UNIKASSEL VERSITÄT

SOFTWARE ENGINEERING
UNIVERSITÄT KASSEL

Umgang mit Parametern

Jens Kosiol

(teilweise basierend auf Folien von Eiben/Smith)

Parameter in Evolutionären Algorithmen

- Ein Evolutionärer Algorithmus hängt von unheimlich vielen Parametern ab, die konfiguriert werden müssen:
  - Welche Variationsoperatoren werden benutzt?
  - Welche Selektionsoperatoren werden benutzt?
  - Wie groß ist die Population?
  - Welche Mutations- und Crossoverrate wird gewählt?
  - …
- Die Wahl der Parameterwerte hat entscheidenden Einfluss auf den Erfolg von evolutionärer Suche.
- Für unterschiedliche Optimierungsprobleme und sogar unterschiedliche Probleminstanzen können unterschiedliche Parameterwerte von Vorteil sein.

Qualitätsmaße für evolutionäre Algorithmen

Je nach Anwendungsfall können unterschiedliche Eigenschaften von evolutionären Algorithmen gewünscht sein:

- Finden einer Lösung möglichst hoher Fitness
- Schnelle Konvergenz zu einer Lösung akzeptabler Qualität
- „Robustes“ Verhalten in Bezug auf
  - Laufzeit
  - Qualität der gefundenen Lösungen
- …

Was eine gute Parameterwahl ist, hängt auch entscheidend vom Einsatzzweck ab!

Unterschiedliche Parameterarten

Symbolische Parameter

- Wahl eine Möglichkeit aus einer endlichen Klasse von Möglichkeiten.
- Raum der Werte hat keine Struktur.
- Beispiele:
  - Art des Crossoveroperators
  - Elitismus: ja oder nein?
  - Selektionsmethode

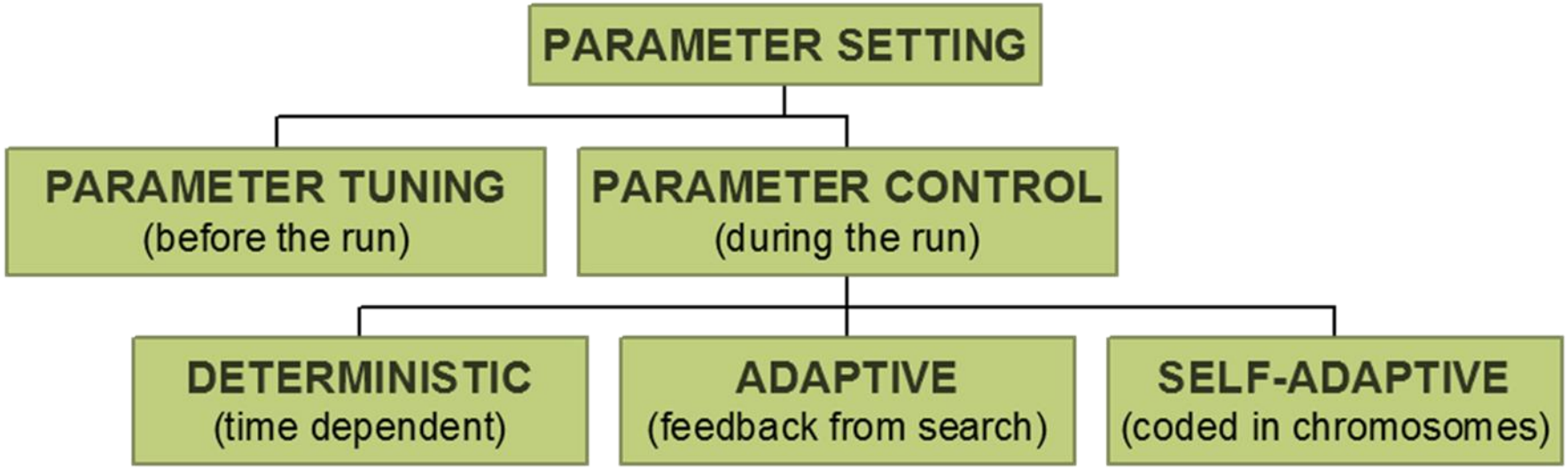
Numerische Parameter

- Parameterart, die Werte in (einer Teilmenge von) z.B. $\mathbb{N}$ oder $\mathbb{R}$ einnimmt.
- Raum der Werte hat (gut durchsuchbare) Struktur.
- Beispiele:
  - Populationsgröße
  - Mutations- und Crossoverrate
  - Turniergröße

Die Wahl eines symbolischen Parameters zieht häufig die Notwendigkeit der Wahl von numerischen Parametern nach sich:

- Turnierselektion $\rightarrow$ Turniergröße?
- k-Punkt-Crossover $\rightarrow$ Welches k?
- …

Taxonomie für Umgang mit Parametern nach Eiben/Smith



Klassischer Tuning-Ansatz

Der klassische Tuning-Ansatz: generate and test. Vor dem eigentlichen Einsatz eines evolutionären Algorithmus werden in Probeläufen verschiedene Parameterwerte ausprobiert und der Erfolg gemessen.

Probleme:

- Zeitintensiv
- Expertenwissen über evolutionäre Algorithmen notwendig
- Wegen Parameter-Interaktionen sind erschöpfende Tests nicht möglich

Modernes Parametertuning: Hyperparameteroptimierung

- Fasse das Finden guter Parameterwerte selbst als Optimierungsproblem auf:
  - Aus Einsatzzweck ergeben sich Optimierungsziele (schnelle Konvergenz, Finden möglichst guter Lösungen, …)
  - Suchraum ergibt sich aus den Wertebereichen der zu optimierenden Parameter; z.B. Mutationswahrscheinlichkeit $p_{\mathrm{m}} \in [0,1] \subset \mathbb{R}$; Populationsgröße $\mu \in \mathbb{N}$.
- Wende Optimierungsverahren an, um optimale Werte zu finden:
  - gradientenbasierte Methoden
  - Evolutionäre Algorithmen
  - …

Grundprobleme Parametertuning

Parametertuning hat die folgenden grundsätzlichen Probleme:

- Gefundene optimale Werte für Parameter sind im Allgemeinen nicht verallgemeinerbar, sondern unterscheiden sich von Problem zu Problem und sogar zwischen verschiedenen Instanzen des gleichen Problems.
- Für viele Parameterwerte (Populationsgröße, Mutationsrate, Crossoverrate, Selektionsdruck) hängt der optimale Wert auch vom Zustand der Suche oder sogar vom betrachteten Individuum ab.

Typischer Ablauf evolutionärer Algorithmus (Wdh.)

Phasen während der Optimierung einer eindimensionalen Fitnesslandschaft

Frühe Phase:

(quasi-)zufällige Verteilung der Population

Mittlere Phase:

Population um Optima herum arrangiert

Späte Phase:

Population bei ausgeprägten Optima konzentriert

Gibt es überhaupt eine optimale Mutationsrate?

- Wäre es nicht vorteilhaft, wenn die Mutationsrate mit dem Lauf des Algorithmus abnimmt?
- Wäre es nicht vorteilhaft, wenn Individuen in der Nähe von lokalen Optima eine niedrigere Mutationsrate hätten als andere Individuen?

Grundsätze der Parameterkontrolle

- Ein Parameter kann variabel sein und seinen Wert während des evolutionären Algorithmus ändern.
- Änderung kann gemäß verschiedener Strategien geschehen:
  - Deterministisch mittels einer zeitabhängigen Funktion
  - Adaptiv, indem Feedback vom Zustand der Suche verwendet wird (Beispiel: Diversität der Population unterschreitet festgelegten Schwellwert → Crossoverrate wird erhöht)
  - Selbstadaptiv: Parameterwerte werden Teil der Kodierung eines Individuums und unterlaufen selbst Evolution

Probleme Parameterkontrolle

- Wahl der Parameter wird teils nicht unnötig, sondern bloß verschoben und ggf. sogar verkompliziert:
  - Statt Crossoverrate bestimme Schwellwert für Diversität, ab dem sich die Crossoverrate erhöhen soll
  - Statt Anzahl der Nachkommen in $(\mu + \lambda)$, bestimme eine Funktion $\lambda(t)$, die Anzahl der Nachkommen in Abhängigkeit von der Iteration t festlegt.
- Bei Selbstadaption wird die Dimension des Suchraums und die Komplexität des zu lösenden Optimierungsproblems erhöht.
- Adaption und Selbstadaption sind von Natur aus spezifisch für einen Lauf eines evolutionären Algorithmus und nicht (bzw. schwer) wiederverwendbar in anderen Kontexten.

Parameterkontrolle oder Parameter-Tuning?

- Beide Ansätze wurden erfolgreich angewendet.
- Hyperparameteroptimierung ist aktives Forschungsfeld, auch in anderen Gebieten (z.B. Machine Learning).
- Eiben und Smith sehen Indizien, dass Parameter-Tuning besser funktioniert als Parameterkontrolle.
- Mit Parameterkontrolle gibt es mehr Erfahrungen in Evolutionären Algorithmen.

Beispiele für Parameterkontrolle

- Anpassen der mutation step size
  - Zeitabhängig (deterministisch)
  - Erfolgsabhängig (adaptiv)
  - Selbstadaptiv
- Anpassen des Selektionsvorgangs (Simulated Annealing)

Mutation von Vektoren reeller Zahlen (Wdh.)

Verbreitetster Mutationsoperator auf Vektoren reeller Zahlen:

- Gegeben Vektor $(v_1, \dots, v_n)$ wird zu jeder Variable v_i ein um 0 normalverteilt gesampelter Wert addiert. Notation:

$$(v_1, \dots, v_n) \mapsto (v_1', \dots, v_n'),$$
$$v_i' = v_i + N(0, \sigma)$$

- Die Standardabweichung σ ist hier der entscheidende Parameter, der das Ausmaß der Änderung kontrolliert; Name: mutation step size. Mutationswahrscheinlichkeit p_{m} wird auf 1 gesetzt (jedes Chromosom mutiert).

Deterministische Variation der mutation step size

Statt die mutation step size einmal zu fixieren, definiere sie als zeitabhängige Funktion

$$\sigma(t) = 1 - 0.9 \times \frac{t}{T},$$

wobei $0 \leq t \leq T$ die Anzahl der Iterationen angibt und T die obere Schranke für Iterationen.

Eigenschaften:

- Mutation step size startet bei 1 und konvergiert streng monoton fallend gegen 0.1. Es ist a priori klar, welchen Wert $\sigma(t)$ wann haben wird.
- Anzahl der zu betrachtenden Parameter ist reduziert.
- Änderungen an der mutation step size sind unabhängig vom Fortschritt der Suche.
- Die mutation step size ist für alle Individuen gleich.

Adaptive Variation der mutation step size

In Evolutionsstrategien wurde lange Rechenbergs $^1/_5$ success rule verwendet: ca. ein Fünftel der Mutationen sollte die Fitness verbessern.

Vorgehen:

Alle n Iterationen wird die mutation step size gemäß folgender Regel aktualisiert:

$$\sigma(t) = \begin{cases} \sigma(t-n)/_c \text{ falls } p_s > {}^1/_5 \\ \sigma(t-n) \cdot c \text{ falls } p_s < {}^1/_5 \\ \sigma(t-n) \text{ falls } p_s = {}^1/_5 \end{cases}.$$

Hierbei berechnet p_s die Anzahl erfolgreicher Mutationen (Mutationen, die die Fitness erhöht haben) über die letzten n Iterationen und $0 < c \leq 1$ ist ein Parameter, der die Stärke der Adaption steuert.

Eigenschaften $^1/_5$ success rule

- Der Wert, den die mutation step size im Laufe der evolutionären Suche annehmen wird, ist nicht deterministisch und durch den Nutzer kaum kontrollierbar.
- Anzahl der Parameter, die ein Nutzer einstellen muss, hat sich erhöht: Startwert für die mutation step size und den Parameter c.
  - Abhängigkeit von einem guten Startwert für die mutation step size sinkt aber.
  - Für die Wahl von c gibt es Empfehlungen aus der Erfahrung ($0.817 \leq c \leq 1$).
- Der Wert der mutation step size ist abhängig vom Zustand der Suche.
- Die mutation step size ist für alle Individuen gleich.

Selbstadaption der mutation step size

Zentrale Idee: mutation step size wird Teil der Kodierung eines Individuums und unterliegt auch der Evolution.

Vorgehen:

- Individuen $(x_1, \dots, x_n)$ werden nun jeweils als $(x_1, \dots, x_n, \sigma)$ kodiert.
- Ablauf Mutation eines Individuums $(x_1, \dots, x_n, \sigma)$:
  - Mutation step size σ wird mutiert zu σ' per $\sigma' = \sigma \cdot \mathrm{e}^{\tau N(0,1)}$, wobei τ proportional zu ${}^{1}/_{\sqrt{n}}$.
  - Falls σ' einen vorher festgelegten Schwellwert ϵ unterschreitet, wird $\sigma' = \epsilon$ gesetzt.
  - Die restlichen Variablen werden mutiert per $x_i' = x_i + N(0, \sigma')$, wobei für jedes x_i neu gesampelt wird.
  - Das Individuum $(x_1', \dots, x_n', \sigma')$ ist der berechnete Nachkomme.
- Die mutation step size wird auch der Rekombination unterworfen, häufig Durchschnitt der Elternwerte.
- Die Fitness eines Individuums $(x_1, \dots, x_n, \sigma)$ wird weiterhin auf der Projektion $(x_1, \dots, x_n)$ ausgewertet; “Güte” der mutation step size wird durch Ablauf indirekt kontrolliert.

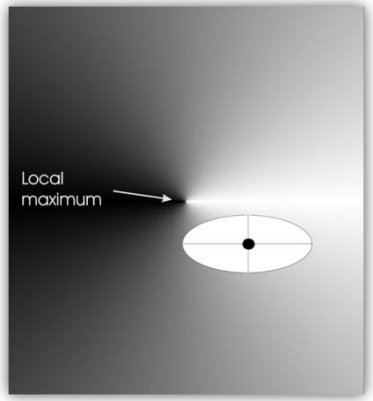
Eigenschaften Selbstadaption der mutation step size

- Der Wert, den die mutation step size im Laufe eines evolutionären Algorithmus annimmt, ist vom Zustand der Suche abhängig und nicht durch den Nutzer kontrollierbar.
- Die mutation step size ist für jedes Individuum spezifisch, aber für jede Variable eines Individuums identisch.
- Die Dimension des Suchraums erhöht sich um 1 und das Problem, das der evolutionäre Algorithmus lösen muss, wird (implizit) komplexer.
- Für das log-normal verteilte Sampeln gilt (für kleines τ):
  - Der Median der Verteilung liegt bei 1, der Erwartungswert nahe bei 1.
  - Ein Wert hat die gleiche Wahrscheinlichkeit, gesampelt zu werden, wie sein Kehrwert. Im Schnitt ist die Mutation der mutation step size also neutral.

Verallgemeinerung der Selbstadaption der mutation step size

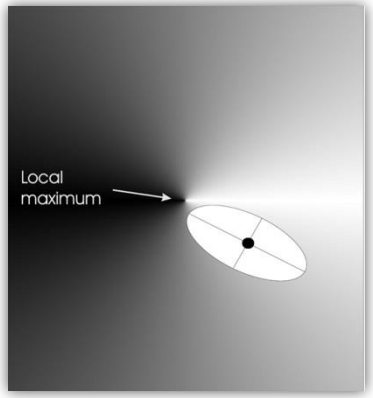
Lernen von n unkorrelierten Schrittgrößen:

- Jede Variable erhält eigene Schrittgröße, die der Mutation unterliegt.
- Individuen werden kodiert als $(x_1, \dots, x_n, \sigma_1, \dots, \sigma_n)$.



Lernen von korrelierten Schrittgrößen:

- Statt n Schrittgrößen wird eine Kovarianzmatrix C gelernt, die (hoffentlich) Abhängigkeiten zwischen den Variablen berücksichtigt.
- Die Problemvariablen werden um einen Wert geändert, der entsprechend der n-dimensionalen Normalverteilung gesampelt wird, den die Kovarianzmatrix C definiert.
- Individuen werden kodiert als $(x_1, \dots, x_n, \sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_{n\cdot(n-1)/2})$
- Algorithmus: Covariance matrix adaptation evolution strategy



Zeitabhängige Anpassung der Selektion

- Das Ergebnis von Turnieren lässt sich randomisieren: Das bessere Element gewinnt nur mit einer gewissen Wahrscheinlichkeit. Diese Wahrscheinlichkeit sollte mit zunehmender Zeit zunehmen.
- Umsetzung: Boltzmann Selektionsmechanismus: Ist Lösung i gegeben und zu entscheiden, ob sie durch die Lösung j ersetzt werden soll, so setze (Minimierungsproblem mit Fitnessfunktion f):

$$P[\text{akzeptiere } j] = \begin{cases} 1 \text{ falls } f(i) \geq f(j) \\ \exp\left(\dfrac{f(i) - f(j)}{K_b \cdot T}\right) \text{falls } f(i) < f(j) \end{cases}.$$

  Hierbei heißt K_b die Boltzmann-Konstante und die Temperatur T nähert sich im Laufe des Algorithmus der 0.
- Wird als Selektionskriterium im Simulated Annealing eingesetzt.