# Fundamente der Computational Intelligence

Dozent: Günter Rudolph
Vertretung: Nicola Beume

Wintersemester 2006/07
Universität Dortmund
Fachbereich Informatik
Lehrstuhl für Algorithm Engineering (LS11)
Fachgebiet *Computational Intelligence*

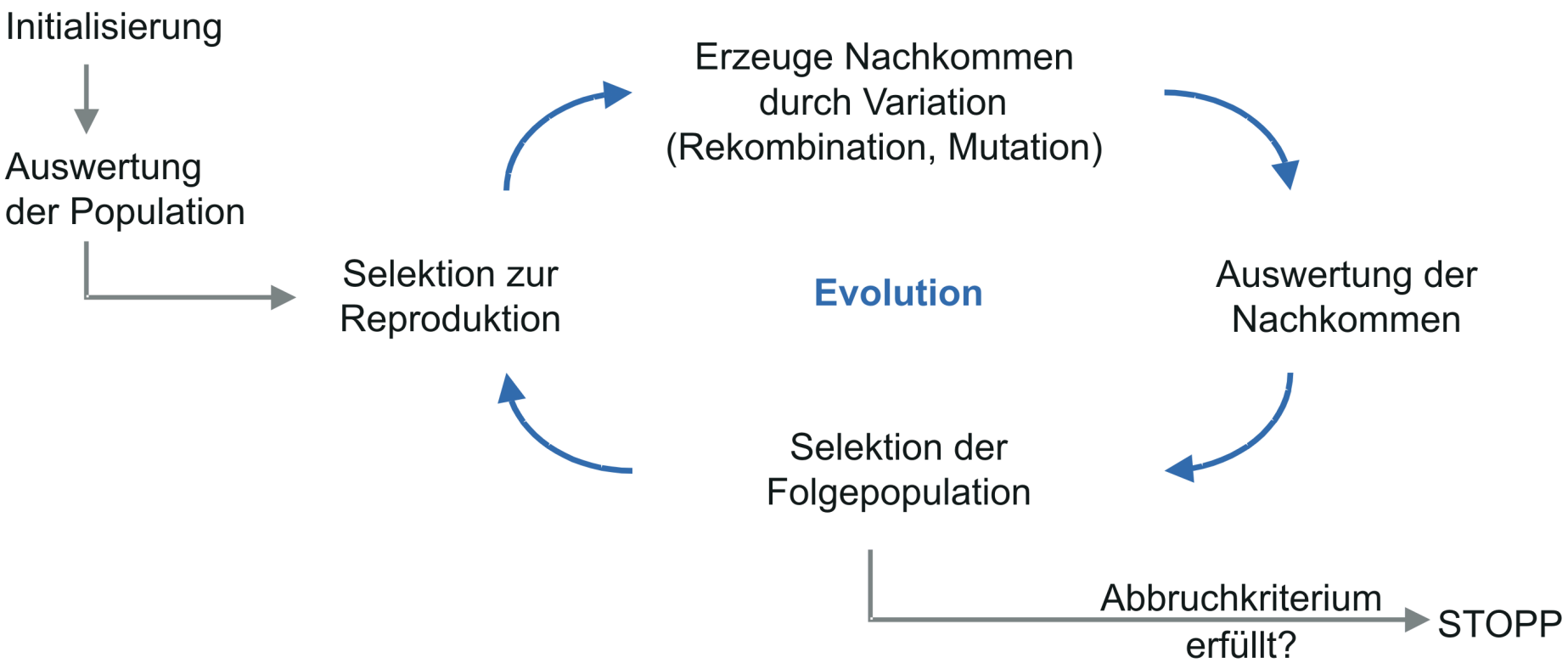
22.11.2006

# Themengebiet II:
# Optimierung mittels evolutionärer Algorithmen (EA)

## Themen von Montag:

- Was ist ein Optimierproblem?
- Welche Optimierprobleme sind schwierig?
- Was ist ein evolutionärer Algorithmus (EA)?
- Fachbegriffe
- Anwendungsgebiet
- Algorithmische Einordnung
- Beispiel: (1+1)EA
- Design-Richtlinien für Operatoren
- Beispiele üblicher Operatoren

# Evolutionäre Algorithmen (EA)



# Nachtrag zu Themen von Montag (20.11.2006)

Rekombination heißt
diskret, falls Wert eines Elter unverändert übernommen wird
arithmetisch, falls Nachkommenwert Vermischung der Elternwerte

Selektion heißt
elitistisch, falls aktuell fitness-beste Individuen immer überleben

Design-Richtlinien:
besagen, was zu tun ist, damit EA Suchraum zufällig durchsucht
Falls Problemwissen vorhanden:
problemspezifische Ausrichtung der Operatoren möglich,
sollte allerdings bewusst geschehen

# Themengebiet II:
# Optimierung mittels evolutionärer Algorithmen (EA)

Themen der heutigen Vorlesung:

- Andere randomisierte (lokale) Suchverfahren
- Parameter und typische Werte
- Theoretische Sichtweise

# Parameter und Basiswerte

Populationgröße: üblicherweise konstant während des Optimierprozesses
$\mu$: Größe der aktuellen Population
$\lambda$: Anzahl erzeugter Nachkommen pro Generation

Notation für Plus-/Komma-Selektion
$(\mu + \lambda)$ EA
$(\mu, \lambda)$ EA, $\lambda \geq \mu$
Selektionsdruck $5 \leq \lambda/\mu \leq 10$

Rekombination
optional, Anwendung mit Wkeit $p_r$
Mutation
wird immer durchgeführt, Anwendung mit Wahrscheinlichkeit 1
Stärke probabilistisch:
MutationsWkeit $p_m$,
Zufallsvariable für Anzahl Basis-Mutationen

# Parametersteuerung

- statisch
  wähle konstante Belegung
- adaptiv
  Veränderung gemäß statischer Regel, abhängig von Optimierverlauf
  z.B. Reduktion der Variation mit steigender Generationszahl
- selbstadaptiv
  Parameter sind Evolution unterworfen
  Idee: gute Individuen vererben ihre Parameterwerte
  mutiere Parameter der Individuen der aktuellen Generation
  erzeuge neue Individuen (durch bereits veränderte Variationsparameter)
  neues Individuum erhält Strategieparameter der/s Elter

# Andere randomisierte Suchverfahren

Lokale Suchverfahren:
1 Suchpunkt, wähle Folgepunkt aus Nachbarschaft von aktuellem
Idee: iterative lokale Verbesserung

Nachbarschaft definiert über Nähe im Raum, gemessen durch Metrik bzw. Norm

Def.: Norm
Funktion $|| \cdot || : S \to \mathbb{R}_0^+$ heißt Norm auf $S :\iff \forall \mathbf{x}, \mathbf{x}' \in S$ gilt:
$||\mathbf{x}|| \geq 0$ und $||\mathbf{x}|| = 0 \iff \mathbf{x} = 0$ Nichtnegativität
$||\alpha\mathbf{x}|| = |\alpha| \cdot ||\mathbf{x}||$ Homogenität
$||\mathbf{x} + \mathbf{x}'|| \leq ||\mathbf{x}|| + ||\mathbf{x}'||$ Dreiecksungleichung

Unterschied zu Metrik: Homogenität statt Symmetrie
auf normiertem Raum ist Metrik definierbar durch $d_S(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||$

Nachbarschaft $N_\alpha(\mathbf{x})$ von $\mathbf{x}$:
$\mathbf{x}' \in N_\alpha(\mathbf{x})$, falls $d_S(\mathbf{x}, \mathbf{x}') \leq \alpha$ $d_S$ Metrik (basierend auf Norm)

# Arten der Nachbarschaftssuche

- erstes
  übernehme jeden erzeugten Punkt als aktuellen
- erste Verbesserung
  wähle ersten gesehenen Punkt mit besserem Funktionswert
  Nachbarschaft deterministisch abgesucht
- erste Verbesserung bei zufälliger Reihenfolge
  wähle zufällige Permutation auf der Nachbarschaft
  wähle ersten gesehenen Punkt mit besserem Funktionswert
- bestes
  betrachte gesamte Nachbarschaft
  wähle besten Punkt als Folgepunkt

# Lokale Suchverfahren

reine Zufallsuche
Iteration: wähle Suchpunkt zufällig gleichverteilt im Suchraum
erster Punkt

zufällige lokale Suche
wähle initialen Suchpunkt $\mathbf{x}_0$ zufällig gleichverteilt
Iteration: wähle Suchpunkt $\mathbf{x}' \in N(\mathbf{x}_t)$ zufällig gleichverteilt
Falls $f(\mathbf{x}')$ besser als $f(\mathbf{x}_t)$: setze $\mathbf{x}_{t+1} = \mathbf{x}'$, sonst $\mathbf{x}_{t+1} = \mathbf{x}_t$
erste Verbesserung

hill climber
wähle initialen Suchpunkt $\mathbf{x}_0$ zufällig gleichverteilt
betrachte Punkte in $N(\mathbf{x}_t)$
wähle $\mathbf{x}_{t+1} = argmax\{f(\mathbf{x}')|\mathbf{x}' \in N(\mathbf{x}_t)\}$
bester Punkt
nur für endliche Nachbarschaft

# Lokale Suchverfahren

## Metropolis

wähle initialen Suchpunkt $\mathbf{x}_0$ zufällig gleichverteilt
Iteration: wähle Suchpunkt $\mathbf{x}' \in N(\mathbf{x}_t)$ zufällig gleichverteilt
Falls $f(\mathbf{x}')$ besser als $f(\mathbf{x}_t)$: setze $\mathbf{x}_{t+1} = \mathbf{x}'$,
sonst übernehme $\mathbf{x}_{t+1} = \mathbf{x}'$ mit Wkeit $e^{(f(\mathbf{x}')-f(\mathbf{x}_t))/T}$
$T$ konstant
Verschlechterungen zugelassen

## Simulated Annealing

wie Metropolis, mit $T$ monoton fallend mit der Zeit (Anzahl Funktionsausw.)
Wkeit für Akzeptanz von Verschlechterung sinkt

## Tabu-Suche

Idee: Archiviere gesehene Lösungen und verbiete wiederholten Besuch
wähle initialen Suchpunkt $\mathbf{x}_0$ zufällig gleichverteilt
Iteration: wähle Suchpunkt $\mathbf{x}' \in N(\mathbf{x}_t)$ zufällig gleichverteilt
Tabuliste speichert letzte $k$ Suchpunkte oder deren Eigenschaften
Kurzzeitgedächtnis

# Komplexitätsmodell für Black-Box-Optimierung

Komplexität: Schwierigkeit von Problem
gemessen in Ressourcen, die zur Lösung benötigt sind
Ressourcen ausgedrückt gemäß Bezugsgröße

Annahme:
Zielfunktionsauswertung zeitaufwändigste Komponente des EA (z.B. Simulation)
Zeit für Zielfunktionsauswertung unbekannt

Anzahl Funktionsauswertungen angemessenes Maß für
zeitliche Ressourcen, verwendete Informationen
formuliert bzgl. Dimension des Suchraums $n$ (#Entscheidungsvariablen)

Funktionenklasse $\mathcal{F}$:
Verallgemeinerung von $f \in \mathcal{F}$
$f$ : Rundreise durch Städte DO, BO, E (Instanz bestimmter Distanzmatrix),
$\mathcal{F}$: Rundreise durch $n$ Städte

Funktionenklasse bekannt, tatsächliche Funktion nicht

# Black-Box-Komplexität

Anzahl Funktionsauswertungen als Komplexitätsmaß:
Maß für Schwierigkeit von Problem

Optimierzeit von BlackBox-Algorithmus $A$ auf Funktion $f$
$T_{A,f} = 1 + \min\{t \mid f(\mathbf{x}_t) = \min\{f(\mathbf{x}) | x \in S\}\}$
Anzahl Funktionsauswertungen zu der erstmalig ein bester Suchpunkt gesehen
$T_{A,f}$ Zufallsvariable,
Erwartungswert $E(T_{A,f})$: erwartete Optimierzeit:

Worst-Case erwartete Optimierzeit:
$T_{A,\mathcal{F}} = \{E(T_{A,f}) \mid f \in \mathcal{F}\}$
erwartete Optimierzeit von schwierigster Instanz aus $\mathcal{F}$

Black-Box-Komplexität von Funktionsklasse $\mathcal{F}$ :
$B_{\mathcal{F}} = \min\{T_{A,\mathcal{F}} \mid$ Black-Box-Algorithmus für $\mathcal{F}\}$
Worst-Case erwartete Optimierzeit des besten Algorithmus für $\mathcal{F}$
Anzahl Funktionsauswertungen des schnellsten Algorithmus auf schwierigster Instanz der Klasse

# Komplexitätsmaß

## Anzahl Funktionsauswertungen als Komplexitätsmaß

Nachteil:
Berechnungen des EA zählen nicht
möglicherweise exponentielle Laufzeit des EA wird nicht mitgerechnet
kleine obere Schranken nicht sehr aussagekräftig

Vorteil:
große Black-Box-Komplexität sehr aussagekräftigt:
Problem bei beliebiger zusätzlicher Laufzeit noch schwierig,
wenig Informationsgehalt pro Funktionsauswertung

# (1+1)EA in $\mathbb{B}^n$

$t = 0$ — Generationenzähler $t$
Wähle $\mathbf{x}_0 \in \mathbb{B}^n$ gleichverteilt zufällig — Initialisierung
$y_0 = f(\mathbf{x}_0)$ — Funktionsauswertung

Tue — Generationsschleife
  $\mathbf{x}' = \text{Standard-Bit-Mutation}(\mathbf{x}_t)$, $p_m = 1/n$ — Variation: Mutation
  $y' = f(\mathbf{x}')$ — Fitness=Zielfunktion
  Falls $y' \leq y_t$ — Minimierung
    $\mathbf{x}_{t+1} = \mathbf{x}'$; $y_{t+1} = y'$
  sonst
    $\mathbf{x}_{t+1} = \mathbf{x}_t$; $y_{t+1} = y_t$ — Folgepopulation: Lösung $\mathbf{x}_{t+1}$
  $t = t + 1$ — Generationenzähler erhöhen
bis Abbruchkriterium erfüllt — Abbruchkriterium

# Eigenschaften in $\mathbb{B}^n$

- Metrik in $\mathbb{B}^n$: Hamming-Abstand
  $H(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{n}(x_i + x_i' - 2x_i x_i')$
- Nachbarschaft: Standard-Bit-Mutation: $N_n$
- Standard-Bit-Mutation mit $p_m = 1/n$
  typische Wkeiten:
  $Pr(\text{bestimmtes Bit mutiert}) = 1/n$
  $Pr(\text{bestimmtes Bit mutiert nicht}) = 1 - 1/n$
  $\mathrm{E}(\#\text{mutierter Bits}) = n \cdot 1/n = 1$
- Selektion elitistisch

# Einfaches Szenario

betrachte einfache Funktion:
OneMax($\mathbf{x}$)= $\sum_{i=1}^{n} x_i$
Maximierung, Optimum: $1^n$
Unimodale Funktion: lokales Optimum = globales Optimum

Def.: lokales Optimum (Maximierung)
Für Funktion $f : \mathbb{B}^n \to \mathbb{R}$ heißt $\mathbf{x} \in \mathbb{B}^n$ lokales Maximum, falls gilt:
$\forall \mathbf{x}' \in N_1(\mathbf{x})$: $f(\mathbf{x}') \leq f(\mathbf{x})$

Welche Informationen für Analyse benötigt?
aktueller Suchpunkt
Vergangenheit bringt keine zusätzliche Information
gedächtnisloser Prozess $\to$ Markov-Kette

# Methode für obere Schranken der Optimierzeit

## Fitnessbasierte Partitionen

fasse Suchpunkte gleicher Fitness zu Partitionen zusammen
numeriere Partitionen gemäß aufsteigender Fitness
alle Elemente höchster Partition optimal (fertig)
Selektion elitistisch: verlasse Partition nur zu besserer

$Pr(\mathbf{x} \text{ mutiert zu } \mathbf{x}') : p_m^{H(\mathbf{x},\mathbf{x}')}(1-p_m)^{n-H(\mathbf{x},\mathbf{x}')}$
mutiere $H(\mathbf{x},\mathbf{x}')$ Bits, mutiere $n - H(\mathbf{x},\mathbf{x}')$ Bits nicht

$s_i$ :VerlassensWkeit für Partition $L_i$
$s_i = \min_{\mathbf{x}\in L_i} \sum_{i<j\leq k} \sum_{\mathbf{x}'\in L_j} p_m^{H(\mathbf{x},\mathbf{x}')}(1-p_m)^{n-H(\mathbf{x},\mathbf{x}')}$
innere Summe: alle $\mathbf{x}'$ einer höheren Partition
äußere Summe: alle höheren Partition
min: ungünstigstes $\mathbf{x}$

erw. Optimierzeit: Summe der Aufenthaltsdauer pro Partition
Aufenthaltsdauer = 1/VerlassensWkeit
$E[T_{(1+1)EA,f}] \leq \sum_{0\leq i<k} s_i^{-1}$

# Obere Schranke für OneMax

## Fitnessbasierte Partitionen

Nützliche Ungleichung: $(1-1/n)^n < 1/e < (1-1/n)^{n-1}$

Vektoren in Partition $i$: $i$ 1-Bits, $n-i$ 0-Bits
mögliche Verbesserung: mutiere $0 \rightarrow 1$, restliche Bits unverändert
($\rightarrow$ Partition verlassen)

$Pr(0 \rightarrow 1) = \#0\text{-Bits} \cdot p_m = (n-i)/n$
Pr(restliche Bits mutieren nicht)$= (1-p_m)^{n-1} = (1-1/n)^{n-1} > 1/e$

untere Schranke für VerlassensWkeit:
$s_i \geq \frac{n-i}{n} \cdot (1-\frac{1}{n})^{n-1} \geq \frac{n-i}{n} \cdot \frac{1}{e} = \frac{n-i}{ne}$

$E(T_{(1+1)EA,OneMax}) \leq \sum_{0 \leq i < n} s_i^{-1} \leq \sum_{0 \leq i < n} \frac{en}{n-i} = en \sum_{1 \leq i \leq n} \frac{1}{i}$
$= enH_n < en(\ln(n)+1) = O(n \log n)$

# Zusammenfassung

Themen der heutigen Vorlesung:

- Parameter und typische Werte
- Andere randomisierte (lokale) Suchverfahren
- Black-Box-Komplexität
- Begriffe der Laufzeitanalyse
- Einfache Laufzeitanalyse in $\mathbb{B}^n$

# Literatur

- EA
  Vorlesungen von Thomas Jansen (in diesem Semester),
  Ingo Wegener, Hans-Georg Beyer
- Lokale Suchverfahren
  Vorlesung von Ingo Wegener (in diesem Semester)
- Black-Box-Komplexität
  *Komplexitätstheorie – Grenzen der Effizienz von Algorithmen*
  von Ingo Wegener