Mitschrift
Organic Computing
Sommersemester 2013

3. August 2013

Inhaltsverzeichnis

1 Einführung: Prinzipien **2**

2 Organische Methoden **4**

2.1 Partikelschwarmoptimierung 4

2.1.1 Verhalten bei Verlassen des Suchraums 7

2.2 Websuche, Data Mining 8

2.2.1 HITS-Algorithmus 9

2.2.2 PageRank 11

2.2.3 Zentralitätsmaße 11

2.3 Ameisen - Algorithmen 12

2.4 Evolutionäre Algorithmen 14

2.4.1 Selektionsverfahren 17

2.4.2 Permutation 18

2.4.3 Mutation 18

2.5 Peer-to-Peer-Netzwerke 19

3 Zusammenfassung **21**

1 Einführung: Prinzipien

Informatik im Zusammenspiel mit der Biologie.

I → B: Informatikmethoden werden eingesetzt um biologische Probleme zu behandeln. Bsp.: DNA-Analyse, räumliche Strukturen von Proteinen.

B → I: Nutzung biologischer Prinzipien in der Informatik insbesondere Organisationsprinzipien für große, komplexe computerisierte Systeme.

Komplexität entsteht durch:

- ungebrochene weitere Miniaturisierung
- Vernetzung vieler „voll ausgestatteter“ Rechner(-systeme)
- Einbettung von HW/SW-Systemen in technische Systeme (Automobil, Flugzeuge, Kühlschrank, Waschmaschine ...)

Schwer zu beherrschen: Fehler/Ausfall haben Auswirkungen, die sich gar nicht vorhersagen lassen („Chaostheorie“[1]). Was genau vorgeht ist nicht immer klar.

Abhilfe sucht man dort, wo die Systementwicklung schon mal erfolgreich durchgeführt worden ist: lebende Systeme.

Ziel: Erkenntnisse über die Funktionsweise lebender Systeme für die Entwicklung künstlicher Systeme nutzbar machen:

- organische Methoden
  - ↱ Schwärme
  - → Genetische/Evolutionäre Algorithmen
  - ↳ Ameisenalgorithmus
- organische Systeme

„Ein ’organischer Computer’ ist ein selbst-organisierendes System, das sich den Umgebungsbedürfnissen dynamisch anpasst.“

Self-*-Eigenschaften

selbst-

- konfigurierend
- optimierend
- heilend
- erklärend
- schützend

Vorteile

+ flexibel
+ robust
+ optimiert sich selbst

Nachteile

– können Fehler machen[2]
– sehr lange Trainingszeiten/Zeiten fürs Umkonfigurieren
– unerlaubte Beeinflussung[3]

[1] nicht lineares Verhalten
[2] Vor allem bei evolutionären Algorithmen: Ampel sollte Fehler nicht probieren müssen
[3] In der Natur z.B. der Kuckuck

Organisationsmuster
Emergenz [4], Autonomie, Föderation
Phänomene sind charakterisiert durch:

- Interaktion
- keine zentrale Kontrolle
- nicht explizit vorherbestimmte Muster

Emergenz bezeichnet das Entstehen neuer Strukturen oder Eigenschaften aus dem **Zusammenwirken** der Elemente in einem komplexen System. Als emergent werden Eigenschaften eines Ganzen bezeichnet, die sich aus den einzelnen Teilen nicht direkt herleiten lassen und nur aus dem Zusammenwirken der Teile, d.h. aus ihrem Prozess (ihrer Interaktion) erklärbar sind. Die emergenten Eigenschaften wirken zurück auf die einzelnen Komponenten.
Typische Phänomene:

- Unvorhersagbarkeit
  Auch mit perfektem Wissen über die einzelnen Komponenten lassen sich manche Eingeschaften des Gesamtsystem erst einmal nicht vorhersagen.
- Irreduzibilität
  Die emergente Eingeschaft lässt sich nicht aus den Eigenschaften der Systembestandteile ableiten. Erst die Interaktion löst die emergente Eigenschaft aus.

Bsp. für emergente Systeme:

- Schwingkreis, insbesondere Resonanz[5]
- Stabilität lebender Systeme gegen Umwelteinflüsse
- darstellende Kunst, insbesondere wenn sie abstrakt ist

Forschungsfragen/-richtungen

- natürliche Phänomene (besser und) quantitativ verstehen. Metriken zur Beurteilung von Selbstorganistations- und Emergenzphänomenen
- System-Architekturen: Obeserver/Controler-Architekturen
- Sicherheit: die selbständige Weiterentwicklung des OC-Systems muss Fehlentwicklungen verhindern
- Einbeziehung von A-priori-Wissen
- Wahrnehmungsfähigkeit → Autonomie und Benutzerinteraktion
- Selbsterklärung

[4] Das Ganze ist mehr als die Summe der Einzelteile, z.B. Google mit Suche über viele Rechnern
[5] z.B. Klappern beim Auto bei bestimmten Geschwindigkeiten oder Schwingen einer Brücke bei Gleichschritt darüber

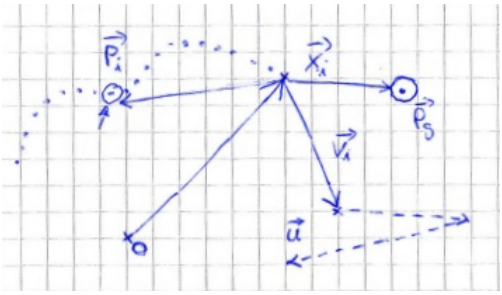
2 Organische Methoden

2.1 Partikelschwarmoptimierung

1995 Erfindung durch Kennedy/Eberhart
2002 Analyse durch Clerc/Kennedy
2003 einfache Analyse durch Trelea

Was wird optimiert?[6]

Gegeben:	eine Funktion $F : \mathbb{R}^d \to \mathbb{R}$ (auf einem Intervall I ggf.) beliebig (als black box)
Gesucht:	$\vec{x}_{\min}$, so dass $F(\vec{x}_{\min}) = \min\{F(\vec{x}) \mid x \in \mathbb{R}^d (\text{bzw. } I)\}$
Ganz allgemein:	evolutionär, d.h. Verbesserung von bereits erzielten Lösungen.
Genauer hier:	Nachahmen und lernen von anderen Individuen einer Population. Fortschritt in Generationen



$\vec{p}_i$ ist die von Individuum i bislang beste gefundene Lösung. $\vec{p}_g$ ist aktuell global beste bekannte Lösung. $\vec{u}$ ist Stelle an der das Individuum als nächstes hingeht.

`rate eine erste Generation an Individuen` $\vec{x}_1^{(0)}, \ldots \vec{x}_s^{(0)} \in \mathbb{R}^d$ `(Position, mögliche Lösung, 0te Generation)`
`rate für jedes Individuum einen Geschwindigkeitsvektor:` $\vec{v}_1^{(0)}, \ldots, \vec{v}_s \in \mathbb{R}^d$
`// alle Individuen zusammen: der Schwarm //`
$\forall i \in \{1, \ldots, s\} : \vec{p}_i := \vec{x}_i^{(0)}$;
$k := 0$;
`repeat`
    `bestimme die beste bisher gefundene Position überhaupt:` $\vec{p}_g$[7]
    `für jedes Partikel i: bestimme die von i bislang beste gefundene Position` $\vec{p}_i$
    Bewegungsgleichungen:
    $\vec{v}_i^{(k+1)} := \vec{a} \odot$[8] $\vec{v}_i + \vec{b}_{loc} \odot \vec{r}_{loc} \odot (\vec{p}_i - \vec{x}_i^{(k)}) + \vec{b}_g \odot \vec{r}_g \odot (\vec{p}_g - \vec{x}_i^{(k)})$
    $\vec{x}_i^{(k+1)} := \vec{c} \odot \vec{x}_i + \vec{d} \odot \vec{v}_i^{(k+1)}$(⊙1 `Zeiteinheit`)[9]

[6] wird gerne in der Prüfung gefragt
[7] Interaktion der Individuen, gerne Klausurfrage
[8] $\odot$ bedeutet komponentenweise Multiplikation
[9] Eine Generation entspricht dem Verstreichen einer Zeiteinheit $\to \vec{v}_i \cdot [t]$ hat Einheit $[m]$

$k := k + 1$

`until Abbruchkriterium erreicht (zB. #Iterationen, Änderungen unter einer Schranke, ...)`

- $\vec{a}$: Moment (konstanter Vektor, den Sie wählen müssen)
- $\vec{b}_{loc}, \vec{b}_g$: Anziehungskraft
- $\vec{c}, \vec{d}$: Momente (wir zeigen oBdA: beide $\vec{1}$)
- $\vec{r}_{loc}, \vec{r}_g \in [0,1]^d$ u.a.r (uniformly at random):
  Zufallsvektoren, die jedes mal ausgewürfelt werden

Ziel: Tradeoff zwischen Exploration[10] und Exploitation[11] [12]
Die Parameter („Moment“) steuern, wie stark man sich auf sich selbst verlässt und wie stark auf die anderen.

Ziel: Analyse der Parameterwahl, so dass der Schwarm konvergiert, d.h. sich „in unendlicher Zeit“ auf eine Lösung festlegt.

Die Vektorkompontenten sind entkoppelt[13], deswegen reicht die Untersuchung des 1-dimensionalen.

$$v^{(k+1)} = a \cdot v^{(k)} + b_{loc} \cdot r_{loc} \cdot (p_{loc}^{(k)} - x^{(k)}) + b_g \cdot r_g \cdot (p_g^{(k)} - x^{(k)})$$
$$x^{(k+1)} = c \cdot x^{(k)} + d \cdot v^{(k+1)}$$

$\boxed{r_{loc} = r_g = \frac{1}{2}}$ (Erwartungswerte)

[10] hingehen wo noch niemand war
[11] untersuche die Umgebung schon gefundener guter Lösungen
[12] wird gerne in der Prüfung gefragt
[13] Dimensionen beeinflussen sich gegenseitig nicht

Mit Substitution:

$$b = \frac{1}{2}(b_{loc} + b_g)$$

$$p^{(k)} = \frac{b_{loc}}{b_{loc} + b_g} \cdot p_{log}^{(k)} + \frac{b_g}{b_{loc} + b_g} \cdot p_g^{(k)}$$

folgt:

$$v^{(k+1)} = a \cdot v^{(k)} + b \cdot (p^{(k)} - x^{(k)})$$

$$x^{(k+1)} = c \cdot x^{(k)} + d \cdot v^{(k+1)}$$

$$\rightarrow v^{(k+1)} = \frac{1}{d}(x^{(k+1)} - c \cdot x^{(k)})$$

$$v^{(k)} = \frac{1}{d}(x^{(k)} - c \cdot x^{(k-1)}) \text{ (Parameterverschiebung)}$$

Einsetzen:

$$x^{(k+1)} + (\underline{bd} - a - c) \cdot x^{(k)} + ac \cdot x^{(k-1)} = \underline{bd} \cdot p^{(k)} \leftarrow \text{diese Position}$$

soll im Unendlichen erreicht werden und <u>fest</u> sein.

Setze $d = 1$ und wähle b „hinterher“ richtig.
Die $x^{(k)}$ sollen im Unendlichen $p^{(.)}$ erreichen:

$$X + (b - a - c)X + acX = bp^{(k)}$$

$$\Leftrightarrow (1 + b - a - c + ac) \cdot X = bp^{(k)}$$

$$0 \stackrel{!}{=} 1 - a - c + ac = (a - 1)(c - 1) \rightarrow \text{setze } c = 1$$

$$\begin{pmatrix} x^{(k+1)} \\ v^{(k+1)} \end{pmatrix} = \begin{pmatrix} 1 - b & a \\ -b & a \end{pmatrix} \cdot \begin{pmatrix} x^{(k)} \\ v^{(k)} \end{pmatrix} + \begin{pmatrix} b \\ b \end{pmatrix} \cdot p^{(k)} \rightarrow p$$

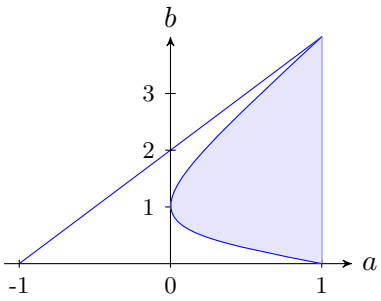
$$Y^{(k+1)} = A \cdot Y^{(k)} + \begin{pmatrix} b \\ b \end{pmatrix} \cdot p \rightarrow \begin{pmatrix} p \\ 0 \end{pmatrix} \text{ Equilibrium / Fixpunkt } [14]$$

Das System konvergiert, wenn die Beträge der (auch komplexen!) Eigenwerte von A alle echt kleiner als 1 sind.

$\det \begin{pmatrix} 1 - b - \lambda & a \\ -b & a - \lambda \end{pmatrix} = (1 - b - \lambda)(a - \lambda) + ab \stackrel{!}{=} 0 \Rightarrow \lambda_{1/2} = \frac{a-b+1}{2} \pm \sqrt{\frac{(a-b+1)^2}{4} - a}$

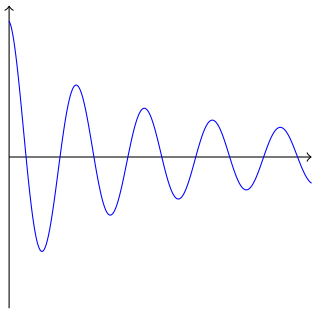
[14] Punkt an dem ein System eingeschwungen ist

Eigenwerte < 1 bei $a < 1$ und $b > 0$, $2a - b + 2 > 0$

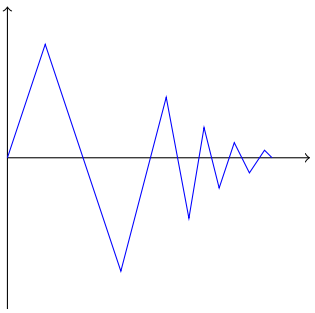


Innerhalb vom Dreieck konvergiert das System. Innerhalb der schraffierten Fläche kommt es zu harmonischen Schwingungen.

Schwingungen:

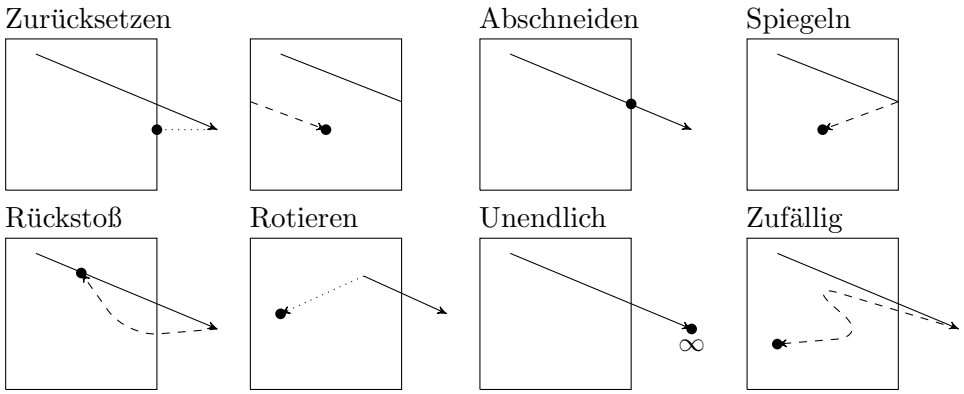


Harmonisch: es gibt komplexe Eigenwerte



Zickzaging: mind. einer der EW hat neg. Realteil

2.1.1 Verhalten bei Verlassen des Suchraums



2.2 Websuche, Data Mining

Das World'Wide'Web als Beispiel eines organischen Systems

- ist ein Hypertext-Corpus mit extremer Komplexität
- User mit ganz verschiedenen und oft gegensätzlichen Zielen erzeugen durch Hyperlinks verbundene Inhalte auf äußerst lokaler Ebene

Fragen und Autoritäten

- Spezifische Anfrage „Does Netscape support the JDK 1.1 codesigning API"
  (Problem: nur ganz wenigen Seiten enthalten die gesuchte Information)
- Thematisch breite Anfragen: „Find Information about the Java programming language"
  (Problem: Überfluss an Webseiten die das Thema behandeln, insbesondere bei rein textbasierten Suchverfahren)
- Ähnlichkeitsanfrage: finde ähnliche Seiten wie www.java.com

Bei den thematischen Anfragen: Herausfinden der relevanten Seiten, der Autoritäten
Problem: wie misst man die emergente Eigenschaft „Autorität"?
Komplikation:

- Die Seite, wo die Anfrage am häufigsten drin auftaucht, muss nicht die mit der höchsten Autorität sein. (Autorität ist kein endogenes Maß, das allein anhand einer Seite bestimmt werden kann.)
- Die Seite mit höchster Autorität muss nicht einmal die Suchwörter enthalten (z.B. www.audi.de und „Automobilhersteller")

Die Link-Struktur auswerten! Sie enthält menschliche Bewertungen
Achtung:

+ Navigationslinks sollten nichts beitragen
+ Werbung??

Erste Heuristik

Gib von allen Seiten, die den Suchstring enthalten, die aus, auf die am häufigsten gezeigt wird. Probleme:

→ audi.de–Problematik
→ populäre Seiten, die häufig verlinkt, werden zu Autoritäten für alles

Hier jetzt: es gibt Autoritäten und die, die sie kennen. (Autoritäten für Autoritäten, Hubs [15])
Wie identifiziert man die beiden?
Achtung: Clustering ist etwas anderes, z.B. die Trennung von Seiten, die z.B. „echte Schlüssel" meinen von denen, die Kryptographie meinen, oder Fenster von Windows (vor allem im englischen)

[15] Nabe, wie das Zentrum von einem Rad mit vielen Speichen, z.B. beim Fliegen erst mal zu einem großen Flughafen, vor dort aus weiter

2.2.1 HITS-Algorithmus

Vorstellung des HITS[16]-Algorithmus von Kleinberg

Gegeben: eine thematisch breite Anfrage mittels Suchstring σ
Ziel: Ausgabe von Seiten mit hoher Autorität

Zuerst: berechne einen Graphen S_σ (Basis) mit folgenden Eigenschaften

(i) S_σ ist relativ klein
(ii) S_σ hat viele relevante Seiten
(iii) S_σ hat viele der Seiten mit höchster Autorität

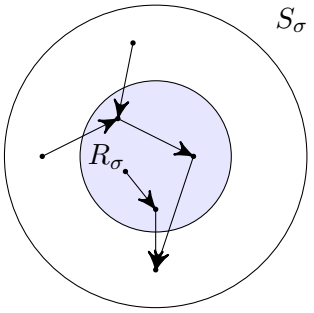
Wir beginnen mit einer textbasierten Suche, die uns die t[17] „besten“ Seiten liefert $\rightarrow$ Wurzel R_σ (erfüllt (i), (ii) aber nicht (iii))

Phänomen: kaum interne Kanten (Links) (Kleinbergs Experiment: für $t = 200$ gabs bis zu ≤ 28 interne Links bei 39800 möglichen internen Links) Seite mit hoher Autorität bzgl. σ nicht in R_σ. Wahrscheinlichkeit ist hoch, dass eine Seite aus R_σ auf sie zeigt.

```
S_σ := R_σ
fuer alle p ∈ R_σ
  S_σ := S_σ ∪ Γ⁺(p)  [18]
  if |Γ⁻(p)| ≤ d
  then S_σ := S_σ ∪ Γ⁻(p)
  else S_σ := S_σ ∪ d zufaellig aus Γ⁻(p) ausgewaehlte Seiten[19]
```

[Kleinbergs. Exp. $t = 200, d = 50$ $S_\sigma \approx 1000$ bis 5000 Seiten]
Noch 2 Heuristiken

- lösche interne (bzgl. Domain) Links (Navigationslinks)
- erlaube aus einer Domain nur m (ca. 4 bis 8) Links auf eine Seite (wegen: „diese Seite wurde erstellt von...“)



Nun: Unterscheiden zwischen relevanten und bloß populär.
Auf S_σ ist schon die Sortierung nach dem Eingangsgrad ganz gut (im Gegensatz zum Gesamtgraphen).

[16] Hypertext-Induced Topic Search
[17] darf man selber wählen
[18] Menge der Nachbarseiten von p (Seiten, auf die p zeigt)
[19] nennt sich Sampling

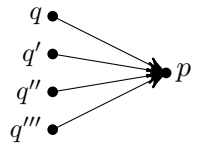
Autorität berechnen

Seiten mit großer Autorität zur Anfrage σ sollten nicht nur großen Eingangsgrad haben; diejenigen, die auf sie zeigen, sollten sich erheblich überlappen; Hubs

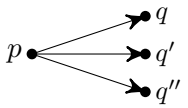
Hubs und Autoritäten verstärken sich gegenseitig.
Jede Seite p hat eine Autorität $x^{<p>}$ und ein Hubgewicht $y^{<p>}$ (Zahlen! ☺) so dass $\sum_{p \in S_\sigma} (x^{<p>})^2 = 1$ und $\sum_{p \in S_\sigma} (y^{<p>})^2 = 1$

Je größer der x-Wert, um so besser zu σ passend die Seite.
y-Wert, ein um so besserer Hub liegt vor.

$$x^{<p>} := \sum_{q:(q \to p)} y^{<q>} \quad [20]$$



$$y^{<p>} := \sum_{q:(p \to q)} x^{<q>} \quad [21]$$



auf Länge 1 normiert.

```
Iterate(k)
  n := |S_σ|; x := 1⃗ ∈ ℝ^n; y := 1⃗ ∈ ℝ^n
  wiederhole k mal
    x^<p> :=
    y^<p> :=
    normiere
  gib die Seite(n) auf mit hoechster Autoritaet in x
```

Wenn A die Adjazenzmatrix[22] von S_σ ist, so haben wir:

$$\vec{x} := A^T \cdot \vec{y}$$
$$\vec{y} := A \cdot \vec{x}$$

Satz: Für $k \to \infty$ konvergieren die Vektoren x und y bei Anwendung von Iterate(k).
Beweis:

$$\vec{y}_k = AA^T \cdot \vec{y}_{k-1} \quad \Rightarrow \vec{y}_k = (AA^T)^k \cdot \vec{1}$$
$$\vec{x}_k = A^T A \cdot \vec{x}_{k-1} \quad \Rightarrow \vec{y}_k = (A^T A)^{k-1} \cdot \vec{1}$$

Da AA^T und $A^T A$ symmetrisch: Konvergenz □.

x konvergiert gegen den prinzipiellen Eigenvektor von $A^T A$
y konvergiert gegen den prinzipiellen Eigenvektor von AA^T

[20] je mehr gute Hubs auf eine Seite zeigen, desto besser
[21] auf je mehr Seiten mit hoher Autorität ein Hub zeigt, desto besser
[22] für die Analyse, praktisch wird obiger Algorithmus verwendet

Der prinzipielle Eigenvektor: Eigenvektor, der zum betragsmäßig größten Eigenwert gehört. Experimente zeigen: $k \approx 20$ bis 30 reicht aus. (Expander: relativ große Teilmenge von Knoten hat viele Nachbarn außerhalb der Knotenmenge)

S_σ ist Teil des WWW, und der WWW-Graph ist ein Expander.

Literatur: Jon M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, J.ACM 1999

2.2.2 PageRank

Eine Alternative zum Hits-Algorithmus ist der PageRank-Algorithmus.

`Initialisiere die Relevanz der Seiten auf` $\frac{1}{\texttt{Anzahl Seiten}}$
`Wähle` d`, zB` $d = \frac{4}{5}$
`while kein Terminierungskriterium erfüllt do`
  `for all Seiten S do`
    `Neue Relevanz von S ist` $\frac{1-d}{\texttt{Anzahl Seiten}} + d \cdot \sum_{\forall R:(R,S)\in E} \frac{\texttt{Relevanz von R}}{\texttt{Ausgangsgrad von R}}$
  `end for`
`end while`

2.2.3 Zentralitätsmaße

Degree Centrality

$$C_D(e) = \frac{\deg(e)}{n-1}$$

Wobei $\deg(e)$ den Grad des Knotens beschreibt (Anzahl der Nachbarn).
Bedeutung: Je mehr Nachbarn ein Knoten e hat, desto wichtiger ist e.

Betweeness Centrality

$$C_B(e) = \frac{2}{(n-1)(n-2)} \cdot \sum_{s\neq e\neq t, t\neq s} \frac{\sigma_{st}(e)}{\sigma_{st}}$$

σ_{st} beschreibt die Anzahl der kürzesten Wege von s nach t, $\sigma_{st}(e)$ beschreibt die Anzahl derer, die dabei über e verlaufen.
Bedeutung: Je mehr Bekanntschaften 'direkt' über einen Knoten e verlaufen, desto wichtiger ist e.

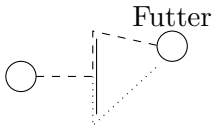
Closeness Centrality

$$C_C(e) = \frac{\sum_{t\neq e} \text{dist}(e,t)}{n-1}$$

$\text{dist}(e,t)$ bezeichnet hier die Länge des kürzesten Pfades zwischen e und t.
Bedeutung: Je 'direkter' viele Leute einen Knoten e kennen, desto wichtiger ist e.

2.3 Ameisen - Algorithmen

1991 von Dorigo entwickelt



Ameisen finden nach einiger Zeit einen sehr kurzen Weg zwischen Nest und Futter.
Autokatalytisch: Ein sich selbst verstärkender Prozess.
Synergie

Die Ameise markiert ihren Weg mittels Pheromonen. Die Pheromone verwittern. Das führt dazu, dass die Konzentration auf kürzeren Wegen stärker ist als auf langen Wegen.

Traveling Salesperson Problem (TSP)
Gegeben ist ein Graph mit Punkten (Knoten) im $\mathbb{R}^2$
Abstand ist der Euklidische Abstand
Gesucht ist eine möglichst kurze Rundereise, die jeden Punkt genau einmal besucht.

$$b_i(t) = \text{ Anzahl der Ameisen auf Punkt } i \text{ zum Zeitpunkt } t$$

$$A = \sum_{i=1}^{n} b_i(t) \text{ bei Punkten } 1, \ldots, n$$

$$\tau_{ij}(t) = \text{Intensität des Pheromons auf der Kante } \{i,j\} \text{ zum Zeitpunkt } t$$

$$\tau_{ij}(t+1) = \underbrace{\rho}_{\text{Verdunstung/Verwitterung}} \cdot \tau_{ij}(t) + \underbrace{[(1-\rho)]}_{\text{möglich, aber muss nicht}} \cdot \Delta\tau_{ij}(t, t+1)$$

$$\Delta\tau_{ij}(t, t+1) = \sum_{k=1}^{A} \Delta\tau_{ij}^{(k)}(t, t+1), \qquad \tau^{(k)} \text{ Name einer Ameise}$$

$$p_{ij}^{(k)}(t) = \begin{cases} \frac{\tau_{ij}(t)^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{j \in \text{allowed}(k)} \tau_{ij}(t)^{\alpha} \cdot \eta_{ij}^{\beta}} & \text{falls } j \in \text{allowed}(k) \\ 0 & \text{sonst} \end{cases}$$

$$\eta_{ij} = \text{ Sichtbarkeit, oft } \eta_{ij} = \frac{1}{d_{ij}} \text{ wobei } d_{ij} \text{ Abstand } i \text{ nach } j$$

Die künstlichen Ameisen haben ein Gedächtnis, sind nicht blind, sie arbeiten Zeit-diskret.

Ant-density

$$\Delta\tau_{ij}^{(k)}(t, t+1) = \begin{cases} Q_1 & \text{Ameise } k \text{ geht von } i \text{ nach } j \text{ zwischen } t \text{ und } t+1 \\ 0 & \text{sonst} \end{cases}$$

Ant-quantity kürzere Kanten werden bevorzugt

$$\Delta\tau_{ij}^{(k)}(t,t+1) = \begin{cases} \frac{Q_2}{d_{ij}} & \text{Ameise } k \text{ geht von } i \text{ nach } j \text{ zwischen } t \text{ und } t+1 \\ 0 & \text{sonst} \end{cases}$$

Jede Ameise berechnet für sich eine mögliche Lösung (Tour).

Ameisen-Kolonie

Um den „Explorationseffekt“ zu unterstützen kann man zusätzlich einen Schwellwert Q einführen, anhanddessen mithilfe eines von Ameise k auf Punkt i gewürfelten Wertes `rnd` bestimmt wird, ob die Ameise $p_{ij}^{(k)}(t)$ zum Punktwechsel verwendet, oder einen anderen Punkt besucht.

`if rnd` $> Q$ `then benutze` $p_{ij}^{(k)}(t)$ `zum Punktwechsel`
$\leq Q$ `then gehe zu` $\underset{j \in allowed(k)}{\texttt{argmax}} \{\tau_{ij}^{\alpha}\}$

Problem: nach n Iterationen hat jede Ameise eine Tour, die „Beste“ ist „Lösung“. Aber was nun?

Ant-cycle

$$L^{(k)} = \text{ Länge der bislang besten Tour der Ameise } k$$

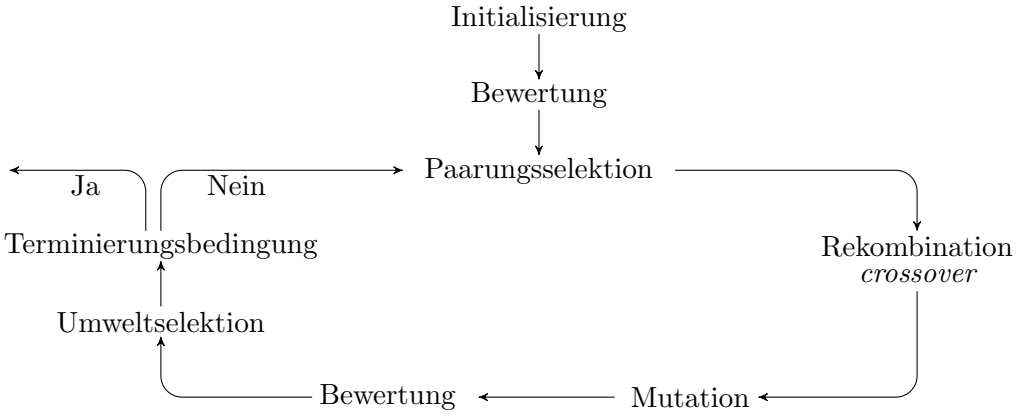
$$\Delta\tau_{ij}^{(k)}(t,t+n) = \begin{cases} \frac{Q_2}{L^{(k)}} & \text{falls Ameise } k \text{ auf ihrer Tour } \{i,j\} \text{ benutzt} \\ 0 & \text{sonst} \end{cases}$$

Ameisen laufen jetzt viele Runden und am Ende wird das beste Ergebnis als Gesamtergebnis ausgegeben.

2.4 Evolutionäre Algorithmen

- Populationskonzept
- Reproduktion durch Vererbung und Variation
- Selektionsprinzip

Zyklus evolutionärer Algorithmen



Allgemeiner EA
Gegeben: Zielfunktion F
Populationsgröße μ: Zahl der Nachkommen λ. Nennt man dann $(\mu + \lambda)$ EA [23]
`t:= 0`
`P(t) := erzeuge eine Population der Größe` μ
$\texttt{Eval}_F\texttt{(P(t))}$
`while Terminierungsbedingung nicht erfüllt do`

- `selektiere Eltern P' für` λ `Nachkommen aus P(t) mittels Selektionsoperator`
- `P'' := erzeuge Nachkommen durch Rekombination aus P'`
- `P''' := mutiere die Induviduen in P'' mittels Mutationsoperator`
- $\texttt{Eval}_F\texttt{(P''')}$
- `t := t+1`
- `P(t) := selektiere` μ `Individuen aus P'''`

`end`
`gib bestes Idividuum aus.`

Genotyp	wie ein Individuum kodiert wird
Phänotyp	das, was dargestellt wird

Wir wollen jetzt einen EA angeben, der das Sortierproblem mit 1 Individuum und 1 Nachkommmen löst: (1+1) EA
Wichtig für uns hierbei: man kann diesen EA analysieren.

[23] das + heißt hier und

Die Eingabe sei eine Permutation π der Zahlen 1 bis n ($\pi \in S_n$)[24]
Verbergen in einer Black Box
Ziel: Permutation finden, so dass daraus auf π angewandt die Folge $1, .., n$ wird.
Wir benötigen eine Zielfunktion, die ein Maß für die Sortiertheit von π' sind. Gebräuchlich sind (Ziel: je größer das Maß, umso sortierter die Folge)

$\mathrm{INV}(\pi')$: Anzahl der Paare $(i,j), i<j,$ mit $\pi'(i) < \pi'(j)$
(Inversion, hier: Zähle die Paare in konkreter Reihenfolge)
$\mathrm{HAM}(\pi')$: Anzahl der Indizes i mit $\pi'(i) = i$
(Schlüssel an richtigen Positionen, HAM = Hamming)
$\mathrm{LAS}(\pi')$: Länge k der längsten aufsteigenden Teilfolge $\pi'(i_1) < ... < \pi'(i_k)$
(longest ascending subsequence)

Unser Algo kann nur Mutationen ausführen. Dafür dürfen als Teil einer Mutation folgende Operationen benutzt werden:

[`swap(i)`	: tausche die Schlüssel an Pos i und $i+1$, falls $i < n$][a]
`exch(i,i)`	: tausche die Schlüssel an Pos i und j
`jump(i,j)`	: Schlüssel an Pos i an die Pos j springt, alle dazwischen werden einen nach vorne bzw. nach hinten verschoben

[a] nur bedingt hilfreich

Beispiel für `jump`: 8 4 3 6 1 7 2 5, `jump(2,6)` $\rightarrow$ 8 3 6 1 7 4 2 5

Der Mutationsoperator arbeitet wie folgt:

- Wähle natürliche Zahl S gemäß einer Poisson-Verteilung mit $\lambda = 1$
  ($\Pr[S=i] = \frac{e^{-\lambda}\cdot\lambda^i}{i!} = \frac{1}{e\cdot i!}$, $E[S]=1$, $\mathrm{Var}[S]=1$)
- `for i:= 1 to S+1 do`
  `wähle gleichverteilt zwei Indizes (i,j), i`$\neq j$`, i,j`$\in \{$`1,..,n`$\}$
  $\begin{cases} \texttt{mit Wkt } 0.5 : \texttt{exch}(i,j) \\ \texttt{mit Wkt } 0.5 : \texttt{jump}(i,j) \end{cases}$

Unser (1+1) EA läuft wie folgt:

- t := 0
- Wähle einer zufällige Permutation π_0
- while $f(\pi_t) <$ opt do[25]
  - erzeuge π' durch Mutation aus π_t, $t := t+1$
  - falls $f(\pi') > f(\pi_{t-1})$
    then $\pi_t := \pi'$
    else $\pi_t := \pi_{t-1}$

[24] symmetrische Gruppe über n Elemente
[25] opt ist bei INV $\frac{n(n-1)}{2}$, sonst n

done

$f \in \{$ INV, HAM, LAS$\}$ fest
Hillclimber, d.h. die Folge $\pi_0, \pi_1, ..$ wird nie schlechter.

Satz: Die erwartete Anzahl an Auswertungen $f(\pi)$ des $(1+1)$EA für $f \in \{\text{INV}, \text{HAM}\}$ ist mindestens $\Omega(n^2)$.

Beweis: Die Wahrscheinlichkeit, dass die erste gewählte Permutation sortiert ist, ist $\frac{1}{n!}$
Die Permutation ist die Lösung, wenn im letzten Schritt der Mutation die sortierte Folge erzeugt wird.
Es gibt höchstens zwei Paare, die das tun, und das je möglicher Operation.
Erfolgswahrscheinlichkeit: $\frac{1}{2} \cdot \frac{2}{n(n-1)} \cdot 2$
$\Rightarrow E[t] \geq \frac{1}{2} n(n-1) = \Omega(n^2)$ $\square$

Satz: Die erwartete Anzahl an Auswertungen $f(\pi)$ des (1+1)EA für $f \in \{\text{INV}, \text{HAM}\}$ ist höchstens $\mathcal{O}(n^2 \log n)$.

Beweis für f = INV

Sei (i, j) ein Indexpaar mit $i < j$ und $\pi(i) > \pi(j)$
Sei a die Anzahl der Schlüssel an den Positionen $i+1, \ldots, j-1$, die kleiner als $\pi(j)$ sind.
Sei b die Anzahl der Schlüssel an den Positionen $i+1, \ldots, j-1$, die zwischen $\pi(j)$ und $\pi(i)$ liegen.
Sei c die Anzahl der Schlüssel an den Positionen $i+1, \ldots, j-1$, die größer als $\pi(i)$ sind.

`exch`(i,j) vergrößert die Zielfunktion um $2b+1$
`jump`(i,j) ändert die Zielfunktion um $a+b-c+1$
`jump`(j,i) ändert die Zielfunktion um $-a+b+c+1$
→ mindestens einer der Werte `jump`(i,j), `jump`(j,i) ist positiv.

Mindestens um 1 wird die Zielfunktion verbessert, falls genau eine lokale Operation erfolgreich ausgeführt wird, [genau eine lokale Operation heißt S=0; $Pr[S=0] = \frac{1}{e}$] und diese Operation ist `exch`(i,j) oder `exch`(j,i) [$\frac{2}{n(n-1)} \cdot \frac{1}{2}$ Wahrscheinlichkeit] oder die Operation ist die "gute" von `jump`(i,j) und `jump`(j,i) [Wahrscheinlichkeit $\frac{2}{2n(n-1)} \cdot \frac{1}{2}$]

Ist die Anzahl der falschherum stehenden Paare m, so ist die Wahrscheinlichkeit, den Wert der Zielfunktion zu verbessern mindestens

$$\frac{1}{e} \cdot m \cdot \left(\frac{1}{n(n-1)} + \frac{1}{2n(n-1)} \right) = \frac{3m}{2n(n-1)e}$$

$\Rightarrow \frac{2}{3} \cdot \frac{1}{m} \cdot e \cdot n(n-1)$ erwartete Schritte, um den Wert der Zielfunktion zu verbessern.

$$\Rightarrow E[t] \leq \sum_{m=1}^{\frac{1}{2}n(n-1)} \frac{2}{3} \cdot \frac{1}{m} \cdot e \cdot n(n-1) \leq \frac{2}{3} \cdot e \cdot n^2 \cdot \underbrace{\sum_{m=1}^{\frac{1}{2}n(n-1)} \frac{1}{m}}_{\text{Harmonische Reihe}, \leq \ln(\frac{1}{2}n(n-1))+1} = \mathcal{O}(n^2 \cdot \log n)$$

Beweis für f = HAM

Ist $HAM(\pi_t) = k$, stehen $n - k$ Schlüssel an falschen Stellen.
Ist der Schlüssel i an Position j ($\pi_t(j) = i$), $i \neq j$ ist auch der Schlüssel an Position i an der falschen Stelle. `exch`(i,j) und `exch`(j,i) verbessen den Wert der Zielfunktion um mindestens 1.
$\Rightarrow$ es gibt mindestens $n - k$ „gute“ `exch`-Operationen.
$\Rightarrow$ Rechnung wie gerade eben: $E[t] \geq 2en^2 \cdot \sum_{k=1}^{n} \frac{1}{k} = \mathcal{O}(n^2 \log n)$ $\square$

Beweis für f = LAS

Sei $i_1 < \cdots < i_k$ mit $\pi(i_1) < \cdots < \pi(i_k)$ eine LAS
$k < n \Rightarrow$ es gibt mindestens eine JUMP-Operation, die die Fitness um 1 erhöht, dh.
$\exists j, i_e$ mit $i_e < j < i_{e+1}$ und $[\pi(j) < \pi(i_e)$ oder $\pi(j) > \pi(i_{e+1})]$
oder $j < i_1$ und $\pi(j) > \pi(i_1)$ oder $j > i_k$ und $\pi(j) < \pi(i_k)$

Wahrscheinlichkeit für korrekten JUMP von j:

Auswahl S = 0	$\frac{1}{e}$
Auswahl von JUMP	$\frac{1}{2}$
Auswahl der korrekten Indizes	$\frac{1}{n(n-1)}$

Insg: $\frac{1}{2en(n-1)}$

Es gibt (n-k) Elemente außerhalb der betrachteten LAS.
$\Rightarrow$ Wahrscheinlichkeit für Verbesserung: $\frac{n-k}{2en(n-1)}$
$\Rightarrow$ Erwartete Wartezeit für nächste Verbesserung: $\frac{2en(n-1)}{n-k}$
$\Rightarrow$ Erwartete Gesamtlaufzeit: $\sum_{k=1}^{n-1} \frac{2en(n-1)}{n-k} = 2en(n-1) \sum_{k=1}^{n-1} \frac{1}{l} = \mathcal{O}(n^2 \log n)$

2.4.1 Selektionsverfahren

Turnierselektion

- wähle mehrere Individuen aus Population
- das bessere Individuum gewinnt (mit einer höheren Wahrscheinlichkeit)

Rangselektion

- erstelle Rangliste für Individuen
- weise jedem Individuum, abhängig von seinem Rang, eine Wahrscheinlichkeit zu

$$\text{z.B. } Pr[I_k] = \frac{2}{N}\left(1 - \frac{k-1}{N-1}\right)$$

2.4.2 Permutation

One-point-crossover

Partially-matched-crossover (PMX)

Edge-recombination-crossover (ERX)

2.4.3 Mutation

Swap mutation operator Vertauschen zweier zufällig gewählter Zahlen.

1	2	3	4	5	6	→	1	2	5	4	3	6

Inversion mutation operator Spiegeln der zwischen 2 zufällig gewählten Zahlen liegenden Sequenz.

1	2	3	4	5	6	→	1	5	4	3	2	6

2.5 Peer-to-Peer-Netzwerke

Notizen: für Mitschrift siehe Folien

Historie

- 2004 besteht ein großteil des Traffics aus Filesharing.
- Erstes derartiges Netzwerk: **Napster**
  - Einen zentralen Server der über alle Clients und alle Daten bescheid weiß
  - Client schickt Anfrage an Server, Clients haben die Daten
  - Datenübertragung geht direkt über Clients
  - Skaliert schlecht, Fehleranfällig
  - Damit eigentlich kein P2P Netz, außer der eigentliche Download
- **Gnutella** soll Probleme von Napster lösen und funktioniert ohne zentrale Strukturen
  - Ist paretoverteilt: von 2 bis n, dann haben 2 ingesamt $\frac{1}{2}$ (Hälfte) Anteil, die n dann $\frac{1}{n}$[26]
  - Anfragen haben immer ein TTL, damit sie nicht zu lokal bleiben („3 ist ein vernüftiger Wert“)
  - Flooding: Netzwerk wird mit Anfragen geflutet, ist zum großen Teil nur noch damit anstelle von Download beschäftigt (Asynchronität: Download bereits stattgefunden, es kommen trotzdem noch Antworten von anderen zurück)
  - Small world hypothesis: Experiment aus den USA bei denen Pakete nicht über die Post über die ganze Welt verschickt werden sollten, mit der Anweisung „gib das jemanden den du kennst „. Die meisten Pakete kamen innerhalb vom 5 Hops an (daher TTL von 3)
  - Verbesserungen: Random walks (Zeit vs. Anzahl der Anfragen) und Caching (entlang den Pfaden vorherige Antworten speichern. Ackermann: eine Lösung für passive Replikate)
  - Weiteres Problem: keine Struktur in der Datenablage (eine „Nein “Antwort muss nicht unbedingt stimmen, wurde das ganze Netz durchsucht?)

DHT

- Wenn nur die Daten gehasht werden, gibt es Probleme wenn Peers hinzukommen oder gehen
- Daher auch noch die Peers hashen

[26] Die reichsten 10% haben 90% des Geldes

Gradoptimierte Netzwerke

- Möglich: konstanter Grad, log. Durchmesser. Bsp: $n = 15$ Knoten, Durchmesser: $6 = 2(\log(n+1) - 1)$
- Problem mit Bäumen: es gibt Hotspots die nicht ausfallen drürfen
- Besser **Butterfly-Graph** was hotspots angeht, dafür schlechter wenn sich Peers spontan anmelden und abmelden wollen, da immer eine bestimmte Anzahl gebraucht wird um eine neue Schicht erstellen zu können (Problem der Skalierbarkeit)
- **Viceroy** (eine Schmetterlingsart)
- Besser als log Durchmesser kann man mit konstantem Grad nicht werden
- Bei CAN war Durchmesser $\mathcal{O}(n^{\frac{1}{2}})$, Viceroy kann das in $\log(n)$
- Wrap-around-Kanten: die untersten Knoten gibt es eigentlich nicht, es sind die selben wie oben (so aber leichter zu zeichnen). Dadurch hat jeder Knoten Grad 4 (Beispiel siehe Folien)
- Konstruktion Butterfly-Graphen: siehe Bild
  - Jeder Knoten hat einer Adresse, i für das Level und ein bitstring der die Spalte angibt
  - Erst mal alle vertical verbinden (mit wrap around)
  - Danach im bitstring das $i$te Bit flippen und mit dem Knoten im nächsten level verbinden
- Wenn man von `000` nach `111` will, muss man nach und nach die Bits korrigieren, d.h. `000`→`100`→`110`→`111` (über Kreuzkanten)
- Aufbau Viceroy:
  - Rückgrad: levels (Knoten die in Kreisen zusammengeschlossen sind)
  - Neuer Knoten wählt erstmal ein level
  - Ein Ring der alle Knoten verbindet
  - Butterfly Netzwerk zwischen den levels
  - Falls ein Knoten in einem level nicht gefunden wird, dann kommen die Kreise ins Spiel (da nochmal Datenstruktur drüber packen die dafür sorgt, dass immer ein weiteres Element gefunden wird)

3 Zusammenfassung

- Allgemeine Begriffsbestimmung für OC
  - Abgrenzung zu anderen Disziplinen
  - self-*-properties
  - Emergenz
- Organische Methoden
  - PSO, Optimierung kontinuierlicher Funktionen
  - Ameisen-Algo., Optimierung diskreter Funktionen
  - Suchverfahren: HITS-Algorithmus, Pagerank
  - Zentralitätamaße
- Organische Systeme
  - P2P-Netzwerke
    - Napster
    - Gnutella
    - CAN
    - Viceroy