INF FAKULTÄT FÜR INFORMATIK

Evolutionäre Algorithmen

Metaheuristiken und verwandte Optimierungsverfahren I/II

Prof. Dr. Rudolf Kruse **Christian Moewes**

{kruse,cmoewes}@iws.cs.uni-magdeburg.de

Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik

Institut für Wissens- und Sprachverarbeitung

Übersicht

Was ist eine Metaheuristik?

- Algorithmus zur näherungsweisen Lösung eines kombinatorischen Optimierungsproblems
- definiert abstrakte Folge von Schritten, die auf beliebige Problemstellungen anwendbar sind
- einzelnen Schritte müssen aber problemspezifisch implementiert werden

$\Rightarrow$ problemspezifische Heuristik

Einsatz von Metaheuristiken

- bei Problemen, wo kein effizienterer Lösungsalgorithmus bekannt
- z.B. kombinatorische Optimierungsprobleme
- Finden einer optimalen Lösung ist in der Regel nicht garantiert
- gute Lösungen können beliebig schlecht verglichen mit optimalen Lösung sein
- Erfolg und Laufzeit hängen ab von
  - Problemdefinition und
  - Implementierung der einzelnen Schritte

$\Rightarrow$ EAs sind auch Metaheuristiken

Übersicht

Lokale Suchalgorithmen

nach [Weicker, 2007]

- geg.: Optimierungsproblem $(\Omega, f, \succ)$
- ges.: finde Element $x \in \Omega$, das f optimiert (max. oder min.)
- o.B.d.A.: finde ein Element $x \in \Omega$, das f maximiert
  (ist f zu minimieren, dann betrachten wir $f' \equiv -f$)

$\Rightarrow$ **lokale Suchalgorithmen** zum Finden von lokalen Optima in Ω

- **Voraussetzung:** $f(x_1)$ und $f(x_2)$ unterscheiden sich für ähnliche $x_1, x_2 \in \Omega$ nicht zu sehr (keine großen Sprünge in f)
- anwendbar für beliebige Ω zum Finden von lokalen Optima

Lokale Suchalgorithmen

- lokale Suchalgorithmen = Sonderfall der EAs
- Population: 1 Lösungskandidaten $\Rightarrow$ verschiedene Konsequenzen
  - Rekombinationsoperator nicht sinnvoll, da nur ein Individuum
  - Veränderungen: Mutations- bzw. Variationsoperator
  - Selektion: neu erzeugtes Individuum anstatt Elternindividuum in nächster Generation?

$\Rightarrow$ ein Basisalgorithmus für alle lokalen Suchalgorithmen

- Varianten durch unterschiedliche Akzeptanzkriterien
- Individuen besitzen i.d.R. keine Zusatzinformationen $\mathcal{Z} = \emptyset$
- Genotyp $\mathcal{G}$ ist (wie gehabt) problemabhangig

Algorithm 1 Basisalgorithmus der lokalen Suche

Input: Zielfunktion F
Output: Lösungskandidaten A

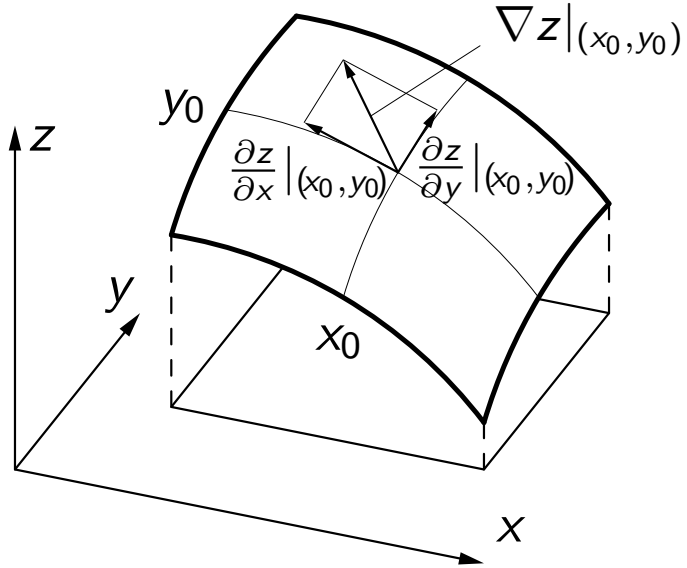
```
 1: t ← 0
 2: A(t) ← erzeuge Lösungskandidat
 3: bewerte A(t) durch F
 4: while Terminierungsbedingung nicht erfüllt {
 5:     B = variiere A(t)
 6:     bewerte B durch F
 7:     t ← t + 1
 8:     if Akz(A(t − 1).F, B.F, t) {          /* Akzeptanzbedingung */
 9:         A(t) ← B
10:     } else {
11:         A(t) ← A(t − 1)
12:     }
13: }
14: return A(t)
```

- *Akz*: untersch. implementiert bei versch. lokalen Suchalgorithmen

Gradientenverfahren

- **Voraussetzung:** $\Omega \subseteq \mathbb{R}^n$ und $f : \Omega \to \mathbb{R}$ ist differenzierbar
- **Gradient:** Differentialoperation, die Vektorfeld erzeugt

$\Rightarrow$ liefert Vektor in Richtung der stärksten Steigung einer Funktion



- Illustration des Gradienten von $z = f(x, y)$ am Punkt(x_0, y_0)

$$\nabla z|_{(x_0,y_0)} = \left(\frac{\partial z}{\partial x}|_{(x_0,y_0)}, \frac{\partial z}{\partial y}|_{(x_0,y_0)} \right)$$

Gradientenverfahren

Idee: von zufälligem Startpunkt, gehe kleine Schritte in Ω jeweils in Richtung des stärksten Anstiegs der Funktion bis (lokales) Maximum erreicht

1. wähle (zufälligen) Startpunkt $\boldsymbol{x}^{(0)} = \left(x_1^{(0)}, \ldots, x_n^{(0)}\right)$
2. bestimme Gradienten am aktuellen Punkt $\boldsymbol{x}^{(t)}$

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

3. gehe kleines Stück in Richtung des Gradienten

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \eta \, \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}^{(t)}\right)$$

   η: Schrittweitenparameter („Lernrate“ in KNNs)
4. wiederhole Schritte 2 und 3 bis Abbruchkriterium erfüllt ist (z.B. best. Anzahl Schritte ausgeführt, aktueller Gradient sehr klein)

Probleme

Wahl des Schrittweitenparameters

- zu kleiner Wert $\Rightarrow$ lange Laufzeit bis Maximum erreicht
- zu großer Wert $\Rightarrow$ Oszillationen, Hin- und Herspringen in Ω
- Lösungen: Momentterm, adaptiver Schrittweitenparameter (siehe Vorlesung „Neuronale Netze“)

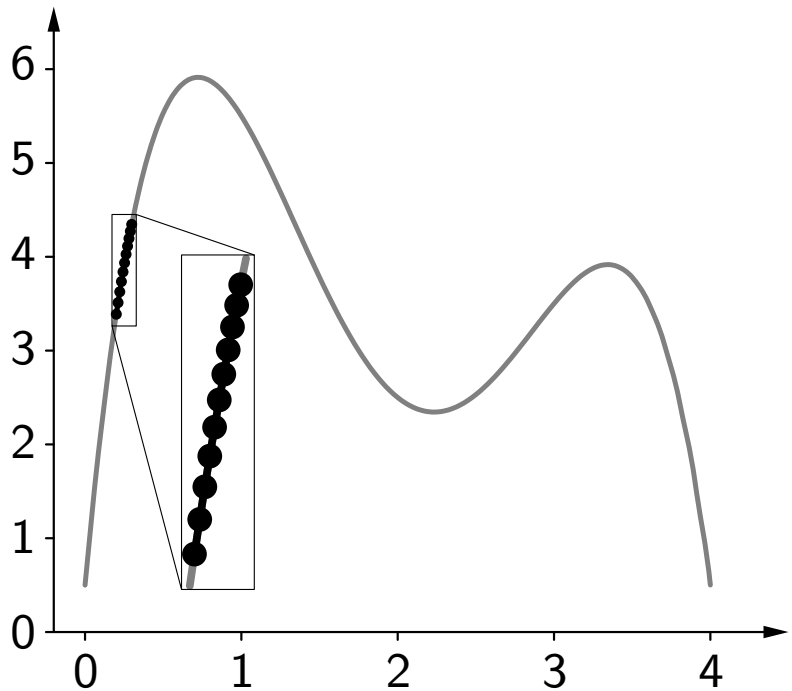
Hängenbleiben in lokalen Maxima

- aufgrund lokaler Steigungsinformationen, vielleicht nur lokales Maximum erreichbar
- Problem kann *nicht* prinzipiell behoben werden
- Chancenverbesserung: mehrfaches Ausführen mit verschiedenen Startwerten

Beispiele

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

t	x_t	$f(x_t)$	$f'(x_t)$	Δx_t
0	0.200	3.388	11.147	0.011
1	0.211	3.510	10.811	0.011
2	0.222	3.626	10.490	0.010
3	0.232	3.734	10.182	0.010
4	0.243	3.836	9.888	0.010
5	0.253	3.932	9.606	0.010
6	0.262	4.023	9.335	0.009
7	0.271	4.109	9.075	0.009
8	0.281	4.191	8.825	0.009
9	0.289	4.267	8.585	0.009
10	0.298	4.340		

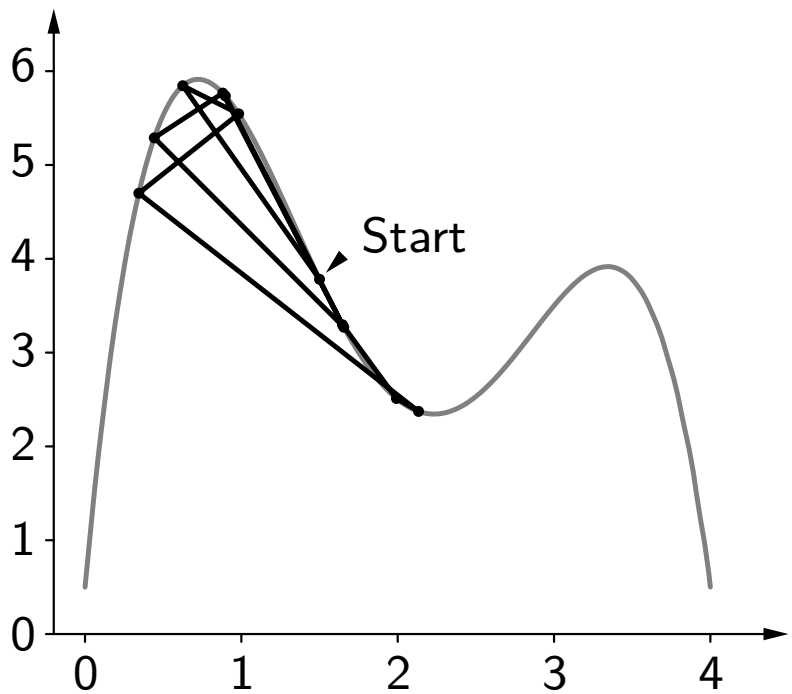


Gradientenaufstieg mit Startwert 0.2 und $\eta = 0.001$

Beispiele

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

t	x_t	$f(x_t)$	$f'(x_t)$	Δx_t
0	1.500	3.781	−3.500	−0.875
1	0.625	5.845	1.431	0.358
2	0.983	5.545	−2.554	−0.639
3	0.344	4.699	7.157	1.789
4	2.134	2.373	−0.567	−0.142
5	1.992	2.511	−1.380	−0.345
6	1.647	3.297	−3.063	−0.766
7	0.881	5.766	−1.753	−0.438
8	0.443	5.289	4.851	1.213
9	1.656	3.269	−3.029	−0.757
10	0.898	5.734		

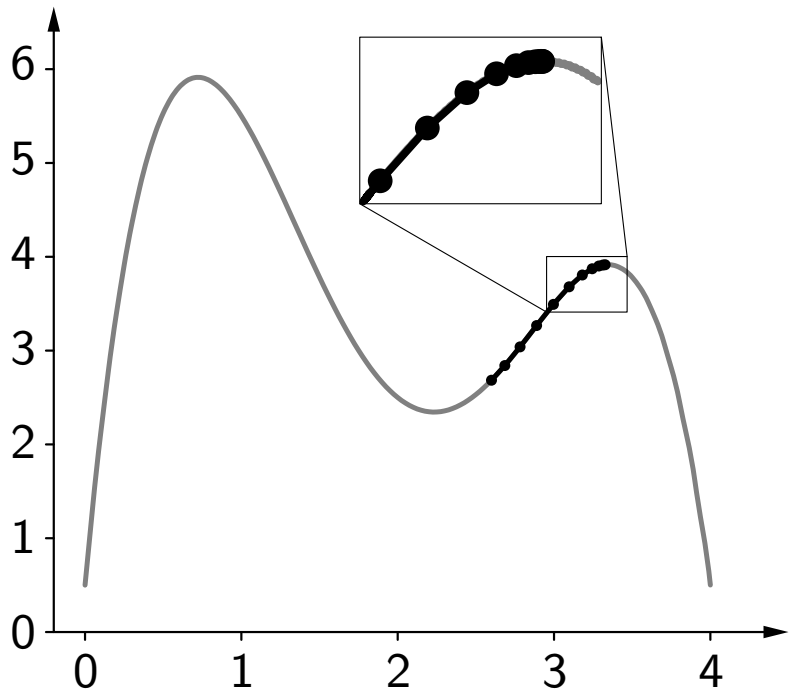


Gradientenaufstieg mit Startwert 1.5 und $\eta = 0.25$

Beispiele

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

t	x_t	$f(x_t)$	$f'(x_t)$	Δx_t
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 2.6 und $\eta = 0.05$

Zufallsaufstieg

Idee: falls f nicht differenzierbar, bestimme Richtung, in der f ansteigt durch Auswerten zufälliger Punkte in Umgebung des aktuellen Punktes

1. wähle (zufälligen) Startpunkt $x_0 \in \Omega$
2. wähle Punkt $x' \in \Omega$ „in der Nähe“ von x_t (z.B. durch zufällige, aber kleine Veränderung von x_t)
3. setze
$$x_{t+1} = \begin{cases} x' & \text{falls } f(x') > f(x_t), \\ x_t & \text{sonst} \end{cases}$$
4. wiederhole Schritte 2 und 3 bis Abbruchkriterium erfüllt

Zufallsaufstieg

Pseudocode für **Akzeptanzbedingung** im **Basisalgorithmus**:

Algorithm 2 Akzeptanzbedingung für Zufallsaufstieg

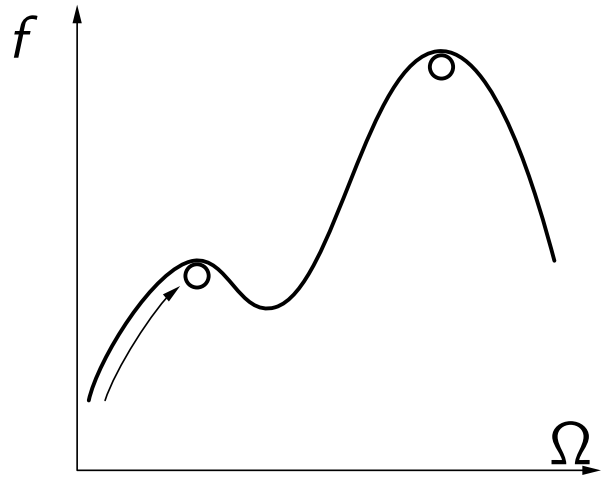
Input: Elterngüte $A.F$, Kindgüte $B.F$, Generation t
Output: **true** oder **false**

1: **return** $B.F \succ A.F$

- **Problem:** Hängenbleiben in lokalen Maxima
- alle folgenden Verfahren versuchen, dieses Problem zu verringern

Simuliertes Ausglühen

- Erweiterung des Zufalls- und Gradientenaufstiegs, die Hängenbleiben vermeidet
- **Idee:** Übergänge von niedrigeren zu höheren (lokalen) Maxima sollen wahrscheinlicher sein als umgekehrt



Prinzip:

- zufällige Varianten der aktuellen Lösung werden erzeugt
- bessere Lösungen werden immer übernommen
- schlechtere Lösungen werden mit bestimmten W'keit übernommen, die abhängt von
  - Qualitätsdifferenz der aktuellen und der neuen Lösung und
  - Temperaturparameter (wird im Laufe der Zeit verringert)

Motivation (Minimierung statt Maximierung)

- physikalische Minimierung der (Atomgitter-)Energie, wenn erhitztes Stück Metall langsam abgekühlt wird
- Prozess nennt sich **Ausglühen** (engl.: *annealing*)
- Zweck: Metall weicher machen durch Aufheben innerer Spannungen und Instabilitäten $\Rightarrow$ leichtere Bearbeitung

Alternative Motivation: (ebenfalls Minimierung)

- Kugel rollt auf unregelmäßig gewellter Oberfläche
- zu minimierende Funktion ist potentielle Energie der Kugel
- anfangs: gewisse kinetische Energie um Anstiege zu überwinden
- Reibung: Energie sinkt $\Rightarrow$ schließlich Ruhe in einem Tal

Achtung: keine Garantie, dass globales Optimum gefunden wird

Simuliertes Ausglühen

1. wähle (zufälligen) Startpunkt $x_0 \in \Omega$
2. wähle Punkt $x' \in \Omega$ „in der Nähe“ des aktuellen Punktes x_t (z.B. durch zufällige, aber kleine Veränderung von x_t)
3. setze

$$x_{t+1} = \begin{cases} x' & \text{falls } f(x') \geq f(x_t), \\ x' \text{ mit Wahrscheinlichkeit } p = e^{-\frac{\Delta f}{kT}} & \\ x_t \text{ mit Wahrscheinlichkeit } 1 - p & \text{sonst.} \end{cases}$$

$\Delta f = f(x_t) - f(x')$	Qualitätsverringerung der Lösung
$k = \Delta f_{\max}$	(Schätzung des) Umfangs der Funktionswerte
T	Temperaturparameter (sinkt im Laufe der Zeit)

4. wiederhole Schritte 2 und 3 bis Abbruchkriterium erfüllt

für kleine T Verfahren geht nahezu in Zufallsaufstieg über

Simuliertes Ausglühen

Algorithm 3 Akzeptanzbedingung für Simuliertes Ausglühen

Input: Elterngüte $A.F$, Kindgüte $B.F$, Generation t

Output: **true** oder **false**

```
1:  if B.F ≻ A.F {
2:      return true
3:  } else {
4:      u ← wähle zufällig aus U([0, 1]) /* Zufallszahl zw. 0 und 1 */
5:      if u ≤ exp(−(A.F−B.F)/(kT_{t−1})) {
6:          return true
7:      } else {
8:          return false
9:      }
10: }
```

Schwellwertakzeptanz

Idee: ähnlich wie beim simulierten Ausglühen auch Akzeptanz schlechterer Lösungen, allerdings mit oberer Schranke für Verschlechterung

1. wähle (zufälligen) Startpunkt $x_0 \in \Omega$
2. wähle Punkt $x' \in \Omega$ „in der Nähe“ des aktuellen Punktes x_t (z.B. durch zufällige, aber kleine Veränderung von x_t)
3. setze

$$x_{t+1} = \begin{cases} x' & \text{falls } f(x') \geq f(x_t) - \theta, \\ x_t & \text{sonst.} \end{cases}$$

   θ Schwellenwert für das Akzeptieren schlechterer Lösungen (wird im Laufe der Zeit (langsam) gesenkt) ($\theta = 0$ entspricht normalem Zufallsaufstieg)
4. wiederhole Schritte 2 und 3 bis Abbruchkriterium erfüllt

Schwellwertakzeptanz

Algorithm 4 Akzeptanzbedingung für Schwellwertakzeptanz

Input: Elterngüte $A.F$, Kindgüte $B.F$, Generation t

Output: **true** oder **false**

1: **if** $B.F \succ A.F$ oder $A.F - B.F \leq \theta$ {
2: **return true**
3: } **else** {
4: **return false**
5: }

Sintflut-Algorithmus

Idee: ähnlich wie beim simulierten Ausglühen auch Akzeptanz schlechterer Lösungen, allerdings mit unterer Schranke

1. wähle (zufälligen) Startpunkt $x_0 \in \Omega$
2. wähle Punkt $x' \in \Omega$ „in der Nähe“ des aktuellen Punktes x_t (z.B. durch zufällige, aber kleine Veränderung von x_t)
3. setze

$$x_{t+1} = \begin{cases} x' & \text{falls } f(x') \geq \theta_0 + t \cdot \eta, \\ x_t & \text{sonst.} \end{cases}$$

   θ_0 untere Schranke für Lösungskandidaten bei $t = 0$ (anfänglicher „Wasserstand“)

   η Schrittweitenparameter („Geschwindigkeit des Regens“)
4. wiederhole Schritte 2 und 3 bis Abbruchkriterium erfüllt

Sintflut-Algorithmus

Algorithm 5 Akzeptanzbedingung für Sintflut-Algorithmus

Input: Elterngüte $A.F$, Kindgüte $B.F$, Generation t

Output: **true** oder **false**

```
1: if B.F ≻ θ_0 + η · t {
2:     return true
3: } else {
4:     return false
5: }
```

Rekordorientiertes Wandern

Idee: ähnlich wie beim Sintflut-Algorithmus wird steigender Wasserpegel genutzt, jedoch an Güte des bisher besten gefundenen Individuums gekoppelt

- mögliche Verschlechterung: stets in Relation zum besten gefundenen Individuum
- nur bei Verbesserung: aktuelles Individuum von Bedeutung
- wie bei Schwellwertakzeptanz: monoton fallende Folge reeller Zahlen regelt Übernahme eines schlechten Individuums

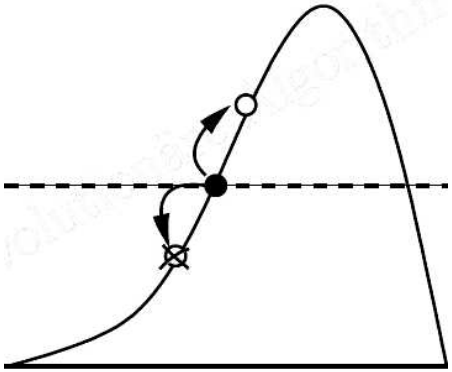
Rekordorientiertes Wandern

Algorithm 6 Akzeptanzbedingung für Rekordorientiertes Wandern

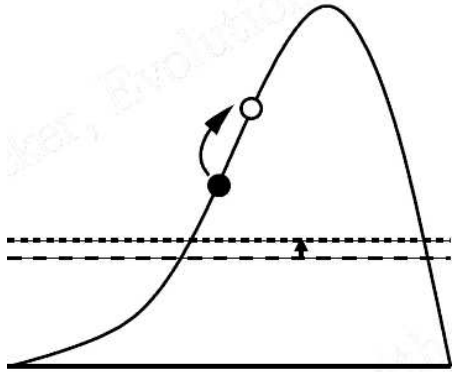
Input: Elterngüte $A.F$, Kindgüte $B.F$, t, beste gefundene Güte F_{best}
Output: **true** oder **false**, F_{best}

```
1: if B.F ≻ F_best {
2:     F_best ← B.F
3:     return true, F_best
4: } else {
5:     if B.F − F_best < T_t {
6:         return true, F_best
7:     }
8: }
9: return false, F_best
```

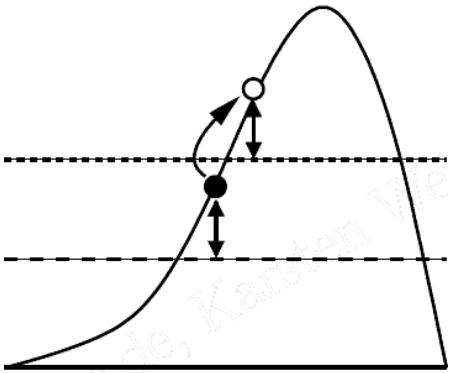
Vergleich lokaler Suchalgorithmen



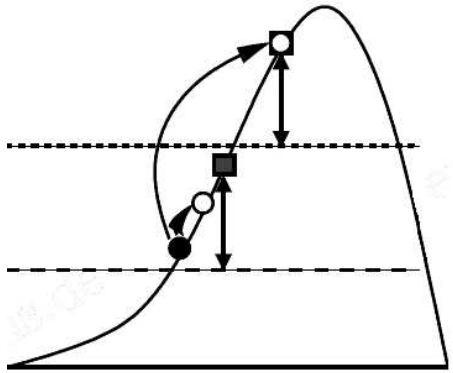
Hillclimbing



Sintflut-Algorithmus



Schwellwertakzeptanz



Rekordorientiertes Wandern

Übersicht

Beispiel: Problem des Handlungsreisenden

engl. Traveling Salesman Problem (TSP)

- **geg.:** Menge von n Städten (als Punkte in einer Ebene) mit Abständen/Kosten der Wege zwischen Städten
- **ges.:** 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:** TSP ist NP-vollständig, d.h. man kennt keinen Algorithmus, der TSP in polynomialer Zeit löst
- **daher:** für großes n ist in annehmbarer Zeit nur Näherungslösung berechenbar (beste Lösung wird vielleicht gefunden)
- **hier:** Lösung mit simuliertem Ausglühen und Zufallsaufstieg

Beispiel: Problem des Handlungsreisenden

1. bringe Städte in zufällige Reihenfolge (zufällige Rundreise)
2. wähle zufällig zweimal 2 Städte, die in aktueller Rundreise aufeinander folgen (alle vier Städte verschieden), trenne Rundreise zwischen Städten jedes Paares und drehe dazwischenliegenden Teil um
3. wenn neue Rundreise besser (kürzer, billiger) als alte, ersetze alte Rundreise durch neue, sonst mit Wahrscheinlichkeit $p = e^{-\frac{\Delta Q}{kT}}$

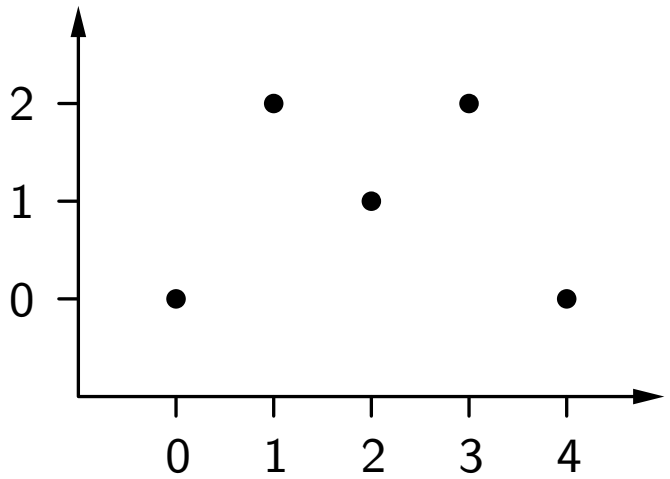
   ΔQ Qualitätsunterschied zwischen alter und neuer Rundreise

   k Spannweite der Rundreisequalitäten (ggf. schätzen, z.B. $k_t = \frac{t+1}{t} \max_{i=1}^{t} \Delta Q_i$, wobei ΔQ_i Qualitätsunterschied im i-ten Schritt und t der aktuelle Schritt)

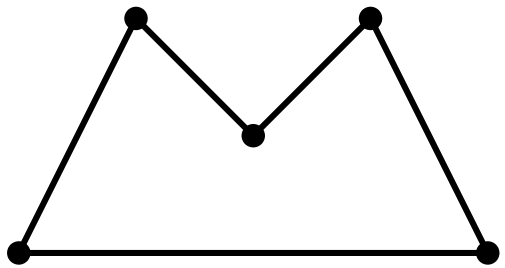
   T Temperaturparameter (mit Zeit fallend, z.B. $T = \frac{1}{t}$)
4. wiederhole die Schritte 2 und 3 bis Abbruchkriterium erfüllt

Beispiel: Problem des Handlungsreisenden

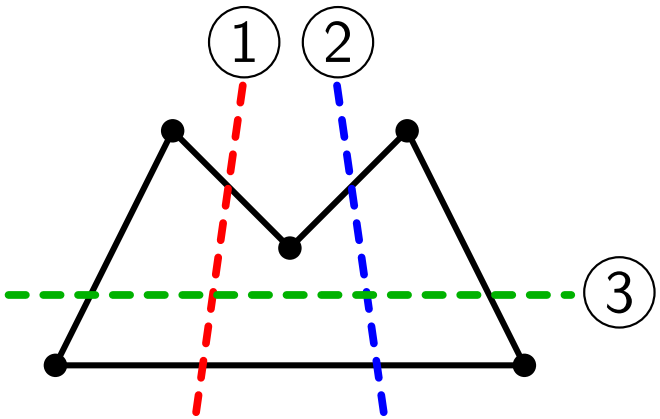
purer Zufallsaufstieg kann in lokalem Minimum hängenbleiben:



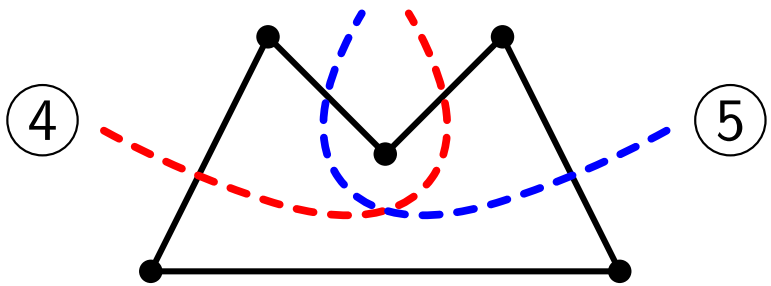
anfängliche Rundreise



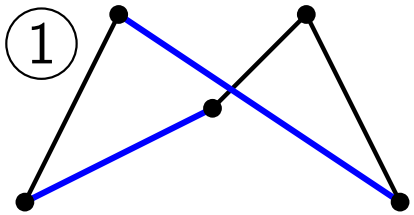
Länge: $2\sqrt{2} + 2\sqrt{5} + 4 \approx 11.30$

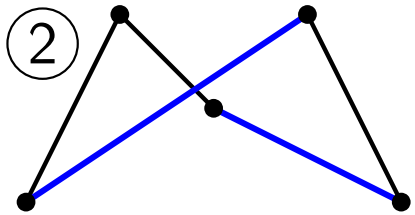


mögliche Teilungen der Rundreise

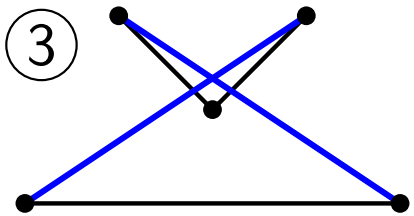


Beispiel: Problem des Handlungsreisenden

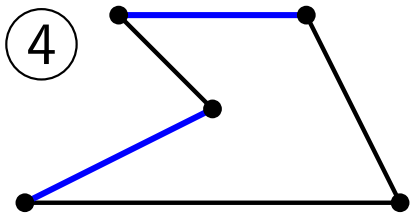


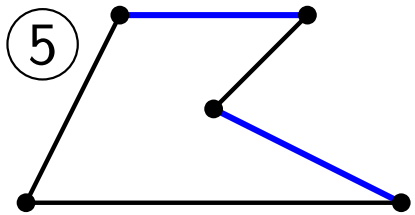


$\sqrt{2} + 3\sqrt{5} + \sqrt{13} \approx 11.73$



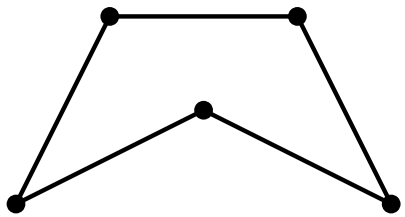
$\sqrt{2} + 2\sqrt{13} + 4 \approx 14.04$





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

global. Optimum:



$4\sqrt{5} + 2 \approx 10.94$

Beispiel: Problem des Handlungsreisenden

- alle Modifikationen der Anfangsrundreise führen zu schlechteren Rundreisen $\Rightarrow$ globales Optimum (von dieser Rundreise) mit reinen Zufallsaufstieg nicht erreichbar
- simulierten Ausglühen akzeptiert dagegen mitunter auch schlechtere Lösungen $\Rightarrow$ Weg zum globalen Optimum (*aber*: keine Garantie, dass dies geschieht!)
- **beachte:** es kann von erlaubten Operationen abhängen, ob Suche in lokalem Optimum hängenbleiben kann:
  - mit weiterer Operation (Ändern der Position einer Stadt in Rundreise bzw. Entfernen von aktueller Position und Einfügen an einer anderen) kein Hängenbleiben im betrachteten Beispiel
  - auch für diese Operationenmenge: Konstruktion eines Beispiels, das in lokalem Minimum hängenbleibt

Übersicht

Tabu-Suche

- lokales Suchverfahren, dass bei Erzeugung neuer Individuen Geschichte berücksichtigt
- *Tabu-Listen* verhindern Rückkehr zu gerade betrachteten Lösungskandidaten

- Tabu-Liste = FIFO-Warteschlange fester Länge
  - Einträge sind ganze Lösungskandidaten oder Aspekte
  - Mutationen dürfen Tabu-Einträge nicht erzeugen
  - meist: FIFO-Liste mit erstrebenswerten Eigenschaften können Tabu brechen
  - auch möglich: neue beste Gesamtgüte bricht Tabu

Beispiel

Graphfärbung mit k Farben, sodass es keine Kante zwischen gleich gefärbten Knoten gibt.

- $\Omega = \{1, \ldots, k\}^n$
- minimiert wird $f(x) = \sum_{(v_i, v_j) \in E} \begin{cases} 1 & \text{falls } x_i = x_j \\ 0 & \text{sonst} \end{cases}$
- Mutation färbt v_i von c auf $d \Rightarrow$ Tabu-Eintrag (i, c)

Algorithm 7 Tabu-Suche

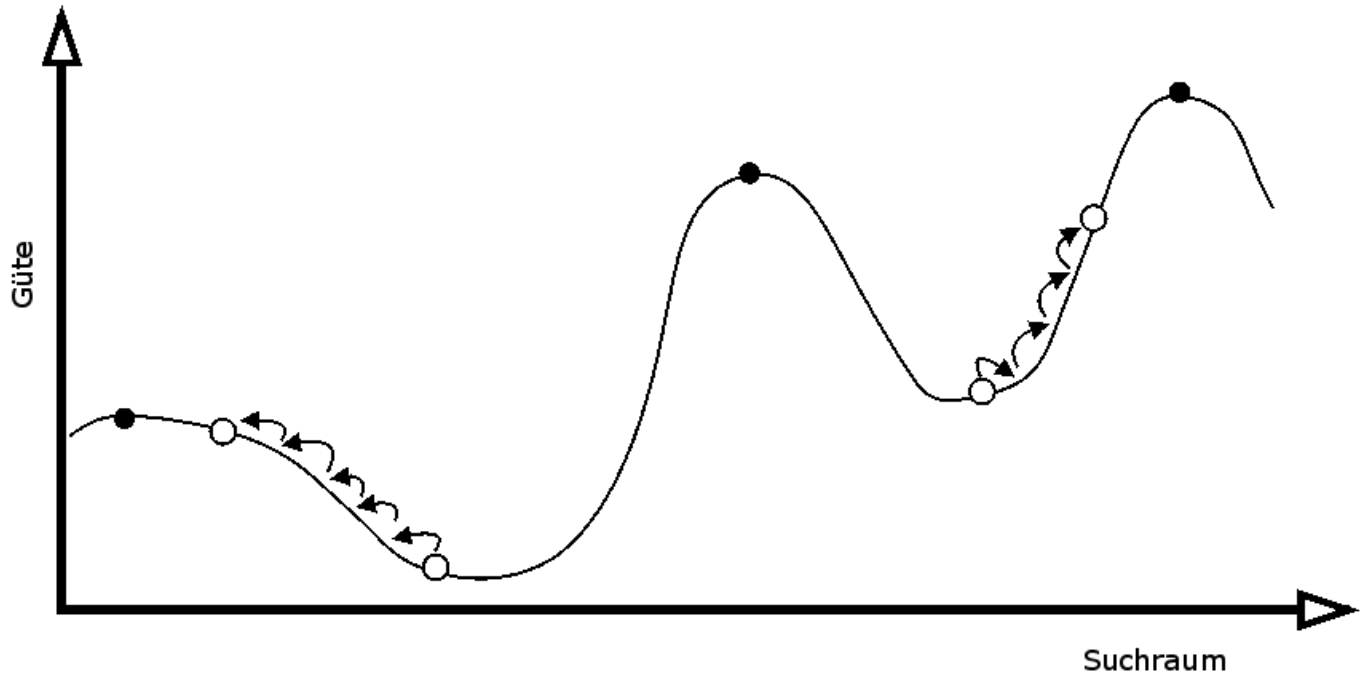
Input: Zielfunktion F
Output: bestes Individuum A_{best}

1: $t \leftarrow 0$
2: $A(t) \leftarrow$ erzeuge zufälligen Lösungskandidaten
3: bewerte $A(t)$ durch F
4: $A_{\text{best}} \leftarrow A(t)$
5: initalisiere Tabu-Liste
6: **while** Terminierungsbedingung nicht erfüllt {
7: $\quad P \leftarrow \emptyset$
8: $\quad$ **while** $|P| < \lambda$ {
9: $\quad\quad B \leftarrow$ mutiere $A(t)$
10: $\quad\quad$ bewerte B durch F
11: $\quad\quad$ **if** $(A(t), B) \notin$ Tabu-Liste oder $B.F \succ A_{\text{best}}.F$ {
12: $\quad\quad\quad P \leftarrow P \cup \{B\}$
13: $\quad\quad$ }
14: $\quad$ }
15: $\quad t \leftarrow t + 1$
16: $\quad A(t) \leftarrow$ bestes Individuum aus P
17: $\quad$ **if** $A(t).F \succ A_{\text{best}}.F$ {
18: $\quad\quad A_{\text{best}} \leftarrow A(t)$
19: $\quad$ }
20: $\quad$ Tabu-Liste $\leftarrow$ aktualisiere durch $(A(t-1), A(t))$
21: }
22: **return** A_{best}

Memetische Algorithmen

- einerseits populationsbasierte Algorithmen
  - Vorteil: breites Durchforsten des Suchraums
  - Nachteil: langsam
- andererseits lokale Suche
  - Vorteil: schnelle Optimierung
  - Nachteil: anfällig für lokale Optima
- **Memetische Algorithmen:** Kombination beider Techniken
- Herkunft des Namens (Richard Dawkins): „Meme“ sind Verhaltenselemente, die individuell erworben werden können (im Gegensatz zu Genen)
- Vorgehen: jede neue Individuum wird sofort lokal optimiert
  - nur wenige Schritte oder
  - bis in lokales Optimum

Beispiel: SAGA



„Simulated Annealing Genetic Algorithm“

Memetische Algorithmen

Algorithm 8 Memetischer Algorithmus

Input: Bewertungsfunktion F
Output: bestes Individuum

1: $t \leftarrow 0$
2: $P(t) \leftarrow$ initalisiere Population der Größe μ
3: $P(t) \leftarrow$ LOKALE-SUCHE(F) für jedes Individuum in $P(t)$
4: bewerte $P(t)$ durch F
5: **while** Terminierungsbedingung nicht erfüllt $\{$
6: $\quad E \leftarrow$ selektiere Eltern für λ Nachkommen aus $P(t)$
7: $\quad P' \leftarrow$ erzeuge Nachkommen durch Rekombination aus E
8: $\quad P'' \leftarrow$ mutiere Individuen in P'
9: $\quad P''' \leftarrow$ LOKALE-SUCHE(F) für jedes Individuum in P''
10: $\quad$ bewerte P''' durch F
11: $\quad t \leftarrow t + 1$
12: $\quad P(t) \leftarrow$ Umweltselektion auf P'''
13: $\}$
14: **return** bestes Individuum aus $P(t)$

Eigenschaften

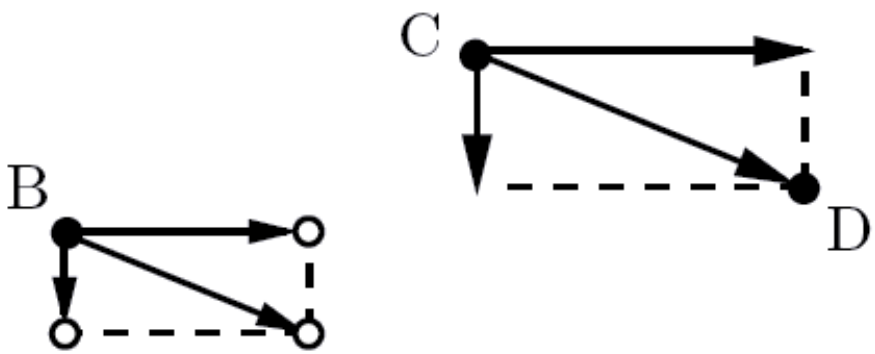
- oft stark beschleunigte Optimierung
- aber: Suchdynamik kann entscheidend eingeschränkt werden
  - Mutation bleibt oft in breiten lokalen Optima
  - Rekombination hat beschränkte Ausgangssituation
  - Teile des Suchraums sind evtl. unerreichbar

- Arbeitsweise entspricht der Lamarckschen Evolution

Differentialevolution

Idee

- keine Anpassung der Schrittweite in $A.S$
- sondern: Relation der Individuen in der Population wird als Grundlage für Schrittweite herangezogen



Differentialevolution

Algorithm 9 DE-Operator

Input: Individuen A, B, C, D

Output: optimiertes Individuum A'

1: index $\leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, l\})$
2: **for each** $i \in \{1, \ldots, l\}$ {
3: $u \leftarrow$ wähle Zufallszahl gemäß $U([0, 1))$
4: **if** $u \leq \tau$ **or** $i =$ index {
5: $A'.G_i \leftarrow B'.G_i + (C.G_i - D.G_i) \cdot \alpha$
6: } **else** {
7: $A'.G_i \leftarrow A.G_i$
8: }
9: }
10: **return** A'

Details

- DE-Operator: Kombination aus Rekombination und Mutation
- Selektion: ein neues Individuum ersetzt Elternindividuum genau dann, wenn es besser ist

Paramter	Wertebereich
Populationsgröße μ	10–100, $10 \cdot n$
Wichtung der Rekombination τ	0.7–0.9
Skalierungsfaktor α	0.5–1.0

Algorithm 10 Differentialevolution

Input: Bewertungsfunktion F
Output: bestes Individuum aus $P(t)$

1: $t \leftarrow 0$
2: $P(t) \leftarrow$ erzeuge Population der Größe μ
3: bewerte $P(t)$ durch F
4: **while** Terminierungsbedingung nicht erfüllt $\{$
5: $\quad P(t+1) \leftarrow \emptyset$
6: $\quad$ **for** $i \leftarrow 1, \ldots, \mu$ $\{$
7: $\quad\quad$ **do** $\{$
8: $\quad\quad\quad A, B, C, D \leftarrow$ selektiere Eltern uniform zufällig aus $P(t)$
9: $\quad\quad$ $\}$ **while** A, B, C, D paarweise verschieden
10: $\quad\quad A' \leftarrow$ DE-Operator(A, B, C, D)
11: $\quad\quad$ bewerte A' durch F
12: $\quad\quad$ **if** $F(A') \succ F(A)$ $\{$
13: $\quad\quad\quad P(t+1) \leftarrow P(t+1) \cup \{A'\}$
14: $\quad\quad$ $\}$ **else** $\{$
15: $\quad\quad\quad P(t+1) \leftarrow P(t+1) \cup \{A\}$
16: $\quad\quad$ $\}$
17: $\quad$ $\}$
18: $\}$
19: $t \leftarrow t+1$
20: **return** bestes Individuum aus $P(t)$

Scatter Search

Zutaten

- Population mit Lösungskandidaten
- Variationsoperatoren
- Selektionsdruck
- zusätzlich: lokale Suche

Aber...

- ein deterministisches Verfahren!
- weiträumige Erforschung des Suchraums:
  - breite Initialisierung
  - systematische Erzeugung neuer Individuen

Ablauf

iterierter Ablauf durch zwei Phasen

1. Erzeugen neuer Individuen und Auswahl derjenigen, die die größte Vielfalt garantieren
2. Rekombination aller „Paarungen“ der gewählten Individuen
3. Selektion der Besten und Iteration bis sich nichts mehr ändert

Beispiel

- reellwertige Problemräume mit $\mathcal{G} = \Omega = \mathbb{R}^n$

Phase 1

1. Diversitätsgenerator erzeugt μ Individuen in P
   zum Beispiel:
   - pro Suchraumdimension Wertebereich in vier Partitionen
   - Häufigkeit der erzeugten Individuen pro Partition merken
   - Partition invers proportional auswählen

2. separat: Population der besten α Individuen wird um die β Individuen aus P erweitert, die $\min_{B \in P_{\text{best}}} d(A.G, B.G)$ maximieren ($A \in P$)

Phase 2

1. Teilmengengenerator wählt Individuen aus Menge der Besten aus
   Beispiel: (hier sehr einfach) alle möglichen Paare
2. Kombinationsoperator anwenden
   (Beispiel: arithmetischer Crossover mit $U\left(\left[-\frac{1}{2}, \frac{3}{2}\right]\right)$)
3. lokal optimieren
4. wenn alle erzeugt, dann Wahl der $\alpha + \beta$ Besten
5. iterieren solange sich Menge der Besten ändert
6. α Besten für nächste Phase 1

Algorithm 11 Scatter Search

Input: Bewertungsfunktion F

$P_{\text{best}} = \emptyset$; $P = \emptyset$
for $t \leftarrow 1, \ldots,$ maxIter {
  while $|P| < \mu$ {
    $A \leftarrow$ erzeuge ein Individuum mit Diversitätsgenerator
    $A \leftarrow$ Lokale-Suche(F) angewandt auf A; bewerte A durch F
    if $A \notin P \cup P_{\text{best}}$ {
      $P \leftarrow P \cup \{A\}$
    }
  }
  if $t = 1$ {
    $P_{\text{best}} \leftarrow$ selektiere α Individuen aus P mit Besten-Selektion
    $P \leftarrow$ streiche Individuen aus P_{best} in P
  }
  for $k \leftarrow 1, \ldots, \beta$ {
    $A \leftarrow$ dasjenige Individuum aus P, das $\min_{B \in P_{\text{best}}} d(A.G, B.G)$ maximiert
    $P \leftarrow$ streiche Individuum A in P
    $P_{\text{best}} \leftarrow P_{\text{best}} \cup \{A\}$
  }
  do {
    $P' \leftarrow \emptyset$; Mengen $\leftarrow$ erzeuge Teilmengen von P_{best} durch Teilmengenoperator
    for each $M \in$ Mengen {
      $A \leftarrow$ wende Kombinationsoperator auf M an
      $A \leftarrow$ Lokale-Suche(F) angewandt auf A; bewerte A durch F
      if $A \notin P_{\text{best}} \cup P'$ {
        $P' \leftarrow P' \cup \{A\}$
      }
    }
    $P_{\text{best}} \leftarrow$ selektiere $\alpha + \beta$ Ind. aus $P_{\text{best}} \cup P'$ mit Besten-Selektion
  } **while** P_{best} hat sich nicht geändert
  $P_{\text{best}} \leftarrow$ selektiere α Individuen aus P mit Besten-Selektion
}
return bestes Individuum in P_{best}

Scatter Search

Empfohlene Parameter

Paramter	Wertebereich
Populationsgröße μ	50–150
Anzahl der besten Individuen α	5–20
Erweiterung der besten Individuen β	5–20

Kulturelle Algorithmen

Motivation:

- weitere Informationsspeicher zusätzlich zur genetischen Ebene
- Kultur bestimmt auf Werten gestütztes Verhalten
- Ebene der Kultur wird in EAs eingeführt

kollektives kulturelles Wissen:

- Speicher: Überzeugungsraum (engl. *belief space*)
- wird durch die besten Individuen einer Generation modifiziert
- situatives und normatives Wissen

Algorithm 12 Kultureller Algorithmus

Input: Bewertungsfunktion F

Output: bestes Individuum in $P(t)$

1: $t \leftarrow 0$
2: $P(t) \leftarrow$ initalisiere Population
3: $\mathcal{BS}(t) \leftarrow$ initialisiere Überzeugungsraum
4: bewerte $P(t)$ durch F
5: **while** Terminierungsbedingung nicht erfüllt $\{$
6: $\quad P' \leftarrow$ bestimme wichtige Individuen aus $P(t)$
7: $\quad \mathcal{BS}(t+1) \leftarrow \mathcal{BS}(t)$ wird durch P' angepasst
8: $\quad P'' \leftarrow$ erzeuge Nachkommen von $P(t)$ auf Basis von $\mathcal{BS}(t+1)$
9: $\quad$ bewerte $P(t)$ durch F
10: $\quad t \leftarrow t+1$
11: $\quad P(t) \leftarrow$ Selektion aus $P' \cup P(t-1)$
12: $\}$
13: **return** bestes Individuum aus $P(t)$

Kulturelle Algorithmen

Situatives Wissen

- für $\Omega = \mathbb{R}^n$: letzten beiden besten Individuen
- Mutation soll sich ggf. in diese Richtung orientieren

Normatives Wissen

- für $\Omega = \mathbb{R}^n$: Ober- und Untergrenze pro Dimension
- größter/kleinster Wert der 20% besten Individuen der letzten Generation ermitteln
- immer: Vergrößerung akzeptieren
- nur bei besserem Gütewert des entsprechenden „Grenzindividuums“: Verkleinerung akzeptieren

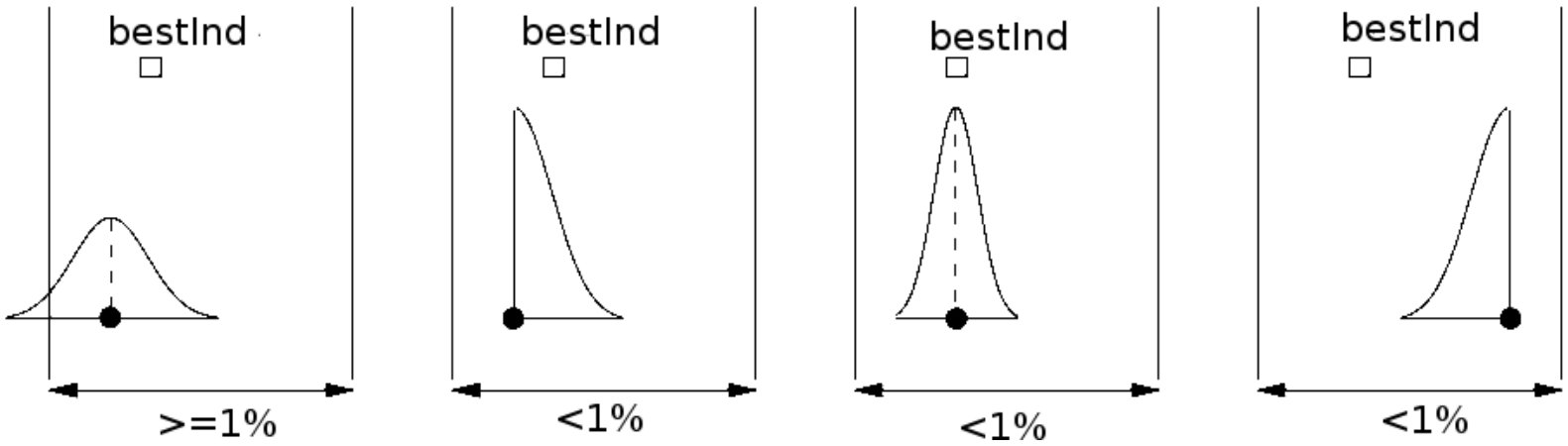
Kulturelle Algorithmen

Stabile Suchraumdimensionen

- wenn Bereich auf $< 1\%$ eingeschränkt und
- letzte beste Individuen innerhalb

Mutation

- falls stabil: am besten Individuum orientieren
- sonst: selbstangepasste Schrittweite



Kulturelle Algorithmen

Algorithm 13 KA-Mutation

Input: Individuum A
Output: Individuum B

$u' \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0,1)$
for each $i \in \{1,\ldots,l\}$ {
  $u_i'' \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0,1)$
  $B.S_i \leftarrow A.S_i \cdot \exp\left(\frac{1}{\sqrt{2l}} \cdot u' + \frac{1}{\sqrt{2\sqrt{l}}} \cdot u_i''\right)$
  $u \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0, B.S_i)$
  if $\mathcal{BS}$ ist stabil für Dimension i {
    switch
    case $A.G_i < \text{BestInd.G}_i : B.G_i \leftarrow A.G_i + |u|$
    case $A.G_i > \text{BestInd.G}_i : B.G_i \leftarrow A.G_i - |u|$
    case $A.G_i = \text{BestInd.G}_i : B.G_i \leftarrow A.G_i + \frac{u}{2}$
  } **else** {
    $B.G_i \leftarrow A.G_i + u$
    $B.G_i \leftarrow \max\{\text{ug}_i, \min\{\text{og}_i, B.G_i\}\}$
  }
}
return B

Literatur zur Lehrveranstaltung

Weicker, K. (2007).
Evolutionäre Algorithmen.
Teubner Verlag, Stuttgart, Germany, 2nd edition.