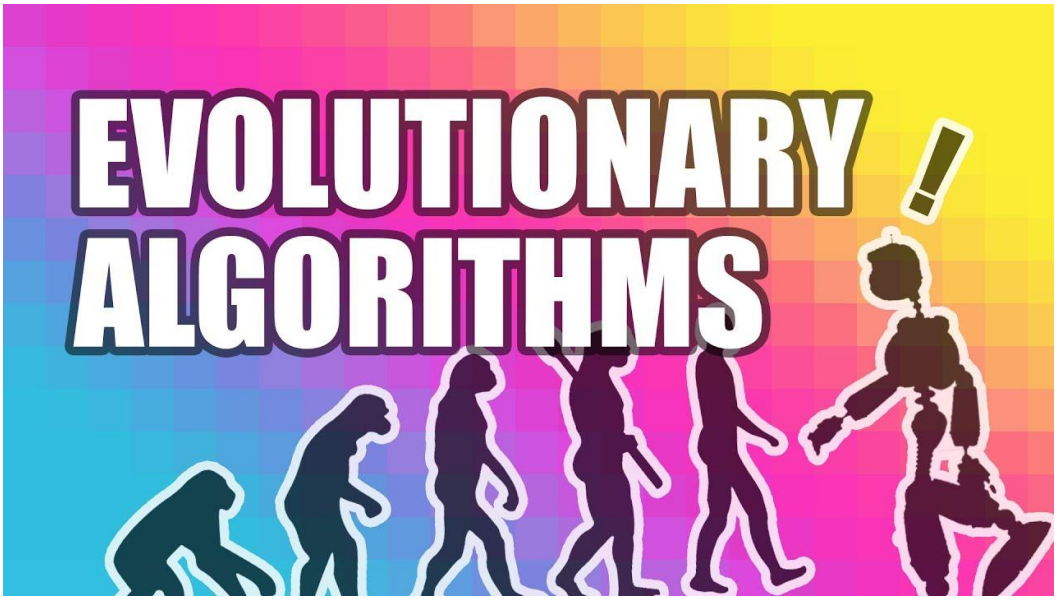
OPDRACHT 10 - EVOLUTIONARY ALGORITHMS



30 JUNI 2024

Naam: Aymane Machrouki en Berke Ozmuk
Docent: Nick Goris

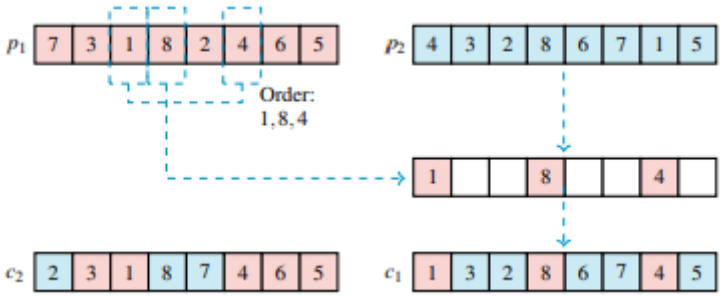
Opdracht 1: Voorbeschouwing

a) Bepaal het phenotype voor dit vraagstuk. Beargumenteer je keuze.
Het phenotype voor dit vraagstuk is de volgorde van de indeling van de stoelen rond om de tafel, waarin de namen van Arthur en zijn twaalf ridders in gegrafeerd zijn.

b) Bepaal een geschikt genotype voor dit vraagstuk. Beargumenteer je keuze.
Een geschikt genotype voor dit vraagstuk is een lijst met de volgorde van alle ridders. Dus een genotype met een lijst van al zijn genen. Het zou er bijvoorbeeld zou uit kunnen zien: [Arthur, Lancelot, Gawain, Geraint;Percival, Bors the Younger, Lamorak, Kay Sir Gareth, Bedivere, Gaheris, Galahad, Tristan] (Dit is overigens niet de volgorde).

c) Bepaal een geschikte fitness functie. Beargumenteer je keuze.
Een geschikt fitness functie voor dit vraagstuk is het volgende: elke plek tussen twee opeenvolgende ridders aan de tafel, zeg tussen ridder A en ridder B, heeft een waarde die gelijk is aan (affiniteit van A naar B) * (affiniteit van B naar A). De fitness functie is de som van de waarden voor al deze "tussenplekken". Des te hoger die som is, des te beter de tafelzetting is.

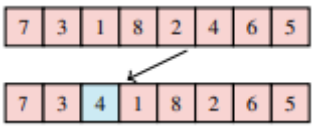
d) Bedenkt geschikte crossover operator(en). Beargumenteer je keuze(s).
De meest geschikte crossover operator voor dit vraagstuk is de **order-based crossover operator**. De order-based crossover werkt als volgt: een willekeurig aantal genen zijn geselecteerd vanuit willekeurige posities van de eerste parent en wordt op dezelfde volgorde herschikt als de tweede parent om zo de child te creëren. De twee child gebeurt op dezelfde manier als de eerste child, maar dan met de parents omgewisseld.



Figuur 1 - Voorbeeld order-based crossover

Met deze crossover geef je de kans om de meest optimale volgorde voor de tafel te bepalen.

e) Bedenk geschikte mutatie operator(en). Beargumenteer je keuze(s).
De meest geschikte mutatie operator voor dit vraagstuk is de **insertion operator.** De insertion operator kiest en verwijdert een willekeurige gene van de chromosoom en voegt het in een ander locatie.



Figuur 2 - Insertion operator

Dit is ideaal, want door deze mutatie oplossing uit te voeren is er een kans om een betere configuratie te krijgen.

Opdracht 2: Coderen

Call-tree

import_csv()

main()

-----evolve()

----------fitness_function()

----------orderbased_crossover()

----------insertion_mutation()

-----fitness_function()

-----print_knights()

Totale uitleg

Als eerst wordt de CSV file geïmporteerd een matrix gecreëerd met de namen van de ridders en hun affiniteiten met de andere ridders (**import_csv()**).

Daarna wordt de **populatie** aangemaakt. De populatie bevat **individuals**. Een individual is een array met alle indexen van de ridders. Dit is gebruikt, zodat het makkelijker is om te coderen. Alle individuals die worden aangemaakt worden geshuffled, zodat de populatie verschillende individuals bevat. Dit is van belang om de meest geschikte oplossing te vinden voor de beste tafelorde.

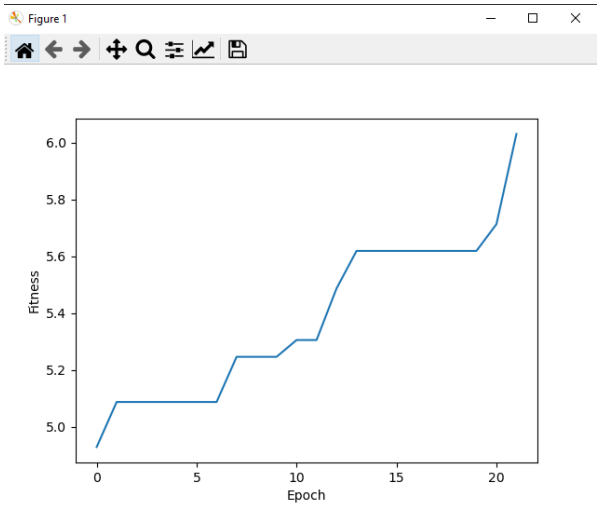
Nadat de populatie is aangemaakt wordt de populatie getrained voor een x aantal generaties om opzoek te gaan naar de beste individuals (beste tafel volgorde). Dit wordt gedaan door middel van de **evolve()** functie. In de **evolve()** functie wordt de huidige populatie aangepast naar een nieuwe populatie. Eerst worden alle individuals van de populatie geranked en gesorteerd op basis van hun affiniteit score. Daarna wordt de nieuwe populatie aangemaakt. De nieuwe populatie bevat een aantal van de beste individuals en een bepaald aantal willekeurige individuals die uit de huidige populatie wordt geselecteerd als **parents** voor de nieuwe generatie. Nadat de nieuwe populatie is aangemaakt worden er twee recombinatie operatoren toegepast op de populatie. De eerste is de **orderbased_crossover()**. Deze operator wordt altijd toegepast. De tweede operator is de **insertion_mutation()**. Deze wordt alleen uitgevoerd als de **mutate** waarde (0,01) groter is dan een willekeurig getal dat een 1 of 0 bevat.

Vervolgens worden de beste individuals per generatie opgeslagen en weergeven. Ook wordt de beste mogelijke tafelorde weergeven.

Resultaten

Nadat de populatie is getrained wordt er getest.

De populatie is op 22 **epochs** getrained en de population size is 100:



Figuur 3 - Grafiek met fitness en epochs

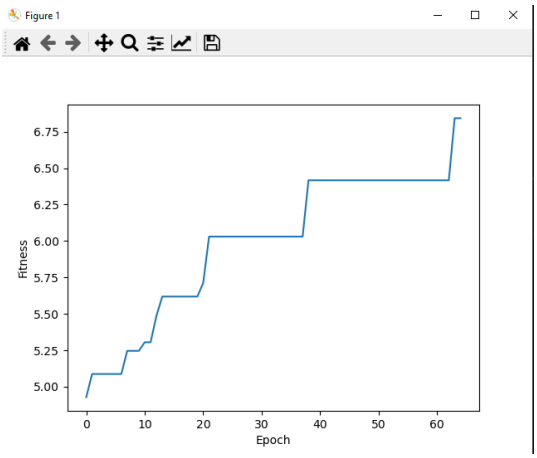
```
Epoch: 0 Fitness: 4.9293000000000005 Individual: [4, 10, 3, 9, 5, 7, 1, 8, 2, 0, 11, 6]
Epoch: 1 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 2 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 3 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 4 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 5 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 6 Fitness: 5.087599999999999 Individual: [8, 5, 1, 0, 7, 3, 9, 2, 6, 11, 10, 4]
Epoch: 7 Fitness: 5.2463999999999995 Individual: [1, 6, 9, 8, 2, 11, 5, 0, 10, 3, 7, 4]
Epoch: 8 Fitness: 5.2463999999999995 Individual: [1, 6, 9, 8, 2, 11, 5, 0, 10, 3, 7, 4]
Epoch: 9 Fitness: 5.2463999999999995 Individual: [1, 6, 9, 8, 2, 11, 5, 0, 10, 3, 7, 4]
Epoch: 10 Fitness: 5.3054000000000014 Individual: [5, 11, 4, 1, 7, 8, 2, 0, 3, 9, 10, 6]
Epoch: 11 Fitness: 5.3054000000000014 Individual: [5, 11, 4, 1, 7, 8, 2, 0, 3, 9, 10, 6]
Epoch: 12 Fitness: 5.4868 Individual: [0, 4, 11, 10, 9, 2, 6, 8, 5, 3, 1, 7]
Epoch: 13 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 14 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 15 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 16 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 17 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 18 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 19 Fitness: 5.6188 Individual: [5, 3, 10, 4, 1, 7, 8, 2, 0, 6, 9, 11]
Epoch: 20 Fitness: 5.7128000000000005 Individual: [8, 2, 9, 7, 4, 5, 0, 11, 10, 3, 1, 6]
Epoch: 21 Fitness: 6.0302 Individual: [3, 9, 10, 4, 6, 8, 2, 0, 5, 11, 1, 7]
```

Figuur 4 - Resultaten best fitness en individual

```
Elapsed time: 0.02600407600402832
```

Figuur 5 - Verstreken tijd

De populatie is op 65 **epochs** getrained en de population size is 100:



Figuur 6 - Grafiek met fitness en epochs

```
Epoch: 50 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 51 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 52 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 53 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 54 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 55 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 56 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 57 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 58 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 59 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 60 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 61 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 62 Fitness: 6.4163 Individual: [0, 7, 1, 9, 2, 8, 5, 11, 10, 3, 4, 6]
Epoch: 63 Fitness: 6.8424 Individual: [10, 9, 1, 7, 4, 3, 0, 2, 8, 6, 5, 11]
```

Figuur 7 - Resultaten laatste 15 epochs met beste fitness en individual

```
Elapsed time: 0.06901478767395020
```

Figuur 8 - Verstreken tijd

Uit de resultaten blijkt dat het algoritme reproduceerbaar is en dat het aan de voorwaarden voldoet: fitness >= 6.0 bij epochs van 22 en fitness >= 6.8 bij epochs van 65.

Opdracht 3: Nabeschouwing

Een gradient descent heeft geen absoluut maximum, maar werkt met meerdere locale optima. Het is dus niet handig om gradient descent toe te passen.