Evolutionäre Algorithmen: Standardalgorithmen

Karsten Weicker

HTWK Leipzig

5. Januar 2013

Überblick

Überblick

Genetische Algorithmen

Grundkonzept

- Primärer Operator: Rekombination
- Hintergrundoperator: Mutation
- Elternselektion: fitnessproportional

Ausprägungen

- ursprünglich: binäre Kodierung
- später auch: reellwertige Zahlen, Permutationen

Typische Parameterwerte

Parameter	Wertebereich
Populationsgröße:	30–100
Rekombinationswahrscheinlichkeit:	0,6–0,9
Mutationsrate:	0,001–0,01, $\frac{1}{l}$

Verallgemeinerung des 1-Punkt-Crossovers

k-Punkt-Crossover(Individuum A, Individuum B)

```
 1  for m ∈ {1, ..., k}
 2  do ⌊ j_m ← wähle Zufallszahl gemäß U({1, ..., l − 1})
 3  sortiere j_1, ..., j_k so, dass j_1 ≤ j_2 ≤ ... ≤ j_k
 4  j_0 ← 0
 5  j_{k+1} ← l
 6  for m ∈ {0, ..., k}
 7  do ⌈ if m mod 2 = 0
 8        then ⌈ for i ∈ {j_m + 1, ..., j_{m+1}}
 9                 do ⌈ C_i ← A_i
10                    ⌊    ⌊ D_i ← B_i
11        else ⌈ for i ∈ {j_m + 1, ..., j_{m+1}}
12                 do ⌈ C_i ← B_i
13           ⌊         ⌊    ⌊ D_i ← A_i
14  return C, D
```

Implementationsdetail: Mutation

EFFIZIENTE-BINÄRE-MUTATION(Individuum A mit $A.G \in \mathbb{B}^l$)

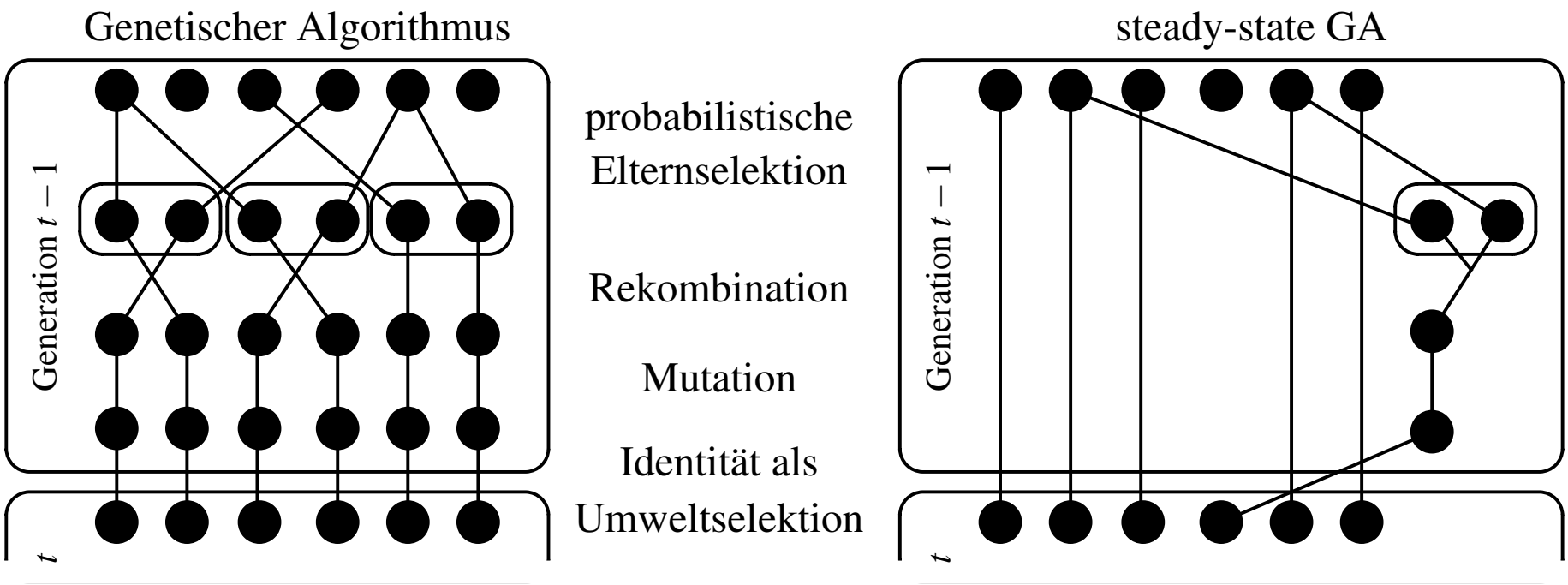
1 $B \leftarrow A$
2 **while** $next \quad (\text{Abstand zur nächsten Mutation}) \quad \leq l$
3 **do** $\ulcorner B_{next} \leftarrow 1 - A_{next}$
4 $\quad u \leftarrow$ wähle zufällig gemäß $U([0,\ 1))$
5 $\quad \llcorner next \leftarrow next + \left\lfloor \frac{\ln(u)}{\ln(1-p_m)} \right\rfloor$
6 $next \leftarrow next - l$
7 **return** B

Überlappende Population

STEADY-STATE-GA(Zielfunktion F)

```
 1  t ← 0
 2  P(t) ← erzeuge Population mit μ (Populationsgröße) Individuen
 3  bewerte P(t) durch F
 4  while Terminierungsbedingung nicht erfüllt
 5  do ⌜ ⟨A, B⟩ ← Selektion aus P(t) mittels FITNESSPROPORTIONALE-SELEKTI
 6        u ← wähle Zufallszahl gemäß U([0, 1))
 7        if u ≤ p_x (Crossover-Wahrscheinlichkeit)
 8        then ⌊ C ← EIN-PUNKT-CROSSOVER(A, B)
 9        else ⌊ C ← B
10        C ← BINÄRE-MUTATION(C)
11        C.F ← bewerte C durch F
12        P' ← entferne das schlechteste Individuum aus P(t)
13        t ← t + 1
14     ⌞ P(t) ← P' ∘ ⟨C⟩
15  return bestes Individuum aus P(t)
```

Ablauf des genetischen Algorithmuses



Reellwertige GAs

Besonderheiten

- Kodierungsprobleme (Hamming-Klippen) werden vermieden
- Standard-Crossover-Operatoren (uniform und k-Punkt) können nicht alle Punkte im Suchraum erreichen
- Daher: Arithmetischer Crossover
- Mutation: keine Gauß-Mutation

Creep Mutation

GLEICHVERTEILTE-REELLWERTIGE-MUTATION(Individuum A)

```
1  for i ∈ {1, ..., l}
2  do ⌜ u ← wähle Zufallszahl gemäß U([0, 1))
3        if u ≤ p_m  (|Mutationswahrscheinlichkeit|)
4        then ⌜ unten ← max{ug_i, A_i − x (|maximale Schrittweite|)}
5                    oben ← min{og_i, A_i + x  }
6     ⌞        ⌞ B_i ← wähle Zufallszahl gemäß U([unten, oben])
7  return B
```

Kombinatorische GAs

Operatoren

- invertierende Mutation
- vertauschende Mutation
- neu: verschiebende Mutation
- neu: mischende Mutation
- Ordnungsrekombination
- Kantenrekombination
- neu: Abbildungsrekombination

Verschiebende Mutation

VERSCHIEBENDE-MUTATION(Individuum A mit $A.G \in S_l$

```
1   B ← A
2   u_1 ← wähle zufällig gemäß U({1, ..., l})
3   u_2 ← wähle zufällig gemäß U({1, ..., l})
4   B_{u_2} ← A_{u_1}
5   if u_1 > u_2
6   then ⌜ for j ∈ {u_2, ..., u_1 − 1}
7        ⌞ do ⌈ B_{j+1} ← A_j
8   else ⌜ for j ∈ {u_1 + 1, ..., u_2}
9        ⌞ do ⌈ B_{j−1} ← A_j
10  return B
```

Mischende Mutation

MISCHENDE-MUTATION(Individuum A mit $A.G \in \mathcal{S}_l$)

1 $B \leftarrow A$
2 $u_1 \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, l\})$
3 $u_2 \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, l\})$
4 **if** $u_1 > u_2$
5 **then** [vertausche u_1 und u_2
6 $\pi \leftarrow$ wähle zufällig aus $U(\mathcal{S}_{u_2-u_1+1})$
7 **for** $j \in \{1, \ldots, u_2 - u_1 + 1\}$
8 **do** [$B_{u_1+j-1} \leftarrow A_{u_1+\pi(j)-1}$
9 **return** B

ABBILDUNGSREKOMBINATION(Individuen A, B mit $A.G, B.G \in \mathcal{S}_l$)

```
 1  for  each i ∈ {1, ..., l}
 2  do ⌊ g(A_i) ← B_i
 3  u_1 ← wähle Zufallszahl gemäß U({2, ..., l − 1})
 4  u_2 ← wähle Zufallszahl gemäß U({2, ..., l − 1})
 5  if u_2 < u_1
 6  then ⌊ vertausche u_1 und u_2
 7  benutzt ← ∅
 8  for  each i ∈ {u_1, ..., u_2}
 9  do ⌈ C_i ← B_i
10     ⌊ benutzt ← benutzt ∪ {B_i}
11  for i ← 1, ..., u_1 − 1, u_2 + 1, ..., l
12  do ⌈ x ← A_i
13     while x ∈ benutzt
14     do ⌊ x ← g(x)
15     C_i ← x
16     ⌊ benutzt ← benutzt ∪ {x}
17  return C
```

Überblick

Evolutionsstrategie

Grundsätzliches

- uniforme Wahl der Eltern
- $\mathcal{G} = \mathbb{R}^l$
- Gauß-Mutation mit Anpassung von σ
- zunächst keine Rekombination
- (Umwelt-)Selektion der Besten

Evolutionsstrategie

Notationen

- (μ, λ)-ES: Selektion nur aus Kindern
- $(\mu + \lambda)$-ES: Selektion aus Kindern und Eltern

Implementationsdetails

- Selektion durch Sortieren: $\mathcal{O}(\lambda \log \lambda)$
- Selektion durch Heap mit Delete-Min: $\mathcal{O}(\lambda + \mu \log \lambda)$

ES-ADAPTIV(Zielfunktion F)

```
 1  t ← 0
 2  σ ← Wert für Anfangsschrittweite
 3  s ← 0
 4  P(t) ← erzeuge Population mit μ (|Populationsgröße|) Individuen
 5  bewerte P(t) durch F
 6  while Terminierungsbedingung nicht erfüllt
 7  do ⌜ P' ← ⟨⟩
 8        for i ∈ {1, ..., λ}
 9        do ⌜ A ← selektiere Elter uniform zufällig aus P(t)
10              C ← GAUSS-MUTATION(A) mit σ
11              bewerte C durch F
12              if C.F ≻ A.F
13              then ⌞ s ← s + 1
14                   ⌞ P' ← P' ∘ ⟨C⟩
15        t ← t + 1
16        if t mod k (|Modifikationshäufigkeit|) = 0
17        then ⌜ σ ← ADAPTIVE-ANPASSUNG(σ, s/(kλ))
18             ⌞ s ← 0
19     ⌞ P(t) ← Selektion aus P' mittels BESTEN-SELEKTION
20  return bestes Individuum aus P(t)
```

Mit Selbstadaptation

ES-SELBSTADAPTIV(Zielfunktion F)

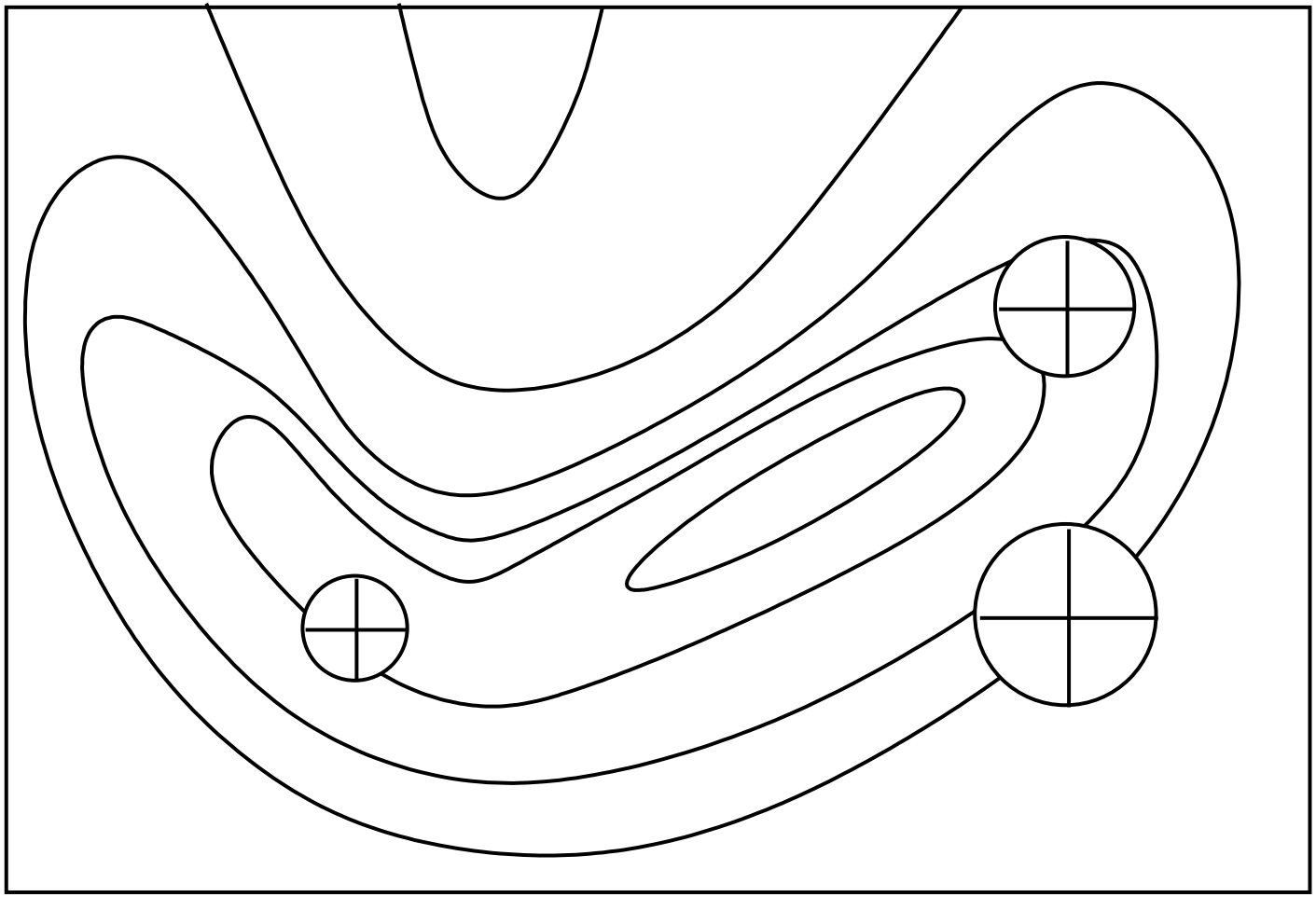
```
 1  t ← 0
 2  P(t) ← erzeuge Population mit μ (⌈Populationsgröße⌋) Individuen
 3  bewerte P(t) durch F
 4  while Terminierungsbedingung nicht erfüllt
 5  do ⌜ P' ← ⟨⟩
 6        for i ∈ {1, ..., λ (⌈Anzahl der Kinder⌋) }
 7        do ⌜ A ← selektiere Elter uniform zufällig aus P(t)
 8              B ← SELBSTADAPTIVE-GAUSS-MUTATION(A)
 9           ⌞ P' ← P' ∘ ⟨B⟩
10        bewerte P' durch F
11        t ← t + 1
12     ⌞ P(t) ← Selektion aus P' mittels BESTEN-SELEKTION
13  return bestes Individuum aus P(t)
```

Mutation mit Selbstadaptation (Wdhg)

SELBSTADAPTIVE-GAUSS-MUTATION(Individuum A)

1 $u \leftarrow \mathcal{N}(0,1)$
2 $B.S \leftarrow A.S \; e^{\frac{1}{\sqrt{l}}u}$
3 **for** $i \in \{1,\ldots,l\}$
4 **do** $\ulcorner$ $u_i \leftarrow$ wähle zufällig gemäß $\mathcal{N}(0, B.S)$
5 $B_i \leftarrow A_i + u_i$
6 $B_i \leftarrow \max\{B_i, ug_i \; (\text{untere Wertebereichsgrenze})\}$
7 $\llcorner$ $B_i \leftarrow \min\{B_i, og_i \; (\text{obere Wertebereichsgrenze})\}$
8 **return** B

Problem angepasster Schrittweiten

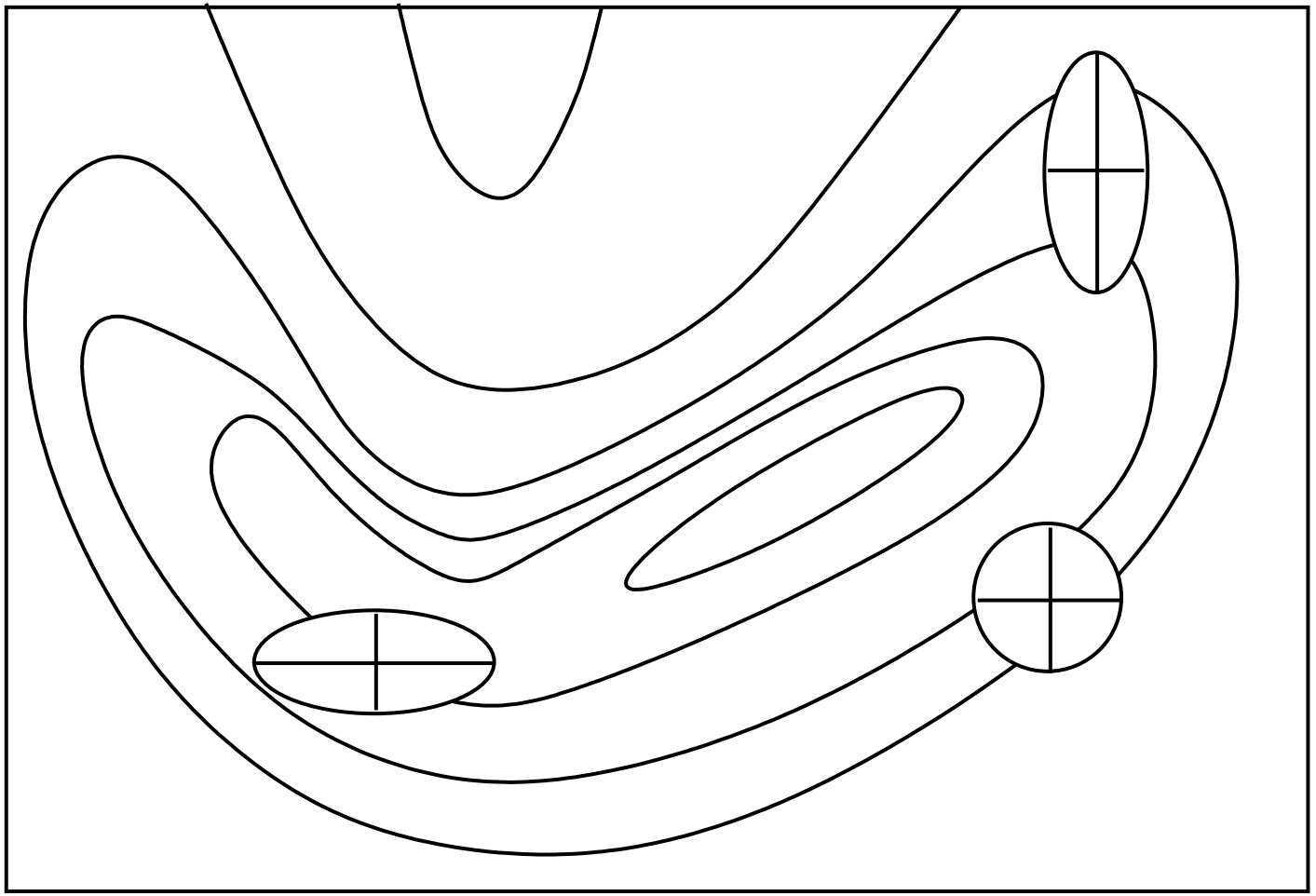


Separate Schrittweiten

Idee

- jede Suchraumdimension erhält eigenen Schrittweitenparameter
- $\mathcal{Z} = (\mathbb{R}^+)^l$
- zufällige Veränderungen aus gemeinsamer und individueller Komponente
- $B.S_i \leftarrow A.S_i \cdot \exp\left(\frac{1}{\sqrt{2l}}u + \frac{1}{\sqrt{2\sqrt{l}}}u'_i\right)$
  mit $u, u_1, \ldots, u_l \sim \mathcal{N}(0,1)$
- Dann analog: $B.G_i \leftarrow A.G_i + \mathcal{N}(0, B.S_i)$

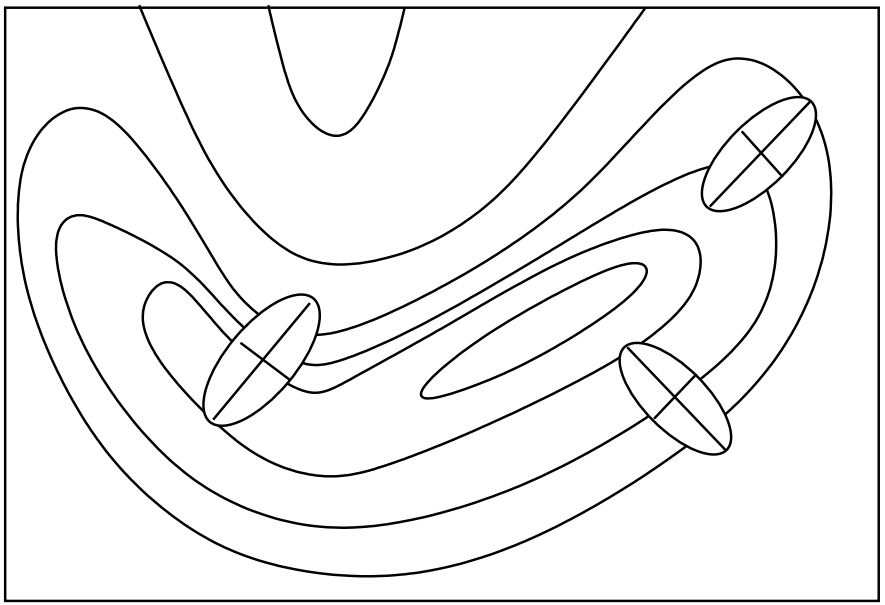
Besser angepasste Schrittweiten



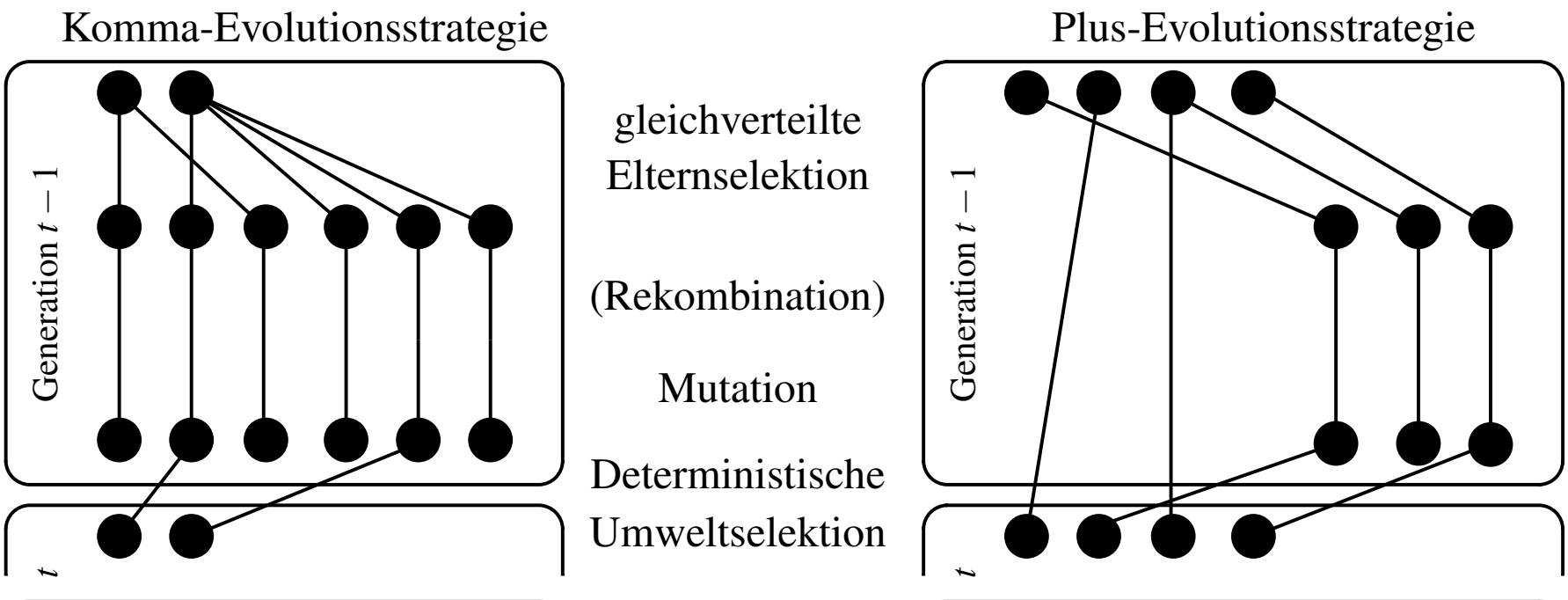
Zusätzliche Ausrichtung im Raum

Idee

- beliebige Orientierung durch weitere $\frac{1}{2} \cdot I \cdot (I - 1)$ Strategieparameter



Ablauf der Evolutionsstrategie



Rekombination

Operatoren

- uniformer Crossover
- arithmetischer Crossover
- aber auch: globale Varianten mit mehr Eltern
- häufig: global arithmetisch auf Strategieparametern, uniform auf Objektvariablen

Notation

- $(\mu/r \overset{+}{,} \lambda)$-ES mit Anzahl der Eltern r pro Rekombination

Globale Rekombination

GLOBALER-ARITHMETISCHER-CROSSOVER(Population $P = \langle A^{(k)} \rangle_{1 \leq}$

1 **for** $i \in \{1, \ldots, l\}$
2 **do** $\lfloor B.G_i \leftarrow \frac{1}{\mu} \cdot \sum_{k=1}^{\mu} A^{(k)}.G_i$
3 **return** B

Typische Parameter für die Komma-ES

Parameter	Wertebereich
Populationsgröße μ:	1–30
Kindindividuen pro Generation:	$5 \cdot \mu$–$7 \cdot \mu$ (Komma) sonst ≥ 1
Rekombinationsrate:	0,0–1,0

Derandomisierte ES

Problem

- große Unsicherheit bei indirekter Bewertung der Strategieparameter
- verkleinertes σ führt nur in Erwartung zu einem kleineren Schritt

Idee

- $(\mu/\mu, \lambda)$-ES mit globalem σ
- σ wird durch echte Schrittlänge der μ besten Kindindividuen angepasst
- Lernmechanismus: Lernrate $c = \frac{1}{\sqrt{l}}$ und Dämpfungsfaktor $d = \sqrt{l}$

DERANDOMISIERTE-ES(Zielfunktion $F : \mathbb{R}^l \to \mathbb{R}$)

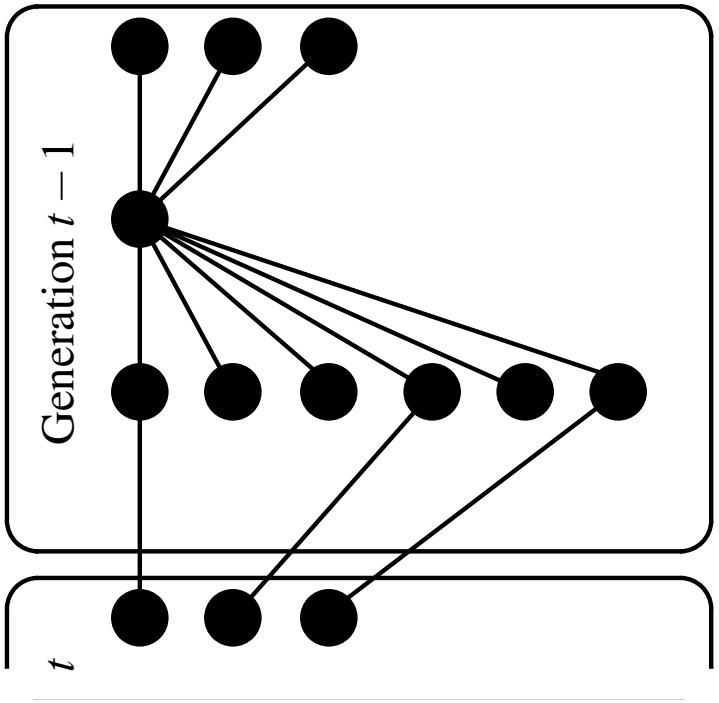
1 $t \leftarrow 0$
2 $P(t) \leftarrow$ erzeuge Population mit μ (Populationsgröße) Individuen
3 $s^{(t)} \leftarrow (0, \ldots, 0) \in \mathbb{R}^l$
4 initialisiere $\sigma^{(t)}$ (globaler Schrittweitenparameter)
5 bewerte $P(t)$ durch F
6 **while** Terminierungsbedingung nicht erfüllt
7 **do** $\ulcorner B \leftarrow$ GLOBALER-ARITHMETISCHER-CROSSOVER($P(t)$)
8 $P' \leftarrow \langle\rangle$
9 **for** $i \in \{1, \ldots, \lambda$ (Anzahl der Kinder) $\}$
10 **do** $\ulcorner C \leftarrow$ GAUSS-MUTATION(B) mit $\sigma^{(t)}$
11 $P' \leftarrow P' \circ \langle C \rangle$
12 $\llcorner z^{(i)} \leftarrow C.G - B.G \in \mathbb{R}^l$
13 bewerte P' durch F
14 $t \leftarrow t + 1$
15 *Indizes* $\leftarrow$ BESTEN-SELEKTION für Individuen in P'
16 $P(t) \leftarrow$ Selektion aus P' gemäß *Indizes*
17 $\overline{z} \leftarrow \frac{1}{\mu} \cdot \sum_{j \in \mathit{Indizes}} z^{(j)}$
18 $s^{(t)} \leftarrow (1 - c$ (Lernfaktor) $) \cdot s^{(t-1)} + \sqrt{c \cdot (2 - c) \cdot \mu} \cdot \overline{z}$
19 $\llcorner \sigma^{(t)} \leftarrow \sigma^{(t-1)} \cdot \exp\left(\frac{\|s^{(t)}\|^2 - l}{2 \cdot D \cdot l}\right)$ (mit Dämpfungsfaktor D)
20 **return** bestes Individuum aus $P(t)$

Ablauf der derandomisierten ES

globale Rekombination

Mutation

Deterministische Umweltselektion mit Anpassung von σ



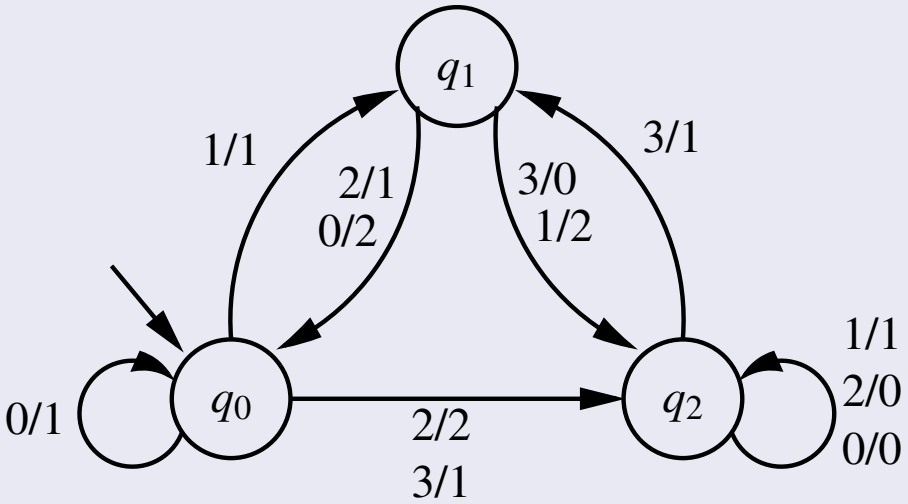
Überblick

Evolutionäres Programmieren

Motivation

- Prognoseaufgaben
- zunächst: endliche Automaten
- später: neuronale Netze

Beispiel



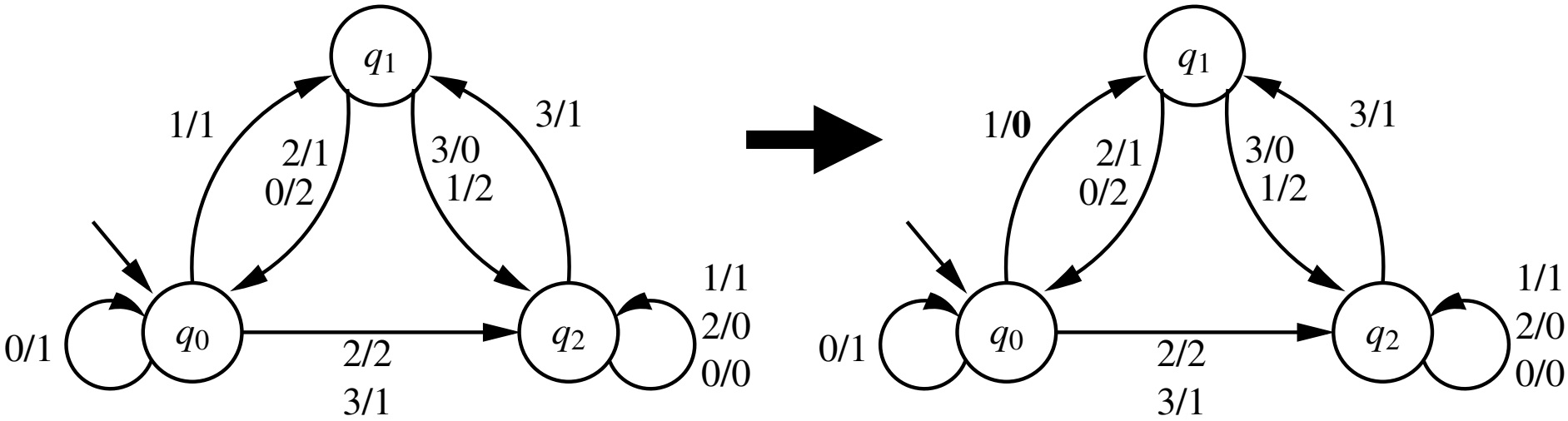
EP-Mutationen

Mutation der Ausgabe

AUTOMATENMUTATION-AUSGABE(Individuum A mit $A.G$

1 $\gamma' \leftarrow \gamma$
2 $q \leftarrow$ wähle zufällig gemäß $U(Q)$
3 $a \leftarrow$ wähle zufällig gemäß $U(\Sigma)$
4 $a' \leftarrow$ wähle zufällig gemäß $U(\Sigma)$
5 $\gamma'(q, a) \leftarrow a'$
6 **return** B mit $B.G = (Q, \delta, \gamma', q_0)$

EP-Mutationen



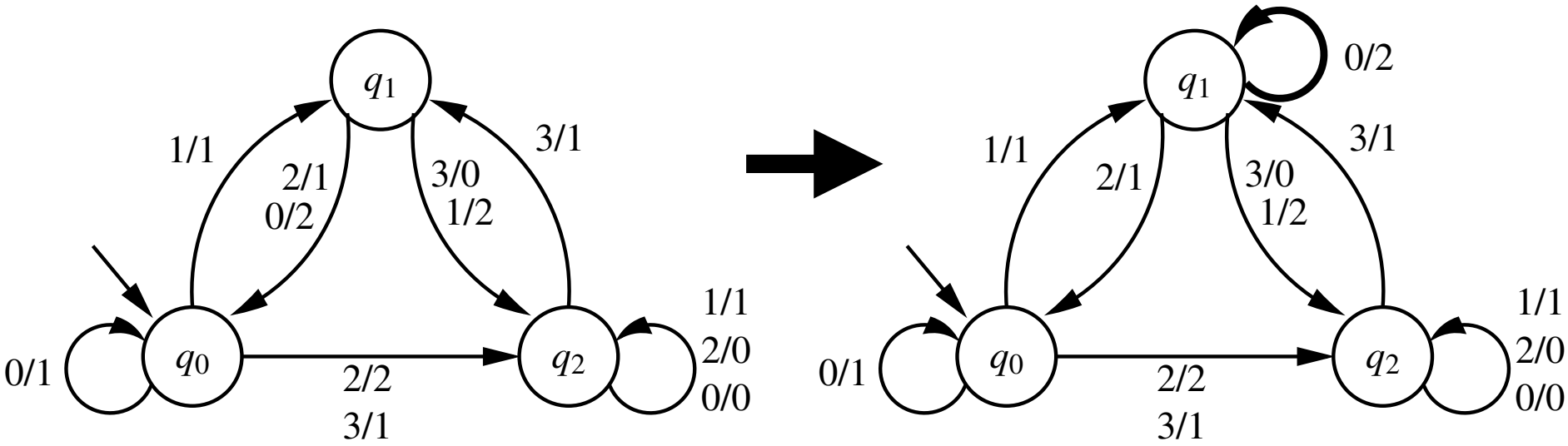
EP-Mutationen

Mutation des Folgezustands

AUTOMATENMUTATION-FOLGEZUSTAND(Individuum A r

1 $\delta' \leftarrow \delta$
2 $q \leftarrow$ wähle zufällig gemäß $U(Q)$
3 $p \leftarrow$ wähle zufällig gemäß $U(Q)$
4 $a \leftarrow$ wähle zufällig gemäß $U(\Sigma)$
5 $\delta'(q, a) \leftarrow p$
6 **return** B mit $B.G = (Q, \delta', \gamma, q_0)$

EP-Mutationen



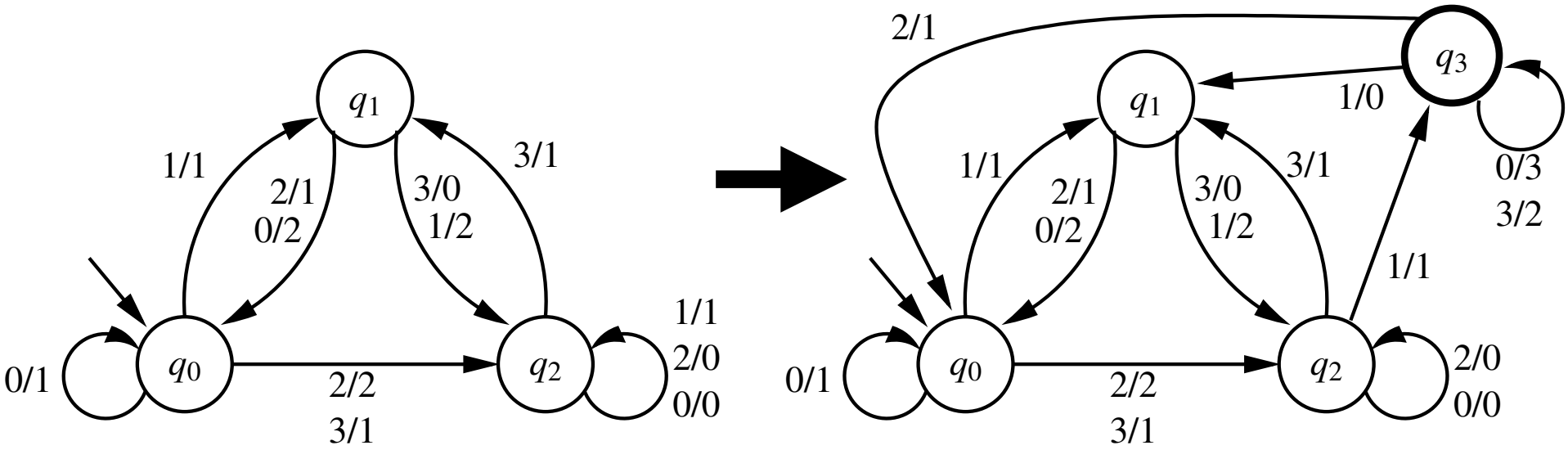
EP-Mutationen

Mutation: Neuer Zustand

AUTOMATENMUTATION-NEUER-ZUSTAND(Individuum A mit $A.G =$

1 $Q' \leftarrow Q \dot{\cup} \{p\}$
2 $\delta' \leftarrow \delta$
3 $\gamma' \leftarrow \gamma$
4 **for each** $a \in \Sigma$
5 **do** $\ulcorner$ $q \leftarrow$ wähle zufällig gemäß $U(Q')$
6 $\quad \delta'(p, a) \leftarrow q$
7 $\quad a' \leftarrow$ wähle zufällig gemäß $U(\Sigma)$
8 $\quad \llcorner\, \gamma'(p, a) \leftarrow a'$
9 $q \leftarrow$ wähle zufällig gemäß $U(Q')$
10 $a \leftarrow$ wähle zufällig gemäß $U(\Sigma)$
11 $\delta'(q, a) \leftarrow p$
12 **return** B mit $B.G = (Q', \delta', \gamma', q_0)$

EP-Mutationen



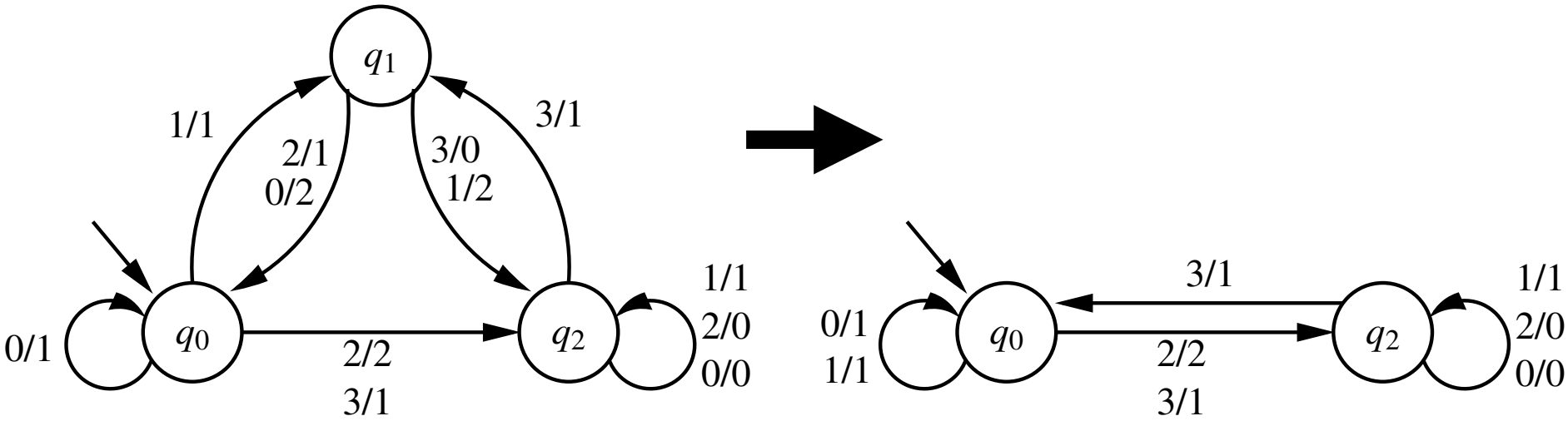
EP-Mutationen

Mutation: Zustand löschen

AUTOMATENMUTATION-ZUSTAND-LÖSCHEN(Individuum A mit $A.G$

1 $p \leftarrow$ wähle zufällig gemäß $U(Q \setminus \{q_0\})$
2 $Q' \leftarrow Q \setminus \{p\}$
3 $\delta' \leftarrow \delta\big|_{Q' \times \Sigma}$
4 $\gamma' \leftarrow \gamma\big|_{Q' \times \Sigma}$
5 **for each** $(q, a) \in Q \times \Sigma$
6 **do** ⌜ **if** $\delta(q, a) = p$
7 **then** ⌜ $p' \leftarrow$ wähle zufällig gemäß $U(Q')$
8 ⌞ ⌞ $\delta'(q, a) \leftarrow p'$
9 **return** B mit $B.G = (Q', \delta', \gamma', q_0)$

EP-Mutationen



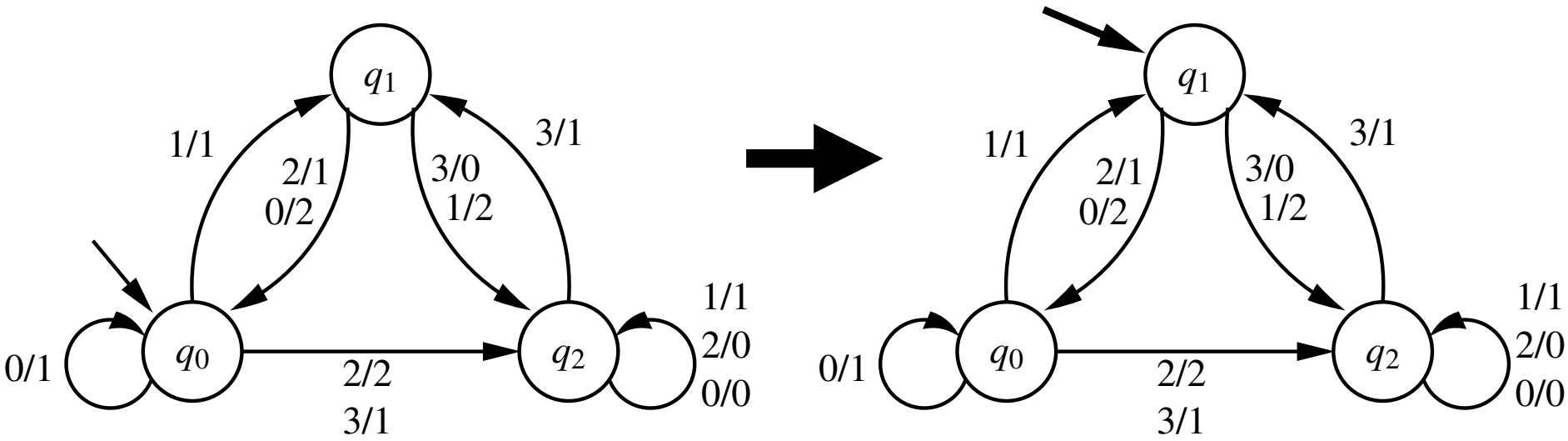
EP-Mutationen

Mutation: Startzustand

AUTOMATENMUTATION-STARTZUSTAND(Individuum A n

1 $q_0' \leftarrow$ wähle zufällig gemäß $U(Q)$
2 **return** B mit $B.G = (Q, \delta, \gamma, q_0')$

EP-Mutationen



EVOLUTIONÄRES-PROGRAMMIEREN-1960ER(Zielfunktion F)

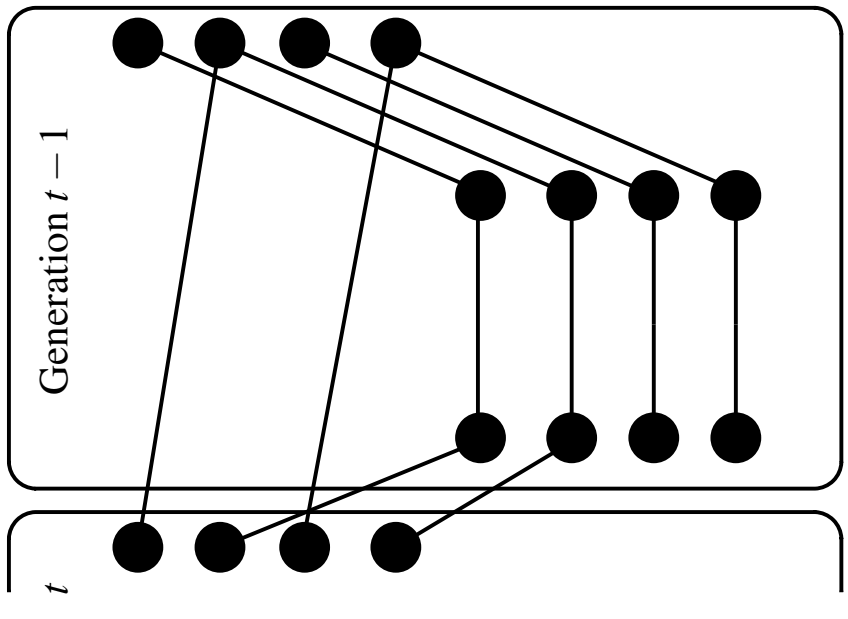
```
 1  t ← 0
 2  P(t) ← erzeuge Population mit μ (|Populationsgröße|) Individuen
 3  bewerte P(t) durch F
 4  while Terminierungsbedingung nicht erfüllt
 5  do ⌜ P' ← P(t)
 6        for j ← 1, ..., μ
 7        do ⌜ (|sei P(t) = ⟨A^(i)⟩_{1≤i≤μ}|)
 8              u ← wähle zufällig gemäß U({1, ..., 5})
 9              switch
10              case u = 1 : B ← AUTOMATENMUTATION-AUSGABE(A^(j))
11              case u = 2 : B ← AUTOMATENMUTATION-FOLGEZUSTAND(A^(j))
12              case u = 3 : B ← AUTOMATENMUTATION-NEUER-ZUSTAND(A^(j))
13              case u = 4 : B ← AUTOMATENMUTATION-ZUSTAND-LÖSCHEN(A^(
14              case u = 5 : B ← AUTOMATENMUTATION-STARTZUSTAND(A^(j))
15              bewerte B durch F
16           ⌞ P' ← P' ∘ ⟨B⟩
17        t ← t + 1
18     ⌞ P(t) ← Selektion aus P' mittels BESTEN-SELEKTION
19  return bestes Individuum aus P(t)
```

Ablauf: evolutionäres Programmieren

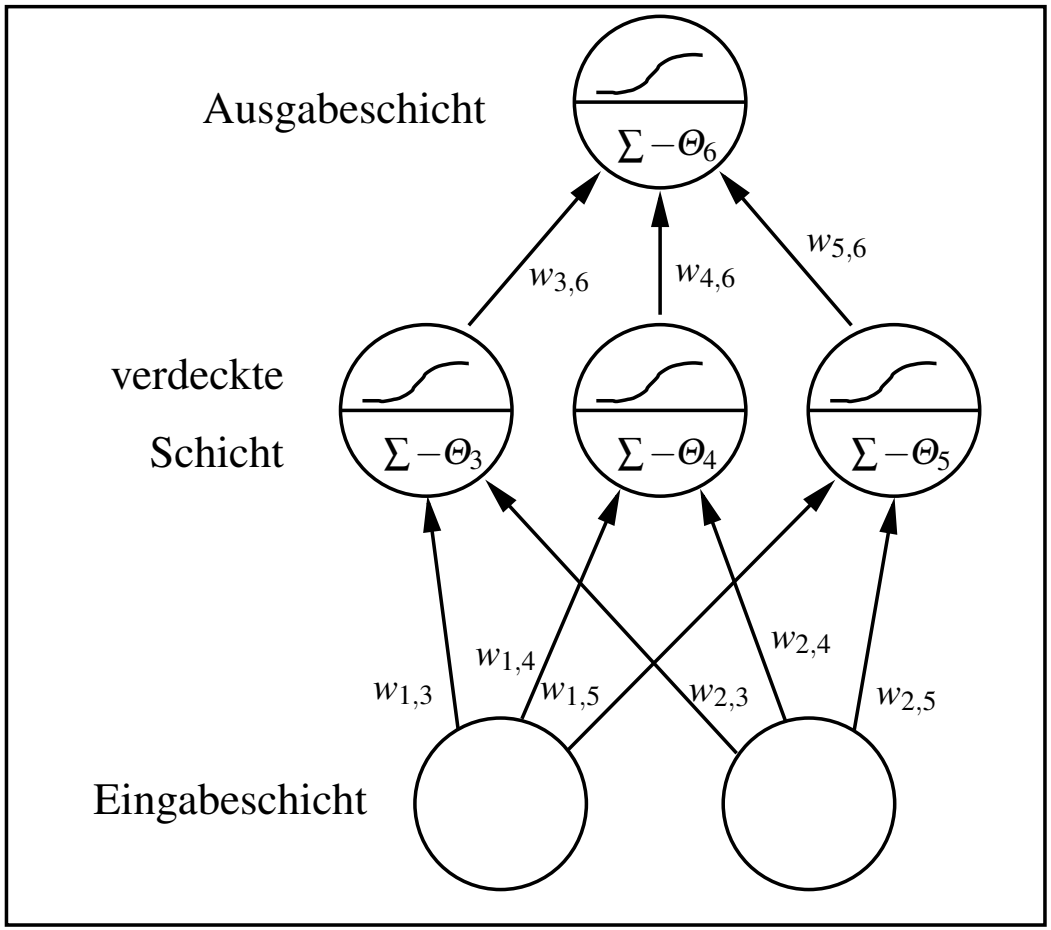
Identität als Elternselektion

Mutation

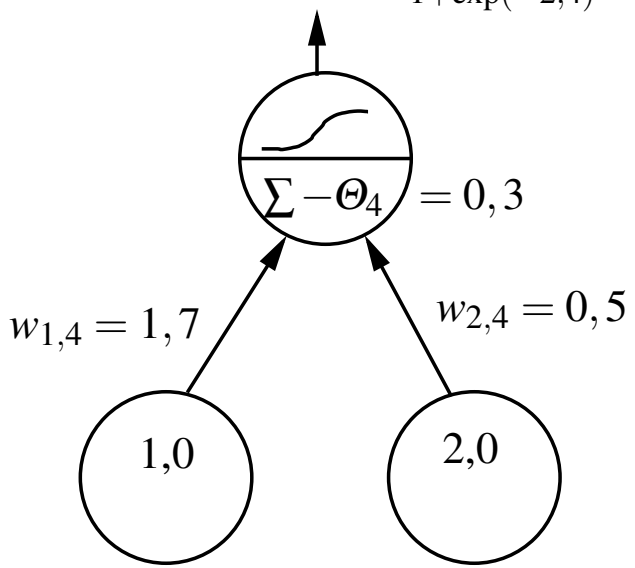
Deterministische Umweltselektion



Neuronale Netze



$$\begin{aligned} &\text{sigmoid}(1,7 \cdot 1,0 + 0,5 \cdot 2,0 - 0,3) \\ = \ &\text{sigmoid}(2,4) = \frac{1}{1+\exp(-2,4)} = 0,125 \end{aligned}$$



Modernisiertes EP

Idee

- die Gewichte $w_{i,j}$ und Schwellwerte Θ_i werden als reellwertige Zahlen gespeichert
- $\mathcal{G} = \mathbb{R}^l$
- Mutation mit Gauß-Verteilung
- alternative Selbstanpassung mit l Strategieparametern

Moderne EP-Mutation

SELBSTADAPTIVE-EP-MUTATION(Individuum A mit $A.G \in \mathbb{R}^l$ und

```
1  for each i ∈ {1, ..., l}
2  do ⌜ u' ← wähle zufällig gemäß N(0, A.S_i · α   (Anpassungsparame
3       B.S_i ← A.S_i + u'
4       B.S_i ← max{B.S_i, ε   (kleinste Standardabweichung) }
5       u ← wähle zufällig gemäß N(0, A.S_i)
6     ⌞ B.G_i ← A.G_i + u
7  return B
```

EVOLUTIONÄRES-PROGRAMMIEREN-1980ER(Zielfunktion F)

```
 1  t ← 0
 2  P(t) ← erzeuge Population mit μ (|Populationsgröße|) Individuen
 3  bewerte P(t) durch F
 4  while Terminierungsbedingung nicht erfüllt
 5  do ⌜ P' ← P(t)
 6        for i ← 1, ..., μ
 7        do ⌜ (|sei P(t) = ⟨A^(i)⟩_{1≤i≤μ}|)
 8              B ← REELLWERTIGE-EP-MUTATION(A^(i))
 9              bewerte B durch F
10           ⌟ P' ← P' ∘ ⟨B⟩
11        t ← t + 1
12     ⌟ P(t) ← Selektion aus P' mittels Q-STUFIGE-TURNIER-SEI
13  return bestes Individuum aus P(t)
```

Typische Parameterwerte

Parameter	Wertebereich
Populationsgröße:	20–200, selten bis 500
Anpassungsstärke α:	0,1–0,4, $\frac{1}{l^2}$
minimale Standardabweichung ϵ:	10^{-5}–10^{-3}
Turniergröße:	5–10

Überblick

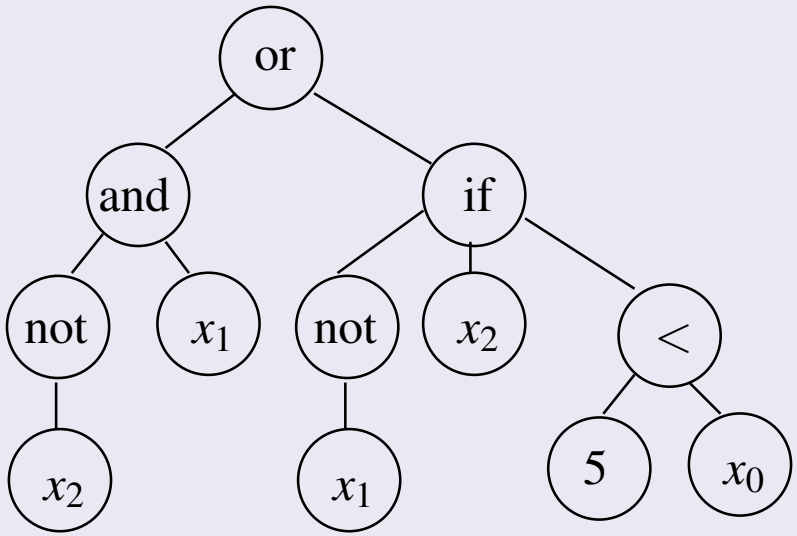
Genetisches Programmieren

Idee

- Computerprogramme werden evolviert
- direkte Darstellung als Syntaxbäume
- anfangs: LISP-Syntax
- später auch: Assembler, Graphen etc.
- immer: Individuen mit variabler Größe
- hier: Bäume in Präfix-Notation
- orientiert am genetischen Algorithmus
- keine Strategieparameter

Beispiel

Syntaxbaum



Präfix-Notation

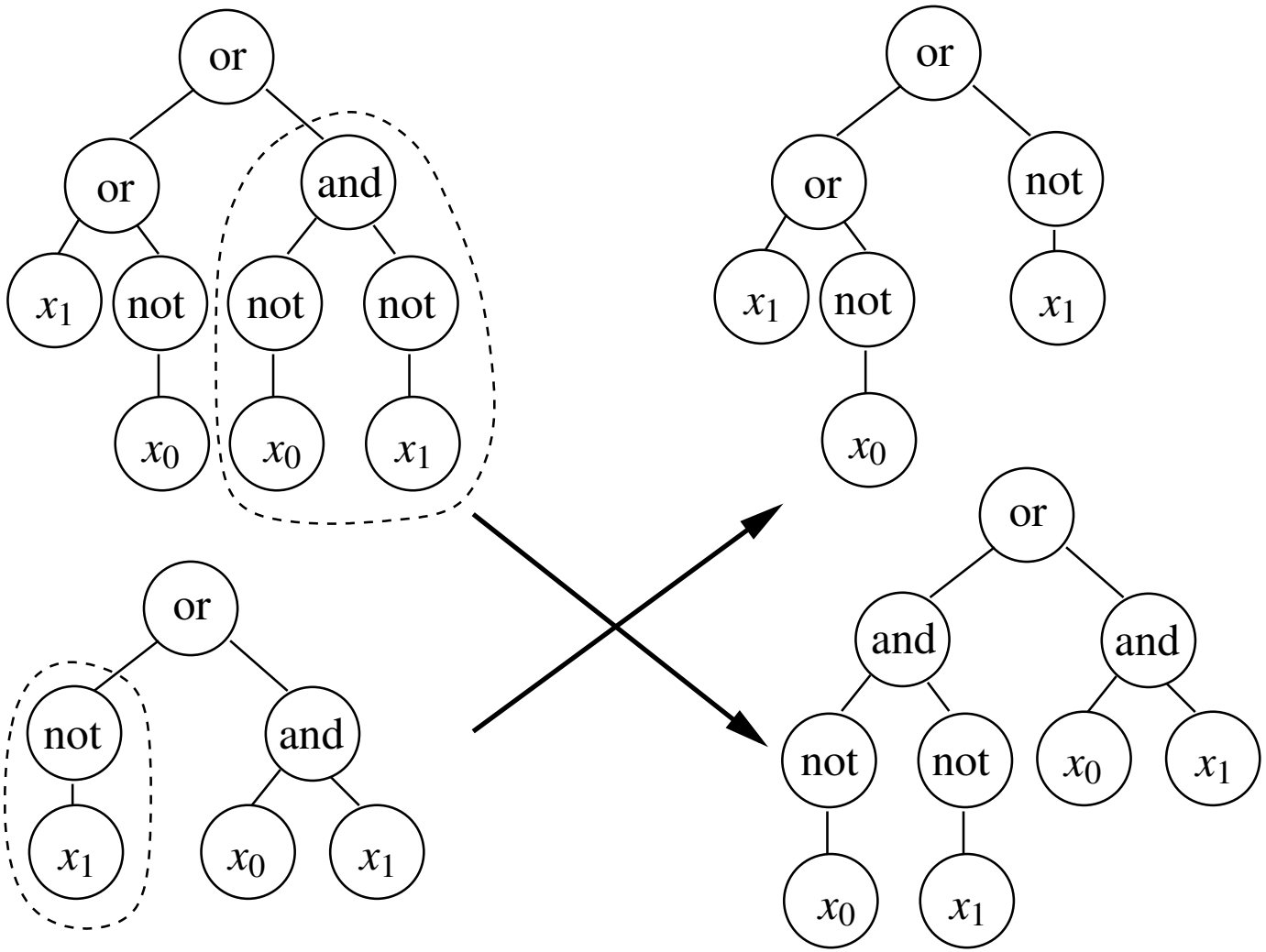
or and not d_2 d_1 if not d_1 d_2 $<$ 5 d_0

GP-Basisoperationen

Basisoperationen

- Entferne($Baum, i$)
- Teilbaum($Baum, i$)
- Erzeugebaum()
- Einfügen($Baum, i, Baum'$)
- Enthalten($Baum, i, j$)
- Größe($Baum$)

Hauptoperator: Rekombination



Hauptoperator: Rekombination

BAUMTAUSCH-REKOMBINATION(Individuen A, B)

1 $i \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, \text{Größe}(A.G)\})$
2 $j \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, \text{Größe}(B.G)\})$
3 $C \leftarrow A$
4 $D \leftarrow B$
5 $Baum \leftarrow \text{Teilbaum}(C.G, i)$
6 $Baum' \leftarrow \text{Teilbaum}(D.G, j)$
7 $C.G \leftarrow \text{Entferne}(C.G, j)$
8 $C.G \leftarrow \text{Einfügen}(C.G, j, Baum')$
9 $D.G \leftarrow \text{Entferne}(D.G, i)$
10 $D.G \leftarrow \text{Einfügen}(D.G, i, Baum)$
11 **return** C, D

Typische Probleme

Größe der Bäume

- können theoretisch beliebig wachsen
- daher oft: maximale Größe oder Baumtiefe vorgegeben
- führt bei Rekombination zu Iterationen

Typkonsistenz

- Annahme: alle Funktionen haben identischen Rückgabetypen
- gilt nur im akademischen Umfeld!
- Typkonsistenz muss zusätzlich beachtet werden

GP-Mutationen

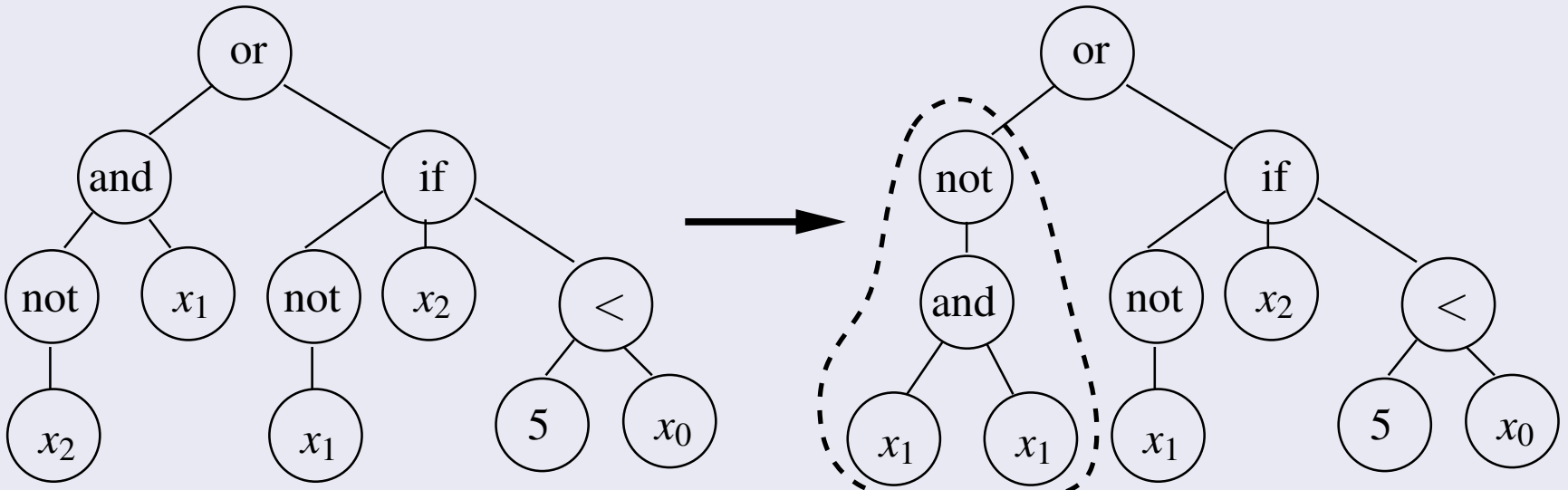
Zufallsbaum

ZUFALLSBAUM-MUTATION(Individuum A)

1 $i \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, \text{Größe}(A.G)\})$
2 $B \leftarrow A$
3 $B.G \leftarrow \text{Entferne}(B.G, i)$
4 $Baum \leftarrow \text{Erzeugebaum}()$
5 $B.G \leftarrow \text{Einfügen}(B.G, i, Baum)$
6 **return** B

GP-Mutationen

Zufallsbaum: Beispiel



GP-Mutationen

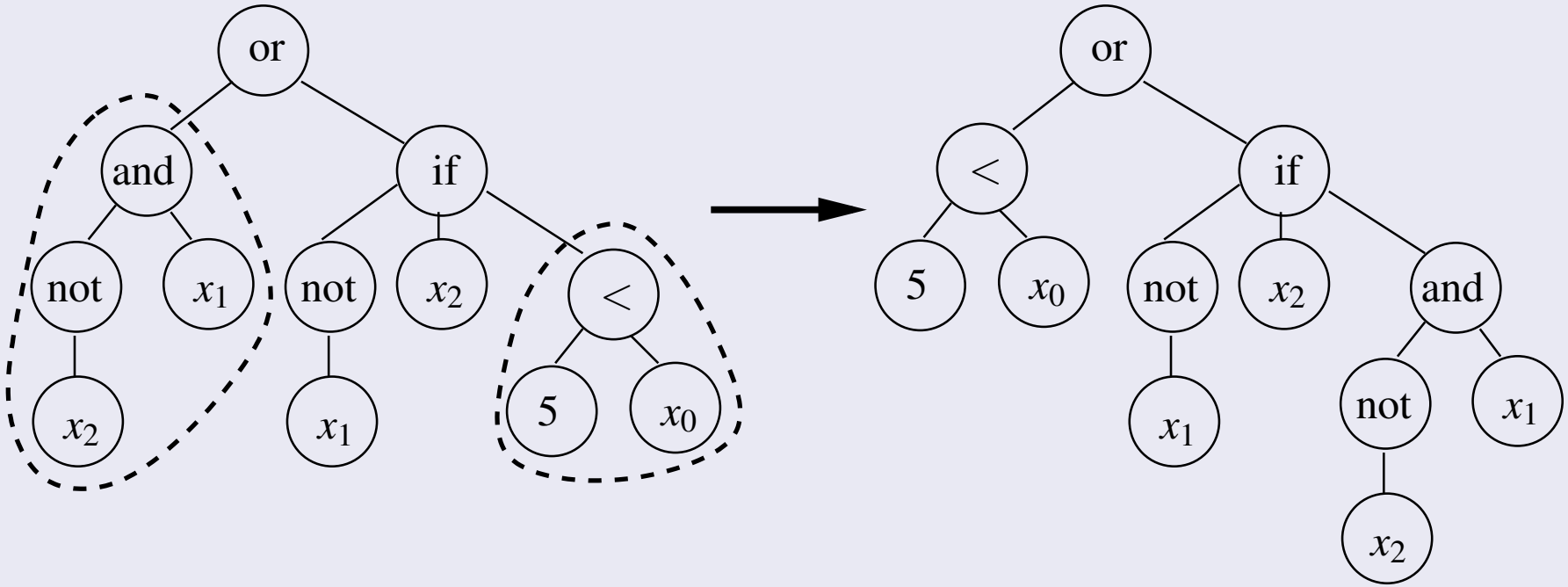
Interne Rekombination

BAUMTAUSCH-MUTATION(Individuum A)

1 **repeat** $\ulcorner\, i \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, \text{Größe}(A.G)\})$
2 $\llcorner\, j \leftarrow$ wähle zufällig gemäß $U(\{1, \ldots, \text{Größe}(A.G)\})$
3 **until** $\neg\text{Enthalten}(A.G, i, j) \wedge \neg\text{Enthalten}(A.G, j, i) \wedge (j > i)$
4 $B \leftarrow A$
5 $Baum \leftarrow \text{Teilbaum}(B.G, j)$
6 $B.G \leftarrow \text{Entferne}(B.G, j)$
7 $Baum' \leftarrow \text{Teilbaum}(B.G, i)$
8 $B.G \leftarrow \text{Entferne}(B.G, i)$
9 $B.G \leftarrow \text{Einfügen}(B.G, i, Baum)$
10 $B.G \leftarrow \text{Einfügen}(B.G, j - \text{Größe}(Baum') + \text{Größe}(Baum), Baum')$
11 **return** B

GP-Mutationen

Interne Rekombination: Beispiel



Initialisierung

. . . der Individuen

- bis Tiefe *mtief* „wachsen“ lassen
- vollständiger Baum mit Tiefe *mtief*

. . . der Population

- erwünscht: Strukturvielfalt
- $2 \cdot$ *mtief* Teile der Population
- Tiefe: $1, \ldots,$ *mtief*
- Verfahren: wachsen, vollständig

Parameterwerte

Parameter	Wertebereich
Populationsgröße:	200–5000
Rekombination/Mutation/Klonen:	80/10/10

Beispiel: Regression

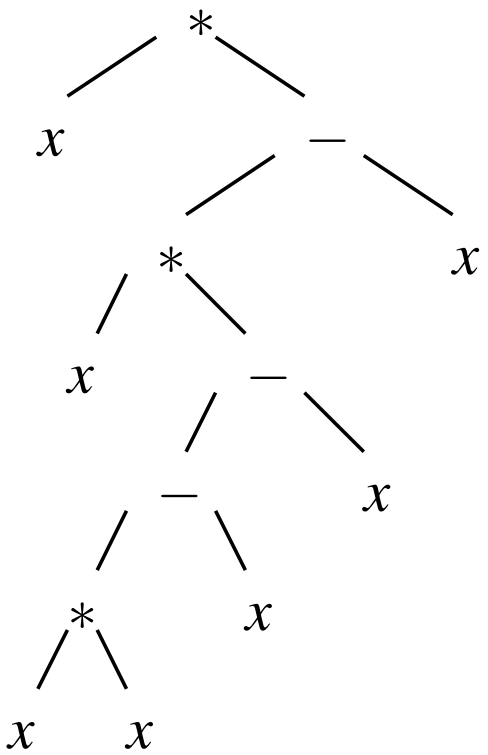
Ziel

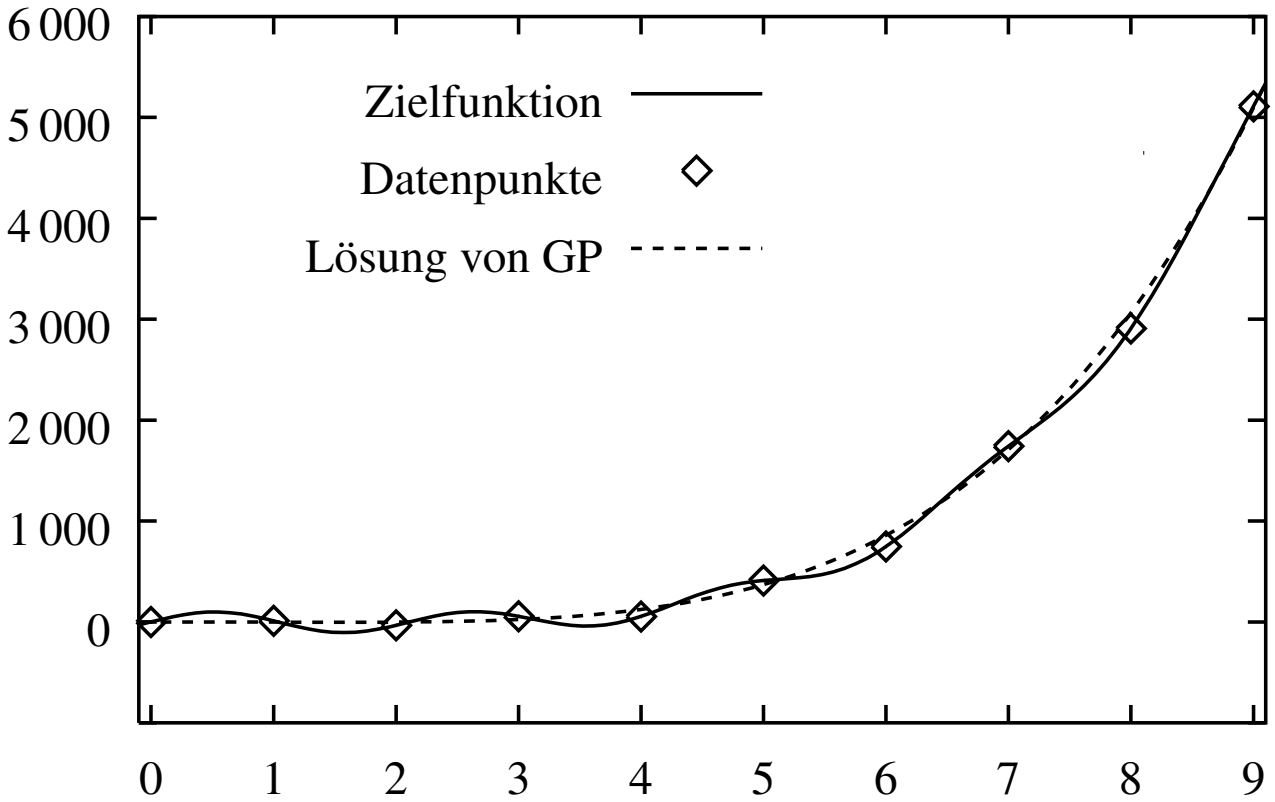
- anzunähern ist: $x^4 - 2x^3 - x^2 - x + 100 \sin(3x)$
- gegeben: Werte an den Stützstellen $\{0,\ 1,\ \ldots,\ 9\}$

Algorithmus

- Terminal: x
- Funktionen: $\{*, +, -, \%\}$
- Populationsgröße: 500
- Generationen: 200

Ergebnis


*
x
—
*
x
x
—
—
x
*
x
x
x


6 000
5 000
4 000
3 000
2 000
1 000
0
0
1
2
3
4
5
6
7
8
9
Zielfunktion
Datenpunkte
Lösung von GP

Überblick

Grundalgorithmus

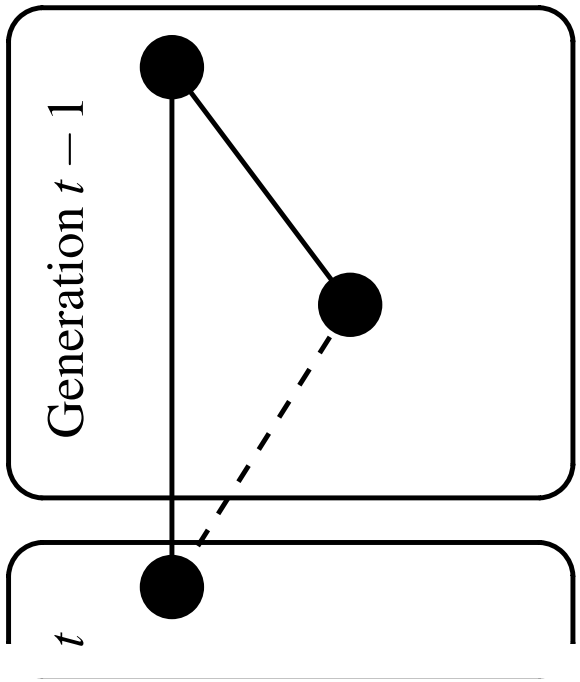
LOKALE-SUCHE(Zielfunktion F)

```
 1  t ← 0
 2  A(t) ← erzeuge Lösungskandidat
 3  bewerte A(t) durch F
 4  while  Terminierungsbedingung nicht erfüllt
 5  do ⌜ B ← variiere A(t)
 6        bewerte B durch F
 7        t ← t + 1
 8        if Akz(A(t − 1).F, B.F, t)  (|Akzeptanzbedingun
 9        then ⌊ A(t) ← B
10           ⌞ else  ⌊ A(t) ← A(t − 1)
11  return A(t)
```

Ablauf

Identität als Elternselektion und Mutation

Akzeptanz-kriterium



Hillclimbing

Idee

- nur Verbesserungen werden akzeptiert

Algorithmus

AKZEPTANZ-HC(Elterngüte $A.F$, Kindgüte $B.F$)

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

Simulated Annealing

Idee

- Verbesserungen immer akzeptieren
- Verschlechterungen mit einer Wahrscheinlichkeit
  - umso geringer je größer die Verschlechterung
  - abnehmend mit voranschreitender Zeit
    ($Temp_{t+1} = Temp_t \cdot \alpha$)

Algorithmus

AKZEPTANZ-SA(Elterngüte $A.F$, Kindgüte $B.F$)

1 **if** $B.F \succ A.F$
2 **then** ⌊ **return** *wahr*
3 **else** ⌜ $u \leftarrow$ wähle zufällig aus $U([0,\ 1])$
4 **if** $u \leq \exp\left(-\frac{d_{euk}(A.F,B.F)}{Temp_{t-1}}\right)$
5 **then** ⌊ **return** *wahr*

Schwellwertakzeptanz

Idee

- Verbesserungen immer
- Verschlechterungen kleiner als ein Schwellwert
- Schwellwert nimmt analog zu Simulated Annealing ab

Algorithmus

AKZEPTANZ-TA(Elterngüte $A.F$, Kindgüte $B.F$)

1 **if** $B.F \succ A.F$ oder $d_{euk}(A.F, B.F) \leq Temp_t$
2 **then** [**return** *wahr*
3 **else** [**return** *falsch*

Sintflutalgorithmus

Idee

- „ansteigender“ Wasserpegel (Gütebereich), der nicht betreten werden darf
- ansonten: alles erlaubt

Algorithmus

AKZEPTANZ-GD(Elterngüte $A.F$, Kindgüte $B.F$)

1 **if** $B.F \succ W$ (Anfangswasserstand) $+ t \cdot R$ (Regengesc
2 **then** [**return** *wahr*
3 **else** [**return** *falsch*

Rekordorientiertes Wandern

Idee

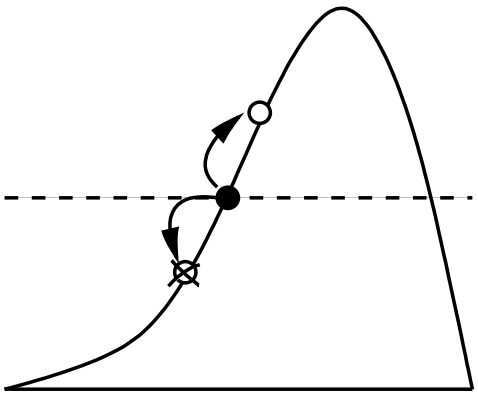
- erlaubte Verschlechterung wird auf das beste bisher gefundene Individuum bezogen
- sonst wie Schwellwertakzeptanz

Algorithmus

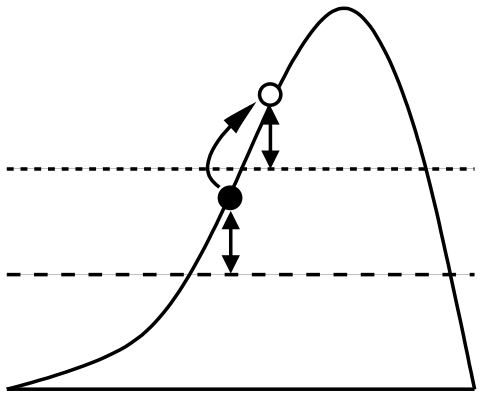
AKZEPTANZ-RR(Elterngüte $A.F$, Kindgüte $B.F$, beste gefundene Gü

```
1  if B.F ≻ besteF
2  then ⌜ besteF ← B.F
3       ⌞ return wahr, besteF
4  else ⌜ if d_euk(B.F, besteF) < Temp_t
5       ⌞ then ⌊ return wahr, besteF
6  return falsch, besteF
```

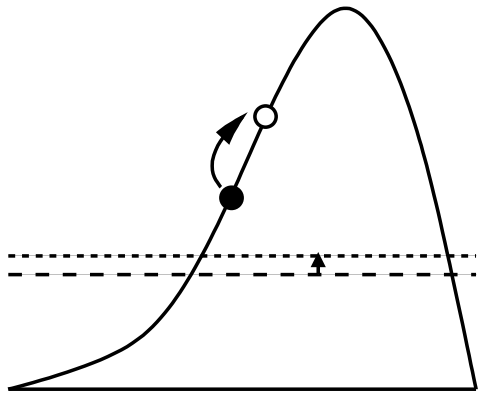
Vergleich



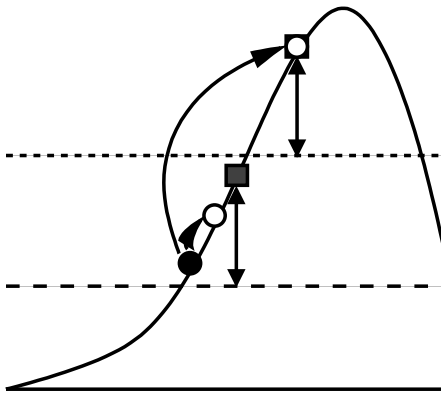
Hillclimbing



Schwellwertakzeptanz



Sintflut-Algorithmus



Rekordorientiertes Wande