Aufg.03)

Link zum Code: https://github.com/rhiever/Data-Analysis-and-Machine-Learning-Projects/blob/master/wheres-waldo-path-optimization/Where's%20Waldo%20path%20optimization.ipynb

1) Finde Waldo

Als erstes: erstellt eine Liste aus zufälligen Waldo Pfaden mit generate_random_population
-> Liste aus zufälligen random_agents

Dann folgt äußere Hauptschleife, die so lange läuft, wie es Generationen gibt.
-> erstellt für jede Generation ein Dictionary.

Erste for-Schleife:
Die Fitness eines Individuums wird ermittelt und in dem Dictionary gespeichert. Doppelte werden ignoriert.

Zweite for Schleife:
sortiert die Individuen nach Fitness. Durch das Teilen der Generationengröße durch 10, speichert nur die ersten/besten 10 (%). Durch enumerate bekommen sie einen rank.

Das if in dieser Schleife printet im Abstand von 10 (%generations_pct) oder wenn die letzte Generation drankommt das beste Individuum (rank=0).

Es wird das aktuelle Individuum der Wiederholung in das Array new_populatation hinzugefügt.

Die nächste for Schleife in der zweiten for Schleife füllt das Array new_population mit 4 weiteren Individuen, welche aus dem aktuellen Individuum mit 1-3 Mutationen bestehen. (Aus jedem aktuellen Individuum werden so insgesamt 5 pro Durchlauf).

Die darauffolgende for Schleife geht diese Individuen noch einmal durch und mutiert sie noch (genau) einmal.

Dann wird die alte Population in einer for Schleife gelöscht. Wobei ich nicht sicher bin, dass diese gebraucht wird, da die neue Population in der Population gespeichert wird.

Selektion: jeweils die Top 10%
Crossover: findet nicht statt
Mutation: 2 mal, einmal aus einem Pfad 4 machen, mehrmals mutieren, dann einmal mutieren
Fitness: Berechnung der Distanz zwischen 2 Punkten. Dies für jeden Punkt im Pfad, die Distanzen werden aufsummiert. Je kleiner, kürzer, desto besser.

2) Evolution Simulator
Die Kodierung eines Individuums scheint eine Klasse Creature zu sein. Diese hat einen Platz in einer Liste und eine Liste von Muskeln. Die Klasse weiß auch, ob die Kreatur lebt, ob sie Wandlungsfähig ist und sie hat eine ID. Nur das float d kann ich nicht zuordnen. (Wahrscheinlich eine Position)
Das Mutieren scheint in Creature modified zu passieren, ich bekomme eine modifizierte Kreatur zurück. Die Methode modifiziert auch die Muskeln. Das Mutieren scheint um einen

MUTABILITY_FACTOR zu passieren, also wie ‚viel' sich die Kreatur noch wandeln kann. Die Fitness oder das Crossover konnte ich im Code nicht finden.

3) Fuzzer: Testmechanismus, der eine Software/ System randomisiert mit ungültigen Eingaben versorgt, bzw. so unerwartet agiert, dass er Schwachstellen, nicht abgefangene Eingaben oder Sicherheitslücken erkennt.

Der american Fuzzy lop beobachtet die Code-Pfade in einem Programm, genutzt durch eine Eingabedatei. Dabei hat es keine Infos über die verwendeten Eingabedaten. Er kann so ohne das Wissen über das Format von gültigen Daten welche erzeugen.
Das Ziel: neue Pfade finden
Kodierung Individuum: eine einzelne Datei, Eingabedatensatz
Selektion: nur neu gefundene Pfade werden weiter genutzt
Mutation: flippen Bits, Bytes löschen oder einfügen
Crossover: eher unüblich? Durch Slicen von Dateien
Fitness: ist fit, wenn er neuen Pfad gefunden hat
So wie ich den lop verstanden habe, nutzt er nur Teile eines EA/GA. Es wird nicht richtig ‚sortiert' wer jetzt besser oder schlechter ist, entscheidend ist nur, ob ein neuer Pfad gefunden wurde, damit ein Individuum als fit gilt.

Anwendungen:
Handelsstragien, zB bei Aktien. Die Parameter, Schwellenwerte wann ge- und verkauft werden sollte, können mit einem EA optimiert werden.

Maschinelles lernen: finden der optimalen Parameter wie Lernrate.

Design: zB hat NASA zur Entwicklung einer Antenne einen EA genutzt. Für eine hocheffiziente Antennenform.

Brückenbau: Position, Form, Gewicht einzelner Bauteile können durch GA optimiert werden.