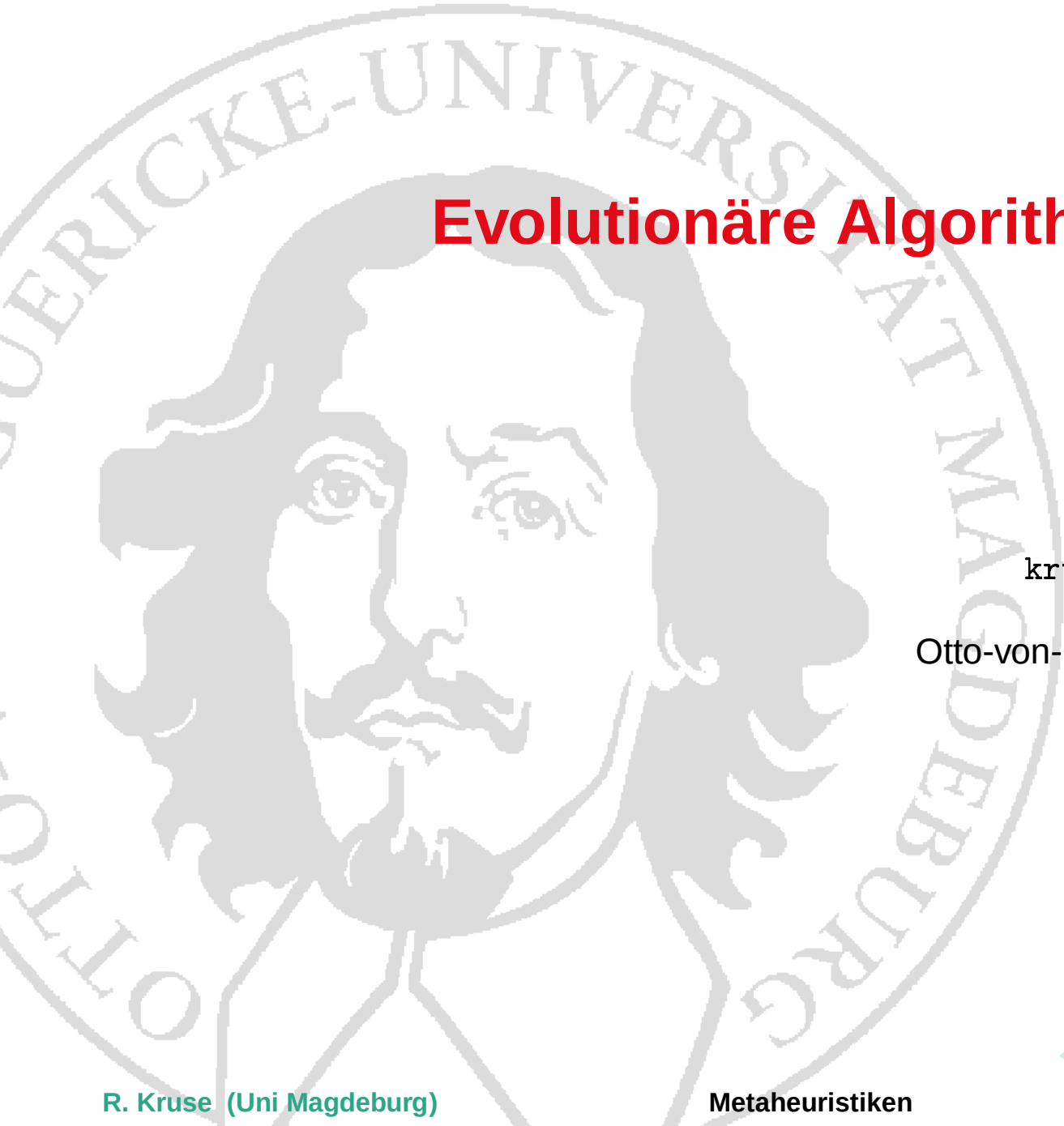


Evolutionäre Algorithmen - Kapitel 15

Metaheuristiken

Prof. Dr. Rudolf Kruse

kruse@iws.cs.uni-magdeburg.de
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

9. Juni 2008

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Was sind Metaheuristiken?

- Problemspezifische Heuristiken
  - nur auf bestimmtes Optimierungsproblem anwendbar
- Metaheuristiken
  - ein Algorithmus zur näherungsweisen Lösung eines kombinatorischen Optimierungsproblems
  - definieren abstrakte Folge von Schritten, die auf beliebige Problemstellung anwendbar sind
- einzelnen Schritte
  - müssen allerdings problemspezifisch implementiert werde

Einsatz von Metaheuristiken

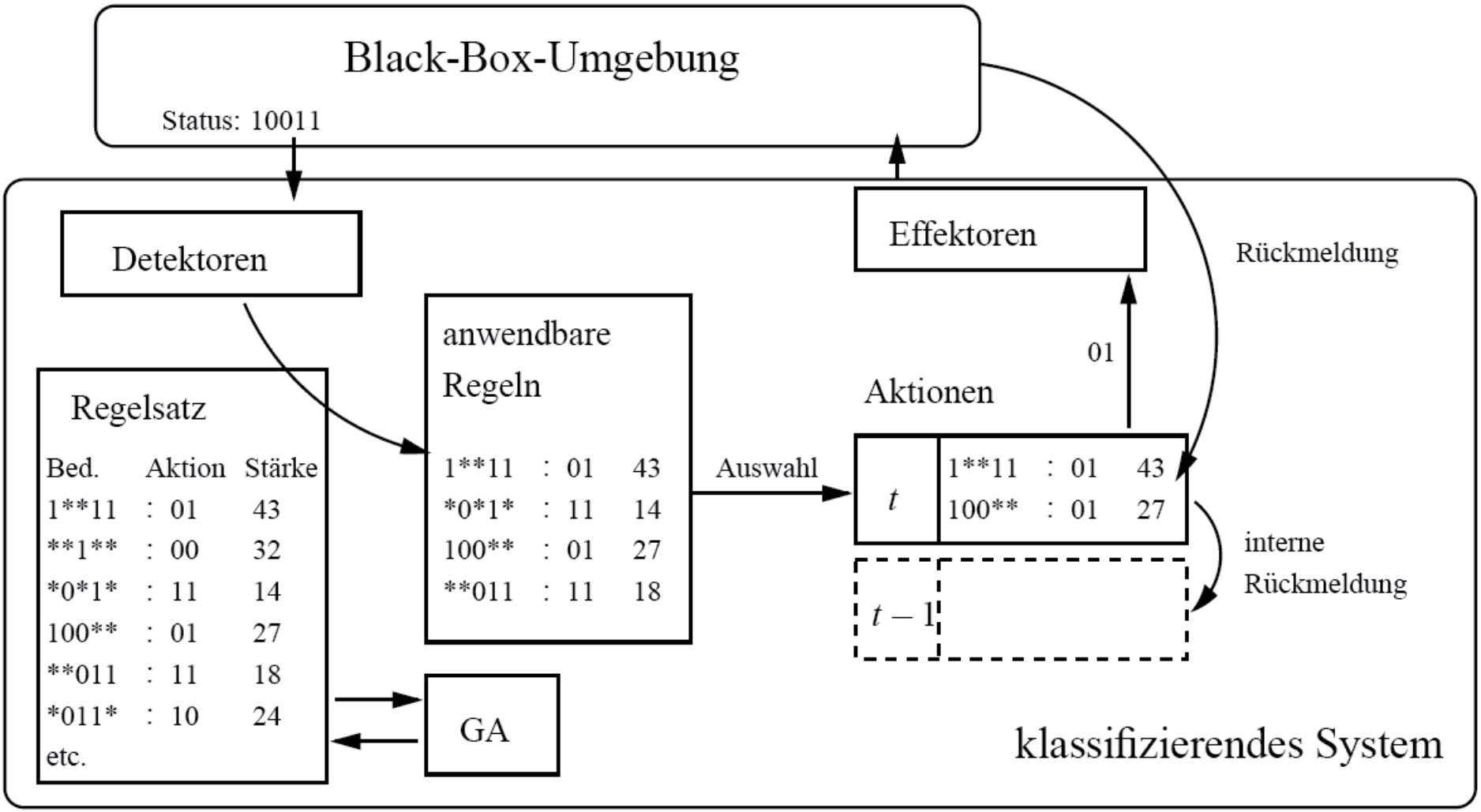
- Einsatz für Probleme,
  - wo kein effizienterer Lösungsalgorithmus bekannt (z.B. kombinatorische Optimierungsprobleme)
- Finden einer optimalen Lösung in der Regel nicht garantiert
- gute Lösungen können beliebig schlecht zu einer Optimallösung sein
- Erfolg und Laufzeit hängen ab von
  - Problemdefinition
  - Implementierung der einzelnen Schritte
- Genetische Algorithmen, schwarmbasierte Optimierungsverfahren sind somit auch Metaheuristiken

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Klassifizierende Systeme - Situation

- Technisches System soll geregelt werden
- Detektoren beobachten Zustand, Effektoren beeinflussen Zustand
- gelegentliche kommt Rückmeldung, wie gut System geregelt wird



Klassifizierende Systeme

Definition: Regeln

- Regel $\text{reg} = (\text{reg.b}, \text{reg.a}, \text{reg.s}) \in \text{Bedingung} \times \text{Aktion} \times \mathbb{R}$ mit
  - Bedingung $\text{reg.b} \in \text{Bedingung} = \{0, 1, *\}^l$
  - Aktion $\text{reg.a} \in \text{Aktion} = \{0, 1\}^m$
  - Stärke $\text{reg.s} \in \mathbb{R}$
- Regelsatz: $\text{regeln} \subset \text{Bedingung} \times \text{Aktion} \times \mathbb{R}$

Definition: Anwendbare Regeln

- Regel aus regeln ist anwendbar unter Statusmeldung $v \in \text{Status} = \{0, 1\}^l$, wenn die Bedingung passt.

$$\text{aktiv}(v) = \left\{ \text{reg} \in \text{regeln} \middle| \bigwedge_{i=1}^{l} (\text{reg.b}_i \neq * \Rightarrow \text{reg.b}_i) = v_i \right\}$$

Klassifizierende Systeme

Definition: Auswahlwahrscheinlichkeit

- aktive Regeln aktiv(v) liegen vor
- mögliche Reaktionen:
  aktion(v) $= \{\text{reg.a} | \text{reg} \in \text{aktiv}(v)\}$
- anwendbare Regeln für $x \in$ aktion(v):
  aktiv$'(v,x) = \{\text{reg} \in \text{aktiv} | \text{reg.a} = x\}$
- fitnessproportionale Auswahlwahrscheinlichkeit:

$$P[x|v] = \frac{\sum_{\text{reg} \in \text{aktiv}'(v,x)} \text{reg.s}}{\sum_{\text{reg} \in \text{aktiv}(v)} \text{reg.s}}$$

Klassifizierende Systeme

Definition: Modifikation der Stärke

- ausgeführte Regeln zur Zeit t : $\text{ausgef}^{(t)}$
- Rückmeldung vom System: $\text{rück} \in \mathbb{R}$
- Modifikation von $\text{reg} \in \text{ausgef}^{(t)}$:

$$\text{reg.s} \leftarrow (1-\alpha) \cdot \text{reg.s} + \alpha \cdot \frac{\text{rück}}{\#\text{ausgef}^{(t)}}$$

- Modifikation von $\text{reg} \in \text{aktiv}(v) \setminus \text{ausgef}^{(t)}$:

$$\text{reg.s} \leftarrow (1-\tau) \cdot \text{reg.s}$$

- Modifikation der letzten Regeln $\text{reg} \in \text{ausgef}^{(t-1)}$:

$$\text{reg.s} \leftarrow \text{reg.s} + \alpha \cdot \beta \cdot \frac{\sum_{\text{reg}' \in \text{ausgef}^{(t)}} \text{reg}'.\text{s}}{\#\text{ausgef}^{(t-1)}}$$

Klassifizierende Systeme

Rolle des Genetischen Algorithmus (GA)

- GA erzeugt neue Regeln im Regelsatz
- steady-state (nur zwei neue Individuen)
- mehrere Lernschritte wechseln mit einer „GA-Generation“
- Auswahl proportional zu Stärkewerten
- Crossover und Mutation, neue Stärke als Durchschnitt der elterlichen Stärke
- proportional zur inversen Stärke gewählter Individuen werden ersetzt

Paramter	Wertebereich
Populationsgröße	400-5000
Lernrate α	0.2
Dämpfungsfaktor β	0.71
Straffaktor τ	0.1

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Tabu-Suche

Grundlegendes

- lokales Sucheverfahren
- ähnelt dem Ablauf einer $(1, \lambda)$-Evolutionsstrategie
- bei der Erzeugung neuer Individuen wird Geschichte berücksichtigt
- sogenannte *Tabu-Listen* verhindern Rückkehr zu gerade betrachteten Lösungskandidaten

Tabu-Suche

Technik

- 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

Tabu-Suche

Beispiel

- Graphfärbung mit k Farben, sodass keine Kante zwischen gleich gefärbten Knoten ist
- $\Omega = \{1, \ldots, k\}^n$
- 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)

Tabu-Suche

TABU-SUCHE(Zielfunktion F)

```
 1  t ← 0
 2  A(t) ← erzeuge zufälligen Lösungskandidaten
 3  bewerte A(t) durch F
 4  bestInd ← A(t)
 5  initalisiere Tabu-Liste
 6  while Terminierungsbedingung nicht erfüllt
 7  do P ← {}
 8     while #P < λ
 9     do B ← MUTATION(A(t))
10        bewerte B durch F
11        if (A(t),B) ∉ Tabu-Liste oder B.F ≻ bestInd.F
12        then P ← P ∘ {B}
13     t ← t + 1
14     A(t) ← bestes Individuum aus P
15     if A(t).F ≻ bestInd.F
16     then bestInd ← A(t)
17     Tabu-Liste ← aktualisiere durch (A(t − 1),A(t))
18  return bestInd
```

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

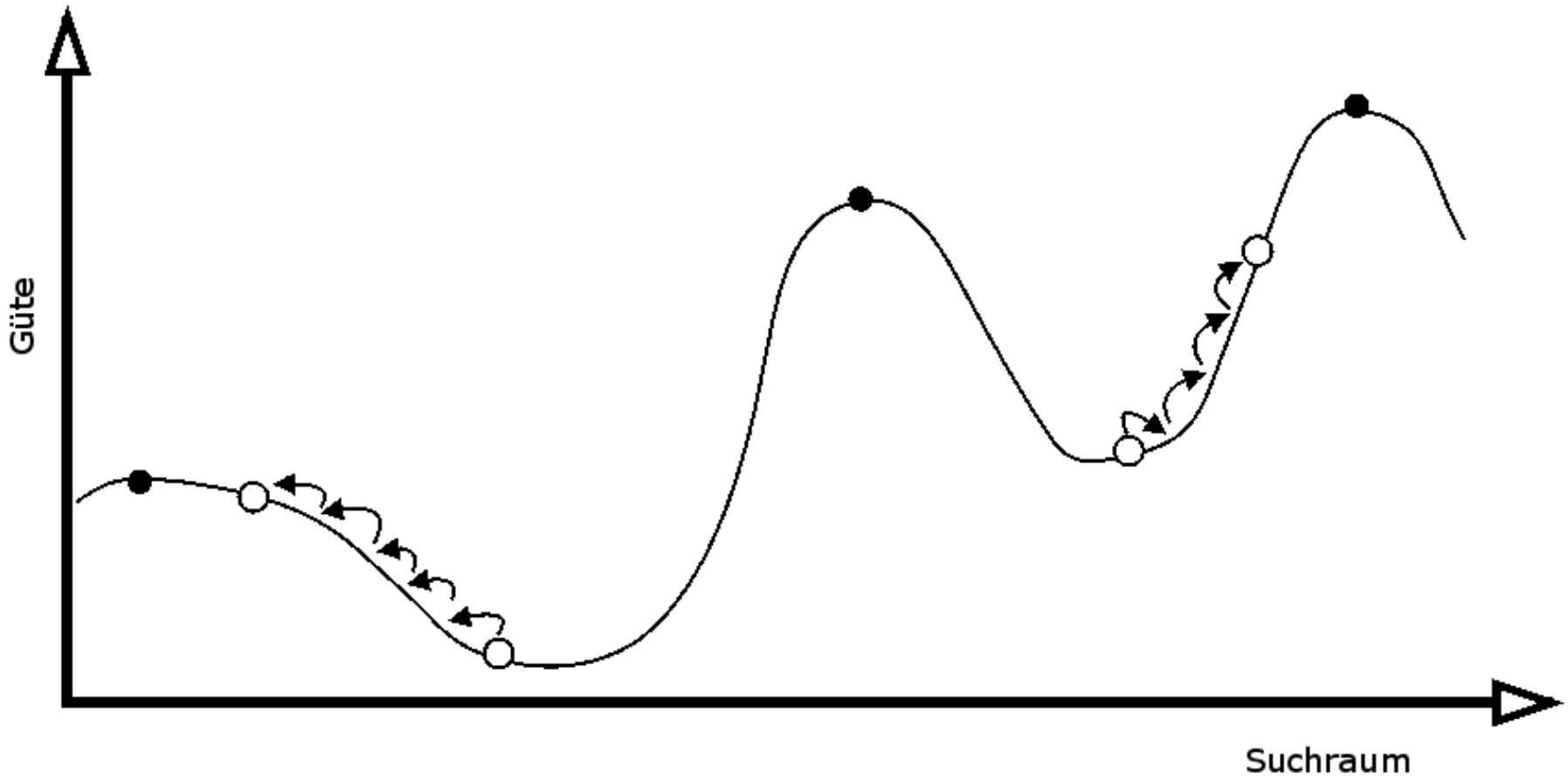
Memetische Algorithmen

Motivation und Grundidee

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

Memetische Algorithmen

Beispiel: SAGA



Genetischer Algorithmus mit Simulated Annealing

Memetische Algorithmen

MEMETISCHER-ALGORITHMUS(Bewertungsfunktion F)

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 **do** $E \leftarrow$ selektiere Eltern für λ Nachkommen aus $P(t)$
7 $P' \leftarrow$ erzeuge Nachkommen durch Rekombination aus E
8 $P'' \leftarrow$ mutiere Individuen in P'
9 $P''' \leftarrow$ LOKALE-SUCHE(F) für jedes Individuum in P''
10 bewerte P''' durch F
11 $t \leftarrow t + 1$
12 $P(t) \leftarrow$ Umweltselektion auf P'''
13 **return** bestes Individuum aus $P(t)$

Memetische Algorithmen

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

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Populationsbasiertes inkrementelles Lernen

PBIL: Grundidee und Operatoren

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

Populationsbasiertes inkrementelles Lernen

PBIL(Bewertungsfunktion F)

1 $t \leftarrow 0$
2 bestInd $\leftarrow$ erzeuge zufälliges Individuum aus $\mathscr{G} = \mathbb{B}^l$
3 $P^{(t)} \leftarrow (0.5, \ldots, 0.5) \in [0,1]^l$
4 **while** Terminierungsbedingung nicht erfüllt
5 **do** $P \leftarrow \{\}$
6 **for** $i \leftarrow 1, \ldots, \lambda$
7 **do** $A \leftarrow$ erzeuge Individuum aus $\mathbb{B}^l$ gemäß $P^{(t)}$
8 $P \leftarrow P \circ \{B\}$
9 bewerte B durch F
10 $\{B\} \leftarrow$ Selektion aus P mittels BESTEN-SELEKTION
11 **if** $F(B) \succ F(\text{bestInd})$
12 **then** bestInd $\leftarrow B$
13 $t \leftarrow t + 1$
14 **for each** $k \in \{1, \ldots, l\}$
15 **do** $P_k^{(t)} \leftarrow B_k \cdot \alpha + P_k^{(t-1)}(1-\alpha)$
16 **for each** $k \in \{1, \ldots, l\}$
17 **do** $u \leftarrow$ wähle Zufallszahl gemäß $U((0,1])$
18 **if** $u < p_m$
19 **then** $u' \leftarrow$ wähle Zufallszahl gemäß $U(\{0,1\})$
20 $P_k^{(t)} \leftarrow u' \cdot \beta + P_k^{(t)}(1-\beta)$
21 **return** bestInd

Populationsbasiertes inkrementelles Lernen

Lernrate α

- niedrig: betont Erforschung
- hoch: betont Feinabstimmung

Paramter	Wertebereich
Populationsgröße λ	20-100
Lernrate α	0.05-0.2
Mutationsrate p_m	0.001-0.02
Mutationskonstate β	0.05

Populationsbasiertes inkrementelles Lernen

Probleme

- genetischer Algorithmus 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

Alternative Verfahren

- bessere Techniken zur Schätzung der Verteilung guter Lösungskandidaten
- insbesondere: interne Abhängigkeiten modellieren (z.B. durch Bayes'sche Netze)
- Beispiel: BOA etc.

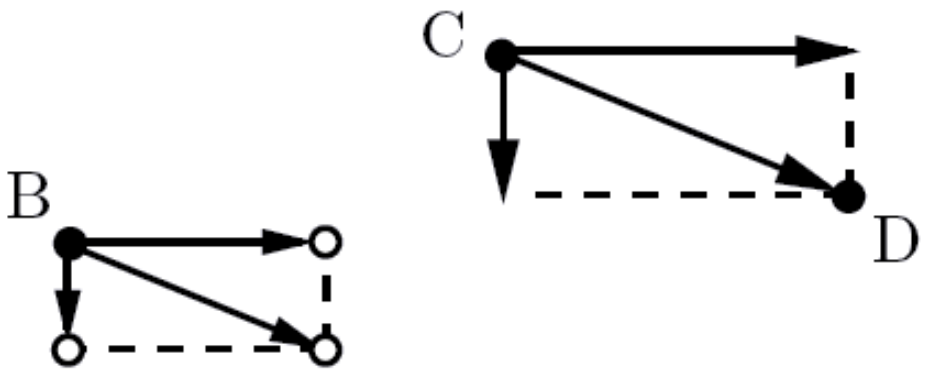
Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Differentialevolution

Idee

- ähnlich zur Evolutionsstrategie
- keine Anpassung der Schrittweite in *A.S*
- sondern: Relation der Individuen in der Population wird als Grundlage für Schrittweite herangezogen



Differentialevolution

DE-OPERATOR(Individuen A, B, C, D)

1 index $\leftarrow$ wähle Zufallszahl gemäß $U(\{1, \ldots, l\})$
2 **for each** $i \in \{1, \ldots, l\}$
3 **do** $u \leftarrow$ wähle Zufallszahl gemäß $U([0, 1))$
4 **if** $u \leq \tau$ **or** $i =$ index
5 **then** $A'_i \leftarrow B'_i + (C_i - D_i) \cdot \alpha$
6 **else** $A'_i \leftarrow A_i$
7 **return** A'

Differentialevolution

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

Differentialevolution

DIFFERENTIALEVOLUTION(Bewertungsfunktion F)

```
 1  t ← 0
 2  P(t) ← erzeuge Population der Größe μ
 3  bewerte P(t) durch F
 4  while Terminierungsbedingung nicht erfüllt
 5  do P(t+1) ← {}
 6     for i ← 1,...,μ
 7     do repeat A,B,C,D ← selektiere Eltern uniform zufällig aus P(t)
 8        until A,B,C,D paarweise verschieden
 9        A' ← DE-OPERATOR(A,B,C,D)
10        bewerte A' durch F
11        if F(A') ≻ F(A)
12        then P(t+1) ← P(t+1) ∘ {A'}
13        else P(t+1) ← P(t+1) ∘ {A}
14     t ← t+1
15  return bestes Individuum aus P(t)
```

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

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

Scatter Search

Ablauf

- iterierter Ablauf durch zwei Phasen
  - Erzeugen neuer Individuen und Auswahl derjenigen, die die größte Vielfalt garantieren
  - Rekombination aller „Paarungen“ der gewählten Individuen
  - Selektion der Besten und Iteration bis sich nichts mehr ändert

Beispiel

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

Scatter Search

Phase 1

- Diversitätsgenerator erzeugt μ Individuen in P
- Beispiel:
  - pro Suchraumdimension Wertebereich in vier Partitionen
  - Häufigkeit der erzeugten Individuen pro Partition merken
  - Partition invers proportional auswählen
- separat: Population der besten α Individuen
  - wird um die β Individuen aus P erweitert, die $\min_{B \in \text{bestP}} d(A.G, B.G)$ maximieren ($A \in P$)

Scatter Search

Phase 2

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

Scatter Search

```
SCATTER-SEARCH(Bewertungsfunktion F)
 1   bestP = {}
 2   P = {}
 3   for t ← 1,...,maxIter
 4   do while #P < μ
 5      do A ← erzeuge ein Individuum mit Diversitätsgenerator
 6         A ← LOKALE-SUCHE(F) angewandt auf A
 7         bewerte A durch F
 8         if A ∉ P ∘ bestP
 9         then P ← P ∘ {A}
10      if t = 1
11      then bestP ← selektiere α Individuen aus P mit BESTEN-SELEKTION
12         P ← streiche Individuum A in P
13      for k ← 1,...,β
14      do A ← dasjenige Individuum aus P, das min_{B∈bestP} d(A.G, B.G) maximiert
15         P ← streiche Individuum A in P
16         bestP ← bestP ∘{A}
17      repeat P' ← {}
18         Mengen ← erzeuge Teilmengen von bestP durch Teilmengenoperator
19         for each M ∈ Mengen
20         do A ← wende Kombinationsoperator auf M an
21            A ← LOKALE-SUCHE(F) angewandt auf A
22            bewerte A durch F
23            if A ∉ bestP ∘P'
24            then P' ← P' ∘ {A}
25         bestP ← selektiere α + β Ind. aus bestP ∘P' mit BESTEN-SELEKTION
26      until bestP hat sich nicht geändert
27      bestP ← selektiere α Individuen aus P mit BESTEN-SELEKTION
28   return bestInd
```

Scatter Search

Empfohlene Parameter

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

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

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

Kulturelle Algorithmen

CULTURAL-ALGORITHMUS(Bewertungsfunktion F)

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

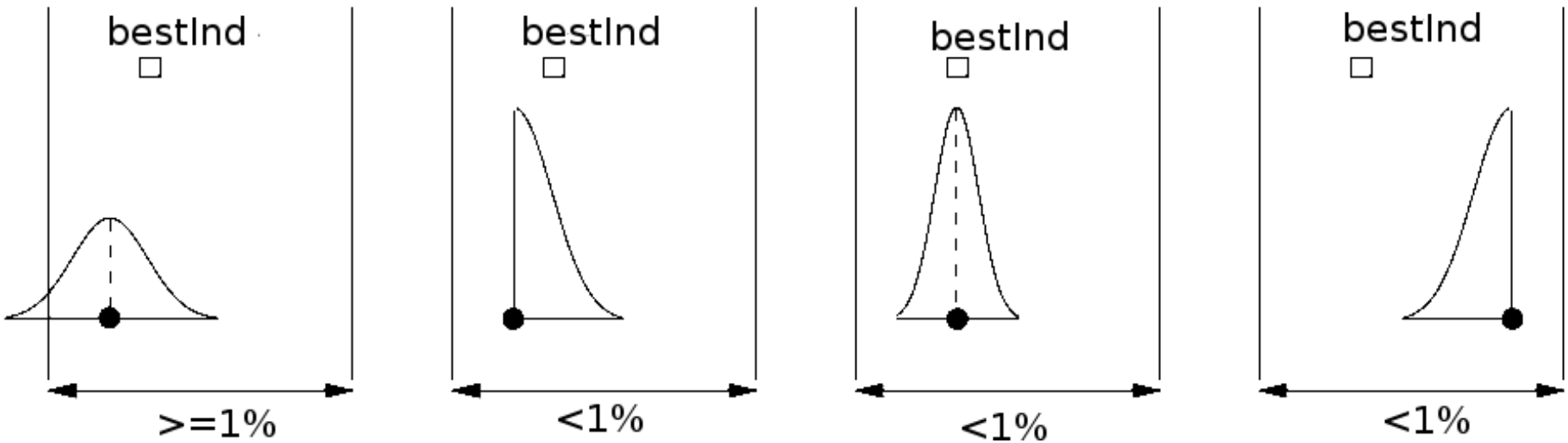
Kulturelle Algorithmen

Stabile Suchraumdimensionen

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

Mutation

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



Kulturelle Algorithmen

CA-MUTATION(Individuum A)

1 $u' \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0,1)$
2 **for each** $i \in \{1,\ldots,l\}$
3 **do** $u_i'' \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0,1)$
4 $B.S_i \leftarrow A.S_i \cdot \exp\left(\frac{1}{\sqrt{2l}} \cdot u' + \frac{1}{\sqrt{2\sqrt{l}}} \cdot u_i''\right)$
5 $u \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0, B.S_i)$
6 **if** $\mathscr{BS}$ ist stabil für Dimension i
7 **then switch**
8 **case** $A.G_i < \text{BestInd.G}_i : B.G_i \leftarrow A.G_i + |u|$
9 **case** $A.G_i > \text{BestInd.G}_i : B.G_i \leftarrow A.G_i - |u|$
10 **case** $A.G_i = \text{BestInd.G}_i : B.G_i \leftarrow A.G_i + \frac{u}{2}$
11 **else** $B.G_i \leftarrow A.G_i + u$
12 $B.G_i \leftarrow \max\{\text{ug}_i, \min\{\text{og}_i, B.G_i\}\}$
13 **return** B

Überblick

1. Was sind Metaheuristiken?
2. Klassifizierende Systeme
3. Tabu-Suche
4. Memetische Algorithmen
5. Populationsbasiertes inkrementelles Lernen
6. Differentialevolution
7. Scatter Search
8. Kulturelle Algorithmen
9. Zusammenfassung

Genetischer Algorithmus

Repräsentation: (klassisch) $\mathbb{B}^l$ mit fester Länge l (auch $\mathbb{R}^l$ und $\mathscr{S}_n$, Dekodierung)

Mutation: Bitflipping, gleichverteilte reellwertige Mutation, spezielle Permutationsoperatoren

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

Selektion: Elternselektion, fitnessproportional oder Turnierselektion

Population: mittelgroße Populationen

Besonderheiten: theoretische Grundlage durch Schema-Theorem

Evolutionsstrategie

Repräsentation: $\mathbb{R}^l$, meist gilt Genotyp = Phänotyp, zusätzliche Strategieparameter

Mutation: Gauß-Mutation, erst Mutation der Strategieparameter

Rekombination: anfangs keine, später arithmetischer und uniformer Crossover, auch globale Varianten, anderer Operator auf Strategieparametern

Selektion: gleichverteilt (Eltern), Komma- bzw. Plus-Selektion (Kinder)

Population: eher kleine Populationen, manchmal $\mu = 1$

Besonderheiten: Selbstadaption für Schrittweiten

Evolutionäres Programmieren

Repräsentation: anfangs endlicher Automat, später $\mathbb{R}^l$ mit Strategieparametern

Mutation: Modifikation der Zustände und Übergänge in Automaten, Gauß-Mutation, gleichzeitige Mutation der Strategieparameter

Rekombination: keine

Selektion: anfangs Plus-Selektion, später 2-fache q-stufige Turnierselektion aus Eltern und Kindern

Population: mittelgroße Populationen

Besonderheiten: es gilt $\mu = \lambda$, Selbstadaption

Genetisches Programmieren

Repräsentation: Bäume aber auch lineare Darstellung variabler Länge

Mutation: internes Umhängen oder Modifikation von Teilbäumen

Rekombination: Austausch von Teilbäumen

Selektion: verschiedene (meist wie genetischer Algorithmus)

Population: sehr große Populationen

Besonderheiten: oft nur ein Operator auf ein Individuum, spezielle Initialisierung, Methoden zur Vermeidung von Introns

Lokale Suche

Repräsentation: beliebig

Mutation: beliebig

Rekombination: keine

Selektion: Verbesserungen immer, Verschlechterungen mit gewisser Wahrscheinlichkeit

Population: ein Individuum

Besonderheiten: zu frühe Konvergenz ist zentrales Problem

Klassifizierendes System

Repräsentation: Regel oder Menge an Regeln, k-är kodiert

Mutation: Bitflipping

Rekombination: k-Punkt-Crossover

Selektion: fitnessproportional

Population: ausreichend für Komplexität der Aufgabe (bei Michigan)

Besonderheiten: Population ist Regelsatz (bei Michigan), Individuum ist Regelsatz (bei Pittsburgh)

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

Populationsbasiertes inkrementelles Lernen

Repräsentation: $\mathbb{B}^l$

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

Differentialevolution

Repräsentation: $\mathbb{R}^l$

Mutation: Mischoperator

Rekombination: Mischoperator

Selektion: Kind ersetzt Elter bei Verbesserung

Population: klein/mittelgrß

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

Ameisenkolonie

Repräsentation: verschiedene

Mutation: jede Ameise konstruiert einen Lösungskandidaten

Rekombination: keine

Selektion: Güte bestimmt Einfluss auf globale Pheromonmenge

Population: Anzahl der Ameisen pro Iterationsschritt

Besonderheiten: kein EA-Schema, globale Pheromonmengen repräsentieren Lösungskandidaten ähnlich zur Statistik in PBIL

Partikelschwarm

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: kein EA-Schema, eher synchrones Durchkämmen des Suchraums