Übungsblatt: Lokale Suche, GA

1 EA.01: Modellierung von GA

1.1 Landkartenfärbeproblem

Problem: Jede Region soll eine Farbe erhalten, wobei benachbarte Regionen nicht die gleiche Farbe haben dürfen. Ziel ist eine gültige Färbung mit möglichst wenigen Farben (Start mit fünf Farben).

Kodierung: Ein Individum ist ein Vektor der Farben aller Regionen:

$$g = [\text{blau, gelb, rot, grün, blau}]$$

Jede Position repräsentiert eine Region, der Wert ist die gewählte Farbe. Damit wird der komplette Zustand kodiert.

Crossover Zwei-Punkt-Crossover: Zwischen zwei zufälligen Positionen wird ein Abschnitt getauscht.

Eltern 1: [blau, **gelb, rot**, grün, blau]
Eltern 2: [rot, **grün, blau**, gelb, rot]
$\Rightarrow$ Kind: [blau, grün, blau, grün, blau]

Durch den Austausch eines Segments entstehen neue Nachbarschaften mit potenziell weniger Konflikten.

Mutation: Ein zufälliges Gen (eine Region) erhält eine neue zufällige Farbe aus dem verfügbaren fünf:

[blau, gelb, rot, grün, blau] $\Rightarrow$ [blau, **violett**, rot, grün, blau]

Es sorgt für Vielfalt und verhindert das Festfahren in lokalen Minima.

Fitnessfunktion

$$\Phi(g) = -10 \cdot \text{Konflikte} - \text{Anzahl genutzter Farben}$$

(Maximierung) Konflikte werden stark bestraft; erst bei konfliktfreier Färbung zählt die Minimierung der Farben.

1.2 8-Queens-Problem

Problem Acht Königinnen auf einem 8×8-Brett platzieren ohne dass sich zwei bedrohen (weder in Zeile, Spalte noch Diagonale).

Kodierung Ein Vektor mit acht Einträgen. Jeder Eintrag gibt die Zeile der Königin in der jeweiligen Spalte an:

$$g = [3, 6, 8, 2, 7, 1, 4, 5]$$

Beispiel: Spalte 1 $\rightarrow$ Zeile 3, Spalte 2 $\rightarrow$ Zeile 6 usw.

Crossover Zwei-Punkt-Crossover, bei dem ein mittleres Segment zwischen den Eltern getauscht wird:

$$\begin{aligned} \text{Eltern 1:} &\quad [3, \mathbf{6, 8, 2}, 7, 1, 4, 5] \\ \text{Eltern 2:} &\quad [5, \mathbf{1, 4, 7}, 2, 8, 3, 6] \\ \Rightarrow \text{Kind:} &\quad [3, 1, 4, 7, 7, 1, 4, 6] \end{aligned}$$

Gute Teilkonfigurationen werden kombiniert, während andere Abschnitte erhalten bleiben.

Mutation Eine zufällige Königin wird in eine andere Zeile derselben Spalte verschoben:

$$[3, 6, 8, 2, 7, 1, 4, 6] \;\Rightarrow\; [3, \mathbf{4}, 8, 2, 7, 1, 4, 6]$$

Kleine lokale Änderungen helfen, Konflikte schrittweise zu reduzieren.

Fitnessfunktion

$$\Phi(g) = \frac{1}{1 + \text{Konfliktspaare}}$$

(Maximierung) Bei null Konflikten ergibt sich $\Phi = 1$. So entsteht eine klare, monotone Skala: weniger Angriffe $\rightarrow$ höhere Fitness.

1.3 Simulated Annealing

Für beide Probleme werden benötigt:

- **Startzustand:** zufällige Färbung bzw. zufällige Damenanordnung.
- **Nachbarschaft:** eine Farbe ändern oder eine Dame in ihrer Spalte verschieben.
- **Energiefunktion:** Anzahl der Konflikte (bei der Färbung zusätzlich Farbanzahl). Diese wird minimiert.
- **Abkühlplan:** $T_k = 0.95 \cdot T_{k-1}$

Idee Zu Beginn (hohe Temperatur) werden auch Verschlechterungen akzeptiert. Beim Abkühlen sinkt die Akzeptanzrate, bis nur noch Verbesserungen angenommen werden. So können lokale Minima überwunden werden, und am Ende bleibt meist eine gute Lösung.

2 EA.02: Implementierung und Auswertung

Implementierung Ist In `GAAllInOne.java`: GA mit Turnierselektion, 1-Punkt-Crossover, Mutation und Elitismus. Beide Probleme (8-Queens, Kartenfärbung) nutzen `runGA`. Färbung: erst mit fünf Farben konfliktfrei, dann Reduktion auf vier, drei, zwei mit Seeding.

Varianten 100 Läufe pro Einstellung:

$$p_c \in \{0.6, 0.7, 0.8\}, \quad p_m \in \{0.01, 0.02, 0.05\}, \quad k \in \{2, 3, 5\}, \quad \mu \in \{50, 100\}.$$

Kenzahlen

$$\text{SR} = \text{Erfolgsquote}, \qquad \text{AES} = \text{durchschnitliche Generationen bei Erfolg}.$$

Beispiel (8-Queens)

p_m	k	SR / AES
0.01	3	0.86 / 118
0.02	3	0.91 / 102

Fazit Mittlere p_m und $k = 3$ funktionieren am besten. Bei der Kartenfärbung sind meist drei Farben ausreichend.

3 EA.03: Anwendungen

1. Waldo (Randal Olson)

- **Kodierung:** Reihenfolge der Suchpunkte als Permutation (Weg durch mögliche Waldo-Positionen).
- **Operatoren:** Tausch von Punkten; Kombination von Teilstrecken zweier Lösungen.
- **Fitness:** Minimierung der erwarteten Suchzeit (kürzerer Weg = besser).

2. Evolution Simulator (carykh)

- **Kodierung:** Vektor mit Eigenschaften der Kreatur (z. B. Segmentlängen, Geschwindigkeit).
- **Operatoren:** Zufällige Änderung einzelner Werte; Mischung zweier Eltern-Genome.
- **Fitness:** Maximierung der zurückgelegten Distanz oder Überlebensdauer.

3. American Fuzzy Lop (AFL)

- **Kodierung:** Eingabedateien als Bytefolgen.
- **Operatoren:** Bitflips, Zahlen ändern, Blöcke löschen/kopieren, Splicing (Zusammenkleben zweier Inputs).
- **Fitness:** Höhere Code-Coverage; neue Programmteile erreichen.

Weitere Anwendungen

- Antennenentwurf (NASA): Beste Form für stärkstes Signal.
- Stunden- und Raumpläne: Keine Überschneidungen.
- Neural Architecture Search: Automatisch beste KI-Netzstruktur finden.
- Chip- und Leiterplatten: Bessere Platzierung und Verkabelung.
- Routenplanung (Logistik): Kürzeste Lieferwege.
- Robotik: Sichere und schnelle Bewegungen lernen.

Gemeinsam: Jede Lösung ist ein individuum, Fitness bewertet die Qualität, Operatoren erzeugen neue Varianten zufällig.