Laboratorium

Algorytm genetyczny cz. 2 – chromosomy niebinarne

Problem inżynieryjny: stop metali

W pewnym zakładzie badawczym inżynierowie próbowali stworzyć bardzo trwały stop sześciu metali. Ilości wszystkich 6 metali w stopie oznaczone zostały symbolami x, y, z, u, v, w i są to liczby z przedziału [0, 1). Okazało się, że wytrzymałość stopu określona jest przez funkcję:

$$endurance(x, y, z, v, u, w) = e^{-2\cdot(y-\sin(x))^2} + \sin(z \cdot u) + \cos(v \cdot w)$$

Obliczenie maksymalnej wytrzymałości (endurance) było dla inżynierów problematyczne. Poproszono Ciebie, eksperta od sztucznej inteligencji, o rozwiązanie problemu.

Zadanie 1

W tym zadaniu rozwiążemy problem inżynieryjny za pomocą algorytmu genetycznego. Naszym celem jest znalezienie odpowiedzi na dwa pytania:

- Jaka jest najlepsza wytrzymałość stopu metali?
- Jakie ilości metali trzeba zmieszać, by uzyskać najbardziej wytrzymały stop?

Przydatny fragment kodu (po imporcie biblioteki math):

```
def endurance(x, y, z, u, v, w):
    return math.exp(-2*(y-math.sin(x))**2)+math.sin(z*u)+math.cos(v*w)
```

Zadanie to jest inne niż te z poprzednich laboratoriów, gdyż musimy się zmierzyć z chromosomami, które nie mają zer i jedynek jak genów. Zamiast nich, mają liczby rzeczywiste (zmiennoprzecinkowe) z przedziału [0, 1).

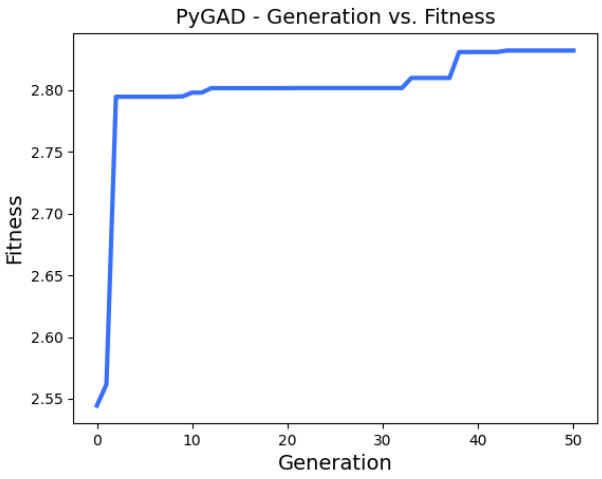
Przykładowy chromosom: $A = [0.09,\ 0.06,\ 0.99,\ 0.98,\ 0.1,\ 0.15]$

Rozwiąż ten problem z użyciem paczki pygad i odpowiedz na pytania zadane na początku tego zadania.

a) Otwórz dokumentację i korzystaj z niej w razie problemów: https://pygad.readthedocs.io/en/latest/README_pygad_ReadTheDocs.html
b) Zdefiniuj poprawnie gene_space.
c) Zdefiniuj sensowną funkcję fitness (to jest u nas banalnie proste 😉)
d) Ustaw sensowne parametry związane z populacją i długością chromosomu (6!).
e) Chromosom jest krótki, mutację trzeba zwiększyć do kilkunastu procent, żeby nie dostawać czerwonego warninga.
```
UserWarning: The percentage of genes to mutate …
```
f) Uruchom algorytm genetyczny. Powtórz kilka razy, by zobaczyć czy wyniki i wykresy się zmieniają.

U mnie najlepszy stop miał około 2.83.

```
Parameters of the best solution : [0.20431118 0.20483286 0.99940818 0.98341734
0.21199392 0.01554426]
Fitness value of the best solution = 2.832060419993197
```



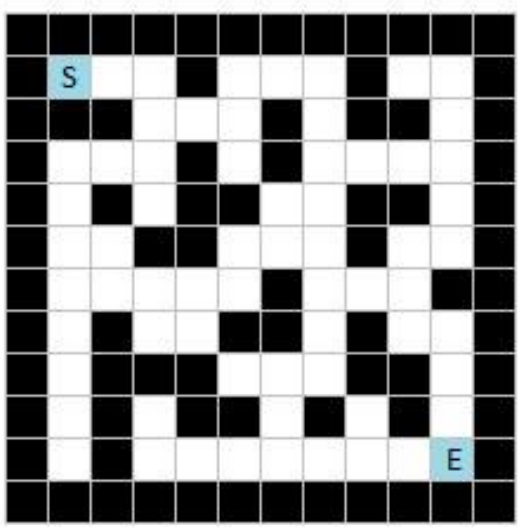
Problem drogi w labiryncie

Dany jest prostokątny labirynt składający się z kwadratowych pól. Niektóre pola są puste i można po nich chodzić (kratki białe), a niektóre pola są ścianami i nie można na nie wchodzić (kratki czarne). Można założyć, że labirynt jest otoczony ramką ścianek, tak aby nie dało się wyjść poza jego granice.

Po labiryncie chodzimy przeskakując z pola na jedno z sąsiednich pól (po lewej, prawej, u góry lub na dole). Nie są legalne skoki na ukos.

W lewym górnym rogu labiryntu znajduje się pole startu (S), a w prawym dolnym rogu pole wyjścia (exit, E). Czy istnieje ścieżka z S do E o określonej maksymalnej długości?

Żeby było łatwiej, weźmy konkretny przykład (instancję problemu). Mamy labirynt 12x12 pól, uwzględniając ramkę ze ścianek, lub 10x10 pól, nie uwzględniając tej ramki. Rysunek:



W labiryncie tym szukamy drogi o maksymalnej liczbie **30 kroków**. Czy istnieje taka droga? Jeśli tak, jak ona wygląda?

Oczywiście, ponieważ labirynt jest łatwy to od razu znajdziemy rozwiązanie. Z bardzo skomplikowanymi labiryntami byłoby jednak gorzej. A jak poradzi sobie algorytm genetyczny z labiryntem?

Zadanie 2

Podobnie jak w zadaniu 1, stosując paczkę pygad, rozwiąż problem szukania drogi w labiryncie za pomocą algorytmu genetycznego.

Aspekty organizacyjne do rozważenia (były omówione dość obszernie na wykładzie)

- Jak zakodować labirynt?
- Jak zakodować chromosomy? Jaką mają długość? Jakie mają geny?
- Jaką wybrać funkcję fitness? Jakie oceny będzie wystawiała?
- Problem wydaje się być cięższy niż poprzednie. Czy warto zwiększyć populację?

Gdy rozstrzygniesz aspekty organizacyjne, przejdź do rozwiązania zadania:

a) Zakoduj labirynt jako macierz.
b) Ustaw odpowiednie gene_space i długości chromosomów.
c) Ustaw sensowne parametry związane z populacją i rodzicami. Możesz przetestować różne konfiguracje.
d) Ustaw sensowny procent na szansie na mutację. Najniższy możliwy, żeby nie wyskakiwał warning.
e) Stwórz funkcję fitness. (Jest to najcięższy punkt tego zadania: sporo programowania, pętle).
f) Jeśli algorytm znajduje rozwiązanie, to wprowadź warunek stopu i zmierz średni czas z 10 uruchomień.