EA.01: Modellierung von GA

Landkarten-Färbeproblem:

Variablen: A, B, C, D, E, F

Werte: {rot, grün, blau, gelb, orange}

Kodierung:

Farbe	Rot	Grün	Blau	Gelb	Orange
Binär-Wert	001	010	100	101	110

Mögliches Individuum I1: {A:001, B:100, C:101, D:100, E:001, F:110}

Mögliches Individuum I2: {A:110, B:100, C:100, D:101, E:001, F:101}

Gewählte Crossover Operator:

- Zweipunktschnitt: An zwei Stellen werden die Individuen aufgeschnitten und rekombiniert. Das ist noch einfach umzusetzen aber etwas vielseitiger als nur einen Schnitt durchzuführen

Beispiel:

- Schnitt I1: {A:001, B:100 **|** C:101, D:100 **|** E:001, F:110}
- Schnitt I2: {A:110, B:100 **|** C:100, D:101 **|** E:001, F:101}
- Rekombination:
  - Kind 1: {A:001, B:100, C:100, D:101, E:001, F:110}
  - Kind 2: {A:110, B:100, C:101, D:100, E:001, F:101}

Gewählte Mutation Operator:

- Konfliktorientierte Mutation: Es werden nur die Farbe der Regionen geändert, die im Konflikt zu einer anderen Region stehen, also dieselbe Farbe besitzen. Das ist zielführend, um Konflikte zu verhindern oder entfernen.
- Mutationswahrscheinlichkeit: $p_{mut} = 0.01$

Beispiel:

- Falls Mutation stattfinden soll:
- Auswahl von Regionen die im Konflikt stehen
- I1: {A:001, **B:100**, C:101, **D:100**, E:001, F:110} -> B und D stehen im Konflikt
- B oder D wird zufällig eine neue Farbe zugewiesen die weniger Konfliktreich ist
- z.B.: B=110 somit kein Konflikt mehr

Fitnessfunktion:

- Die Fitnessfunktion soll jedem Individuum eine Reelle Zahl geben die Aussagt, wie nah das Individuum dem gewollten Ergebnis kommt.

- Die Reelle Zahl wird über die Anzahl der Konflikte in einem Individuum berechnet, also wenn es z.B.: in I1: {**A:001**, **B:100**, C:101, **D:100**, **E:001**, F:110} Konflikte zwischen A und B und B und D gibt dann wäre der Reelle Wert bei 2
- Bedeutet das je kleiner die Zahl, umso besser ist das Individuum und das gewollte Ergebnis wäre bei einem Individuum erreicht, das den Fitnesswert 0 hat

8-Queens-Problem:

Schachbrett Koordinaten:

- X-Achse [A, B, C, D, E, F, G, H]
- Y-Achse [0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000]
- Durch das Koordinaten System lässt sich die Position jeder Dame auf dem Schachbrett bestimmen. Man arbeitet aber nur mit dem Binärzahlen, also der Y-Achse und nicht den Buchstaben. Die Reihenfolge der Buchstaben im Individuum ist also Fest und kann nicht verändert werden

Kodierung:

- Ich arbeite bei den Beispielen mit Dezimal Zahlen weiter (ist einfacher)
- Mögliches Individuum I1: [1, 2, 3, 4, 5, 6, 7, 8]
- Mögliches Individuum I1: [4, 2, 8, 3, 5, 1, 6, 7]

Gewählte Crossover Operator:

- OrderCrossover: Es werden zwei Schnittstellen vom ersten Individuum ausgewählt und ins Kind kopiert, dann die restlichen Zahlen, wie sie im Individuum 2 vorkommen in derselben Reihenfolge, ins Kind packen. Dass sorgt dafür, dass keine Zahl doppelt vorkommt und die Struktur der Individuen vorhanden bleibt

  Beispiel:

- I1: [1, 2, 3 **|** 4, 5, 6 **|** 7, 8] -> Schnittstellen wählen
- Kind_1: [_, _, _, 4, 5, 6, _, _] -> aus I1 ins Kind kopiert
- Kind_1: [3, 2, 8, 4, 5, 6, 1, 7] -> aufgefüllt mit I2
- Kind_2: [_, _, _, 3, 5, 1, _, _] -> aus I2 ins Kind kopiert
- Kind_2: [6, 7, 8, 3, 5, 1, 2, 4] -> aufgefüllt mit I1

Gewählte Crossover Mutation:

- Tausch Mutation: Es werden zufällig immer zwei Gene Gleichzeitig mutiert. Das sorgt dafür das keine Zahlen doppelt vorkommen. Die Zahlen tauschen nur die Position

  Beispiel:

- I1: [1, **2**, 3, 4, **5**, 6, 7, 8] -> 2 und 5 wurden zufällig gewählt
- Kind: [1, **5**, 3, 4, **2**, 6, 7, 8} -> Zahlen von 2 und 5 wurden getauscht

Fitnessfunktion:

- Die Fitnessfunktion soll die Menge der Konflikte eines Individuum zählen. Je mehr Damen eine oder mehrere andere Damen angreifen je schlechter ist das Individuum.

EA0.3: Anwendungen

1. Heres Waldo:
    a. Kodierung: Individuum besteht aus einer Reihenfolge an Punkten, wo sich Waldo befinden kann.
    b. Operatoren:
        i. Crossover: Sein Crossover ist die shuffle_mutation(agent_gnome) Funktion bei der einen Teilweg von dem ganzen Pfad nimmt und sie dann mit einem anderen Teilweg austauscht.
        ii. Mutation: Bei der mutate_agent(agent_gnome, max_mutations=3) Funktion tauscht, in dem ganzen Pfad, nur einen Punkt mit einem anderen aus.
    c. Fitnessfunktion: Die Funktion compute_fitness(solution) benutzt die Funktion calculate_distance(x1, x2, y1 ,y2) um die Distanz zweier Punkte zu berechnen und sie dann mit dem Fitnessscore zu Addieren. Dabei ist es besser je kleiner die Distanz ist. Bedeutet das je kleiner der Fitnessscore ist je besser ist das Individuum.
2. Evolution Simulator:
    a. Kodierung: 1000 Kreaturen pro Individuum.
    b. Operatoren: Es werden 500 Kreaturen zufällig getötet, wobei die schnelleren Kreaturen eine höhere Chance aufs Überleben haben. Danach werden wieder 500 neue generiert.
    c. Fitnessfunktion: Die Fitnessfunktion testet jede Kreatur wie viele Meter es in 15 Sekunden sich bewegen kann. Je weiter sie sich bewegen können, je besser ist die Kreatur. Da sich die Kreaturen auch nach hinten bewegen können gelten diese als die Schlechtesten.
3. American fuzzy lop (AFL): AFL generiert Testeingaben und misst, wie sich diese auf die Programmausführung auswirken, also ob und wie welche Pfade im Code abgedeckt werden.
    a. Kodierung: Die Population besteht aus Seeds, wobei jeder Seed eine menge von Eingabedaten enthält die neue Pfad-Abdeckungen liefern
    b. Operatoren:
        i. Selektion: Es werden nur Seeds ausgewählt die zu neuer Pfad-Abdeckung führen
        ii. Crossover: Zwei Seeds werden miteinander kombiniert
        iii. Mutation: Bit-Manipulation ähnlich wie in GA
    c. Fitnessfunktion: Eingaben die neue Pfade abdecken werden als ‚fitter' betrachtet
4. Weitere EA/GA:
    a. Verifikation und Optimierung von Prototypen: EAs generieren und optimieren automatisch Prototypen, die bestimmte Anforderungen erfüllen, um diese zu

verbessern z.B.: bei der Geschwindigkeit von Mikroprozessoren oder dem Stromverbrauch vom Mobiltelefons.

b. Finanzwesen: In der Finanzwelt werden mit EAs Aktienmärkte analysiert, spieltheoretische Analysen oder agentenbasierte Simulationen entworfen und Portfolios für maximalen Gewinn und minimales Risiko optimiert.