Algorytmy genetyczne
Minimalizacja funkcji typu f(x,y)

Wykonawca: Tomasz Kąkol

Laboratorium: 6,

Data wykonania: 03.01.2019

1. Cel

Celem wykonanego zadania (zgodnie z instrukcją przeznaczaną do tego laboratorium) było stworzenie programu umożliwiającego znalezienie minimum funkcji (o nieznanej definicji) dzięki zastosowaniu metody algorytmu genetycznego.

2. Parametry programu

Zgodnie z poleceniem, poszukiwano minimum dla **funkcji numer 16**.

Główne parametry użyte w budowie algorytmu genetycznego zostały zdefiniowane w pliku main.cpp. W celu znalezienia jak najlepszego rozwiązania dobierałem różne wartości dla różnych parametrów, a następnie porównywałem otrzymane rezultaty i szukając jak najlepszego i jak najbardziej stabilnego rozwiązania (odpornego na utratę poprawnego rozwiązania przy relatywnie niewielkiej zmianie wartości parametrów w otoczeniu badanego rozwiązania) .

Podczas badań uwzględniłem poniższe parametry oraz przyjmowałem dla nich następujące wartości:

Wielkość populacji: 30 lub 50 lub 80 lub 100

Metody selekcji: ‘rank’ lub ‘roulette’ lub ‘tournament’

Metody krzyżowania: ‘onepoint’ lub ‘twopoint’ lub ‘evenodd’ lub ‘unioform’

Prawdopodobieństwo mutacji: 0.01 lub 0.05 lub 0.1

Prawdopodobieństwo krzyżowania: 0.3 lub 0.4 lub 0.5 lub 0.6 lub 0.7 lub 0.8

3. Rezultaty

W celu prezentacji wyników opiszę zachowanie się algorytmu w zależności od zastosowanej metody selekcji wraz ze zmianami wartości pozostałych parametrów użytych wraz z tymi metodami.

Wykonany program tworzy pliki typu .txt, których przykładowa nazwa ‘best_0.01_0.30_100_tournament_uniform.raw’ oznacza, że jest to plik zawierający:

- Wartości ‘X’ i ‘Y’ dla najlepszego rozwiązanie, dla przypadku, gdy wartości parametrów to
  - Prawdopodobieństwo mutacji: 0.01
  - Prawdopodobieństwo krzyżowania: 0.3
  - Wielkość populacji: 100
  - Metody selekcji: ‘tournament’
  - Metody krzyżowania: ‘unioform’

W plikach podobne do ‘tmp_0.01_0.30_100_tournament_uniform.raw’ znajdują się informację o:

- Najlepszym osobniku w danym pokoleniu
- Medianie
- Wartości offline max
- Wartości offline min
- Wartości online

Poniższe badania wykonałem dla wielkości populacji ‘100’, ale wartość tego parametru jest edytowalna i nie ma problemu by przeprowadzić równocześnie badania na różnych wielkościach populacji.

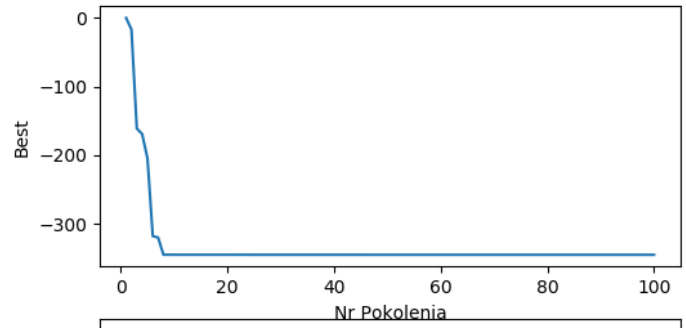
A. Metoda rankingowa

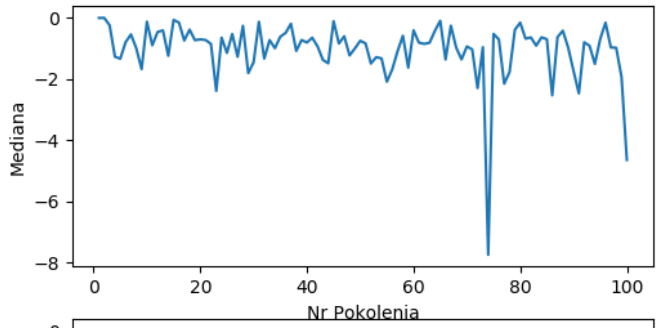
Wykorzystanie metody rankingowej pozwala otrzymać całkiem dobrą zbieżność już po ok 10 pokoleniach przy zastosowaniu dużego współczynnika mutacji i współczynnika krzyżowania. Na podstawie otrzymanych wyników wnioskuję, że w tym przypadku zastosowanie różnych metody krzyżowania nie wpływa negatywnie na stabilność rozwiązaniem. Rozmiar populacji natomiast ma wpływ na polepszenie wyników, przy dobrze dobranych poprzednich

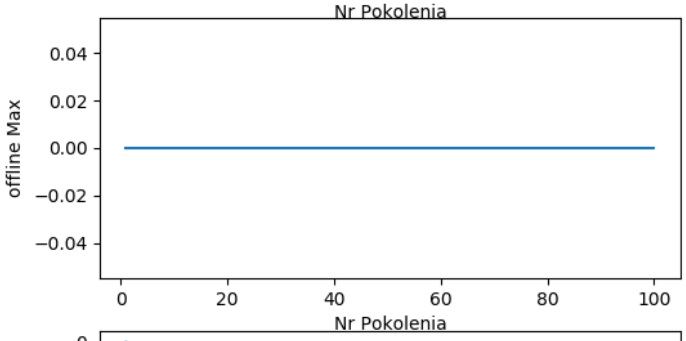
parametrach. Zbieżność jest podobna jak w przypadku metody ruletkowej, jednak metoda rankingowa jest mniej korzystna wydajnościowo.

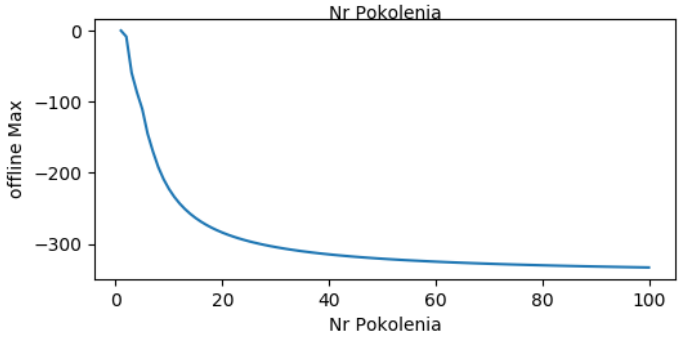
P. mutacji	P. krzyżowania	Metody krzyżowania	**X**	**Y**
0.10	0.3	twopoint	0.781349	0.001434
0.10	0.4	twopoint	0.768761	0.000290
0.10	0.5	twopoint	0.770096	0.002007
0.10	0.6	twopoint	-2.346710	-3.124718
0.10	0.7	twopoint	0.768570	0.000671
0.10	0.8	twopoint	0.796036	-0.227837
0.05	0.3	twopoint	0.781349	0.000099
0.10	0.4	onepoint	0.769333	0.000862
0.10	0.4	evenodd	0.781540	0.001434
0.10	0.5	evenodd	-2.339081	0.001053
0.10	0.6	evenodd	0.781921	0.002769
0.10	0.7	evenodd	0.781540	0.001053
0.01	0.7	evenodd	-2.343849	0.000099
0.01	0.6	evenodd	-2.343849	0.000099
0.01	0.5	evenodd	0.781158	-3.124908
0.01	0.5	Uniform	0.769142	0.000099
0.10	0.5	Uniform	-2.343849	0.000099
0.10	0.4	Uniform	0.768379	0.002579
0.10	0.6	Uniform	0.781349	0.001434
0.10	0.7	Uniform	0.781731	0.000099
0.10	0.8	Uniform	0.768761	0.000099
0.01	0.8	Uniform	0.781349	-3.124908
0.01	0.7	Uniform	0.781349	-3.124908
0.01	0.6	Uniform	0.769142	0.000099
0.01	0.5	Uniform	0.769142	0.000099
0.01	0.4	Uniform	0.781158	-3.124908
0.01	0.3	uniform	0.781349	0.000099

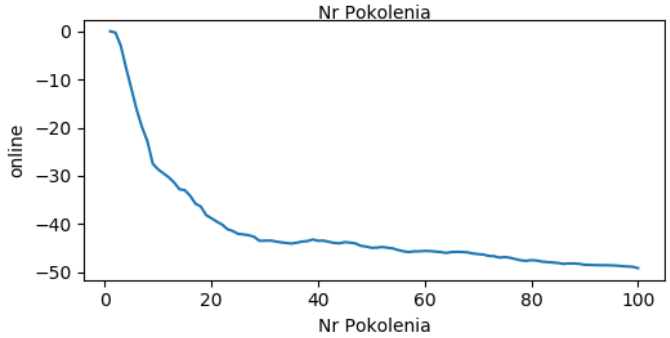
Przykładowe rezultaty dla metody ruletkowej (przypadek tmp_0.10_0.80_100_rank_uniform.raw). Poniższy wykres przedstawia pojedyncze wykonanie metody z danymi parametrami:











W celu sprawdzenia stabilności wyników dla tej metody i parametrów, zapamiętuje te parametry i wykonuję 20 razy cały algorytm i wyznaczam odchylenia standardowe itd.

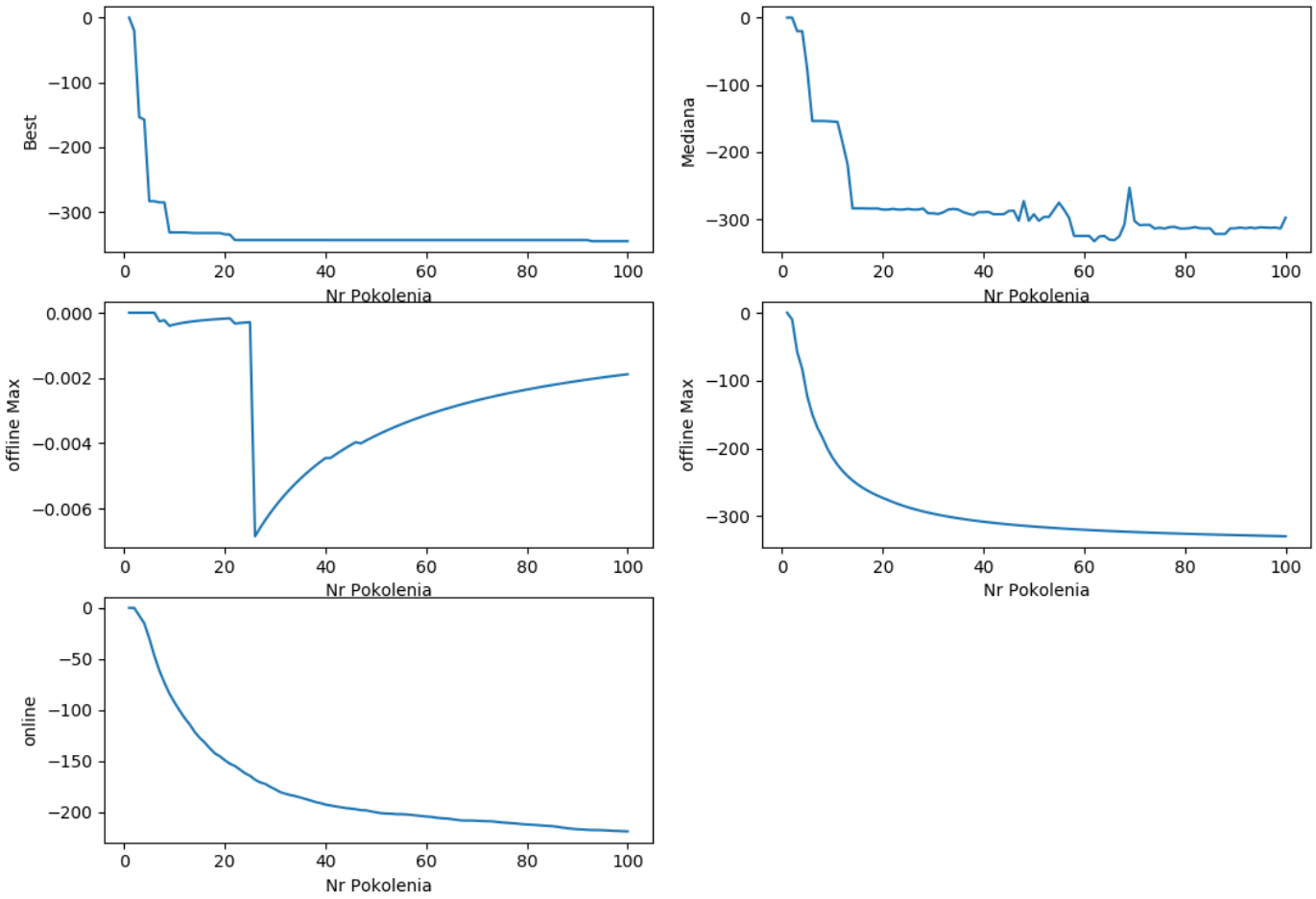
B. Metoda ruletki

Rezultaty są zbliżone do pozostałych metod, natomiast metoda ruletki przy małej populacji osobników daje lepszą zbieżność online niż metoda rankingowa oraz jest wydajniejsza, gdyż zgodnie z ideą tej metody, nie sortujemy osobników (jest to operacją czasochłonną).

P. mutacji	P. krzyżowania	Metody krzyżowania	**X**	**Y**
0.01	0.3	uniform	-2.338318	0.001625
0.01	0.5	uniform	-2.338890	0.000862
0.01	0.6	uniform	0.781349	-3.124527
0.01	0.8	uniform	0.763420	0.000099
0.10	0.3	uniform	0.767426	-0.006393
0.10	0.5	uniform	-2.339844	0.000862
0.10	0.6	uniform	0.781540	-0.003532
0.10	0.8	uniform	0.784973	0.000099
0.01	0.3	onepoint	0.781349	0.000099
0.01	0.5	onepoint	0.781349	0.000099
0.01	0.6	onepoint	0.781731	0.003342
0.01	0.8	onepoint	0.781349	0.000099
0.10	0.3	onepoint	0.769333	0.000481
0.10	0.5	onepoint	0.781540	0.000290
0.10	0.6	onepoint	0.781731	0.003342
0.10	0.8	onepoint	0.781921	0.000671
0.01	0.3	twopoint	0.781349	-3.124908
0.01	0.5	twopoint	0.781349	0.000099
0.01	0.6	twopoint	0.781349	0.000099
0.01	0.8	twopoint	0.781349	0.000099

0.10	0.3	twopoint	0.781349	0.001244
0.10	0.5	twopoint	0.781921	0.000481
0.10	0.6	twopoint	0.782303	0.001625
0.10	0.8	twopoint	0.772003	0.002579
0.01	0.3	evenodd	-2.339081	0.000099
0.01	0.5	evenodd	-2.343658	-3.124908
0.01	0.6	evenodd	0.781349	-3.124908
0.01	0.8	evenodd	-2.344231	0.002007
0.10	0.3	evenodd	0.771240	0.003151
0.10	0.5	evenodd	0.775436	-3.121094
0.10	0.6	evenodd	0.767426	0.000481
0.10	0.8	evenodd	0.765709	0.003532

Przykładowe rezultaty dla metody ruletkowej (przypadek tmp_0.01_0.80_100_roulette_uniform.raw).



Wykres ‘offline Max’ ma wartości na osi Y bliskie zero, dlatego ‘skok’ nie jest niepokojący.

Już po około 10 pokoleniach wartość najlepszego osobnika jest relatywnie bliska końcowemu. W tym przypadku dobór relatywnie wysokiego współczynnika krzyżowania (0.8) jest korzystny. Mediana zgodnie z oczekiwaniami maleje wraz ze wzrostem pokolenia.

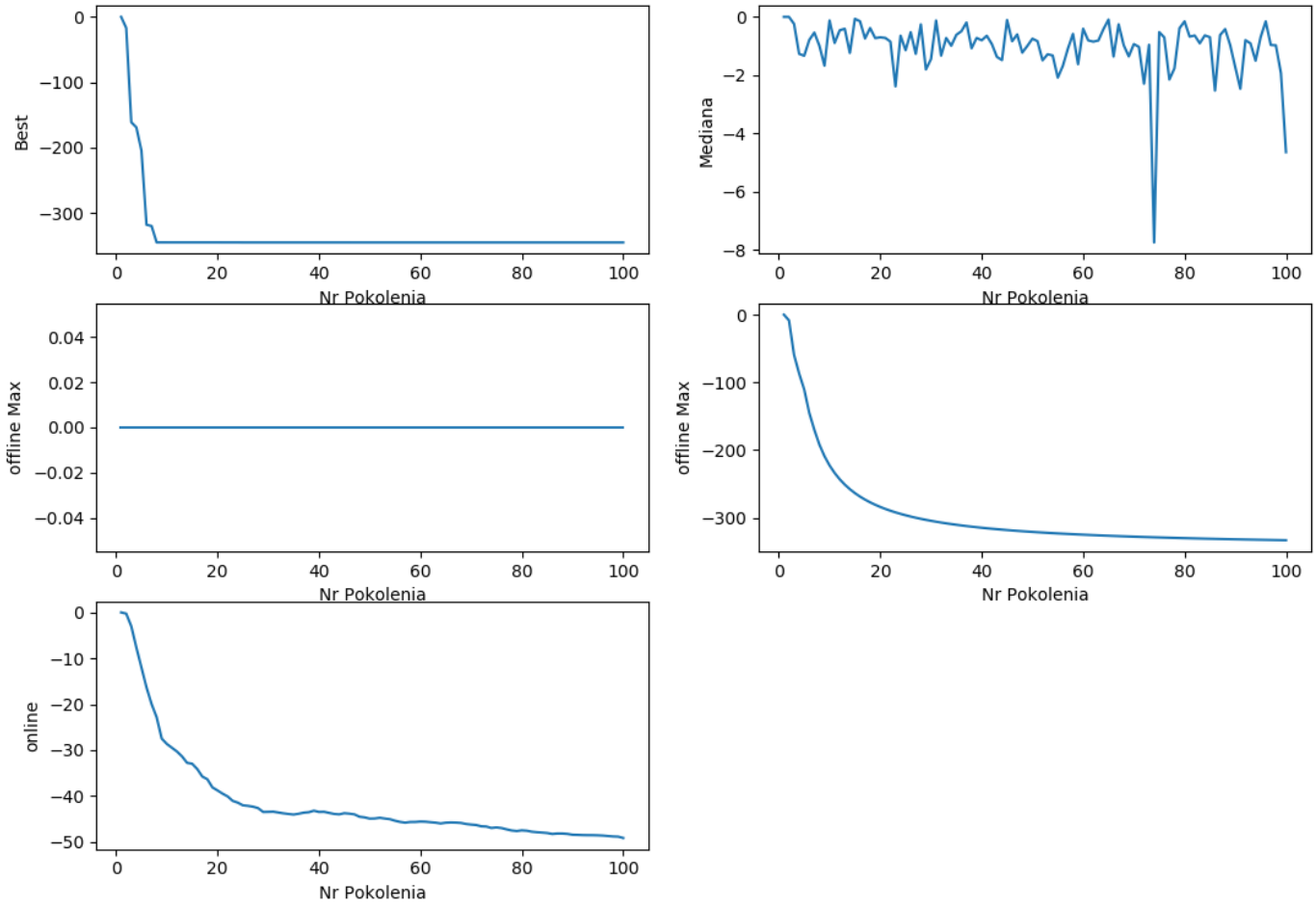
C. Metoda turniejowa

W przypadku metody turniejowej (dla naszej zadanej funkcji) rezultaty są bardzo podobne w porównaniu z innymi metodami. W tym przypadku jest bliskie prawdopodobieństwo, że zarówno dla małych jak i dużych wartości krzyżowania może wystąpić utknięcie algorytmu w minimum lokalnym. W metodzie turniejowej dla mojego przypadku, po przeanalizowaniu wyników stwierdziłem, że niska wartość parametru mutacji jest tutaj trochę niekorzystna. Wartości pozostałych parametrów nie mają decydującego wpływu na rezultaty końcowe.

P. mutacji	P. krzyżowania	Metody krzyżowania	**X**	**Y**

0.01	0.5	uniform	0.769142	0.000099
0.01	0.5	onepoint	-2.346138	-3.124908
0.01	0.5	twopoint	0.781349	0.000099
0.05	0.5	twopoint	0.781349	0.000099
0.05	0.5	onepoint	0.781158	-3.124908
0.05	0.5	uniform	0.781349	0.000290
0.10	0.5	uniform	0.782112	0.001053
0.10	0.5	onepoint	0.769524	0.000481
0.10	0.5	twopoint	0.768761	0.000481
0.01	0.3	twopoint	-2.339462	0.000099
0.01	0.4	twopoint	0.781349	0.000099
0.01	0.5	twopoint	0.781349	0.000099
0.01	0.6	twopoint	0.781349	0.000099
0.01	0.7	twopoint	-2.343849	0.000099
0.01	0.8	twopoint	0.781349	0.000099

Przykładowe rezultaty dla metody ruletkowej (przypadek tmp_0.01_0.50_100_tournament_twopoint.raw)

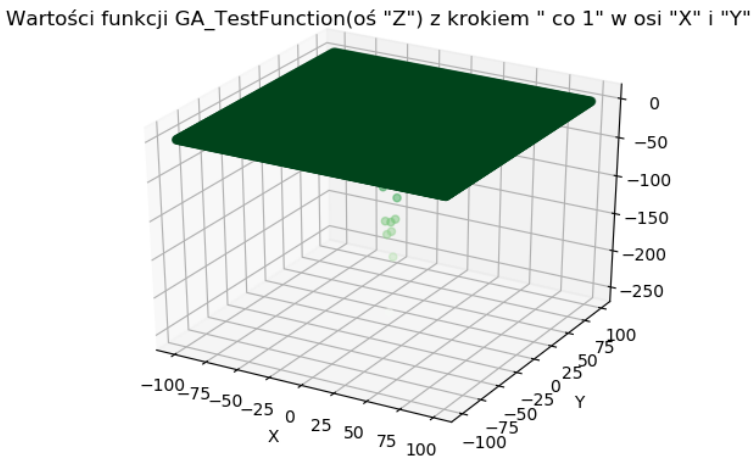


4. Wnioski

Na podstawie otrzymywanych rezultatów, wnioskuję że badana funkcja ma ekstremum w okolicy początku układu współrzędnych. Analizując nieznaną funkcję różnymi metodami oraz różnymi wartościami parametrów, można stwierdzić że najlepsze wyniki online daje metoda ruletkowa przy stosunkowo dużej wartości krzyżowania, natomiast dobre wyniki offline można uzyskać dla każdej z 3 zastosowanych metod. W przypadku badanej funkcji wybrałbym za najlepszą metodę ruletkową, ponieważ ta metoda w porównaniu z metodą turniejową umożliwia z większym prawdopodobieństwem przetrwanie gorszych osobników w populacji, natomiast w porównaniu z metodą rankingową będzie bardziej wydajna obliczeniowo.

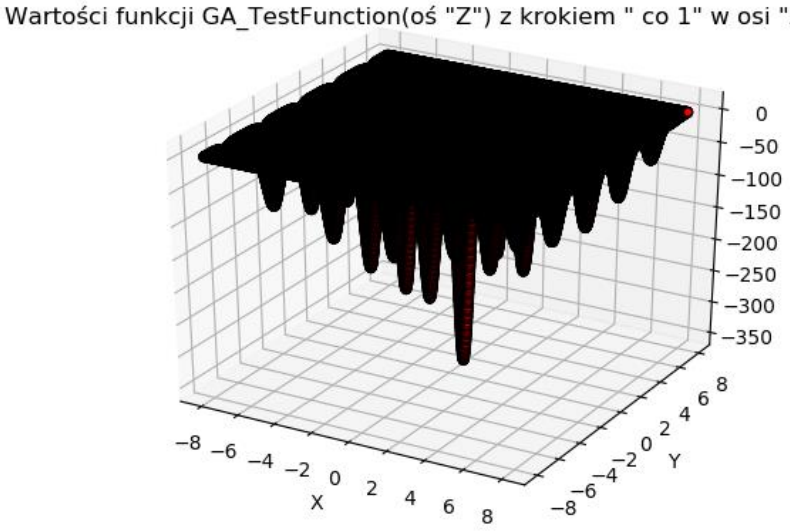
5. Dodatek

W celu wstępnej oceny poprawności otrzymywanych rezultatów, wykonałem zobrazowanie nieznanej przez nas płaszczyzny (**funkcji - numer 16**) w widoku 3D. Do wykonania tego zadania stworzyłem plik ***analiza.c*** oraz ***myplot.py***. Początkowo sprawdziłem wartości ‘min X’, ‘max X’, ‘min Y’ i ‘max Y’. Dla funkcji nr 16 wynoszą one odpowiednio -100, 100, -100 i 100. W tym celu wykonałem próbkowanie funkcji ciągłej z krokiem ‘1’. Uzyskałem w ten sposób płaszczyznę:



Ten obraz oraz poniższe **służą wyłącznie do wykonania subiektywnej oceny poprawności otrzymywanych rezultatów**. Tak wykonania operacja próbkowania funkcji ciągłej może być powodem utraty informacji o minimum globalnym tej funkcji, ale **zobrazowanie tej funkcji ma nam pomóc w ocenie otrzymanych rezultatów otrzymanych z użyciem algorytmu genetycznego**. W celu wykonania dokładniejszej analizy graficznej, zmniejszyłem obszar, w którym przypuściłem, że znajduje się ekstremum oraz zmniejszyłem krok próbkowania funkcji z wartości ‘1’ do ‘0.001’

Poniższa analiza wskazuje, że możemy założyć, że nasz algorytm genetyczny ma zdolność do poprawnej identyfikacji minimum/maksimum globalnego zadanej funkcji z relatywnie małą niedokładnością (nie znamy prawdziwej wartości ekstremum funkcji). Poniżej przedstawione rysunki powodują u nas przeświadczenie, że poszukiwane ekstremum naszej funkcji znajduje się w bliskiej okolicy punktu $X \approx 0$ i $Y \approx 0.76 \pm 0.03$



Wartości funkcji GA_TestFunction(oś "Z") z krokiem " co 1" w osi "X" i "Y"

Wartości funkcji GA_TestFunction(oś "Z") z krokiem " co 1" w osi "X" i "Y"