WSI Ćwiczenie 2
Algorytm genetyczny

Autor:
Jakub Kowieski

Przedmiot:
WSI, semestr: 21Z

Numer Albumu:
310765

Celem ćwiczenia była implementacja algorytmu genetycznego Hollanda oraz zastosowanie go do znalezienia minimum funkcji f dla wektora czterech liczb całkowitych z zakresu [−16, 16).

Stałe:

RANGE = 10000 - liczba ewolucji (iteracji) algorytmu genetycznego
params - lista parametrów zdefiniowanych (znajduję się w ./utils/params.py): (populacja, prawdopodobieństwo krzyżowania, prawdopodobieństwo mutacji)

Założenia:

[1, 0, 1, 0, 0, 1, …] - populacja zdefiniowana jako lista 20 bitów, gdzie każde kolejne 5 bitów tworzy **individual** (osobnik) - liczbę z przedziału [−16, 16) zakodowaną za pomocą kodu graya
fitting_function - oblicza ocenę (rating) najpierw obliczającą wartość funkcji f(x), po czym robimy funkcję przeciwną g(x) = - f(x), by zmienić problem na maksymalizację funkcji **g(x)**, i na koniec odejmujemy od wszystkich wartości funkcji g(x) najmniejszą występującą wartość - 1 (dodatkowe -1 jest by nie miał on zerowego prawdopodobieństwa i nie wystąpił wyjątek, gdy wszystkie osobniki w populacji są takie same)
Gdy występuje nieparzysta ilość osobników przy operacji krzyżowania, wtedy jeden z osobników po prostu w niej nie uczestniczy.

Posiadanie modułów:

- numpy
- Matplotlib
- pytest-3

Testowanie i symulacje:

pytest-3 tests - bardzo proste testy czy algorytm działa poprawnie
python3 table.py - tworzy plik csv w folderze ./csv z wynikami eksperymentów dla parametrów
python3 simulation.py - służy do generowania populacji, znajdowania parametrów dla eksperymentów oraz wyświetlania wykresu średniej wartości funkcji g(x).

Legenda:

(Population Size, Crossing Prob, Mutation Prob) - taki zapis to odwołanie się do eksperymentu dla zadanych parametrów

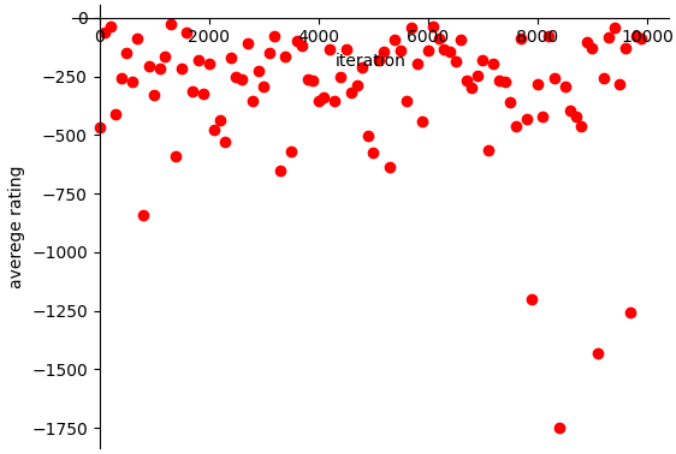
---

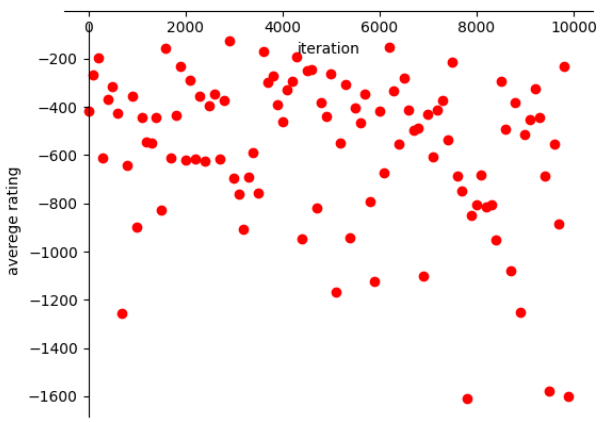
Eksperymenty:

Population Size	Crossing Prob	Mutation Prob	Arithmetic Average	Standard deviation
13	1	0.01	-1.55	0.85
8	0.4	0.2	-1.31	0.53
22	0.8	0.05	-0.99	0.0
15	0.3	0.1	-0.99	0.0
10	0.5	0.15	-0.99	0.0
6	0.6	0.4	-1.67	0.53
19	0.4	0.15	-1.07	0.27
17	0.9	0.08	-0.99	0.0
25	0.5	0.01	-0.99	0.0
8	0.5	0.5	-2.2	0.84

Wnioski:

Algorytm genetyczny Hollanda działa dobrze dla małych wartości prawdopodobieństwa mutacji, a szczególnie dla dużych populacji. Dla mniejszych ma także duże znaczenie, tylko większe wartości mutacji prawdopodbieństwa mogą pomóc z szukaniem rozwiązań dla bardziej oddalonych niż dosyć mała populacja. Dla (8, 0.4, 0.2) i (8, 0.5, 0.5) widzimy że zwiększenie wartości aż o 0.3 pogarsza nam wyniki. Jest to spowodowane zbyt szerokim szukaniem najlepszego osobnika:



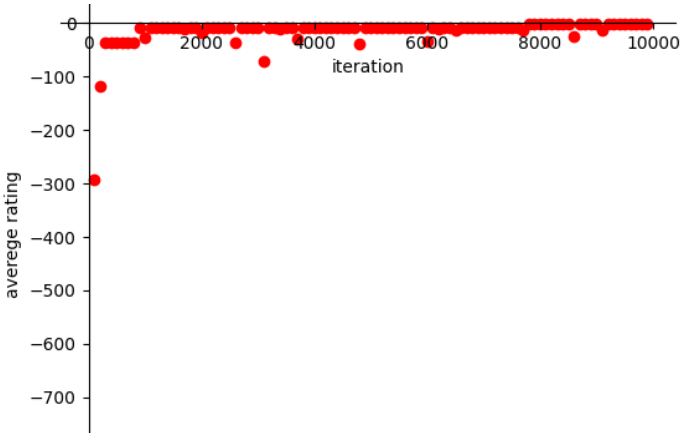


(8, 0.4, 0.2)

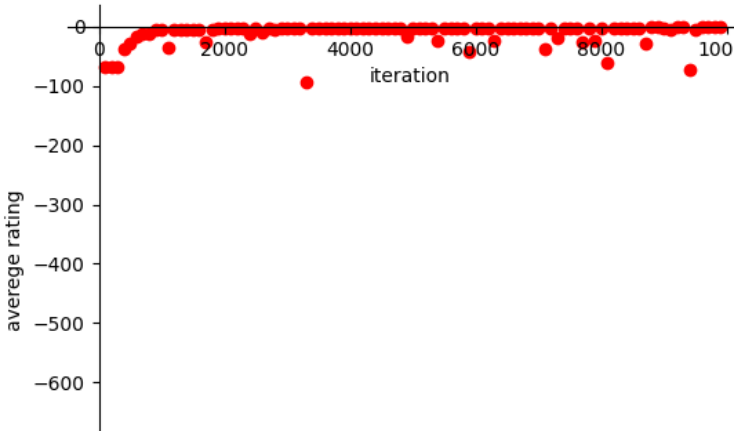
(8, 0.5, 0.5)

Prawdopodobieństwo krzyżowania ma o wiele mniejszy wpływ niż mutacji jeśli chodzi o zauważalną różnicę, ale nadal ważne jest by móc zmienić populację względem samej siebie. Przy większych wartościach (13, 1, 0.01) populacja zmienia się gwałtowniej, a dla (25, 0.5, 0.01) wartości 2 razy mniejszej łagodniej, ale płynniej.

(13, 1, 0.01)

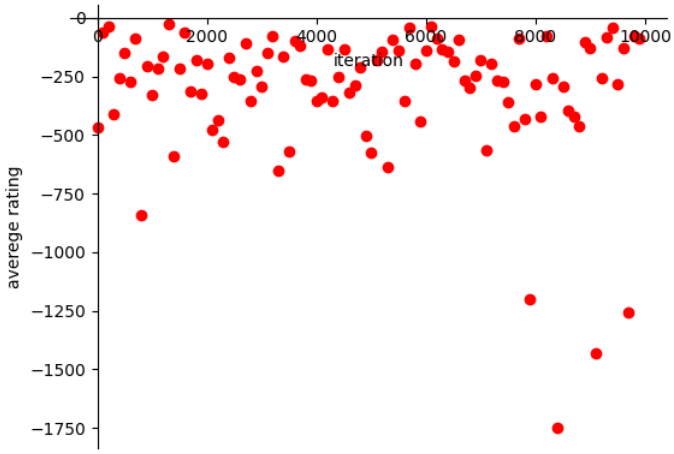


(25, 0.5, 0.01)

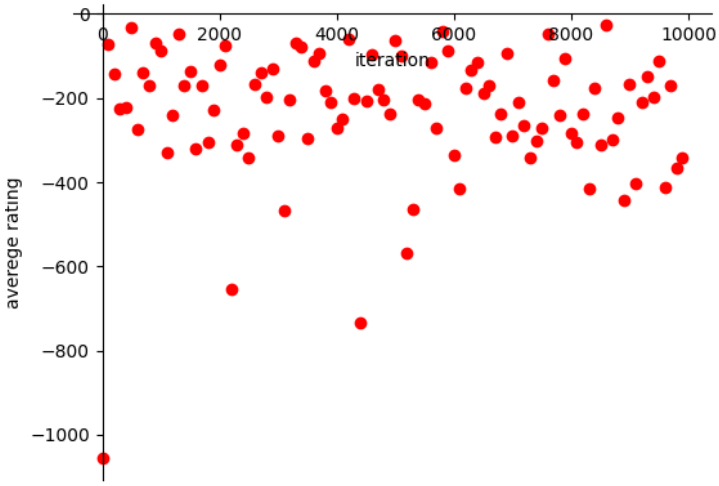


Rozmiar populacji nie zmienia charakterystyki średniej wartości funkcji g(x), ale im większa tym możliwa szybsze znalezienie najlepszego osobnika.

(8, 0.4, 0.2)



(19, 0.4, 0.15)



Wnioski:

Algorytm Hollanda działa bardzo dobrze dla odpowiednich parametrów, im większa populacja tym lepiej (oczywiście im większa tym czas wykonywania się algorytmu jest większy), wartość prawdopodobieństwo krzyżowania zależy od przypadku i jak chcemy by się zmieniała populacja, a prawdopodobieństwo mutacji powinno być małe aby algorytm mógł nadążać z segregacją lepszych osobników od gorszych, ale nie ma że im mniejsza tym lepsza (np. Dla (13, 1, 0.01) algorytm wcale nie zachowuje się najlepiej).