ZDANIE SPRAWY

Algorytmy Ewolucyjne - zadanie 1 Przez

Wojciech Szade

Wydział Elektryczny PW
Informatyka Stosowana

Spis treści

1 Cel zadania

Celem zadania było napisanie programu umożliwiającego znalezienie maksimum funkcji dopasowania jednej zmiennej określonej dla liczb całkowitych w zadanym zakresie przy pomocy elementarnego algorytmu genetycznego (reprodukcja z użyciem nieproporcjonalnej ruletki, krzyżowanie proste, mutacja równomierna).
Program powinien umożliwiać użycie różnych funkcji dopasowania, populacji o różnej liczebności oraz różnych parametrów operacji genetycznych (krzyżowania i mutacji). Program powinien zapewnić wizualizację wyników w postaci wykresów średniego, maksymalnego i minimalnego przystosowania dla kolejnych populacji oraz wykresu funkcji w zadanym przedziale. Program przetestować dla funkcji f(x)= -0.4x2+3x+8 dla x= -1, 0, ... 21

2 Instrukcja obsługi

Dla uniwersalności i łatwości dopasowywania parametrów stworzyłem plik config.json, w którym przekazuje się wszystkie parametry pracy programu: liczbę generacji, wielkość populacji, współczynnik krzyżowania i mutacji - oraz parametry dotyczące funkcji: zakres i funkcję (którą podajemy po prostu wpisując działanie, np. 2x**2+x).
Cały plik wykorzystuje format .json.
Ponadto do programu dostarczany jest też plik requirements.txt, który zawiera wszystkie użyte biblioteki pythonowe, potrzebne do uruchomienia programu.
Jednak zgodnie z wymaganiami - dostarczam wersję projektu, która może zostać otworzona przez program Visual Studio i od razu uruchomiona.
W celu zmiany konfiguracji - należy jedynie edytować plik config.json.

```
{
"num_generations": 100,
"population_size": 100,
"crossover_rate": 0.75,
"mutation_rate": 0.25,
"lower_bound": -1,
"upper_bound": 21,
"function": "-0.4*x**2+3*x+8"
}
```

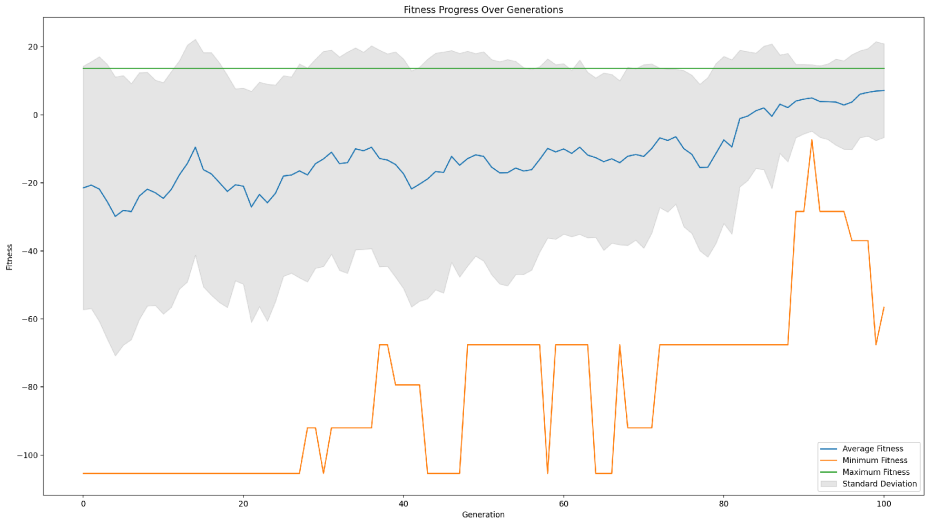
Liczba generacji (num_generations), wielkość populacji (population_size), dolna granica zakresu (lower_bound) oraz górna granica zakresy (upper_bound) muszą być liczbami całkowitymi.
Współczynniki krzyżowania (crossover_rate) i mutacji (mutation_rate) to liczby typu float.
Funkcję podajemy korzystając z podstawowych działań w języku Python, korzystając ze znaków "-", "+", "*", "**", "x". Program zawiera kod walidujący poprawność podanych wartości.

3 Opis wykonanych symulacji

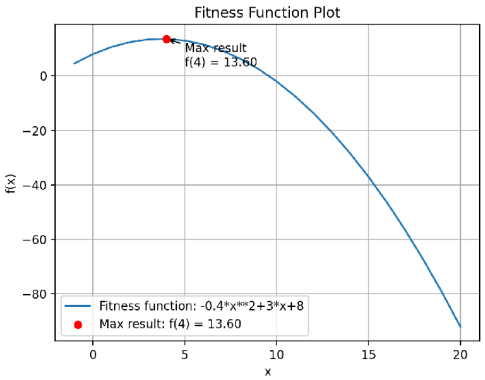
3.1 Symulacja 1

Pierwsza symulacja jaką wykonamy, będzie niejako wzorcowa, dla typowych wartości parametrów prawdopodobieństw mutacji i krzyżowania oraz wartości maksymalnej liczby pokoleń.
Wybrane dane przedstawiają się następująco:

- Liczba generacji: 100,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.8,
- Współczynnik mutacji: 0.1



(a) Wykres przystosowania dla kolejnych generacji



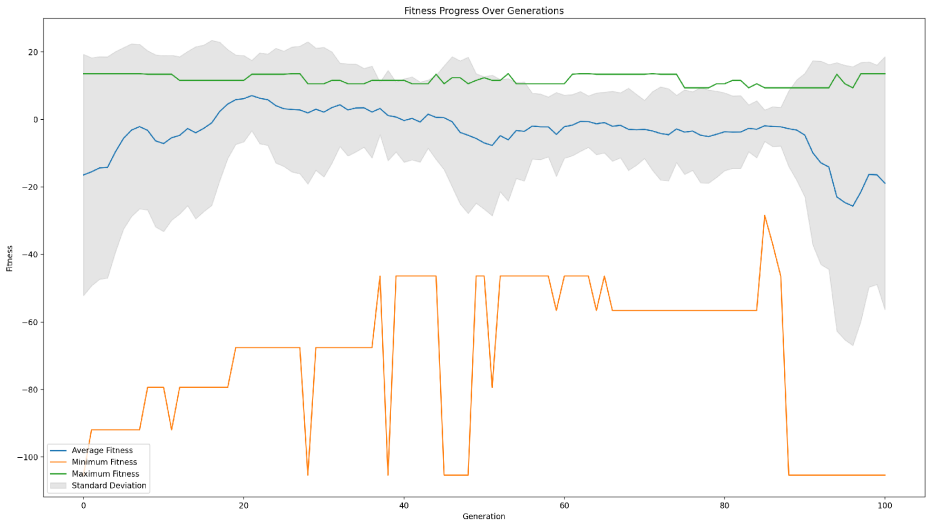
(b) Wykres funkcji

Jak widać osiągnięty wynik jest na prawdę dobry, a sam algorytm doszedł do dobrych wyników w stosunkowo liniowy sposób, będąc lepszy z generacji na generację.

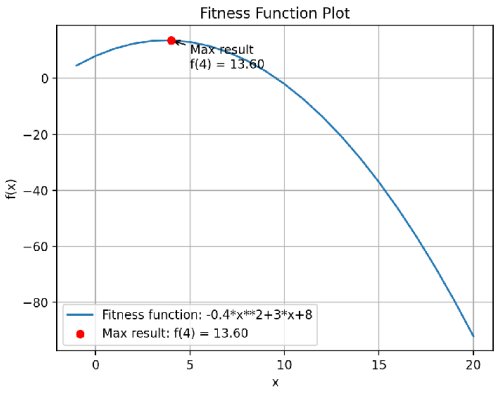
3.2 Symulacja 2

W tej symulacji ustawimy bardzo wysoki współczynnik krzyżowania, nie zmieniając reszty parametrów.

- Liczba generacji: 100,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.99,
- Współczynnik mutacji: 0.1



(a) Wykres przystosowania dla kolejnych generacji



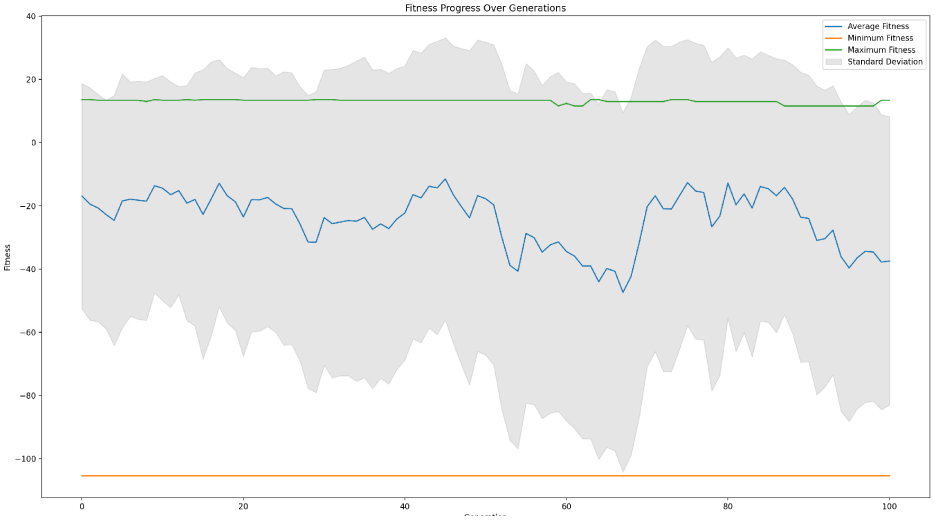
(b) Wykres funkcji

Osiągnięty rezultat, jest ten sam - ale ewidentnie, droga programu nie była taka liniowa. Właściwie, to wyniki stawały się wręcz coraz gorsze. Można od razu zauważyć, że tak wysoki współczynnik krzyżowania wypada niekorzystnie.

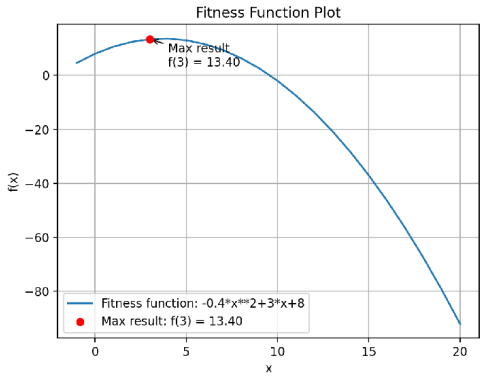
3.3 Symulacja 3

W tej symulacji ustawimy dosyć niski współczynnik krzyżowania - 0.5, nie zmieniając reszty parametrów.

- Liczba generacji: 100,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.5,
- Współczynnik mutacji: 0.1



(a) Wykres przystosowania dla kolejnych generacji



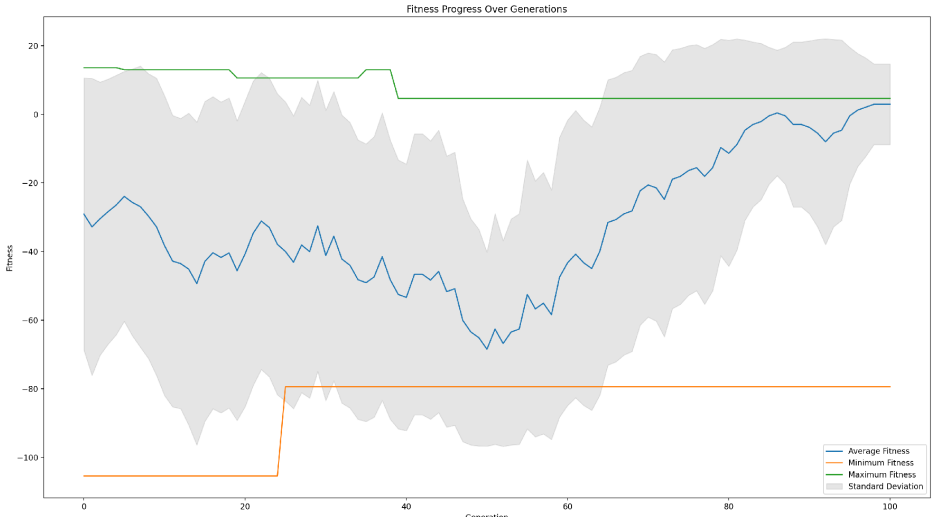
(b) Wykres funkcji

Od razu rzuca się w oczy bardzo wysoka dewiacja wyników. Minima i maksyma są praktycznie takie same przez cały program, bo skrajne wartości nie są prawidłowo odrzucane. Sama średnia nie wypada najgorzej, ale jest mniej przewidywalna.

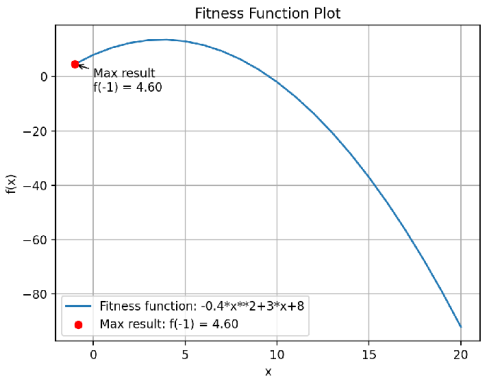
3.4 Symulacja 4

W tej symulacji zmienimy parametr współczynnika mutacji, przywracając współczynnik krzyżowania do zdrowej normy. Ustalimy bardzo niską wartość.

- Liczba generacji: 100,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.8,
- Współczynnik mutacji: 0.01



(a) Wykres przystosowania dla kolejnych generacji



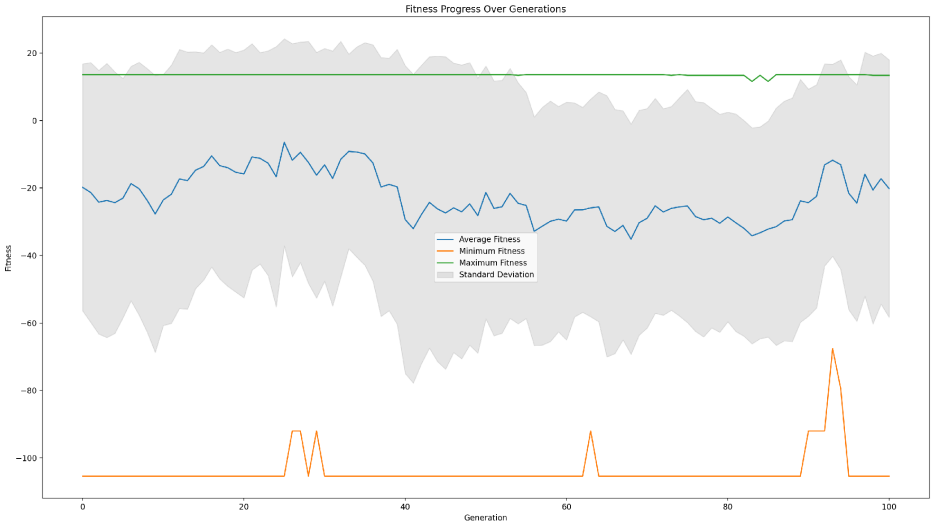
(b) Wykres funkcji

Po zmianie mutacji na bardzo niską, wykres wygląda co najmniej dziwnie. Początkowo wyniki spadają, ale około połowy, średnia znacząco urosła, osiągając finalnie wysokie dopasowanie. Jednak o dziwo, znaleziona wartość nie jest najlepsza - co widać na drugim wykresie. To udowadnia, że mutacja to ważny element tego algorytmu genetycznego.

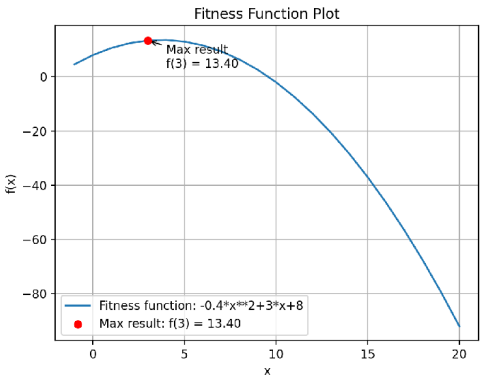
3.5 Symulacja 5

Z racji na ciekawy wynik bardzo niskiego współczynnika mutacji - teraz ustawimy go na wartość trzykrotnie wyższą niż normalnie. Nie jest to, aż tak ekstremalna modyfikacja wartości, więc uzyskany wynik może być ciekawy.

- Liczba generacji: 100,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.8,
- Współczynnik mutacji: 0.3



(a) Wykres przystosowania dla kolejnych generacji



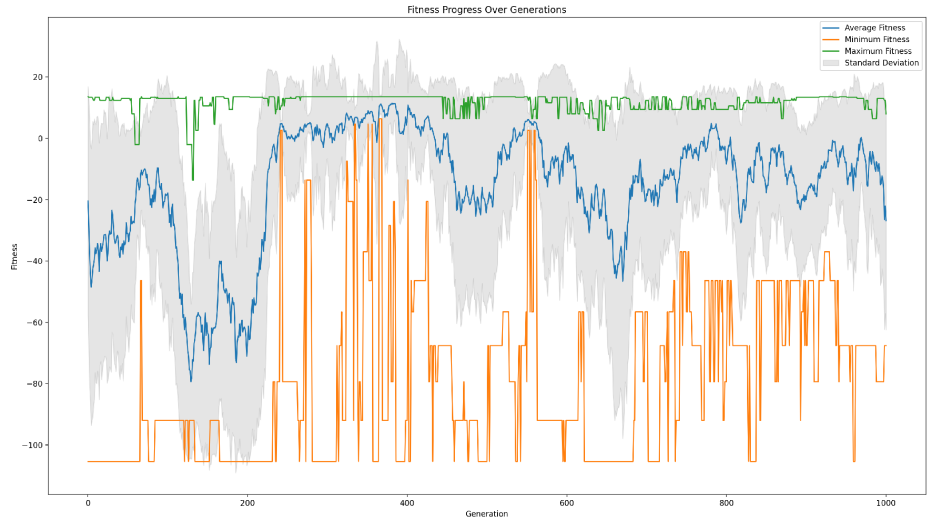
(b) Wykres funkcji

Wykres wygląda po prostu jak gorsza wersja pierwszej symulacji. Minimalne i maksymalne wyniki nie zazębiają się i wartości pozostają cały czas dosyć średnie. Finalny wynik nie jest tak dobry, jak mógłby być.

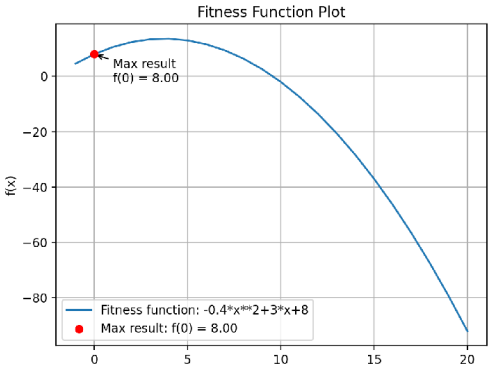
3.6 Symulacja 6

Teraz zmienimy parametr liczby generacji - sprawdzimy co się stanie, gdy będzie ich bardzo dużo. Teoretycznie - powinno skutkować to coraz lepszym wynikiem. Sprawdzimy to.

- Liczba generacji: 1000,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.8,
- Współczynnik mutacji: 0.1



(a) Wykres przystosowania dla kolejnych generacji



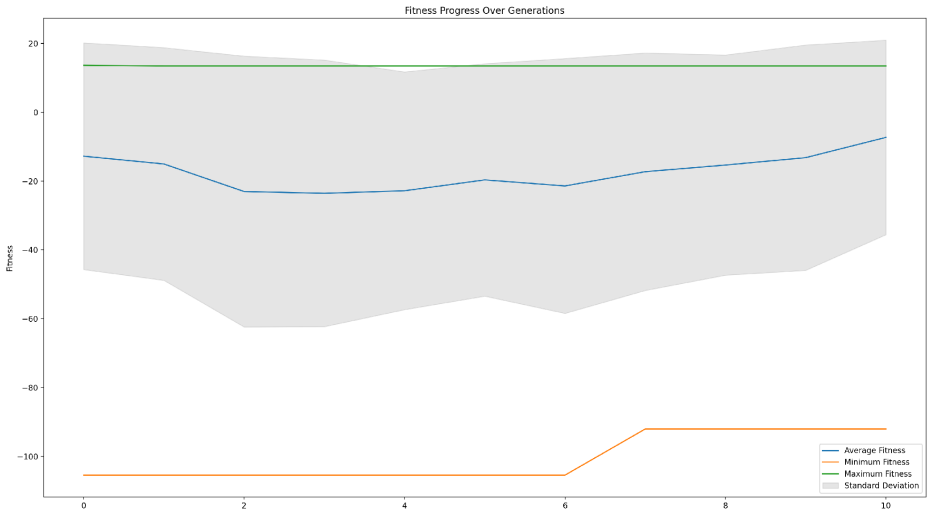
(b) Wykres funkcji

Jak można się było spodziewać, sam wykres stracił na czytelności, przez ilość generacji. Same wartości na przemian spadają i rosną, nie zachowując wcale linearnego wzrostu. Możliwe, że można by było uzyskać lepszy wynik przez lepsze dopasowanie pozostałych współczynników do takiej symulacji. Co ciekawe, finalny wynik jest do tej pory najgorszym, a same najlepsze dopasowania są stosunkowo niskie. Okazuje się, że liczba symulacji to niekoniecznie klucz do sukcesu.

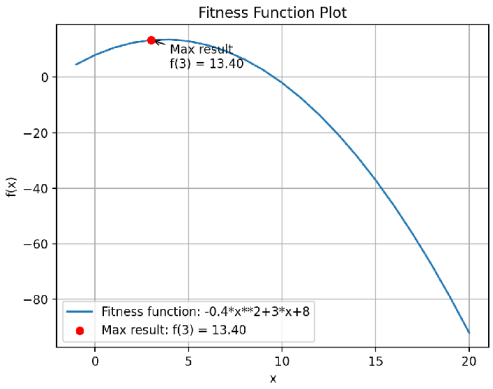
3.7 Symulacja 7

Na koniec sprawdzimy jeszcze sytuację, w której generacji jest bardzo mało - 10. Będzie jasno widać, co dzieje się w każdej generacji. Wynik raczej nie będzie bardzo dobry - ale będzie mało zaszumiony.

- Liczba generacji: 10,
- Wielkość populacji: 100,
- Współczynnik krzyżowania: 0.8,
- Współczynnik mutacji: 0.1



(a) Wykres przystosowania dla kolejnych generacji



(b) Wykres funkcji

Dla odmiany wykres jest bardzo czytelny. Widać spokojny wzrost dopasowania. Sam wynik jest naprawdę niezły, ale obawiam się, że mógłby on znacząco się różnić, dla każdego uruchomienia programu.