



OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

INF

FAKULTÄT FÜR
INFORMATIK

Evolutionäre Algorithmen

Kapitel 5: Metaheuristik 2

Sanaz Mostaghim

Lehrstuhl für Intelligente Systeme

SS 2016

Outline

Schwarm- und populationsbasierte Optimierung

Populationsbasiertes inkrementelles Lernen

Teilchenschwarmoptimierung

Ameisenkolonieoptimierung

Organic Computing

Zusammenfassung

Schwarm- und populationsbasierte Optimierung

Schwarm-Intelligenz

- ▶ Bereich der KI, der intelligente Multi-Agentensysteme entwickelt
- ▶ 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 dieser Arten können recht komplexe Aufgaben lösen (Finden von Nahrungsquellen, Wegesuche, Nestbau, etc.), indem sie kooperieren.

Wesentliche Ideen

- ▶ i.A. ziemlich einfache Einzelindividuen mit begrenzten Fähigkeiten
- ▶ Koordination ohne zentrale Steuerung, lediglich Selbstorganisation
- ▶ Austausch von Informationen zwischen Individuen (Kooperation)

Klassifizierung der Verfahren nach Art des Informationsaustausches

Verfahren

Genetische/Evolutionäre Algorithmen

- ▶ biologisches Vorbild: Evolution der Lebewesen
- ▶ Informationsaustausch durch Rekombination der Genotypen
- ▶ jedes Individuum ist ein Lösungskandidat

Populationsbasiertes inkrementelles Lernen

- ▶ biologisches Vorbild: Evolution der Lebewesen
- ▶ Informationsaustausch durch Häufigkeiten in der Population
- ▶ jedes Individuum ist ein Lösungskandidat

Verfahren

Teilenschwarmoptimierung

- ▶ 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

Populationsbasiertes inkrementelles Lernen

(PBIL)

- ▶ genetischer Algorithmus ohne Population
- ▶ stattdessen: nur Populationsstatistik speichern \Rightarrow bei $\mathcal{G} = \{0, 1\}^L$ für alle L Bits die Häufigkeit der „1“
- ▶ konkrete Individuen (z.B. für Bewertung) werden zufällig gemäß der Häufigkeiten erzeugt
- ▶ Rekombination: uniformer Crossover \Rightarrow implizit bei Erzeugung eines Individuums
- ▶ Selektion: Wahl des besten Individuums B für Aktualisierung der Populationsstatistik $Pr_k^{(t)} \leftarrow B_k \cdot \alpha + Pr_k^{(t-1)}(1 - \alpha)$
- ▶ Mutation: Bit-Flipping \Rightarrow leichte zufällige Verschiebung der Populationsstatistik

Algorithm 1 PBIL

Input: Bewertungsfunktion F

Output: bestes Individuum A_{best}

```
1:  $t \leftarrow 0$ 
2:  $A_{\text{best}} \leftarrow$  erzeuge zufälliges Individuum aus  $\mathcal{G} = \{0, 1\}^L$ 
3:  $Pr^{(t)} \leftarrow (0.5, \dots, 0.5) \in [0, 1]^L$ 
4: while Terminierungsbedingung nicht erfüllt {
5:    $P \leftarrow \emptyset$ 
6:   for  $i \leftarrow 1, \dots, \lambda$  {
7:      $A \leftarrow$  erzeuge Individuum aus  $\{0, 1\}^L$  gemäß  $Pr^{(t)}$ 
8:      $P \leftarrow P \cup \{A\}$ 
9:   }
10:  bewerte  $P$  durch  $F$ 
11:   $B \leftarrow$  selektiere die besten Individuen aus  $P$ 
12:  if  $F(B) \succ F(A_{\text{best}})$  {
13:     $A_{\text{best}} \leftarrow B$ 
14:  }
15:   $t \leftarrow t + 1$ 
16:  for each  $k \in \{1, \dots, L\}$  {
17:     $Pr_k^{(t)} \leftarrow B_k \cdot \alpha + Pr_k^{(t-1)}(1 - \alpha)$ 
18:  }
19:  for each  $k \in \{1, \dots, L\}$  {
20:     $u \leftarrow$  wähle Zufallszahl gemäß  $U((0, 1))$ 
21:    if  $u < p_m$  {
22:       $u' \leftarrow$  wähle Zufallszahl gemäß  $U(\{0, 1\})$ 
23:       $Pr_k^{(t)} \leftarrow u' \cdot \beta + Pr_k^{(t)}(1 - \beta)$ 
24:    }
25:  }
26: }
27: return  $A_{\text{best}}$ 
```

PBIL: Typische Parameter

Lernrate α

- ▶ niedrig: betont Erforschung
- ▶ hoch: betont Feinabstimmung

Parameter	Wertebereich
Populationsgröße λ	20–100
Lernrate α	0.05–0.2
Mutationsrate p_m	0.001–0.02
Mutationskonstante β	0.05

PBIL: Probleme

- ▶ GA kann Abhängigkeit zwischen einzelnen Bits lernen
- ▶ PBIL betrachtet einzelnen Bits isoliert

Beispiel:

Population 1					Population 2			
1	1	0	0	Individuum 1	1	0	1	0
1	1	0	0	Individuum 2	0	1	1	0
0	0	1	1	Individuum 3	0	1	0	1
0	0	1	1	Individuum 4	1	0	0	1
0.5	0.5	0.5	0.5	Populationsstatistik	0.5	0.5	0.5	0.5

- ▶ gleiche Populationsstatistik kann unterschiedliche Populationen repräsentieren

PBIL: Alternative Verfahren

- ▶ bessere Techniken zur Schätzung der Verteilung guter Lösungskandidaten
- ▶ insbesondere: interne Abhängigkeiten modellieren (z.B. durch Bayes'sche Netze)
- ▶ Beispiel: *Bayes'scher Optimierungsalgorithmus (BOA)*
 - ▶ initiale Population wird zufällig erzeugt
 - ▶ Population wird für bestimmte Anzahl von Iterationen aktualisiert mittels Selektion und Variation
 - ▶ Selektion wie gehabt
 - ▶ Variation konstruiert nach Selektion ein Bayes'sche Netz als Modell von vielversprechend Lösungskandidaten
 - ▶ neue Lösungskandidaten werden dann durch Stichprobe vom Bayes'sche Netz generiert

Teilenschwarmoptimierung



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



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

- ▶ Fische, Vögel suchen in Schwärmen nach ergiebigen Futterplätzen
- ▶ Orientierung anhand individueller Suche (kognitiver Anteil) und an anderen Mitgliedern des Schwarmes in ihrer Nähe (sozialer Anteil)
- ▶ außerdem: Leben im Schwarm schützt Individuen gegen Fressfeinde

Teilenschwarmoptimierung

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 Mitgliedern
 - ▶ **Ansatz:** verwende statt nur einzelnen aktuellen Lösungskandidaten „Schwarm“ von m Lösungskandidaten
 - ▶ **Voraussetzung:** $\Omega \subseteq \mathbb{R}^n$ und somit zu optimierende (o.B.d.A.: zu maximierende) Funktion $f : \mathbb{R}^n \rightarrow \mathbb{R}$
 - ▶ **Vorgehen:** fasse jeden Lösungskandidaten als „Teilchen“ auf, das Ort \mathbf{x}_i im Suchraum und Geschwindigkeit \mathbf{v}_i hat ($i = 1, \dots, m$)
- ⇒ vereinigt Elemente der bahnorientierten Suche (z.B. Gradientenverfahren) und populationsbasierter Suche (z.B. EA)

Teilenschwarmoptimierung

Aktualisierung für Ort und Geschwindigkeit des i -ten Teilchens:

$$\mathbf{v}_i(t+1) = \alpha \mathbf{v}_i(t) + \beta_1 \left(\mathbf{x}_i^{(\text{lokal})}(t) - \mathbf{x}_i(t) \right) + \beta_2 \left(\mathbf{x}^{(\text{global})}(t) - \mathbf{x}_i(t) \right)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t)$$

- ▶ Parameter: β_1, β_2 zufällig in jedem Schritt, α mit t abnehmend
- ▶ $\mathbf{x}_i^{(\text{lokal})}$ ist **lokales Gedächtnis** des Individuums (Teilchens): der beste Ort im Suchraum, den Teilchen bisher besucht hat, d.h.

$$\mathbf{x}_i^{(\text{lokal})} = \mathbf{x}_i \left(\arg \max_{u=1}^t f(\mathbf{x}_i(u)) \right)$$

- ▶ $\mathbf{x}^{(\text{global})}$ ist **globales Gedächtnis** des Schwarms: der beste Ort im Suchraum, den Individuum des Schwarms bisher besucht hat (beste bisher gefundene Lösung), d.h.

$$\mathbf{x}^{(\text{global})}(t) = \mathbf{x}_j^{(\text{lokal})}(t) \quad \text{mit} \quad j = \arg \max_{i=1}^m f \left(\mathbf{x}_i^{(\text{lokal})} \right)$$

Algorithm 2 Teilenschwarmoptimierung

```
1: for each Teilchen  $i$  {
2:    $\mathbf{x}_i \leftarrow$  wähle zufällig im Suchraum  $\Omega$ 
3:    $\mathbf{v}_i \leftarrow 0$ 
4: }
5: do {
6:   for each Teilchen  $i$  {
7:      $y \leftarrow f(\mathbf{x}_i)$ 
8:     if  $y \geq f(\mathbf{x}_i^{(\text{lokal})})$  {
9:        $\mathbf{x}_i^{(\text{lokal})} \leftarrow \mathbf{x}_i$ 
10:    }
11:    if  $y \geq f(\mathbf{x}_i^{(\text{global})})$  {
12:       $\mathbf{x}_i^{(\text{global})} \leftarrow \mathbf{x}_i$ 
13:    }
14:  }
15:  for each Teilchen  $i$  {
16:     $\mathbf{v}_i(t+1) \leftarrow \alpha \cdot \mathbf{v}_i(t) + \beta_1 (\mathbf{x}_i^{(\text{lokal})}(t) - \mathbf{x}_i(t)) + \beta_2 (\mathbf{x}_i^{(\text{global})}(t) - \mathbf{x}_i(t))$ 
17:     $\mathbf{x}_i(t+1) \leftarrow \mathbf{x}_i(t) + \mathbf{v}_i(t)$ 
18:  }
19: } while Terminierungskriterium ist nicht erfüllt
```

Erweiterungen

- ▶ **Beschränkter Suchraum:** ist Ω echte Teilmenge des \mathbb{R}^n (z.B. Hyperwürfel $[a, b]^n$), so werden Teilchen an Grenzen des Suchraums reflektiert
- ▶ **Lokale Umgebung eines Teilchens:** statt des globalen Gedächtnisses des Schwarms wird bestes lokales Gedächtnis nur eines Teils des Schwarms verwendet, z.B. Teilchen, die sich in näherer 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 suboptimalen Lösungen soll verhindert werden, dazu kann z.B. bei Aktualisierung der Geschwindigkeit zusätzliche Zufallskomponente eingeführt werden, welche Diversität erhöht

Ameisenkolonieoptimierung



© PeIA <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 Wege zu Futterplätzen mit Duftstoffen (Pheromonen), sodass andere Ameisen der Kolonie diese Futterplätze auch finden können
- ▶ Weglängen zu Futterplätzen werden annähernd minimiert

Ameisenkolonieoptimierung

Ant Colony Optimization [Dorigo and Stützle, 2004]

Motivation: Ameisen einiger Arten finden kürzeste Wege zu Futterquellen durch Legen und Verfolgen von Pheromon („Duftmarken“)

- ▶ intuitiv: kürzere Wege erhalten in gleicher Zeit mehr Pheromon
- ▶ Wege werden zufällig nach vorhandener Pheromonmenge gewählt (es ist um so wahrscheinlicher, dass ein Weg gewählt wird, je mehr Pheromon sich auf dem Weg befindet)
- ▶ Menge des ausgebrachten Pheromons kann von Qualität und Menge des gefundenen Futters abhängen

Grundprinzip: Stigmergie (engl. stigmergy)

- ▶ Ameisen kommunizieren indirekt über Pheromonablagerungen
- ▶ Stigmergie (indirekte Kommunikation durch Veränderung der Umgebung) ermöglicht global angepasstes Verhalten aufgrund lokaler Informationen

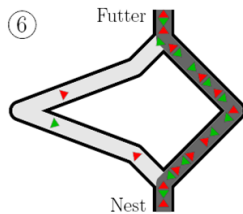
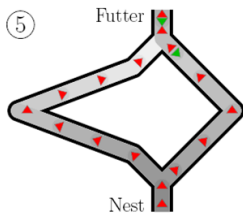
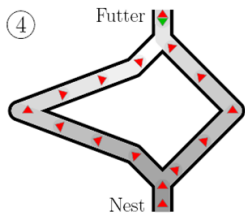
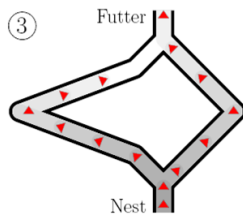
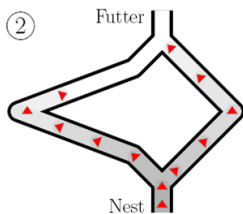
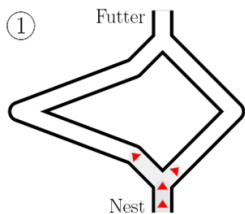
Doppelbrückenexperiment [Goss et al., 1989]

- ▶ Ameisennest und Futterquelle werden durch Doppelbrücke verbunden (beiden Zweige der Brücke sind verschieden lang)
- ▶ Experiment mit argentinischer Ameise *Iridomyrmex Humilis*: diese Ameisenart ist (wie fast alle anderen auch) fast blind (Ameisen können also nicht sehen, welcher Weg kürzer ist)
- ▶ schon nach wenigen Minuten: in meisten Versuchen benutzten fast alle Ameisen kürzeren Weg

Erklärung

- ▶ Auf kürzerem Weg erreichen Ameisen Futter schneller (Ende des kürzeren Weges erhält daher (am Anfang) mehr Pheromon)
- ▶ Auf Rückweg wird wegen entstandener Pheromondifferenz mit höherer W'keit kürzerer Weg gewählt (führt zu einer Verstärkung der Pheromondifferenz)

Doppelbrückenexperiment



Doppelbrückenexperiment: Prinzip

- ▶ kürzerer Weg wird systematisch verstärkt (Autokatalyse):
mehr Pheromon auf Weg \longleftrightarrow mehr Ameisen wählen Weg
- ▶ beachte: kürzerer Weg wird nur gefunden, weil Ameisen
sowohl auf Hin- als auch auf Rückweg Pheromon ablegen
- ▶ wird z.B. nur auf Hinweg Pheromon abgelegt:
 - ▶ auf Hinweg zur Futterquelle kann keiner der beiden Wege
bevorzugt werden, da keine Pheromondifferenz vorliegt oder
systematisch entsteht
 - ▶ am Vereinigungspunkt der Brücken verringert sich Verhältnis
der Pheromonmengen im Laufe der Zeit und verschwindet
schließlich nahezu
 - ▶ durch zufällige Fluktuationen in Wegewahl konvergiert
Wegesuche ggf. dennoch zufällig(!) auf einen der beiden
Brückenzweige
- ▶ analog (symmetrische Situation), wenn Pheromon nur auf
Rückweg abgelegt

Doppelbrückenexperiment

- ▶ **beachte:** kürzerer Weg wird gefunden, weil schon zu Beginn beide Zweige der Brücke zur Verfügung stehen und auf beiden kein Pheromon liegt
 - ▶ Ende des kürzeren Weges wird früher von mehr Ameisen erreicht
- ⇒ unterschiedliche Mengen an Pheromon auf beiden Wegen
- ⇒ sich selbst verstärkender Prozess
- ▶ **Fragen:** Was passiert, wenn durch Veränderung der Umgebung neuer Weg möglich wird, der kürzer ist als bisheriger?
 - ▶ Wird auf diesen kürzeren Weg gewechselt?

Antwort: Nein! [Goss et al., 1989]

- ▶ ist erst einmal ein Weg etabliert, so wird dieser beibehalten
- ▶ Nachweis durch 2. Brückenexperiment: anfangs nur längerer Brückenast da, kürzerer später hinzugefügt
- ▶ Mehrheit der Ameisen benutzen weiter längeren Weg, nur seltenen Wechsel auf kürzeren Weg

Natürliche und künstliche Ameisen

abstrahiere zu Suche nach bestem Weg in gewichtetem Graphen

- ▶ **Problem:** Kreise, die sich selbst verstärken (durchläuft Ameise Kreis, erzeugt sie durch abgelegtes Pheromon Tendenz, 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 Suche auf am Anfang konstruierte Lösungskandidaten (vorzeitige Konvergenz)
- ▶ **Abhilfe:** Pheromonverdunstung (spielt in Natur geringe Rolle)

Nützliche Erweiterungen/Verbesserungen

- ▶ abgelegte Pheromonmenge hängt von Lösungsgüte ab
- ▶ Einbringen von Heuristiken in Kantenwahl (z.B. Kantengewicht)

Ameisenkolonieoptimierung

- ▶ **Voraussetzungen:** kombinatorisches Optimierungsproblem mit konstruktiver Methode, zur Erzeugung eines Lösungskandidaten
- ▶ **Vorgehen:** Lösungen werden durch Folge von Zufallsentscheidungen konstruiert, wobei jede Entscheidung Teillösung erweitert
- ▶ Entscheidungsfolge = Pfad in Entscheidungsgraphen (auch: Konstruktionsgraphen)
- ▶ Ameisen sollen Pfade durch Entscheidungsgraphen erkunden und besten (kürzesten, billigsten) Weg finden
- ▶ Ameisen markieren benutzte Kanten des Graphen mit Pheromon \Rightarrow andere Ameisen werden zu guten Lösungen geleitet
- ▶ Pheromon verdunstet in jeder Iteration, damit einmal ausgebrachtes Pheromon System nicht zu lange beeinflusst („Vergessen“ veralteter Information)

Anwendung auf TSP

- ▶ Darstellung des Problems durch $n \times n$ Matrix $\mathbf{D} = (d_{ij})_{1 \leq i, j \leq n}$
- ▶ n Städte mit Abständen d_{ij} zwischen Städte i und j
- ▶ beachte: \mathbf{D} kann asymmetrisch sein, aber
 $\forall i \in \{1, \dots, n\} : d_{ii} = 0$
- ▶ Pheromoninformation als $n \times n$ Matrix $\Phi = (\phi_{ij})_{1 \leq i, j \leq n}$
- ▶ Pheromonwert $\phi_{ij} (i \neq j)$ gibt an, wie wünschenswert es ist, Stadt j direkt nach Stadt i zu besuchen (ϕ_{ii} nicht benötigt)
- ▶ Φ muss nicht notwendig symmetrisch sein/gehalten werden
- ▶ alle ϕ_{ij} werden mit gleichen kleinen Wert initialisiert (anfangs liegt auf allen Kanten gleiche Menge Pheromon)
- ▶ Ameisen durchlaufen (durch Pheromon) Hamiltonkreise (sie markieren Kanten des durchlaufenden Hamiltonkreises mit Pheromon, wobei ausgebrachte Pheromonmenge der Lösungsqualität entspricht)

Lösungskonstruktion

- ▶ jede Ameise hat „Gedächtnis“ C , welche Indizes der noch nicht besuchten Städte enthält
 - ▶ jede besuchte Stadt wird aus Menge C entfernt
 - ▶ Gedächtnis gibt es im biologischen Vorbild nicht!
1. Ameise wird in zufällig bestimmter Stadt gesetzt (Anfang der Rundreise)
 2. Ameise wählt noch nicht besuchte Stadt und begibt sich in diese: in Stadt i wählt Ameise (unbesuchte) Stadt j mit W 'keit

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

3. wiederhole Schritt 2 bis alle Städte besucht

Pheromonaktualisierung

1. Verdunstung/Evaporation

alle ϕ_{ij} werden um Bruchteil η (evaporation) verringert:

$$\forall i, j \in \{1, \dots, n\} : \phi_{ij} = (1 - \eta) \cdot \phi_{ij}$$

2. Verstärkung konstruierter Lösungen

Kanten der konstruierten Lösungen werden mit zusätzlicher Menge an Pheromon belegt, die Lösungsqualität entspricht:

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

Π_t ist Menge der im Schritt t konstruierten Rundreisen (Permutationen), Qualitätsfunktion: z.B. inverse Reiselänge

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

„Je besser die Lösung, desto mehr Pheromon erhalten deren Kanten.“

Problem des Handlungsreisenden

Algorithm 3 Ameisenkolonieoptimierung für das TSP

```
1: initialisiere alle Matrixelemente  $\phi_{ij}$ ,  $1 \leq i, j \leq n$ , auf kleinen Wert  $\epsilon$ 
2: do {
3:   for each Ameise {                               /* konstruiere Lösungskandidaten */
4:      $C \leftarrow \{1, \dots, n\}$                    /* Menge der zu besuchenden Städte */
5:      $i \leftarrow$  wähle zufällig Anfangsstadt aus  $C$ 
6:      $C \leftarrow C \setminus \{i\}$                  /* entferne sie aus den unbesuchten Städten */
7:     while  $C \neq \emptyset$  {                       /* solange nicht alle Städte besucht wurden */
8:        $j \leftarrow$  wähle nächste Stadt der Reise aus  $C$  mit  $W$ 'keit  $p_{ij}$ 
9:        $C \leftarrow C \setminus \{j\}$                /* entferne sie aus den unbesuchten Städten */
10:       $i \leftarrow j$                                /* und gehe in die ausgewählte Stadt */
11:    }
12:  }
13:  aktualisiere Pheromon-Matrix  $\Phi$  nach Lösungsgüte
14: } while Terminierungskriterium ist nicht erfüllt
```

Erweiterungen und Alternativen

- ▶ **Bevorzuge nahe Städte:** (analog zur NächstenNachbarHeuristik)
gehe von Stadt i zu Stadt j mit W' keit

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

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

- ▶ **Tendiere zur Wahl der besten Kante:** (greedy)
mit W' keit p_{exploit} gehe von Stadt i zur Stadt j_{best} mit

$$j_{\text{best}} = \arg \max_{j \in C} \phi_{ij} \quad \text{bzw.} \quad j_{\text{best}} = \arg \max_{j \in C} \phi_{ij}^{\alpha} \tau_{ij}^{\beta}$$

und benutze p_{ij} mit W' keit $1 - p_{\text{exploit}}$

- ▶ **Verstärke beste bekannte Rundreise:** (elitist)
Lege zusätzliches Pheromon auf besten bisher bekannten Rundreise ab (z.B. Bruchteil Ameisen, die sie zusätzlich ablaufen)

Erweiterungen und Alternativen

Rangbasierte Aktualisierung

- ▶ lege Pheromon nur auf Kanten der besten m Lösungen der letzten Iteration ab (und ev. auf besten bisher gefundenen Lösung)
- ▶ Pheromonmenge hängt vom Rang der Lösung ab

Strenge Eliteprinzipien

- ▶ lege Pheromon nur auf besten Lösung der letzten Iteration ab
- ▶ lege Pheromon nur auf besten bisher gefundenen Lösung ab

Erweiterungen und Alternativen

Minimale/maximale Pheromonmenge

- ▶ begrenze Pheromonmenge einer Kante nach unten/oben
- ⇒ Mindest-/Maximalwertigkeit für Wahl einer Kante
- ⇒ bessere Durchforstung des Suchraums, ggf. schlechtere Konvergenz

Eingeschränkte Verdunstung/Evaporation

- ▶ Pheromon verdunstet nur von Kanten, die in Iteration benutzt wurden
- ⇒ bessere Durchforstung des Suchraums

Lokale Verbesserungen der Rundreise

- ▶ Verknüpfung mit lokaler Lösungsverbesserung ist oft vorteilhaft: Vor Pheromonaktualisierung wird erzeugte Rundreise lokal optimiert (einfache Modifikationen werden auf Verbesserung überprüft)
- ▶ lokale Optimierungen benutzen z.B. folgende Operationen:
 - ▶ **Rekombination nach Entfernen von zwei Kanten** (2-opt)
entspricht dem „Umdrehen“ einer Teil-Rundreise
 - ▶ **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: nur auf beste gefundene Lösung oder die in einer Iteration beste Lösung angewandt

Allg. Anwendung auf Optimierungsprobleme

▶ Grundsätzliches Prinzip

formuliere Problem als Suche in (Entscheidungs-)Graphen, Lösungskandidaten müssen durch Kantenmengen beschreibbar sein, (beachte: es muss sich nicht notwendigerweise um Pfade handeln!)

▶ **Allgemeine Beschreibung:** im folgenden werden 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

▶ algorithmisches Vorgehen ist i.W. analog zu Vorgehen beim TSP

Allg. Anwendung auf Optimierungsprobleme: TSP

- ▶ *Knoten und Kanten des Entscheidungs-/Konstruktionsgraphen*: die zu besuchenden Städte und ihre Verbindungen, die Verbindungen sind gewichtet (Abstand, Zeit, Kosten)
- ▶ *einzuhaltende Nebenbedingungen*: besuche jede Stadt genau 1x
- ▶ *Bedeutung des Pheromons auf den Kanten*: wie wünschenswert ist es, Stadt j nach Stadt i zu besuchen
- ▶ *nutzbare heuristische Hilfsinformation*: Abstand der Städte, bevorzuge nahe Städte
- ▶ *Konstruktion eines Lösungskandidaten*: ausgehend von zufällig gewählter Stadt wird stets zu einer weiteren, noch nicht besuchten Stadt fortgeschritten

Allg. Anwendung auf Optimierungsprobleme

Verallgemeinertes Zuordnungsproblem

Ordne n Aufgaben zu m Arbeiter (Personen, Maschinen):
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 = Knoten des Konstruktionsgraphen (Kanten tragen die Zuordnungskosten d_{ij})
- ▶ jede Aufgaben muss genau einem Arbeiter zugeordnet werden (Kapazitäten der Arbeiter nicht überschreiten)
- ▶ Pheromonwerte auf Kanten beschreiben, wie wünschenswert Zuordnung einer Aufgabe an Arbeiter ist
- ▶ inverse absolute oder relative r_{ij} oder inverse d_{ij}
- ▶ wähle schrittweise Kanten (müssen keinen Pfad bilden), übergehe Kanten von bereits zugeordneten Aufgaben (bestrafe Lösungen, die Nebenbedingungen verletzen (Kostenerhöhung))

Allg. Anwendung auf Optimierungsprobleme

Rucksackproblem

Wähle aus n Objekten mit zugeordnetem Wert w_i , Gewicht g_i , Volumen v_i , etc. $1 \leq i \leq n$, Teilmenge maximalen Wertes aus, sodass Maximalwerte für Gewicht, Volumen, etc. eingehalten

- ▶ jedes Objekt = Knoten des Konstruktionsgraphen (Knoten tragen Objektwerte w_i , Kanten nicht benötigt)
- ▶ Maximalwerte für Gewicht, Volumen, etc. müssen eingehalten werden
- ▶ Pheromonwerte: nur Knoten zugeordnet (sie beschreiben, wie wünschenswert Auswahl des zugehörigen Objektes)
- ▶ Verhältnis von Objektwert zu relativem Gewicht, Volumen, etc. wobei in den Verhältnissen die Maximalwerte berücksichtigt werden können
- ▶ wähle schrittweise Knoten aus, wobei in jedem Schritt sichergestellt, dass Maximalwerte eingehalten

Konvergenz der Suche

betrachte „Standardverfahren“ mit folgenden Eigenschaften:

- ▶ Verdunstung des Pheromons mit konstantem Faktor von allen Kanten
- ▶ nur auf Kanten des besten, bisher gefundenen Lösungskandidaten wird Pheromon abgelegt (strenges Eliteprinzip)
- ▶ \exists Untergrenze ϕ_{\min} für Pheromonwerte der Kanten, welche nicht unterschritten wird
- ▶ Standardverfahren konvergiert in W' keit gegen Lösung, d.h. mit $t \rightarrow \infty$ geht W' keit, dass Lösung gefunden wird, gegen 1
- ▶ lässt man Untergrenze ϕ_{\min} für Pheromonwerte „genügend langsam“ gegen 0 gehen ($\phi_{\min} = \frac{c}{\ln(t+1)}$ mit Schrittzahl t und Konstante c), kann man zeigen, dass für $t \rightarrow \infty$ jede Ameise der Kolonie Lösung mit gegen 1 gehender W' keit konstruiert

Zusammenfassung

- ▶ Schwarm- und populationsbasierte Algorithmen: **Heuristiken zur Lösung von Optimierungsproblemen**
- ▶ Ziel: Finden guter Näherungslösungen
- ▶ man versucht **Problem der lokalen Optima** zu verringern (durch bessere Durchforstung/Exploration des Suchraums)
- ▶ wichtig: **Informationsaustausch** zwischen Individuen (je nach Prinzip: verschiedene Algorithmentypen)
- ▶ **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)

Outline

Schwarm- und populationsbasierte Optimierung

Organic Computing

Zusammenfassung

Motivation

In Zukunft werden voneinander unabhängige Systeme

- ▶ miteinander kommunizieren können,
- ▶ sich automatisch an ihre Umgebung anpassen müssen,
- ▶ in vielen Bereichen der Gesellschaft zum Einsatz kommen.

Beispiele solcher Systeme:

- ▶ Verwalten von Peer-to-Peer-Netzen
- ▶ Lernfähige Verkehrssteuerung
- ▶ Auskundschaften von Gebieten durch Roboter
- ▶ Automatisierung von Fabrikabläufen
- ▶ Management von regenerativen Energiequellen
- ▶ Selbstheilung von Systemfehlern in Automobilen

Organic Computing

- ▶ Ziel: eigenständige Organisation, Konfiguration und Reparatur dieser komplizierten Rechnersysteme
- ▶ Lösung: *Organische Computersysteme* (engl. Organic Computing)
 - ▶ passen sich an veränderte Bedingungen (Umwelteinflüsse) an,
 - ▶ sind von Vorbildern der Natur inspiriert

Probleme:

- ▶ Steuerung und Kontrolle dieser Systeme wird schwieriger, da sie *emergente Verhaltensweisen* entwickeln, d.h. vorher nicht vorhandenes Verhalten kann jederzeit auftreten
- ▶ Emergenzen können positiv sein, falls System richtig auf unerwartete Situation reagiert. Sie können aber auch fatal sein. Beispielsweise lernende Verkehrssteuerung: alle Ampeln auf grün geschaltet, weil alle Fahrzeuge dadurch freie Fahrt bekämen und Staus reduziert würden

Organic Computing: Mehr als nur Optimierung

- ▶ Ameisen übernehmen im Staat Rollen wie Soldat oder Arbeiter
 - ▶ sinkt z.B. Zahl der Soldaten unter bestimmte Schwelle, werden aus Arbeitern Soldaten
- ⇒ Sensoren könnten neue Aufgaben übernehmen wenn andere ausfallen
- ▶ Systeme könnten Aufgaben, die sie bereits mehrmals gelöst haben, bei jedem Mal effizienter und effektiver lösen
-
- ▶ seit 2004 existiert ein von DFG geförderter Bereich zum Thema Organic Computing
 - ▶ Literatur: [Würtz, 2008] auch online unter <http://www.springerlink.com/content/978-3-540-77656-7>

Outline

Schwarm- und populationsbasierte Optimierung

Organic Computing

Zusammenfassung

Genetischer bzw. Evolutionärer Algorithmus

Repräsentation: (klassisch) $\{0, 1\}^L$ mit fester Länge L (auch \mathbb{R}^L und \mathcal{S}_n , Dekodierung)

Mutation: Bitflipping, gleichverteilte reellwertige Mutation, spezielle Permutationsoperatoren

Rekombination: k -Punkt- und uniformer Crossover, arithmetischer Crossover, Ordnungsrekombination

Selektion: Elternselektion, fitnessproportional oder Turnirselektion

Population: mittelgroße Populationen

Besonderheiten: theoretische Grundlage durch Schema-Theorem
(*wird noch behandelt in der Vorlesung*)

Lokale Suche

Repräsentation: beliebig

Mutation: beliebig

Rekombination: keine

Selektion: Verbesserungen immer, Verschlechterungen mit gewisser W'keit

Population: ein Individuum

Besonderheiten: zu frühe Konvergenz ist zentrales Problem

Tabu-Suche

Repräsentation: phänotypnah

Mutation: unumkehrbare durch Tabu-Listen

Rekombination: keine

Selektion: bestes Individuum

Population: ein Elter, mehrere Kinder

Besonderheiten: bestes gefundenes Individuum wird zusätzlich gespeichert

Memetischer Algorithmus

Repräsentation: beliebig

Mutation: wird mit lokaler Suche verknüpft

Rekombination: beliebig

Selektion: beliebig

Population: beliebig

Besonderheiten: beliebig

Differentialevolution

Repräsentation: \mathbb{R}^L

Mutation: Mischoperator

Rekombination: Mischoperator

Selektion: Kind ersetzt Elter bei Verbesserung

Population: klein/mittelgroß

Besonderheiten: Mutation nutzt Populationsinformation

Scatter Search

Repräsentation: \mathbb{R}^L und andere

Mutation: keine

Rekombination: Teilmengenoperator und Kombination

Selektion: Selektion der Besten

Population: mittelgroß

Besonderheiten: viele Varianten, deterministisches Verfahren

Kultureller Algorithmus

Repräsentation: \mathbb{R}^L und andere

Mutation: nutzt Information des Überzeugungsraums

Rekombination: keine

Selektion: Umweltselektion

Population: mittelgroß

Besonderheiten: Überzeugungsraum speichert normatives und situationsbezogenes Wissen

Populationsbasiertes inkrementelles Lernen

Repräsentation: $\{0, 1\}^L$

Mutation: Änderung in der Populationsstatistik

Rekombination: implizit

Selektion: bestes Kindindividuum geht in Statistik ein

Population: wird durch Populationsstatistik ersetzt

Besonderheiten: benötigte Individuum werden aus der Statistik zufällig erzeugt

Ameisenkolonieoptimierung

Repräsentation: verschiedene

Mutation: jede Ameise konstruiert einen Lösungskandidaten

Rekombination: keine

Selektion: Güte bestimmt Einfluss auf globale Pheromonmenge

Population: Anzahl der Ameisen pro Iterationsschritt

Besonderheiten: globale Pheromonmengen repräsentieren Lösungskandidaten ähnlich zur Statistik in PBIL

Teilenschwarmoptimierung

Repräsentation: \mathbb{R}^L

Mutation: basiert auf Trägheit und Orientierung an Nachbarn



Rekombination: keine

Selektion: Orientierung am Besten (Population/eigene Historie)

Population: klein/mittelgroß

Besonderheiten: eher synchrones Durchkämmen des Suchraums

Literatur zur Lehrveranstaltung

-  Dorigo, M. and Stützle, T. (2004).
Ant Colony Optimization.
MIT Press, Cambridge, MA, USA.
-  Goss, S., Aron, S., Deneubourg, J., and Pasteels, J. M.
(1989).
Self-organized shortcuts in the argentine ant.
Naturwissenschaften, 76:579–581.
-  Kennedy, J. and Eberhart, R. (1995).
Particle swarm optimization.
In *Proceedings of the IEEE International Conference on Neural Networks*, page 1942–1948, Perth, Australia. IEEE Press.
-  Würtz, R. P., editor (2008).
Organic Computing.
Understanding Complex Systems. Springer Berlin / Heidelberg.