenden Teil um.
3. Wenn die so entstehende Rundreise besser (kürzer, billiger) ist als die alte, ersetze die alte Rundreise durch die neue.
4. Wiederhole die Schritte 2 und 3, bis ein Abbruchkriterium erfüllt ist.

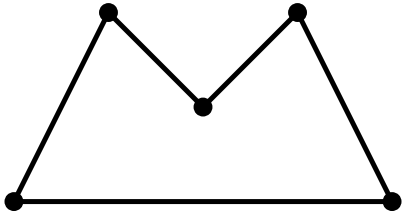


Beispiel: Problem des Handlungsreisenden

- Lokale Verbesserung kann in einem lokalen Minimum hängenbleiben.
  Dazu ein einfaches Beispiel mit 5 Städten:

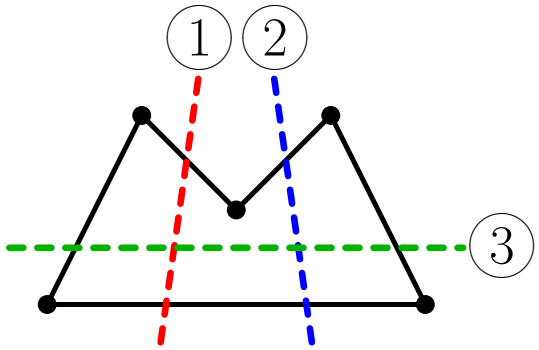


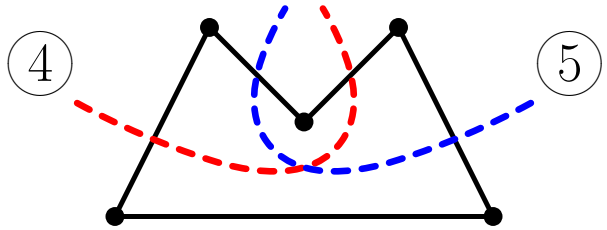
anfängliche Rundreise



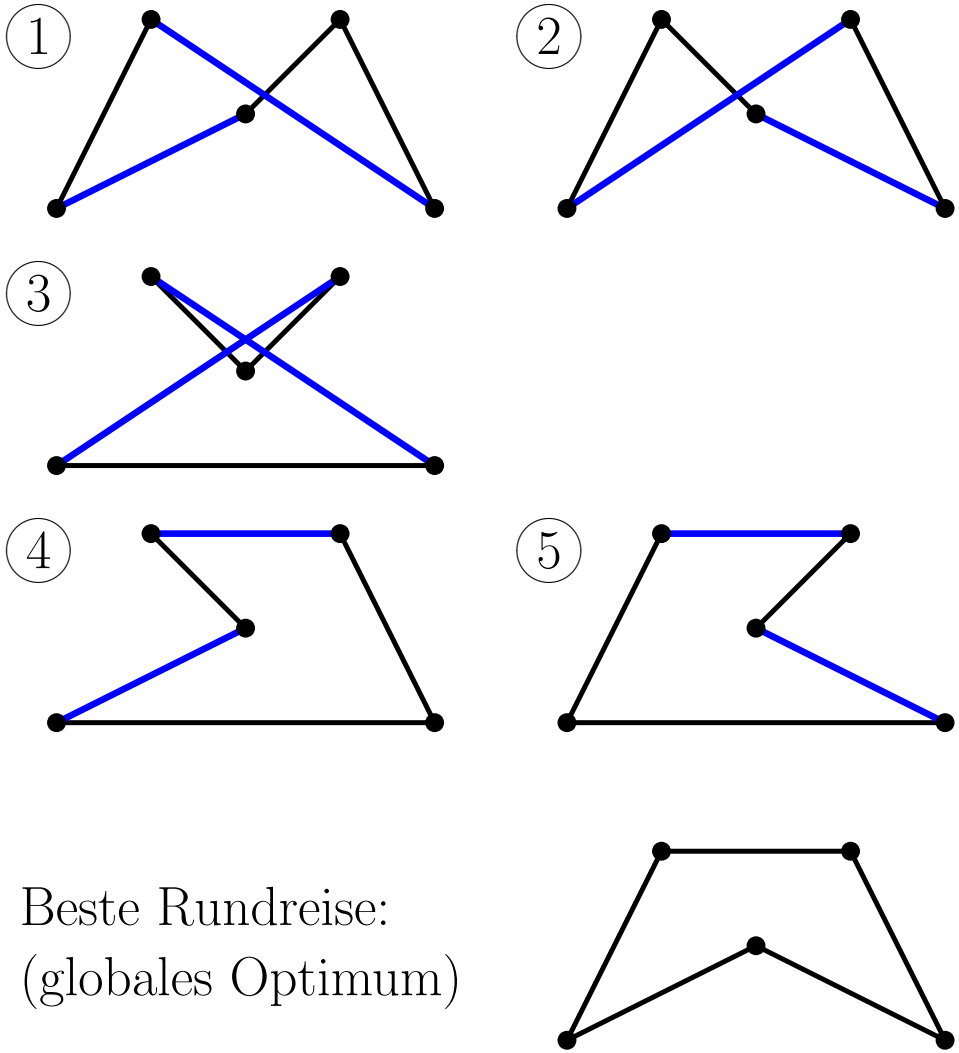
Länge: $2\sqrt{2} + 2\sqrt{5} + 4 \approx 11.30$

mögliche Teilungen der Rundreise





Beispiel: Problem des Handlungsreisenden



Länge: $\sqrt{2} + 3\sqrt{5} + \sqrt{13} \approx 11.73$

Länge: $\sqrt{2} + 2\sqrt{13} + 4 \approx 14.04$

Länge: $\sqrt{2} + 2\sqrt{5} + 2 + 4 \approx 11.89$

Beste Rundreise:
(globales Optimum)

Länge: $4\sqrt{5} + 2 \approx 10.94$

Beispiel: Problem des Handlungsreisenden

- Alle Modifikationen der Anfangsrundreise führen zu Rundreisen, die schlechter sind. Das globale Optimum kann daher, ausgehend von dieser Rundreise, mit rein lokalen Verbesserungen nicht gefunden werden.

- **Beachte:** Es kann von der betrachteten Menge an Operationen (definieren die Umgebung/Nachbarschaft einer Rundreise) abhängen, ob die Suche in einem lokalen Optimum hängenbleiben kann:

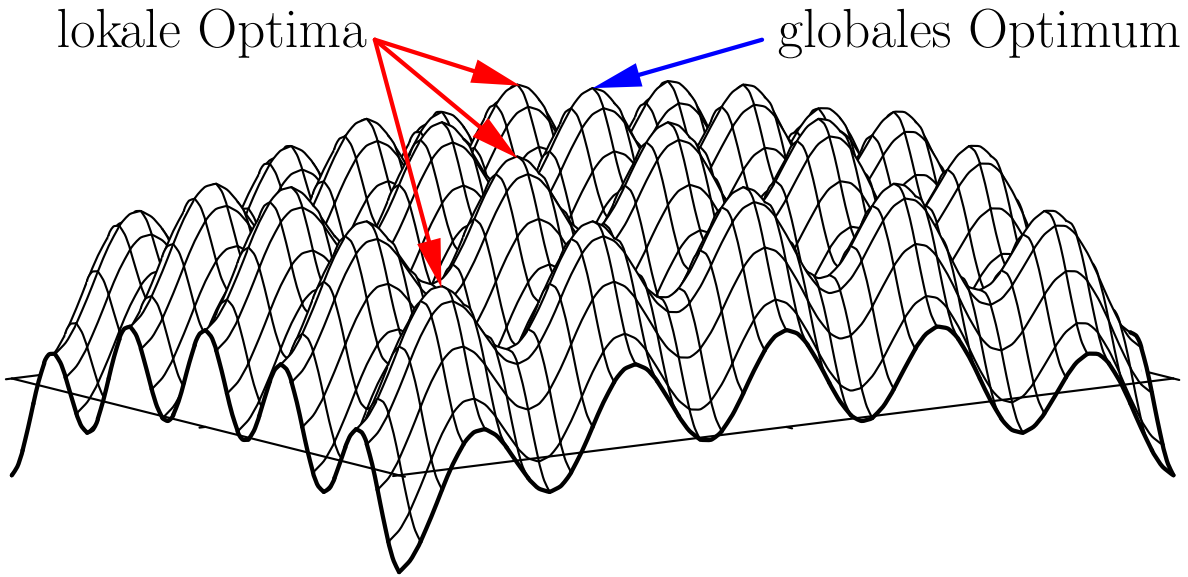
  Läßt man als weitere Operation zu, daß die Position einer Stadt in der Rundreise geändert wird (Entfernen von der aktuellen Position und Einfügen an einer anderen), so tritt im betrachteten Beispiel kein Hängenbleiben mehr auf.

  Auch für diese Operationenmenge/Nachbarschaft läßt sich jedoch ein Beispiel konstruieren, in dem die Suche in einem lokalen Minimum hängenbleibt.

- **Extremfall:** Erlaube als Operation eine beliebige Umordnung der Städte. Kein lokales Optimum mehr (da jede Rundreise erzeugt werden kann), aber dafür gibt es „zu viele“ mögliche Modifikationen einer Rundreise.

Das Problem lokaler Optima

- Erlaubt man in jedem Schritt nur eine Verbesserung der Lösung, erreicht man u.U. nur ein lokales Optimum.
- Das globale Optimum zu finden, kann sogar sehr unwahrscheinlich werden, nämlich wenn die zu optimierende Funktion viele lokale Optima besitzt.



$$f(x,y) = \frac{1}{2}(\cos(5\pi x) + \cos(5\pi y)) - x^2 - y^2 + 2, \qquad (x,y) \in [0,1]^2$$

Einige klassische Ansätze zum Problem lokaler Optima

- **Grundprinzip:** Erlaube auch Verschlechterungen der Lösung.
- **Simuliertes Ausglühen** (s': modifizierter Lösungskandidat)

$$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i), \\ s' & \text{mit Wahrscheinlichkeit } p = e^{-\frac{\Delta f}{kT}}, \\ s_i & \text{mit Wahrscheinlichkeit } 1-p, \end{cases} \quad \text{sonst.}$$

  $\Delta f = f(s_i) - f(s')$, $k = \Delta f_{\max}$ (Schätzung der) Spannweite der Funktion.
  T Temperaturparameter; wird im Laufe der Zeit (langsam) gesenkt.

- **Akzeptieren mit Schwellenwert**

$$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq f(s_i) - \theta, \\ s_i, & \text{sonst.} \end{cases}$$

  θ wird im Laufe der Zeit (langsam) gesenkt.

- **Sintflut-Algorithmus**

$$s_{i+1} = \begin{cases} s', & \text{falls } f(s') \geq \theta, \\ s_i, & \text{sonst.} \end{cases}$$

  θ wird im Laufe der Zeit (langsam) erhöht.

Klassische Optimierungsverfahren: Probleme

- Alle bisher erwähnten Verfahren suchen im wesentlichen **lokal**:
  - Es wird stets nur ein aktueller Lösungskandidat betrachtet.
  - Der aktuelle Lösungskandidat wird nur geringfügig verändert.
- **Nachteil:** Es wird u.U. nur ein kleiner Teil des Suchraums betrachtet.
- **Abhilfe:** Mehrere Läufe des Verfahrens mit verschiedenen Startpunkten.
  Nachteil: Keine Informationsübertragung von einem Lauf zum nächsten.
- Beachte: Große Veränderungen des Lösungskandidaten, bis hin zu völliger Neuberechnung, sind nicht sinnvoll, da dann zu wenig/keine Information von einem Lösungskandidaten zum nächsten weitergegeben wird.

⇒ **Wichtig sind:**
  - großräumige Abdeckung des Suchraums (Exploration)
  - Zusammenhang der erzeugten Lösungskandidaten
  - Informationsaustausch zwischen parallelen Prozessen

Schwarm- und populationsbasierte Optimierung

Swarm Intelligence
Bereich der KI, in dem intelligente Multi-Agenten-Systeme entwickelt werden.
Inspiration durch das Verhalten bestimmter Tierarten, speziell

- sozialer Insekten (z.B. Ameisen, Termiten, Bienen etc.) und
- in Schwärmen lebender Tiere (z.B. Fische, Vögel etc.).

Tiere solcher Arten können recht komplexe Aufgaben bewältigen (Finden von Nahrungsquellen, Wegesuche, Nestbau etc.), indem sie zusammenarbeiten.

Wesentliche Ideen

- Einzelindividuen sind i.a. ziemlich einfach, haben nur begrenzte Fähigkeiten.
- Koordination meist ohne zentrale Steuerung, sondern durch Selbstorganisation.
- Austausch von Informationen zwischen Individuen; Kooperation.

Man kann die Verfahren nach der Art des Informationsaustauschs klassifizieren.

Schwarm- und populationsbasierte Optimierung

- **Genetische/Evolutionäre Algorithmen**
  - Biologisches Vorbild: Evolution der Lebewesen
  - Informationsaustausch: durch Rekombination der Genotypen
  - Jedes Individuum ist ein Lösungskandidat
- **Teilchenschwarmoptimierung**
  - Biologisches Vorbild: Futtersuche von Fisch- und Vogelschwärmen
  - Informationsaustausch: über einfache Aggregation der Einzellösungen
  - Jedes Individuum ist ein Lösungskandidat
- **Ameisenkolonialgorithmen**
  - Biologisches Vorbild: Wegesuche zu Futterquellen durch Ameisen
  - Informationsaustausch: über Veränderung der Umgebung (Stigmergie; erweiterter Phänotyp nach Dawkins)
  - Individuen konstruieren Lösungskandidaten

Genetische/Evolutionäre Algorithmen

- **Grundsätzliches Prinzip** der biologischen Evolutionstheorie:

  Durch zufällige Variation entstehende vorteilhafte Eigenschaften werden durch natürliche Auslese ausgewählt.

  (Individuen mit vorteilhaften Eigenschaften haben bessere Fortpflanzungs- und Vermehrungschancen — „differentielle Reproduktion“.)

- Ein genetischer Algorithmus besteht aus:
  - einer **Kodierungsvorschrift** für die Lösungskandidaten,
  - einer Methode, eine **Anfangspopulation** zu erzeugen,
  - einer **Bewertungsfunktion (Fitneßfunktion)** für die Individuen,
  - einer **Auswahlmethode** auf der Grundlage der Fitneßfunktion,
  - **genetischen Operatoren**, die die Lösungskandidaten verändern,
  - Werten für verschiedene **Parameter** und einem **Abbruchkriterium**.

Grundstruktur eines genetischen Algorithmus

```
procedure evolution_program;
begin
  t := 0;                                  (* initialisiere den Generationenzähler *)
  initialize pop(t);                       (* erzeuge die Anfangspopulation *)
  evaluate pop(t);                         (* und bewerte sie (berechne Fitneß) *)
  while not termination criterion do       (* solange Abbruchkriterium nicht erfüllt *)
    t := t + 1;                            (* zähle die erzeugte Generation *)
    select pop(t) from pop(t − 1);         (* wähle Individuen nach Fitneß aus *)
    alter pop(t);                          (* wende genetische Operatoren an *)
    evaluate pop(t);                       (* bewerte die neue Population *)
  end                                      (* (berechne neue Fitneß) *)
end
```

- Durch die Auswahl wird eine Art „Zwischenpopulation“ von Individuen mit (im Durchschnitt) hoher Fitneß erzeugt.
- Nur die Individuen der Zwischenpopulation können Nachkommen bekommen.

Teilchenschwarmoptimierung

© Eric T. Schulz http://www.eeb.uconn.edu/courses/eeb296/

© Ariel Bravy http://www.skphoton.com/albums/

- Fische und Vögel suchen in Schwärmen nach ergiebigen Futterplätzen.
- Neben individueller Suche (kognitiver Anteil) orientieren sie sich außerdem an den anderen Mitgliedern des Schwarms in ihrer Nähe (sozialer Anteil).
- Das Leben im Schwarm hat außerdem die Funktion, die Individuen gegen Freßfeinde zu schützen.

Teilchenschwarmoptimierung

Particle Swarm Optimization [Kennedy and Eberhart 1995]

- **Motivation:** Verhalten von z.B. Fischschwärmen bei der Futtersuche: Zufälliges Ausschwärmen, aber stets auch Rückkehr zum Schwarm. Informationsaustausch zwischen den Mitgliedern des Schwarms.
- **Ansatz:** Statt nur einem einzelnen aktuellen Lösungskandidaten wird ein „Schwarm“ von m Lösungskandidaten verwendet.
- **Voraussetzung:** Der Suchraum ist reellwertig, d.h. $S \subseteq \mathbb{R}^n$, und folglich die zu optimierende (o.B.d.A.: zu maximierende) Funktion $f : \mathbb{R}^n \to \mathbb{R}$.
- **Vorgehen:** Jeder Lösungskandidat wird als „Teilchen“ aufgefaßt, das einen Ort $\vec{x}_i$ im Suchraum und eine Geschwindigkeit $\vec{v}_i$ hat, $i = 1, \ldots, m$.
- Teilchenschwarmoptimierung kann als ein Verfahren gesehen werden, das Elemente der bahnorientierten Suche (z.B. Gradientenverfahren) und populationsbasierter Suche (z.B. genetische Algorithmen) zusammenbringt.

Teilchenschwarmoptimierung

- **Aktualisierungsformeln** für Ort und Geschwindigkeit des i-ten Teilchens:

$$\begin{aligned}\vec{v}_i(t+1) &= \alpha \cdot \vec{v}_i(t) + \beta_1 \cdot \left(\vec{x}_i^{\,(\text{lokal})}(t) - \vec{x}_i(t)\right) + \beta_2 \cdot \left(\vec{x}^{\,(\text{global})}(t) - \vec{x}_i(t)\right) \\ \vec{x}_i(t+1) &= \vec{x}_i(t) + \vec{v}_i(t)\end{aligned}$$

- Parameterwahl: β_1, β_2 zufällig in jedem Schritt, α mit der Zeit abnehmend.

- $\vec{x}_i^{\,(\text{lokal})}$ ist das **lokale Gedächtnis** des Individuums (Teilchens).
  Es ist der beste Ort im Suchraum, den das Teilchen bisher besucht hat, d.h.

$$\vec{x}_i^{\,(\text{lokal})}(t) = \vec{x}_i \left(\text{argmax}_{u=1}^{t} f(\vec{x}_i(u))\right)$$

- $\vec{x}^{\,(\text{global})}$ ist das **globale Gedächtnis** des Schwarms.
  Es ist der beste Ort im Suchraum, den ein Individuum des Schwarms bisher besucht hat (bester bisher gefundener Lösungskandidat), d.h.

$$\vec{x}^{\,(\text{global})}(t) = \vec{x}_j^{\,(\text{lokal})}(t) \qquad \text{mit} \qquad j = \text{argmax}_{i=1}^{m}\, f\left(\vec{x}_i^{\,(\text{lokal})}(t)\right).$$

Teilchenschwarmoptimierung: Pseudocode

procedure pso; (∗ particle swarm optimization ∗)

for each particle i **do begin** (∗ initialisiere den Ort jedes Teilchens ∗)
choose random $\vec{x}_i$; $\vec{v}_i = \vec{0}$; **end**; (∗ zufällig im Suchraum ∗)
repeat (∗ bewege die Teilchen des Schwarms ∗)
for each particle i **do begin** (∗ durchlaufe die Teilchen ∗)
$y := f(\vec{x}_i)$; (∗ berechne Funktion am Ort des Teilchens ∗)
if $y \geq f\left(\vec{x}_i^{\,(\text{lokal})}\right)$ **then** $\vec{x}_i^{\,(\text{lokal})} := \vec{x}_i$;
if $y \geq f\left(\vec{x}^{\,(\text{global})}\right)$ **then** $\vec{x}^{\,(\text{global})} := \vec{x}_i$;
end; (∗ aktualisiere lokales/globales Gedächtnis ∗)
for each particle i **do begin** (∗ durchlaufe die Teilchen erneut ∗)
$\vec{v}_i := \alpha \cdot \vec{v}_i + \beta_1 \cdot \left(\vec{x}_i^{\,(\text{lokal})} - \vec{x}_i\right) + \beta_2 \cdot \left(\vec{x}^{\,(\text{global})} - \vec{x}_i\right)$;
$\vec{x}_i := \vec{x}_i + \vec{v}_i$; (∗ aktualisiere die Geschwindigkeit ∗)
end; (∗ und den Ort jedes Teilchens ∗)
until stopping criterion is met;

Teilchenschwarmoptimierung: Erweiterungen

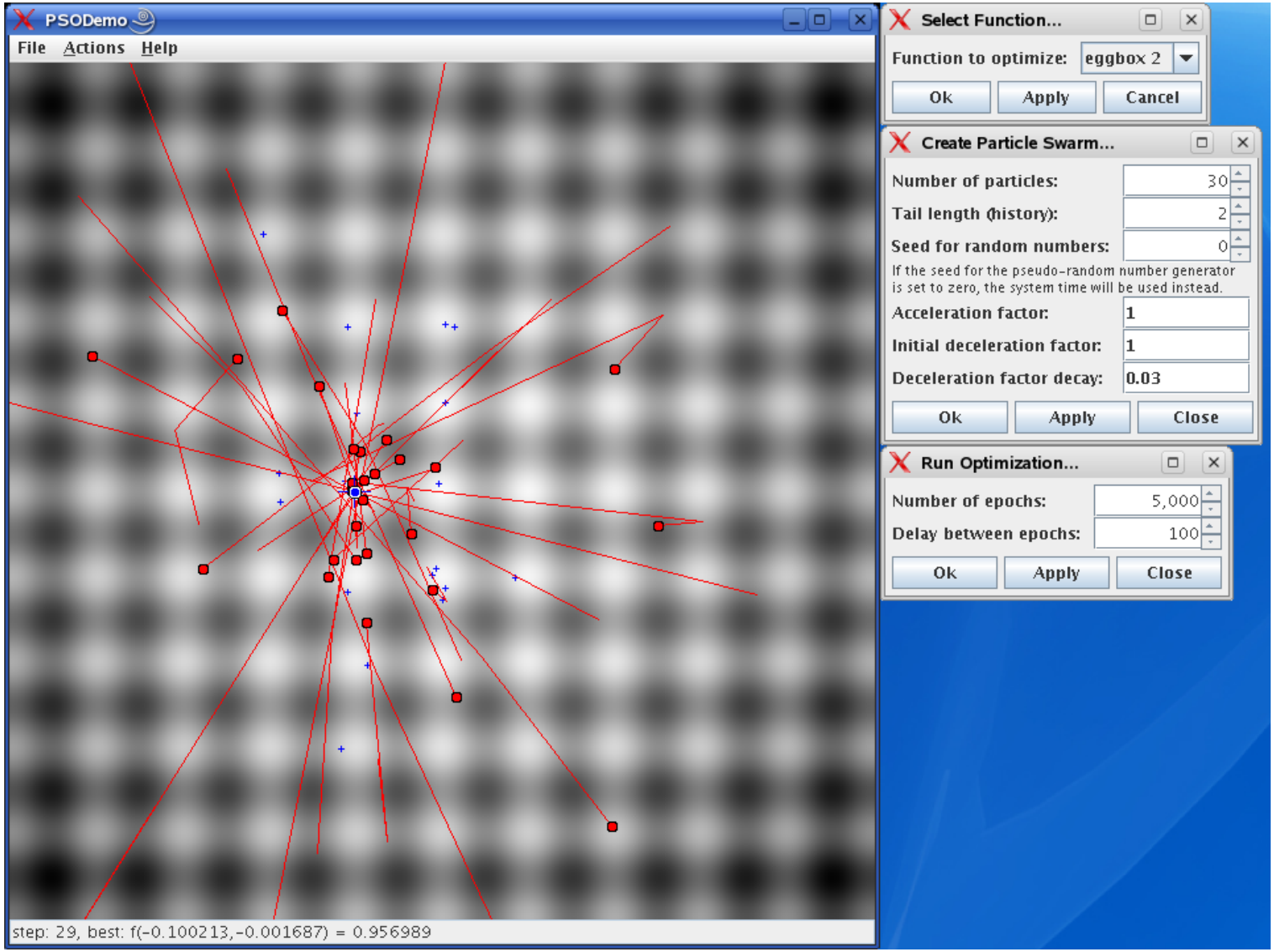
- **Beschränkter Suchraum**
  Ist der Suchraum eine echte Teilmenge des $\mathbb{R}^n$ (z.B. ein Hyperwürfel $[a, b]^n$), so werden die Teilchen an den Grenzen des Suchraums reflektiert.

- **Lokale Umgebung eines Teilchens**
  Statt des globalen Gedächtnisses des Schwarms wird das beste lokale Gedächtnis nur eines Teils des Schwarms verwendet, z.B. der Teilchen, die sich in der näheren Umgebung des zu aktualisierenden Teilchens befinden.

- **Automatische Parameteranpassung**
  Z.B. Anpassung der Schwarmgröße: Teilchen, deren lokales Gedächtnis deutlich schlechter ist als das der Teilchen in ihrer Nähe, werden entfernt.

- **Diversitätskontrolle**
  Vorzeitige Konvergenz auf suboptimale Lösungen soll verhindert werden.
  Dazu kann z.B. bei der Aktualisierung der Geschwindigkeit eine zusätzliche Zufallskomponente eingeführt werden, die die Diversität erhöht.

Teilchenschwarmoptimierung



Ameisenkolonieoptimierung

© PeTA http://www.helpingwildlife.com/ants.asp

© NickLyonMedia http://nicklyon.orchardhostings4.co.uk

- Da gefundenes Futter zur Versorgung der Nachkommen zum Nest transportiert werden muß, bilden Ameisen Transportstraßen.
- Dazu markieren sie die Wege zu Futterplätzen mit Duftstoffen (Pheromonen), so daß andere Ameisen der Kolonie diese Futterplätze auch finden können.
- Die Weglängen zu den Futterplätzen werden annähernd minimiert.

Ameisenkolonieoptimierung

Ant Colony Optimization [Dorigo *et al.* 1996, Dorigo and Gambardella 1997]

- **Motivation:** Ameisen einiger Arten finden kürzeste Wege zu Futterquellen durch Legen und Verfolgen von Pheromonmarkierungen („Duftmarken“).
  - Intuitiv: Kürzere Wege erhalten in gleicher Zeit mehr Pheromon.
  - Wege werden zufällig nach der vorhandenen Pheromonmenge gewählt. (Es ist um so wahrscheinlicher, daß ein Weg gewählt wird, je mehr Pheromon sich auf dem Weg befindet.)
  - Die Menge des ausgebrachten Pheromons kann von der Qualität und der Menge des gefundenen Futters abhängen.

- **Grundprinzip: Stigmergie** (stigmergy)
  Zur Wegesuche kommunizieren Ameisen indirekt über Pheromonablagerungen.

  Stigmergie (indirekte Kommunikation durch Veränderung der Umgebung) ermöglicht global angepaßtes Verhalten auf Grund lokaler Informationen.

Ameisenkolonieoptimierung

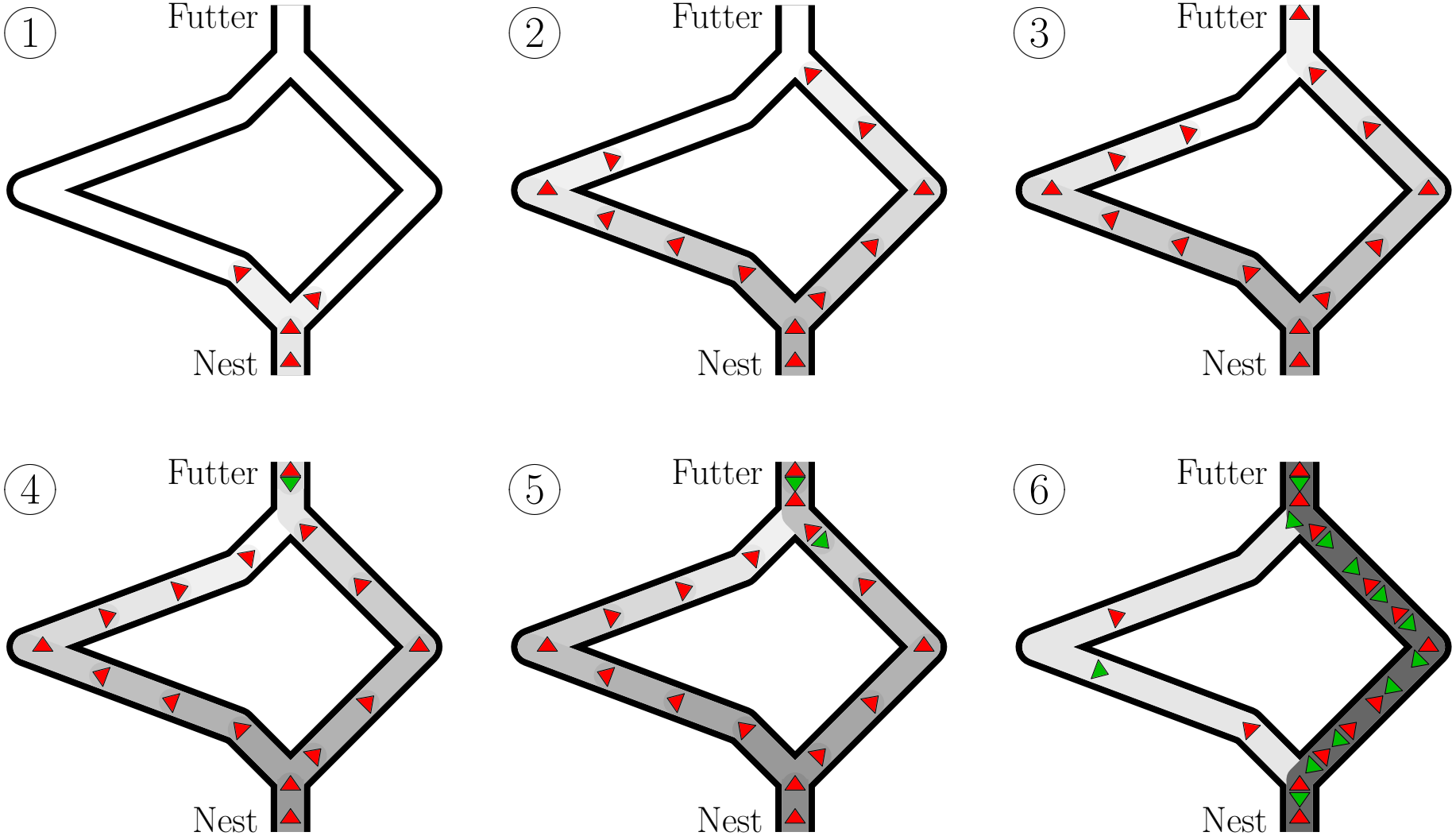
Doppelbrückenexperiment [Goss *et al.* 1989]

- Ameisennest und Futterquelle werden durch eine Doppelbrücke verbunden.
  Die beiden Zweige der Brücke sind verschieden lang.
- Experiment mit der argentinischen Ameise *Iridomyrmex Humilis*.
  Diese Ameisenart ist (wie fast alle anderen auch) fast blind.
  (Die Ameisen können also nicht sehen, welches der kürzere Weg ist.)
- In den meisten Versuchen benutzten schon nach wenigen Minuten
  fast alle Ameisen den kürzeren Weg.

Erklärung

- Auf dem kürzeren Weg erreichen die Ameisen das Futter schneller.
  Das Ende des kürzeren Weges erhält daher (am Anfang) mehr Pheromon.
- Auf dem Rückweg wird wegen der entstandenen Pheromondifferenz
  mit höherer Wahrscheinlichkeit der kürzere Weg gewählt.
  Dies führt zu einer Verstärkung der Pheromondifferenz.

Doppelbrückenexperiment



Doppelbrückenexperiment

- **Prinzip:** Der kürzere Weg wird systematisch verstärkt (Autokatalyse):

  mehr Pheromon auf Weg $\rightleftharpoons$ mehr Ameisen wählen Weg

- **Beachte:** Der kürzere Weg wird nur gefunden, weil die Ameisen sowohl auf dem Hin- als auch auf dem Rückweg Pheromon ablegen.

- Wird z.B. nur auf dem Hinweg Pheromon abgelegt:
  - Auf dem Hinweg zur Futterquelle kann keiner der beiden Wege bevorzugt werden, da keine Pheromondifferenz vorliegt oder systematisch entsteht.
  - Am Vereinigungspunkt der Brücken verringert sich das Verhältnis der Pheromonmengen im Laufe der Zeit und verschwindet schließlich nahezu.
  - Durch zufällige Fluktuationen in der Wegewahl konvergiert die Wegesuche ggf. dennoch zufällig(!) auf einen der beiden Brückenzweige.

- Analoge Argumente (symmetrische Situation) können angewandt werden, wenn Pheromon nur auf dem Rückweg abgelegt wird.

Doppelbrückenexperiment

- **Beachte:** Der kürzere Weg wird gefunden, weil schon zu Beginn beide Zweige der Brücke zur Verfügung stehen und auf beiden kein Pheromon liegt.

  Das Ende des kürzeren Weges wird früher von mehr Ameisen erreicht.
  Dies führt zu unterschiedlichen Mengen an Pheromon auf den beiden Wegen, was zu einen sich selbst verstärkenden Prozeß führt.

- **Fragen:** Was passiert, wenn durch Veränderung der Umgebung ein neuer Weg möglich wird, der kürzer ist als der bisherige?
  Wird auf diesen kürzeren Weg gewechselt?

- **Antwort:** *Nein!* [Goss *et al.* 1989]
  Ist erst einmal ein Weg etabliert, so wird dieser beibehalten.

- Nachweis durch zweites Brückenexperiment, in dem am Anfang nur der längere Brückenzweig zur Verfügung steht und der kürzere später hinzugefügt wird.

  Die Mehrheit der Ameisen benutzen weiter den längeren Weg.
  Nur in seltenen Fällen wird auf den kürzeren Weg gewechselt.

Natürliche und künstliche Ameisen

- Abstrahiere die Situation zu einer Suche nach einem besten Weg in einem gewichteten Graphen.
- **Problem:** Kreise, die sich selbst verstärken
  Durchläuft die Ameise einen Kreis, erzeugt sie durch das abgelegte Pheromon eine Tendenz, den Kreis erneut zu durchlaufen.

  Abhilfe: Ablegen von Pheromon erst nach Konstruktion des ganzen Weges.
  Entfernen von Kreisen, bevor Pheromon abgelegt wird.
- **Problem:** Ggf. konzentriert sich die Suche auf am Anfang konstruierte Lösungskandidaten (vorzeitige Konvergenz).

  Abhilfe: Pheromonverdunstung (spielt in der Natur nur geringe Rolle)
- **Nützliche Erweiterungen/Verbesserungen**
  - Abgelegte Pheromonmenge hängt von der Lösungsgüte ab.
  - Einbringen von Heuristiken in die Kantenwahl (z.B. Kantengewicht).

Ameisenkolonieoptimierung

- **Voraussetzungen**
  - Es handelt sich um ein kombinatorisches Optimierungsproblem.
  - Es gibt eine konstruktive Methode, um Lösungskandidaten zu erzeugen.
- **Vorgehen:** Lösungen werden mit Hilfe einer Folge von Zufallsentscheidungen konstruiert, wobei jede Entscheidung eine Teillösung erweitert.
- Die Entscheidungsfolge kann als Pfad in einem Entscheidungsgraphen (auch: Konstruktionsgraphen) gesehen werden.
- Die Ameisen sollen Pfade durch den Entscheidungsgraphen erkunden und den besten (kürzesten, billigsten) Weg finden.
- Die Ameisen markieren die benutzten Kanten des Graphen mit Pheromon. Dadurch werden andere Ameisen zu guten Lösungen geleitet.
- Pheromon verdunstet in jeder Iteration, damit einmal ausgebrachtes Pheromon das System nicht zu lange beeinflußt („Vergessen“ veralteter Information).

Ameisenkolonieoptimierung

Anwendung auf das Problem des Handlungsreisenden

- Darstellung des Problems durch eine $n \times n$ Matrix $\mathbf{D} = (d_{ij})_{1 \leq i,j \leq n}$.
  n ist die Anzahl der Städte und d_{ij} der Abstand der Städte i und j.
  (Beachte: $\mathbf{D}$ kann asymmetrisch sein, aber $\forall i; 1 \leq i \leq n : d_{ii} = 0$.)
- Darstellung der Pheromoninformation als $n \times n$ Matrix $\mathbf{\Phi} = (\phi_{ij})_{1 \leq i,j \leq n}$.
  Ein Pheromonwert ϕ_{ij}, $i \neq j$, gibt an, wie wünschenswert es ist,
  die Stadt j direkt nach der Stadt i zu besuchen. (ϕ_{ii} wird nicht benötigt.)
  Die Matrix muß nicht notwendig symmetrisch sein/gehalten werden.
- Alle Matrixelemente ϕ_{ij} werden mit dem gleichen kleinen Wert initialisiert.
  (Am Anfang liegt auf allen Kanten die gleiche Menge Pheromon.)
- Ameisen durchlaufen (mit Hilfe des Pheromons) Hamiltonkreise.
  Sie markieren die Kanten des durchlaufenen Hamiltonkreises mit Pheromon,
  wobei die ausgebrachte Pheromonmenge der Qualität der Lösung entspricht.

Lösungskonstruktion

Jede Ameise hat als „Gedächtnis“ eine Menge C, die die Indizes der noch nicht besuchten Städte enthält. Jede besuchte Stadt wird aus dieser Menge entfernt.

(Dieses Gedächtnis gibt es im biologischen Vorbild nicht!)

1. Eine Ameise wird in eine zufällig bestimmte Stadt gesetzt.
   Diese Stadt ist der Anfang der Rundreise.

2. Die Ameise wählt eine noch nicht besuchte Stadt und begibt sich in diese.
   In Stadt i wählt die Ameise die (unbesuchte) Stadt j mit der Wahrscheinlichkeit

$$p_{ij} = \frac{\phi_{ij}}{\sum_{k \in C} \phi_{ik}},$$

   wobei C die Menge der Indizes der noch nicht besuchten Städte ist und ϕ_{ij} die Menge an Pheromon auf der Verbindung von Stadt i nach Stadt j.

3. Wiederhole Schritt 2, bis alle Städte besucht wurden.

Pheromonaktualisierung

1. **Verdunstung/Evaporation**
   Alle Pheromonwerte werden um einen Bruchteil η (evaporation) verringert:

$$\forall i, j; 1 \leq i, j \leq n: \quad \phi_{ij} := (1 - \eta) \cdot \phi_{ij}$$

2. **Verstärkung konstruierter Lösungen**
   Die Kanten der konstruierten Lösungen werden mit einer zusätzlichen Menge an Pheromon belegt, die der Lösungsqualität entspricht.

$$\forall \pi \in \Pi_t: \quad \phi_{\pi(i)\pi((i \bmod n)+1)} := \phi_{\pi(i)\pi((i \bmod n)+1)} + Q(\pi)$$

   Π_t ist die Menge der im Schritt t konstruierten Rundreisen (Permutationen). Als Qualitätsfunktion kann man z.B. die inverse Reiselänge verwenden:

$$Q(\pi) = c \cdot \left(\sum_{i=1}^{n} d_{\pi(i)\pi((i \bmod n)+1)} \right)^{-1}$$

   Anschaulich: Je besser die Lösung, um so mehr Pheromon erhalten die Kanten.

Problem des Handlungsreisenden: Pseudocode

```
procedure aco_tsp;                    (* ant colony optimization for *)
                                      (* the traveling salesman problem *)
initialize pheromone;                 (* initialisiere alle Matrixelemente φ_ij, *)
repeat                                (* 1 ≤ i, j ≤ n, auf einen kleinen Wert ε *)
    for each ant do                   (* konstruiere Kandidatenlösungen *)
        C := {1, ..., n};             (* Menge der zu besuchenden Städte *)
        choose i ∈ C;                 (* wähle die Anfangsstadt und *)
        C := C − {i};                 (* entferne sie aus den unbesuchten Städten *)
        while C ≠ ∅ do begin          (* solange nicht alle Städte besucht wurden *)
            choose j ∈ C with probability p_ij;
            C := C − {i};             (* wähle die nächste Stadt der Reise, *)
            i := j;                   (* entferne sie aus den unbesuchten Städten, *)
        end;                          (* und gehe in die ausgewählte Stadt *)
    endfor;
    update pheromone;                 (* aktualisiere Matrix Φ nach Lösungsgüten *)
until stopping criterion is met;
```

Erweiterungen und Alternativen

- **Bevorzuge nahe Städte** (analog zur Nächster-Nachbar-Heuristik)
  Gehe von Stadt i zu Stadt j mit der Wahrscheinlichkeit

$$p_{ij} = \frac{\phi_{ij}^{\alpha}\, \tau_{ij}^{\beta}}{\sum_{k \in C} \phi_{ik}^{\alpha}\, \tau_{ik}^{\beta}},$$

  wobei C die Menge der Indizes der unbesuchten Städte und $\tau_{ij} = d_{ij}^{-1}$ ist.

- **Tendiere zur Wahl der besten Kante** (greedy)
  Mit Wahrscheinlichkeit p_{exploit} gehe von der Stadt i zur Stadt j_{best} mit

$$j_{\text{best}} = \text{argmax}_{j \in C}\, \phi_{ij} \qquad \text{bzw.} \qquad j_{\text{best}} = \text{argmax}_{j \in C}\, \phi_{ij}^{\alpha}\, \tau_{ij}^{\beta},$$

  und benutze Wahrscheinlichkeiten p_{ij} mit Wahrscheinlichkeit $1 - p_{\text{exploit}}$.

- **Verstärke beste bekannte Rundreise** (elitist)
  Lege zusätzliches Pheromon auf der besten bisher bekannten Rundreise ab.
  Kann z.B. als Bruchteil Ameisen angegeben werden, die sie zusätzlich ablaufen.

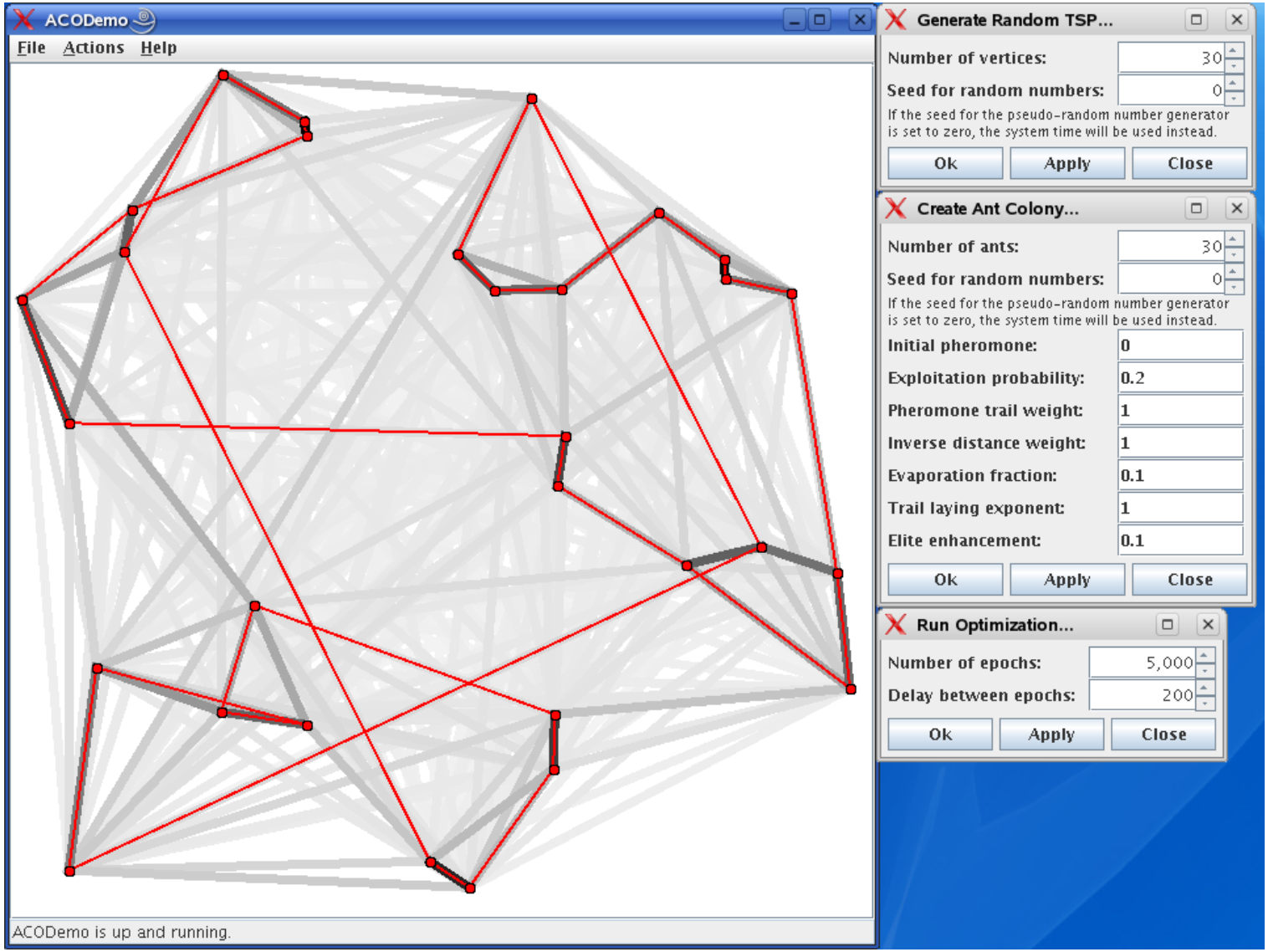
Erweiterungen und Alternativen

- **Rangbasierte Aktualisierung**
  Lege Pheromon nur auf den Kanten der besten m Lösungen der letzten Iteration ab (und eventuell auf der besten bisher gefundenen Lösung).
  Die Pheromonmenge hängt vom Rang der Lösung ab.
- **Strenge Eliteprinzipien**
  - Lege Pheromon nur auf der besten Lösung der letzten Iteration ab.
  - Lege Pheromon nur auf der besten bisher gefundenen Lösung ab.
- **Minimale/maximale Pheromonmenge**
  Begrenze die Pheromonmenge einer Kante nach unten/oben.
  $\Rightarrow$ Mindest-/Maximalwahrscheinlichkeit für die Wahl einer Kante
  $\Rightarrow$ bessere Durchforstung des Suchraums, ggf. schlechtere Konvergenz
- **Eingeschränkte Verdunstung/Evaporation**
  Pheromon verdunstet nur von Kanten, die in der Iteration benutzt wurden.
  $\Rightarrow$ bessere Durchforstung des Suchraums

Lokale Verbesserungen der Rundreise

- Verknüpfung mit **lokaler Lösungsverbesserung** ist oft vorteilhaft:
  Vor der Pheromonaktualisierung wird eine erzeugte Rundreise lokal optimiert, indem einfache Modifikationen auf Verbesserung überprüft werden.
- Lokale Optimierungsansätze können z.B. folgende Operationen benutzen:
  - **Rekombination nach Entfernen von zwei Kanten** (2-opt)
    entspricht dem „Umdrehen“ einer Teil-Rundreisen
  - **Rekombination nach Entfernen von drei Kanten** (3-opt)
    entspricht dem „Umdrehen“ zweier Teil-Rundreisen
  - **Eingeschränkte Rekombination** (2.5-opt)
  - **Austausch benachbarter Städte**
  - **Permutation benachbarter Triplets**
- „Teure“ lokale Optimierungen sollten nur auf die beste gefundene Lösung oder die in einer Iteration beste Lösung angewandt werden.

Ameisenkolonieoptimierung



Allgemeine Anwendung auf Optimierungsprobleme

- **Grundsätzliches Prinzip**
  Formuliere das Problem als Suche in einem (Entscheidungs-)Graphen.
  Lösungskandidaten müssen durch Kantenmengen beschreibbar sein.
  (Beachte: Es muß sich nicht notwendigerweise um Pfade handeln!)
- **Allgemeine Beschreibung**
  Es werden im folgenden jeweils angegeben:
  - Knoten und Kanten des Entscheidungs-/Konstruktionsgraphen
  - Einzuhaltende Nebenbedingungen
  - Bedeutung des Pheromons auf den Kanten (und evtl. Knoten)
  - Nutzbare heuristische Hilfsinformation
  - Konstruktion eines Lösungskandidaten
- Das algorithmische Vorgehen ist im wesentlichen analog
  zum Vorgehen beim das Problem des Handlungsreisenden.

Allgemeine Anwendung auf Optimierungsprobleme

Problem des Handlungsreisenden

- *Knoten und Kanten des Entscheidungs-/Konstruktionsgraphen:*
  die zu besuchenden Städte und ihre Verbindungen,
  die Verbindungen sind gewichtet (Abstand, Zeit, Kosten)
- *Einzuhaltende Nebenbedingungen:*
  jede Stadt muß genau einmal besucht werden
- *Bedeutung des Pheromons auf den Kanten:*
  wie wünschenwert es ist, Stadt j nach Stadt i zu besuchen
- *Nutzbare heuristische Hilfsinformation:*
  Abstand der Städte; bevorzuge nahe Städte
- *Konstruktion eines Lösungskandidaten:*
  ausgehend von einer zufällig gewählten Stadt wird stets zu einer weiteren, noch nicht besuchten Stadt fortgeschritten

Allgemeine Anwendung auf Optimierungsprobleme

Verallgemeinertes Zuordnungsproblem

n Aufgaben müssen an m Arbeiter (Personen, Maschinen) zugeordnet werden. Minimierung der Summe der Zuordnungskosten d_{ij} unter Einhaltung maximaler Kapazitäten ρ_j bei gegebenen Kapazitätskosten r_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$.

- Jede Aufgabe und jeder Arbeiter ist ein Knoten des Konstruktionsgraphen, die Kanten tragen die Zuordnungskosten d_{ij}.
- Jede Aufgabe muß genau einem Arbeiter zugeordnet werden.
  Die Kapazitäten der Arbeiter dürfen nicht überschritten werden.
- Die Pheromonwerte auf den Kanten beschreiben,
  wie wünschenswert die Zuordnung einer Aufgabe an einen Arbeiter ist.
- Inverse absolute oder relative Kapazitätskosten oder inverse Zuordnungskosten.
- Es werden schrittweise Kanten ausgewählt (müssen keinen Pfad bilden),
  wobei Kanten von bereits zugeordneten Aufgaben übergangen werden.
  Nebenbedingungen verletztende Lösungen werden bestraft (Kostenerhöhung).

Allgemeine Anwendung auf Optimierungsprobleme

Rucksackproblem

Aus n Objekten mit zugeordnetem Wert w_i, Gewicht g_i, Volumen v_i etc., $1 \leq i \leq n$, soll eine Teilmenge maximalen Wertes ausgewählt werden, so daß Maximalwerte für das Gewicht, das Volumen etc. eingehalten werden.

- Jedes Objekt ist ein Knoten des Konstruktionsgraphen.
  Die Knoten tragen die Objektwerte w_i. Kanten werden nicht benötigt.
- Die Maximalwerte für Gewicht, Volumen etc. müssen eingehalten werden.
- Pheromonwerte sind nur den Knoten zugeordnet. Sie beschreiben,
  wie wünschenswert die Auswahl des zugehörigen Objektes ist.
- Verhältnis von Objektwert zu relativem Gewicht, Volumen etc.,
  wobei in den Verhältnissen die Maximalwerte berücksichtigt werden können.
- Es werden schrittweise Knoten ausgewählt, wobei in jedem Schritt sichergestellt wird, daß die Maximalwerte eingehalten werden.

Konvergenz der Suche

- Betrachte „Standardverfahren“ mit den folgenden Eigenschaften:
  - Verdunstung des Pheromons mit konstantem Faktor von allen Kanten.
  - Nur auf den Kanten des besten, bisher gefundenen Lösungskandidaten wird Pheromon abgelegt (strenges Eliteprinzip).
  - Es gibt eine Untergrenze $\phi_{\min}$ für die Pheromonwerte der Kanten, die nicht unterschritten werden darf.

- Dieses Standardverfahren konvergiert in Wahrscheinlichkeit gegen die Lösung. (D.h., mit gegen unendlich gehender Zahl der berechneten Schritte geht die Wahrscheinlichkeit, daß die Lösung gefunden wird, gegen 1.)

- Läßt man die Untergrenze $\phi_{\min}$ für die Pheromonwerte „genügend langsam“ gegen 0 gehen ($\phi_{\min} = \frac{c}{\ln(t+1)}$ mit der Schrittzahl t und einer Konstante c), kann man sogar zeigen, daß für gegen unendlich gehende Schrittzahl jede Ameise der Kolonie die Lösung mit gegen 1 gehender Wahrscheinlichkeit konstruiert.

Zusammenfassung

- Schwarm- und populationsbasierte Algorithmen sind **Heuristiken zur Lösung von Optimierungsproblemen**.
  Es geht darum, gute Näherungslösungen zu finden.
- Man versucht das **Problem der lokalen Optima** zu verringern
  (durch bessere Durchforstung/Exploration des Suchraums).
- Wichtig ist der **Informationsaustausch** zwischen den Individuen.
  Je nach Prinzip können verschiedene Algorithmentypen unterschieden werden.
- **Teilchenschwarmoptimierung**
  - Optimierung einer Funktion mit reellen Argumenten.
  - Informationsaustausch durch Orientierung an Nachbarn.
- **Ameisenkolonieoptimierung**
  - Suche nach besten Wegen (abstrakt: in einem Entscheidungsgraphen).
  - Informationsaustausch durch Veränderung der Umgebung (Stigmergie).

Demonstrationsprogramme

Die vorgeführten Demonstrationsprogramme sind verfügbar unter den URLs:

- **Teilchenschwarmoptimierung** (particle swarm optimization)

  `http://fuzzy.cs.uni-magdeburg.de/~borgelt/psopt.html`

- **Ameisenkolonieoptimierung** (ant colony optimization)

  `http://fuzzy.cs.uni-magdeburg.de/~borgelt/acopt.html`