

WSI – ćwiczenie 2.

Algorytmy ewolucyjne

Denys Fokashchuk, 323944

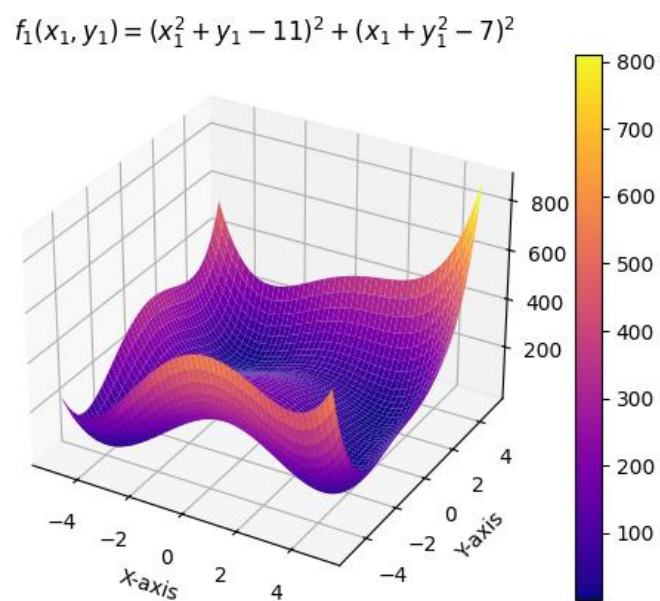
Zadanie

Należy zaimplementować algorytm ewolucyjny służący do minimalizacji sumy funkcji f_1 oraz f_2 .

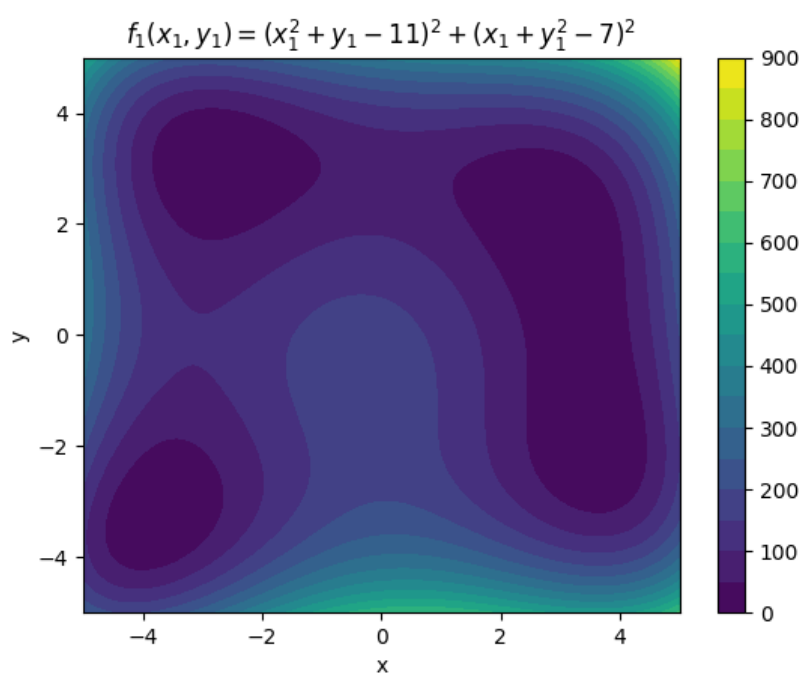
$$f_1(x_1, y_1) = (x_1^2 + y_1 - 11)^2 + (x_1 + y_1^2 - 7)^2$$

$$f(x_2, y_2) = 2 * x_2^2 + 1.05 * x_2^4 + \frac{x_2^6}{6} + x_2 y_2 + y_2^2$$

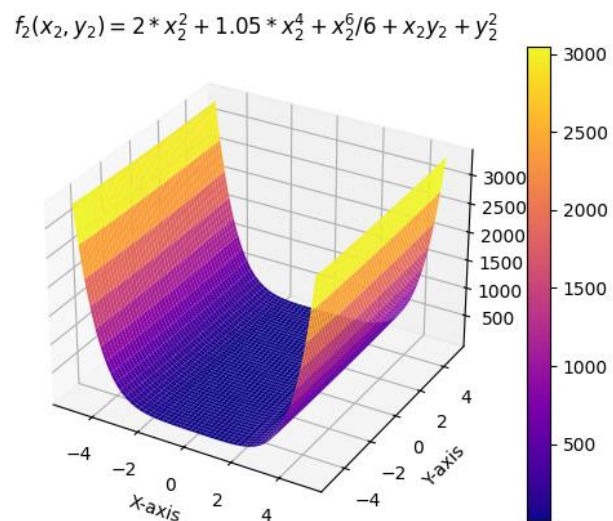
Wykresy funkcji f_1, f_2



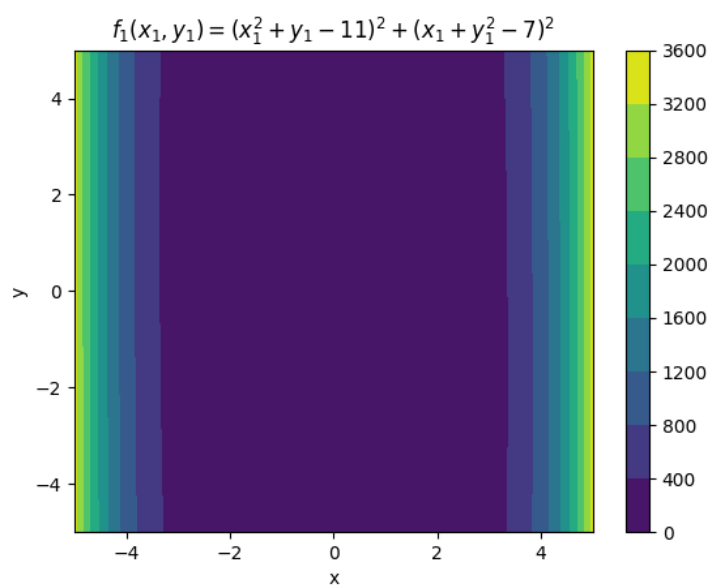
Rysunek 1. Wykres powierzchni funkcji $f_1(x)$



Rysunek 2. Wykres konturowy funkcji $f_1(x)$



Rysunek 3. Wykres powierzchni funkcji $f_2(x)$



Rysunek 4. Wykres konturowy funkcji $f_2(x)$

Ogólne uwagi odnośnie do mojego rozwiązania zadania

1. *algorithm.py*:

Główne funkcje potrzebne do realizacji algorytmu ewolucyjnego mieszczą się w tym pliku. Stworzyłem klasę *Individual* (osobnik), która zawiera tylko pole *genome* (genom). Z jednej strony to może wydać się bezsensowne, jednak jeśli osobnik będzie potrzebował więcej informacji niż jednego genomu, to będzie mniej pracy do refaktoringu.

2. *functions.py*

Plik służący do definicji funkcji f_1 i f_2 , a także określa dziedzinę ww. funkcji.

3. *plot.py*

Mieści funkcji niezbędne do robienia wykresów powierzchni, konturowych oraz słupkowych. Nie wywoła funkcji *pyplot.show()*, więc należy to dopisać ręcznie po wywołaniu funkcji rysowania.

4. *utility.py*

Posiada funkcje pomocnicze, a także funkcję do zapisywania wyników do pliku z rozszerzeniem .csv.

5. *main.py*

Skrypt, który należy uruchamiać w celu zobaczenia działania algorytmu. Jest podzielony na sekcje, które należy „odkomentować” i komentować. Są niezależne i wykonują różne czynności: uruchamiają działania algorytmu, robią wykresy oraz zapisują dane do pliku. Każda sekcja została poprzedzona komentarzem z opisem co robi poszczególna sekcja kodu.

Działanie algorytmu

Selekcja turniejowa ($k=2$)

Selekcja turniejowa działa w taki sposób, że losuje się dwóch osobników z całej populacji ze zwróceniem (czyli jeden osobnik może zostać wylosowany kilkakrotnie) i osobnik będący lepszym (w danym zadaniu lepszy osobnik, to ten, który ma mniejszą ocenę) przechodzi do następnego pokolenia. Operacja powtarza się n razy, gdzie n – to liczba osobników w populacji. Na rysunku 5 oraz 6 pokazano przykładową populację, składającą się z 20 osobników, do selekcji turniejowej i po niej odpowiednio. W każdym wierszu pokazano genom osobnika, a obok wartość kosztu. Obie populacje są posortowane rosnąco według kosztu.

```
[-2.7682, 2.47409, -0.1178, 0.45969] cost: 14.23088170577331
[-3.93117, -1.95485, 0.4741, 1.54441] cost: 60.41599433963357
[-4.21509, -4.20178, -0.22249, -3.95068] cost: 64.64060470965467
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-4.19474, -1.73857, -0.76266, 0.14363] cost: 91.83911097948209
[4.24695, 0.34707, 1.67637, 3.80119] cost: 99.88198157561632
[-0.49876, 0.74473, -0.22104, 4.51827] cost: 167.8676843640418
[-1.66099, 1.66126, 2.6142, 2.30862] cost: 205.38762776163077
[0.23047, -2.61464, 1.72062, -3.64254] cost: 210.3689234298733
[3.83727, -4.16864, -1.13996, -4.00375] cost: 227.59098911525678
[3.06848, 0.23615, 3.24998, 4.08126] cost: 381.42352812859735
[3.99845, 4.49646, -0.25747, 1.82969] cost: 389.3752551311614
[1.16955, -3.72374, 3.26647, -2.7265] cost: 584.8085864245987
[4.85949, -4.25587, 3.31602, 1.19764] cost: 700.9137578364581
[2.36891, -4.96555, -2.93301, 0.80284] cost: 707.5279005013942
[0.8812, 4.25058, -4.2284, -2.98964] cost: 1524.027303409817
[0.66801, -0.4083, -4.27026, -3.13982] cost: 1577.6519164807332
[2.49211, 4.60993, 4.266, -3.7137] cost: 1667.0349300555006
[3.49731, -4.13602, 4.31372, 3.17664] cost: 1691.993128806762
[-4.22053, 1.73458, 4.64739, 4.24048] cost: 2390.3821663561775
```

Rysunek 5. Przykładowa populacja przed selekcją

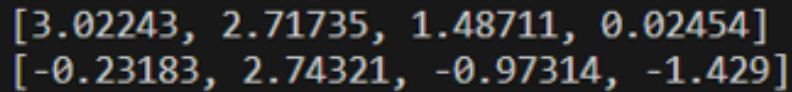
```
[-2.7682, 2.47409, -0.1178, 0.45969] cost: 14.23088170577331
[-3.93117, -1.95485, 0.4741, 1.54441] cost: 60.41599433963357
[-4.21509, -4.20178, -0.22249, -3.95068] cost: 64.64060470965467
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-2.69143, -2.68935, -1.67236, 2.29451] cost: 66.47152710851569
[-4.19474, -1.73857, -0.76266, 0.14363] cost: 91.83911097948209
[-4.19474, -1.73857, -0.76266, 0.14363] cost: 91.83911097948209
[-0.49876, 0.74473, -0.22104, 4.51827] cost: 167.8676843640418
[-1.66099, 1.66126, 2.6142, 2.30862] cost: 205.38762776163077
[0.23047, -2.61464, 1.72062, -3.64254] cost: 210.3689234298733
[3.06848, 0.23615, 3.24998, 4.08126] cost: 381.42352812859735
[1.16955, -3.72374, 3.26647, -2.7265] cost: 584.8085864245987
[4.85949, -4.25587, 3.31602, 1.19764] cost: 700.9137578364581
[0.8812, 4.25058, -4.2284, -2.98964] cost: 1524.027303409817
[2.49211, 4.60993, 4.266, -3.7137] cost: 1667.0349300555006
[2.49211, 4.60993, 4.266, -3.7137] cost: 1667.0349300555006
[-4.22053, 1.73458, 4.64739, 4.24048] cost: 2390.3821663561775
```

Rysunek 6. Przykładowa populacja po selekcji

Można zauważyć, że po selekcji więcej osobników mających mniejszy koszt przeszło do następnej populacji. Jednak może się wydarzyć taka sytuacja, iż najlepszy osobnik nie został w ogóle wylosowany i w takim razie on nie przejdzie do następnej generacji. Z powyższych dwóch rysunków można zauważyć, iż najgorszy osobnik też przeszedł, a spowodowano to tym, iż podczas jednego turnieju został on wylosowany dwukrotnie.

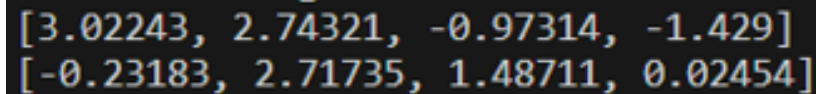
Krzyżowanie jednopunktowe

Celem krzyżowania jest przekazanie dwóm potomkom cech obu rodziców. Losowo wybiera się dwóch rodziców oraz punkt przecięcia. Następnie pierwszy potomek ma kod rodzica od początku do punktu przecięcia, a następnie ma kod rodzica 2. Drugi potomek dostaje kod drugiego rodzica do punktu przecięcia, a następnie kod pierwszego rodzica. Genom rodziców i potomków po krzyżowaniu jednopunktowym zilustrowano na rysunkach 7 i 8. Punkt przecięcia w danym przypadku następuje po 1. genie.



[3.02243, 2.71735, 1.48711, 0.02454]
[-0.23183, 2.74321, -0.97314, -1.429]

Rysunek 7. Genom dwóch rodziców

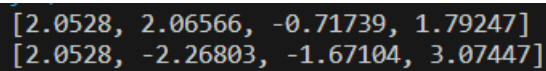


[3.02243, 2.74321, -0.97314, -1.429]
[-0.23183, 2.71735, 1.48711, 0.02454]

Rysunek 8. Genom dwóch potomków uzyskanych po krzyżowaniu

Mutacja gaussowska

Celem mutacji jest generacja genomu, który jest bliski do będącego przed mutacją. Każdy gen z określonym prawdopodobieństwem może zostać zmutowany. Jeśli mutacja zachodzi, to do aktualnej wartości genu dodaje się liczba losowa z rozkładu normalnego, która została przeskalowana przez pewną liczbę, nazywaną siłą mutacji. Przykład działania zaimplementowanej przeze mnie mutacji z prawdopodobieństwem mutacji równym 0.7 oraz siłą mutacji 2 zilustrowano na rysunku 9. Pierwszy wiersz pokazuje genom przed mutacją, a drugi – po. Można zauważyć, iż pierwszy gen nie został zmutowany, a inne zostały, przy czym różnica wartości poszczególnych genów przed mutacją i po jest różna, ponieważ za każdym razem losuje się nową wartość z rozkładu normalnego.



```
[2.0528, 2.06566, -0.71739, 1.79247]  
[2.0528, -2.26803, -1.67104, 3.07447]
```

Rysunek 9. Przykładowy gen przed i po mutacji

Sukcesja generacyjna

Są różne typy sukcesji, czyli określenie tego, które osobniki przejdą do następnego pokolenia. Sukcesja generacyjna nie zachowa najlepszych osobników, które zostaną przechowane w kolejnej populacji, w przeciwieństwie do sukcesji elitarniej. W sukcesji generacyjnej przechodzą mutanci do następnej iteracji, a więc w tym zadaniu nie trzeba było tworzyć osobnej funkcji *generational_succession*, bo jej działanie sprowadzałoby się do tego, iż przyjmowała populację mutantów, i przekazywała ich dalej. Jednak jeśli będzie potrzeba zmienić wariant sukcesji, to będzie to łatwo zrobić. Należy tylko określić parametr *succession_func* w funkcji *run_simulation*. Zrobiłem to w celu tworzenia bardziej ogólnego algorytmu ewolucyjnego.

Szukanie zestawu dobrych hiperparametrów

W celu znalezienia dobrych hiperparametrów postanowiłem zrobić po 5 symulacji działania algorytmu dla każdego wylosowano-wybranego zestawu hiperparametrów (jest ich 7 i oczywiście wartości wybrane są zgodnie ze zdrowym rozsądkiem), a następnie sprawdziłem dla których hiperparametrów wartość znalezionych minimów dla funkcji f_1 jest zazwyczaj najlepsza. Kod, który został użyty do symulacji ułożono w sekcji 7 w pliku main.py. Otrzymane dane zostały zaprezentowane w tabeli 1.

prawdopodobieństwo krzyżowania	prawdopodobieństwo mutacji	siła mutacji	liczba osobników w populacji	liczba iteracji	x1	y1	x2	y2	wartość $f_1(x_1, y_2)$	wartość $f_2(x_2, y_2)$	wartość
0.8	0.2	1.8	100	100	3.0008	2.0447	0.1167	0.0272	0.0354	0.0313	0.0668
0.8	0.2	1.8	100	100	2.9898	2.0652	0.1077	-0.2119	0.065	0.0454	0.1105
0.8	0.2	1.8	100	100	2.9521	2.0272	-0.0409	0.2945	0.0703	0.078	0.1484
0.8	0.2	1.8	100	100	3.011	1.9229	-0.0093	-0.1893	0.0851	0.0378	0.1229
0.8	0.2	1.8	100	100	3.5457	-1.9022	-0.091	0.0433	0.136	0.0146	0.1504
0.7	0.1	1.5	50	20	2.9457	2.1036	-0.0492	0.1817	0.1856	0.0289	0.2144
0.7	0.1	1.5	50	20	3.0162	2.0387	-0.4027	0.3787	0.0483	0.3436	0.3919
0.7	0.1	1.5	50	20	3.6775	-1.6704	-0.4456	0.5215	1.0119	0.4794	1.4911
0.7	0.1	1.5	50	20	3.098	1.9287	-0.0369	-0.1933	0.3102	0.0472	0.3573
0.7	0.1	1.5	50	20	2.9705	2.1231	-0.0282	0.0733	0.2313	0.0049	0.2364
0.5	0.4	2.8	60	90	-2.7529	3.0737	-0.1853	0.7348	0.2142	0.4737	0.6878
0.5	0.4	2.8	60	90	3.0292	1.8272	0.5122	0.5791	0.3996	1.2319	1.6315
0.5	0.4	2.8	60	90	-2.8436	3.0609	-0.2488	-0.082	0.2467	0.155	0.4019
0.5	0.4	2.8	60	90	2.9435	2.3755	0.4744	-0.4202	2.5186	0.4824	3.0016
0.5	0.4	2.8	60	90	3.0225	1.918	-0.3205	0.8237	0.0921	0.6312	0.7234
0.95	1	1.2	20	200	3.4612	-1.7688	-0.7927	-0.277	0.7906	2.009	2.7995
0.95	1	1.2	20	200	3.6113	-1.7133	-0.1851	0.8002	0.3132	0.562	0.8751
0.95	1	1.2	20	200	-2.9584	2.978	-0.0944	0.9428	1.721	0.8178	2.5395
0.95	1	1.2	20	200	3.6516	-1.5854	-0.1699	-0.7036	1.2577	0.6732	1.9312
0.95	1	1.2	20	200	-2.593	3.1143	-0.0596	-0.1352	1.3616	0.0335	1.395
0.4	0.9	1.3	68	80	3.5476	-1.7817	-0.4265	-0.7301	0.1158	1.244	1.3595
0.4	0.9	1.3	68	80	3.6208	-1.8134	0.2654	0.7534	0.0963	0.9137	1.0101
0.4	0.9	1.3	68	80	-2.7142	3.2001	0.0313	-1.0702	0.4646	1.1138	1.5789
0.4	0.9	1.3	68	80	3.4199	-1.8738	-0.0163	0.0605	1.3926	0.0032	1.3954
0.4	0.9	1.3	68	80	3.6072	-1.8637	-0.118	-0.5684	0.0285	0.4182	0.4467
0.9	0.2	2.7	200	200	2.9829	2.0129	0.1925	-0.2878	0.0092	0.103	0.1121
0.9	0.2	2.7	200	200	3.0083	2.0189	0.0086	-0.1329	0.0118	0.0167	0.0285
0.9	0.2	2.7	200	200	2.9588	1.9842	-0.1262	0.037	0.0791	0.0288	0.108
0.9	0.2	2.7	200	200	3.0459	2.0024	0.1126	-0.0533	0.0814	0.0224	0.1038
0.9	0.2	2.7	200	200	2.9997	2.022	-0.2397	0.0625	0.0082	0.1073	0.1155
0.1	1	2.6	200	280	2.9465	2.0719	0.2234	-0.5026	0.1179	0.2428	0.3607
0.1	1	2.6	200	280	2.8911	2.21	-0.0615	0.2472	0.7872	0.0535	0.8405
0.1	1	2.6	200	280	3.0736	2.0614	0.0941	0.0183	0.3628	0.0198	0.3823
0.1	1	2.6	200	280	2.8821	1.9634	0.2543	-0.1663	0.6022	0.1191	0.7208
0.1	1	2.6	200	280	3.6002	-1.7794	0.6168	-0.8498	0.0877	1.12	1.2076

Tabela 1. Eksperyment szukania najlepszych hiperparametrów

Dlatego, że z treści zadania wynika, iż najbardziej interesuje nas wartość znalezionego minimum funkcji f_1 , to kolumnę, która temu odpowiada jest różnokolorowa w celu ułatwienia czytania danych. Można zauważyć, że pierwszy oraz

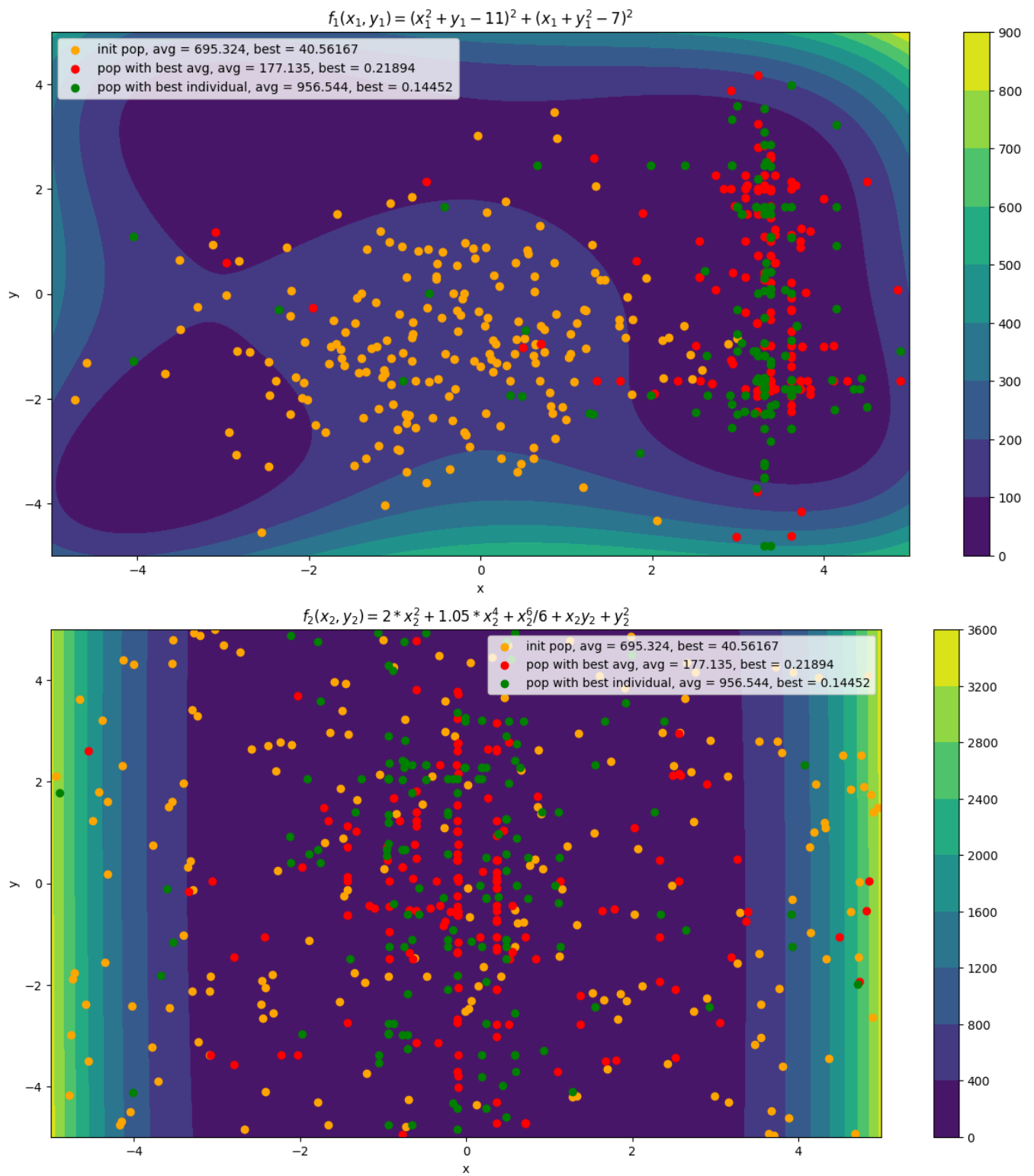
szósty zestaw hiperparametrów dają relatywnie dobry wynik. Liczba osobników w populacji z szóstego zestawu hiperparametrów jest na 100 osobników większa oraz liczba iteracji jest na tyleż większa, jednak to nie jest duża strata wydajności dla algorytmu, więc w celu dalszego badania algorytmu będę próbować ulepszyć te hiperparametry. Dodatkowo z tej tabeli można zauważyć, że kiedy prawdopodobieństwo mutacji jest duże, a prawdopodobieństwo krzyżowania jest małe, to wynik jest dość zły (jak dla ostatniego zestawu hiperparametrów) mimo wielkiej liczby osobników w populacji oraz dużej liczby iteracji.

Badanie wybranego zestawu hiperparametrów

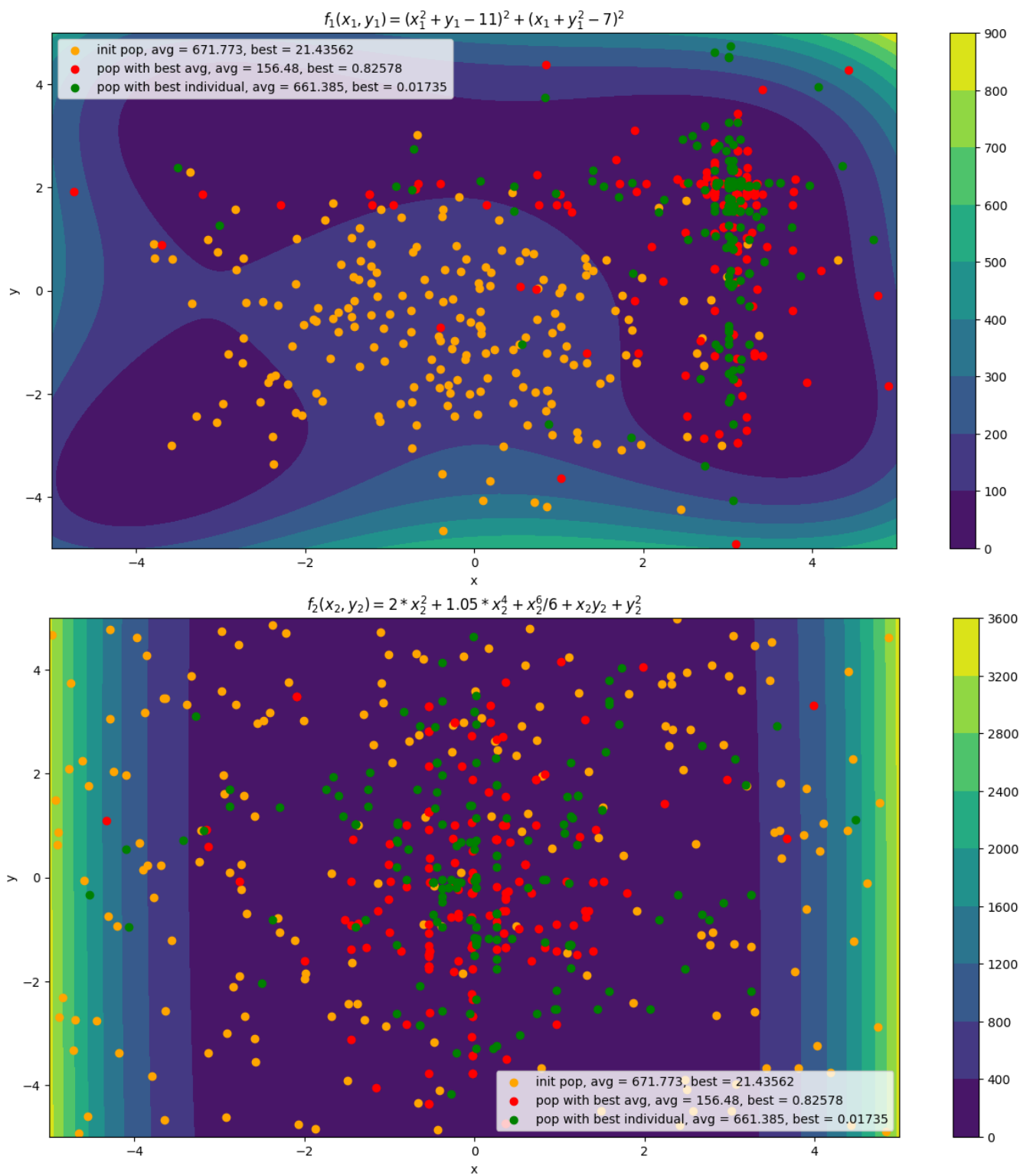
Wybrany zestaw hiperparametrów to:

- prawdopodobieństwo krzyżowania = 0.9
- prawdopodobieństwo mutacji = 0.2
- siła mutacji = 2.7
- liczba osobników w populacji = 200
- liczba iteracji = 200

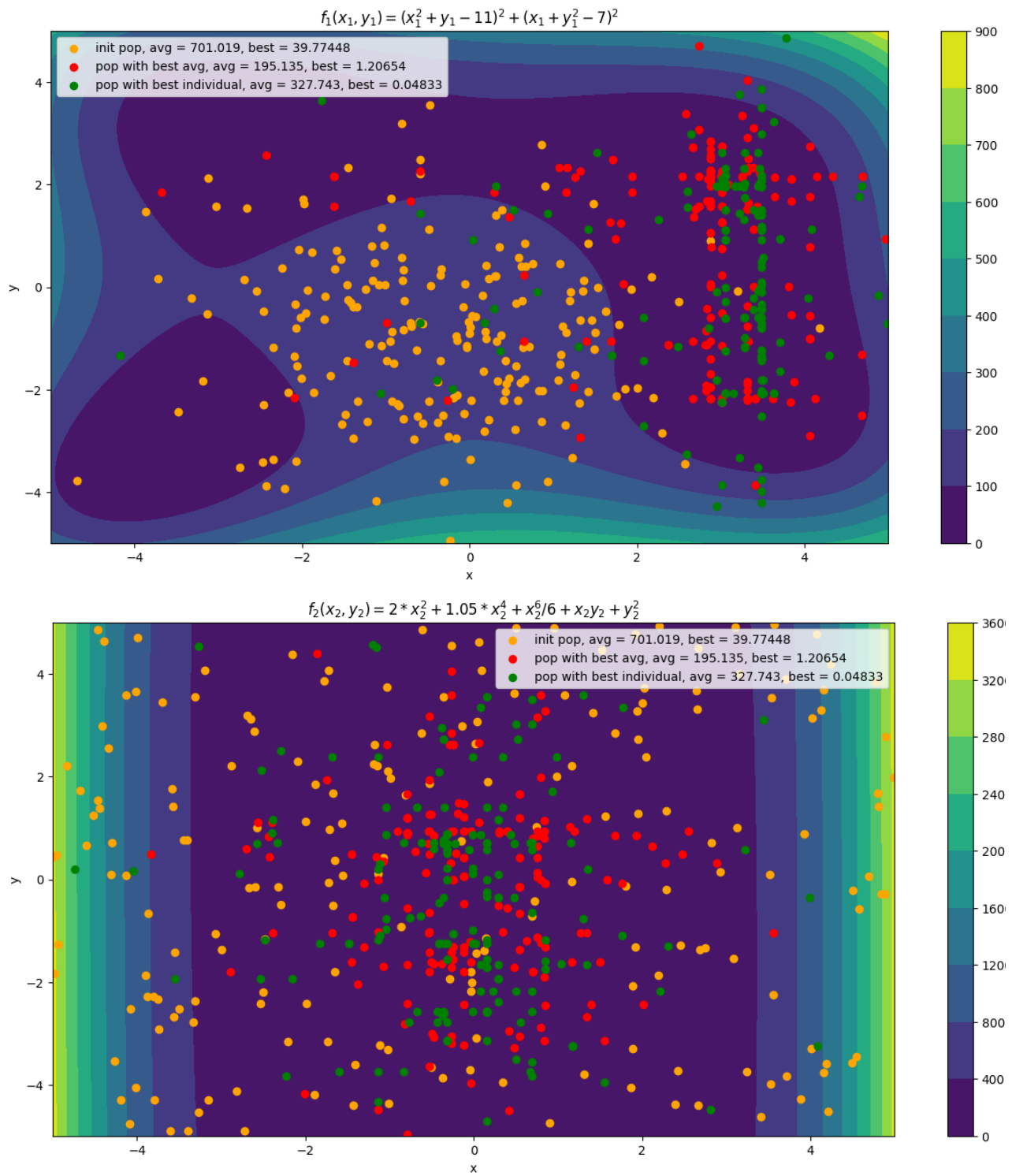
Uruchomiłem 3 razy algorytm dla wybranego zestawu danych w przypadku dystrybucji populacji początkowej zgodnemu z rozkładem normalnym wokół punktów $x_1 = -0.3$, $y_1 = -0.9$ z odchyleniem standardowym = 1.5. A dystrybucja początkowa dla punktów x_2 i y_2 jest jednorodna po całym obszarze -5×5 . Wykresy z populacją początkową, populacją z najlepszym osobnikiem oraz populacją z najlepszą przeciętną oceną pokazano na rysunkach 10-12 dla funkcji f_1 i poniżej dla funkcji f_2 .



Rysunek 10. Test algorytmu ewolucyjnego, próba 1

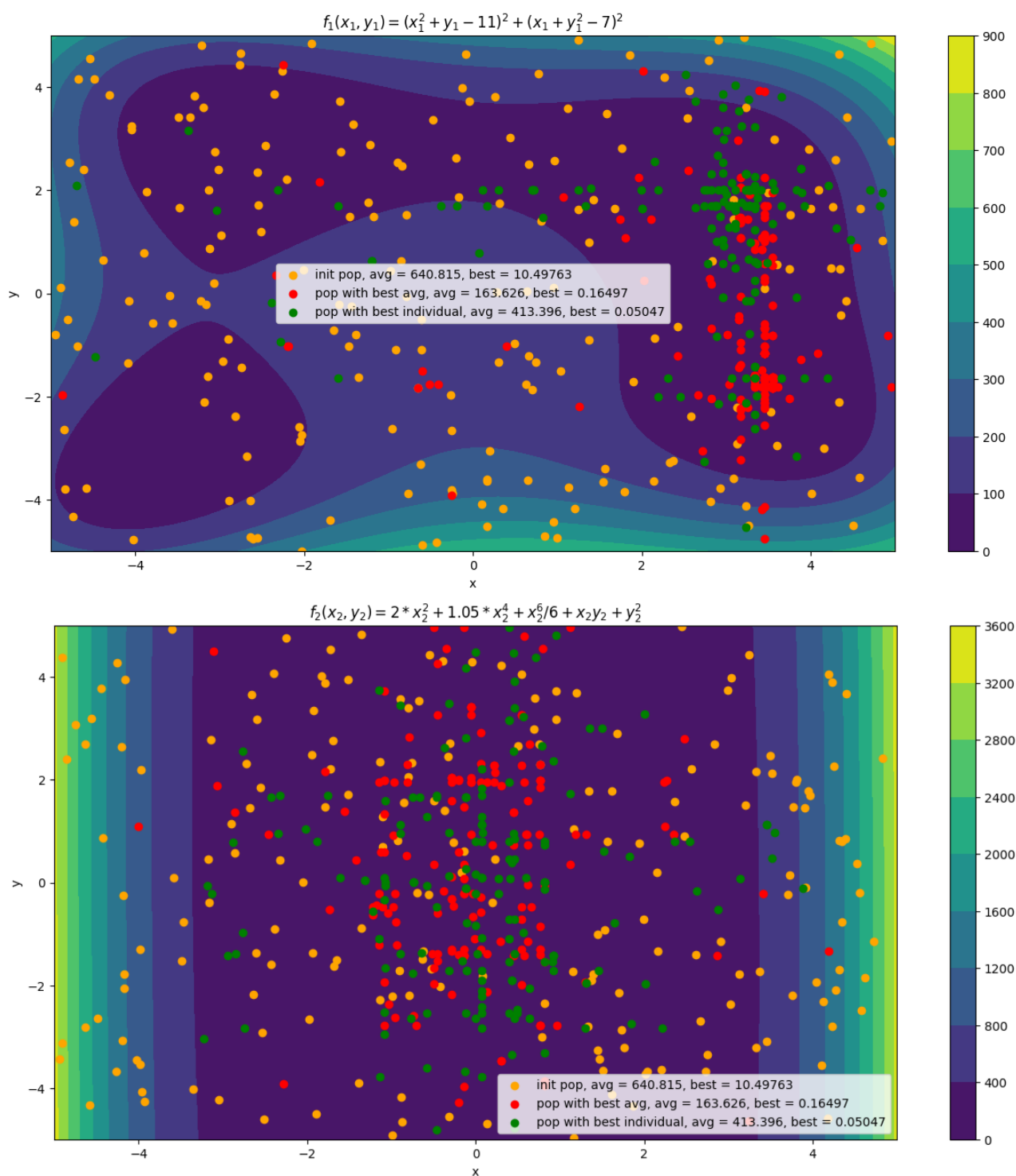


Rysunek 11. Test algorytmu ewolucyjnego, próba 2

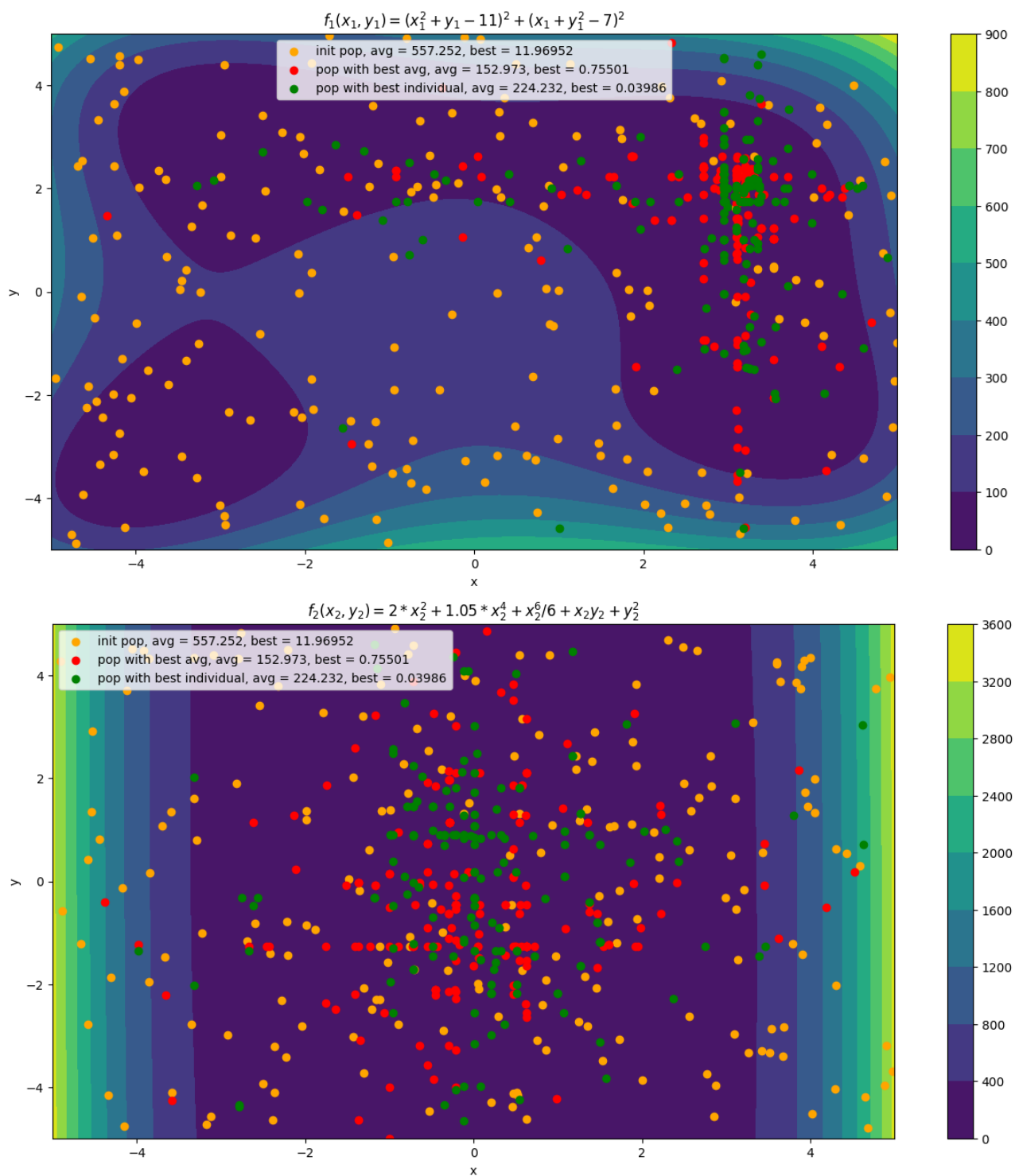


Rysunek 12. Test algorytmu ewolucyjnego, próba 3

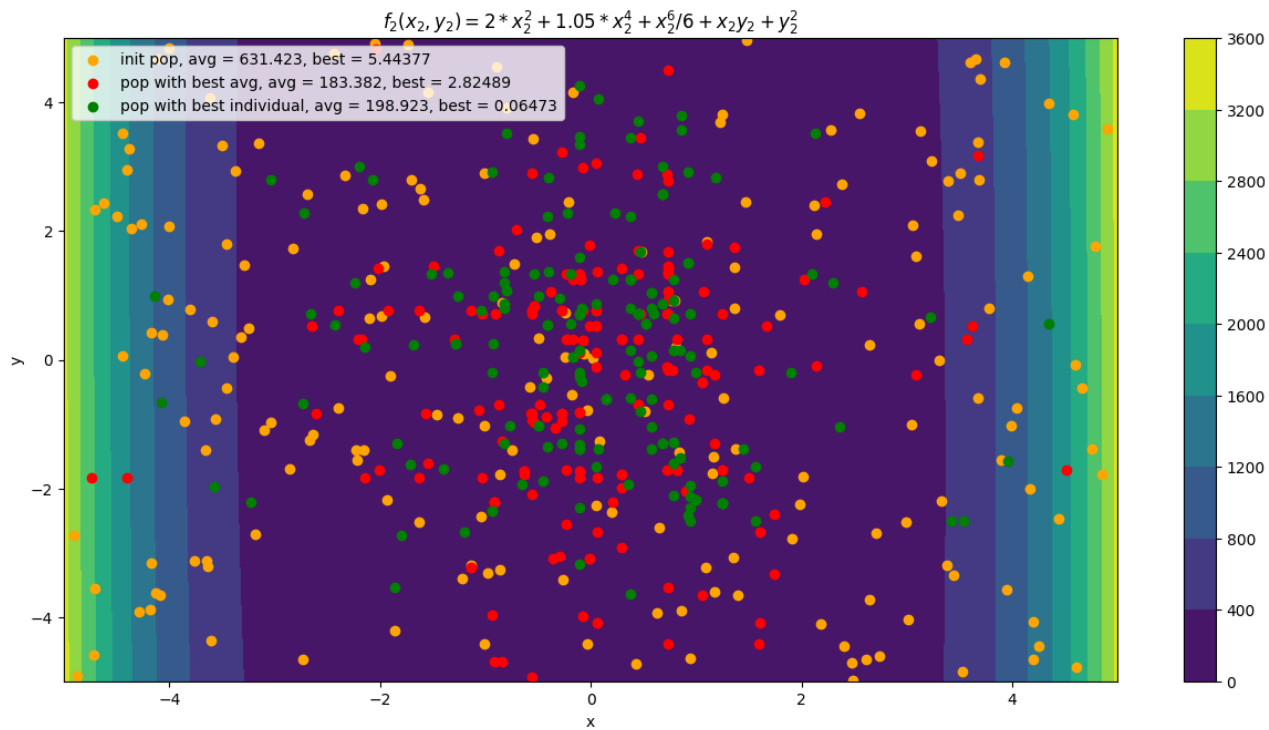
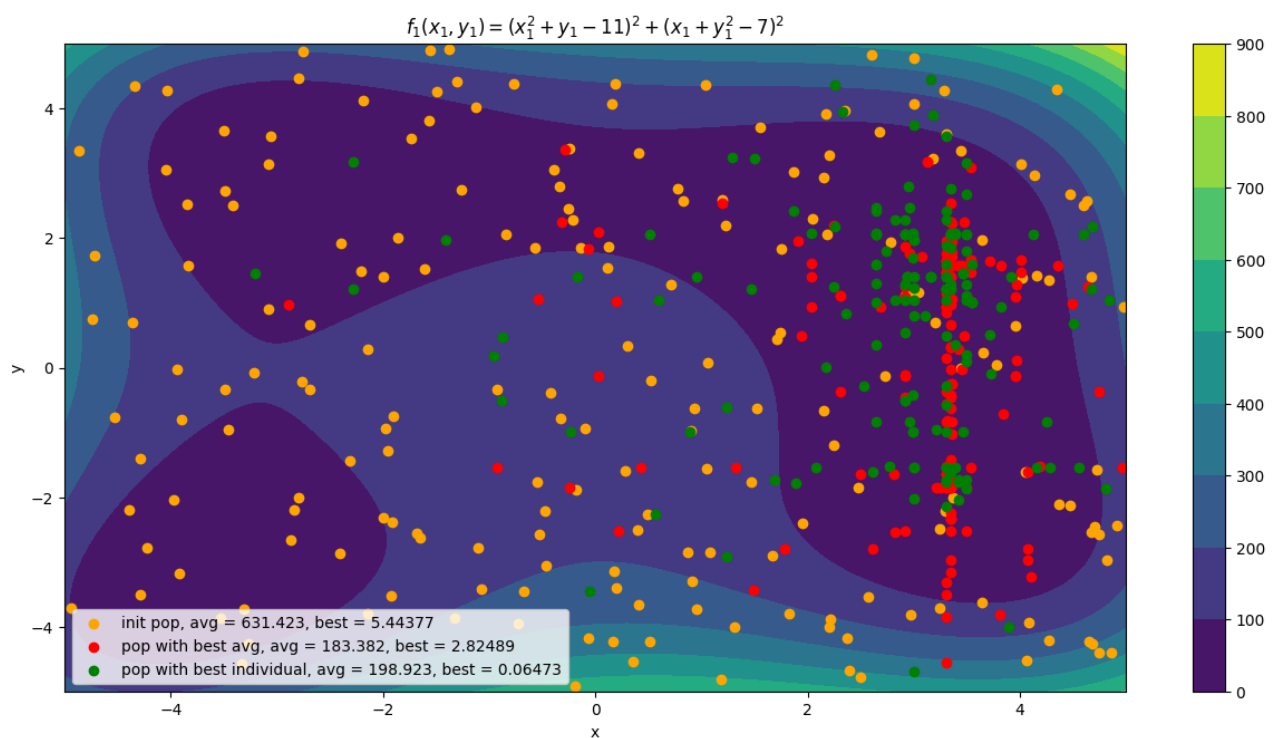
Następnie zrobiłem jeszcze 3 testy, tylko, zmieniłem implementację dystrybucji początkowej: jest jednorodna dla wszystkich punktów. Dany eksperyment został zilustrowany na rysunkach 13-15.



Rysunek 13. Test algorytmu ewolucyjnego, próba 1



Rysunek 14. Test algorytmu ewolucyjnego, próba 2



Rysunek 15. Test algorytmu ewolucyjnego, próba 3

Przeanalizowawszy dane, umieszczone na wykresach na rysunkach 10 – 15 można dojść do wniosków, że:

1. dla populacji, mającej najlepszą przeciętną ocenę wartość przeciętnego kosztu jest podobna dla obu różnych dystrybucji;
2. wartość najlepszego osobnika z wszystkich iteracji populacji prawie się nie różni przy porównaniu tych dwóch różnych dystrybucji populacji początkowych;
3. wartość najlepszego osobnika w pierwszej populacji w przypadku jednorodnej dystrybucji po całym obszarze jest lepsza od dystrybucji początkowej dla rozkładu normalnego dla punktów $x_1=-0.3$, $y_1=-0.9$. Spowodowano to kształtem funkcji.

Reasumując, można powiedzieć, że w danym przypadku dystrybucja początkowa nie bardzo ma wpływ, jednak w niektórych przypadkach jej wpływ może być znaczący. Na przykład, dla przypadku dystrybucji początkowej z rozkładem normalnym wokół jakiegoś minimum lokalnego algorytm ewolucyjny raczej nie byłby w stanie „wyjść” z tego minimum, a więc prawdopodobnie znajdzie minimum lokalny, wokół którego ta dystrybucja początkowa była.

Badanie zmiany prawdopodobieństwa oraz siły mutacji

W tym zadaniu rozważę sześć przypadków dla których prawdopodobieństwo mutacji oraz siła mutacji będzie różna. Dla każdego przypadku zrobię 5 testów, a następnie zachowam te wyniki do pliku. Otrzymane wyniki można zobaczyć na tabeli 2.

prawdopodo- bieństwo krzyżowani	prawdopodo- bieństwo mutacji	siła mutacji	liczba osobników w populac	liczba iteracji	x1	y1	x2	y2	wartość f1(x1, y1)	wartość f2(x2, y2)	wartość ć
0.9	0.2	2.7	200	200	3.0227	2.0085	-0.2346	0.0419	0.0243	0.1052	0.1296
0.9	0.2	2.7	200	200	2.9885	2.0291	0.1147	-0.2514	0.0128	0.0609	0.0737
0.9	0.2	2.7	200	200	2.9882	1.9638	0.0432	-0.1076	0.0355	0.0107	0.0462
0.9	0.2	2.7	200	200	2.994	2.0366	0.0959	-0.0337	0.0201	0.0164	0.0364
0.9	0.2	2.7	200	200	3.5811	-1.905	-0.1093	-0.0396	0.0507	0.0299	0.0805
0.9	0.5	2.7	200	200	2.9213	2.037	-0.1989	1.0132	0.189	0.9058	1.0946
0.9	0.5	2.7	200	200	3.1091	2.0636	0.3678	-0.0135	0.6681	0.2854	0.9532
0.9	0.5	2.7	200	200	-2.8477	3.1918	-0.1472	-0.4892	0.2062	0.3552	0.5613
0.9	0.5	2.7	200	200	3.0752	1.9726	-0.2656	-0.2682	0.1856	0.2895	0.4749
0.9	0.5	2.7	200	200	3.6077	-1.851	-0.2587	-0.0271	0.0282	0.1463	0.1746
0.9	0.8	2.7	200	200	-2.7673	3.0884	-0.5491	0.469	0.1168	0.6655	0.7821
0.9	0.8	2.7	200	200	-3.8832	-3.3196	-0.279	-0.0475	0.5957	0.1776	0.7731
0.9	0.8	2.7	200	200	2.8491	1.9743	0.4505	-0.4457	0.8891	0.4484	1.338
0.9	0.8	2.7	200	200	3.656	-1.8023	-0.4284	0.1361	0.3273	0.3637	0.6905
0.9	0.8	2.7	200	200	3.6158	-1.6936	0.2044	0.217	0.4109	0.1768	0.5878
0.9	0.2	0.5	200	200	3.0005	1.991	0.0042	-0.0126	0.0013	0.0001	0.0014
0.9	0.2	0.5	200	200	3.5826	-1.8437	0.0018	0.0159	0.0004	0.0003	0.0007
0.9	0.2	0.5	200	200	2.9988	1.9985	-0.0096	0.0335	0.0001	0.001	0.0011
0.9	0.2	0.5	200	200	3.004	1.9937	0.0285	-0.0258	0.0008	0.0016	0.0023
0.9	0.2	0.5	200	200	3.0038	2.0015	-0.0032	0.0173	0.0007	0.0003	0.0009
0.9	0.5	0.9	200	200	2.9909	1.9947	0.066	-0.0163	0.0045	0.0079	0.0125
0.9	0.5	0.9	200	200	3.5833	-1.8401	-0.0228	0.1654	0.0009	0.0246	0.0256
0.9	0.5	0.9	200	200	2.9847	1.9984	0.1814	0.0613	0.0092	0.0818	0.091
0.9	0.5	0.9	200	200	3.0374	1.9183	-0.0821	0.235	0.1007	0.0495	0.1501
0.9	0.5	0.9	200	200	3.6049	-1.895	-0.1698	-0.1129	0.0484	0.0905	0.139
0.9	0.8	1.3	200	200	2.967	2.0458	-0.3025	0.1156	0.046	0.1703	0.2163
0.9	0.8	1.3	200	200	3.0037	2.0016	-0.2251	-0.4404	0.0007	0.3971	0.3978
0.9	0.8	1.3	200	200	3.0228	2.0381	-0.3978	0.0746	0.062	0.3193	0.3815
0.9	0.8	1.3	200	200	3.0546	1.9555	0.2006	-0.4514	0.0966	0.1954	0.2919
0.9	0.8	1.3	200	200	-2.8107	3.1332	0.2781	-0.1283	0.0011	0.1418	0.1429

Tabela 2. Badanie zmiany wartości prawdopodobieństwa mutacji oraz siły mutacji

Z powyższych danych można z łatwością zauważyć, że najlepsza ocena jest dla czwartego zestawu wybranych parametrów (prawdopodobieństwo mutacji = 0.2, siła mutacji = 0.5). Przed przeprowadzeniem testu myślałem, iż najlepszy wynik będzie przy małym prawdopodobieństwie mutacji oraz relatywnie dużej sile mutacji, jednak okazało się trochę po innemu. Z tego można dojść do wniosku, że algorytm ewolucyjny (przynajmniej dla tego zadania) działa wspornie przy małym prawdopodobieństwie oraz małej sile mutacji. Po tym eksperymencie postanowiłem użyć algorytmu ewolucyjnego bez mutacji i wynik był znacznie gorszy, co pokazano na tabeli 3.

prawdopodobieństwo krzyżowania	prawdopodobieństwo mutacji	siła mutacji	liczba osobników w populacji	liczba iteracji	x1	y1	x2	y2	wartość f1(x1, y1)	wartość f2(x2, y2)	wartość
0.9	0	0	200	200	3.3485	-1.7079	0.1959	-0.37	2.776	0.1427	2.9197
0.9	0	0	200	200	-2.8012	2.9803	-0.017	0.2284	0.8745	0.0489	0.9229
0.9	0	0	200	200	2.9942	1.9921	-0.1266	0.0343	0.0032	0.0292	0.0324
0.9	0	0	200	200	-2.8529	3.1977	0.0258	-0.4232	0.2521	0.1695	0.4215
0.9	0	0	200	200	2.9195	2.1297	0.0949	-0.2737	0.3274	0.067	0.3946
0.9	0	0	200	200	2.9731	2.0359	-0.0493	0.2444	0.0295	0.0525	0.082
0.9	0	0	200	200	-2.8382	3.1668	-0.2751	0.634	0.0856	0.385	0.4705
0.9	0	0	200	200	2.399	1.4492	0.0968	-0.0128	20.6607	0.0178	20.6801
0.9	0	0	200	200	-3.7811	-3.2408	-0.1536	0.3631	0.0806	0.1238	0.2046
0.9	0	0	200	200	2.7485	1.9495	-0.184	0.0505	2.4421	0.0622	2.5049

Tabela 3. Wyniki algorytmu ewolucyjnego bez mutacji

Badanie zmiany prawdopodobieństwa krzyżowania

W tym zadaniu rozważę 4 przypadki dla których prawdopodobieństwo krzyżowania będzie różne. Dla każdego przypadku zrobię 5 testów, a następnie zachowam te wyniki do pliku. Otrzymane wyniki można zobaczyć na tabeli 4.

prawdopodo- bieństwo krzyżowan	prawdopodo- bieństwo mutacji	siła mutacji	liczba osobników w populacji	liczba iteracji	x1	y1	x2	y2	wartość f1(x1, y1)	wartość f2(x2, y2)	wartość
0.8	0.2	0.5	200	200	3.0002	1.9945	-0.0157	-0.0213	0.0005	0.0013	0.0018
0.8	0.2	0.5	200	200	3.587	-1.8553	-0.0056	-0.0096	0.001	0.0002	0.0012
0.8	0.2	0.5	200	200	3.5801	-1.8422	0.0182	0.0021	0.0013	0.0007	0.002
0.8	0.2	0.5	200	200	2.9944	2.008	-0.0176	0.0284	0.0014	0.0009	0.0023
0.8	0.2	0.5	200	200	3.0087	1.9849	-0.0092	-0.0117	0.004	0.0004	0.0045
0.85	0.2	0.5	200	200	2.9945	2.0137	-0.0081	0.0024	0.0028	0.0001	0.0029
0.85	0.2	0.5	200	200	3.0019	2.0009	-0.0257	0.0363	0.0002	0.0017	0.0019
0.85	0.2	0.5	200	200	3.0071	1.9967	-0.0148	-0.044	0.0016	0.003	0.0046
0.85	0.2	0.5	200	200	2.9975	2.009	-0.0204	0.0254	0.0012	0.001	0.0021
0.85	0.2	0.5	200	200	2.9945	1.9992	-0.0102	0.0244	0.0012	0.0006	0.0018
0.9	0.2	0.5	200	200	2.9927	2.0044	0.0301	-0.0361	0.0017	0.002	0.0037
0.9	0.2	0.5	200	200	2.998	1.9934	-0.0137	-0.003	0.0011	0.0004	0.0016
0.9	0.2	0.5	200	200	3.0092	1.9956	0.0039	-0.0403	0.0027	0.0015	0.0041
0.9	0.2	0.5	200	200	2.9987	1.9984	-0.0193	-0.0059	0.0001	0.0009	0.001
0.9	0.2	0.5	200	200	2.9987	1.994	0.0148	0.0245	0.0008	0.0014	0.0022
0.95	0.2	0.5	200	200	3.0047	1.9957	-0.0022	-0.0007	0.0007	0	0.0007
0.95	0.2	0.5	200	200	3.0041	1.993	-0.0133	0.0368	0.0009	0.0012	0.0021
0.95	0.2	0.5	200	200	2.9994	2.0055	-0.0031	0.0292	0.0005	0.0008	0.0012
0.95	0.2	0.5	200	200	3.5894	-1.8545	0.0267	-0.0009	0.0017	0.0014	0.0031
0.95	0.2	0.5	200	200	2.9987	2.0077	0.0124	0.0088	0.0009	0.0005	0.0014

Tabela 4. Badanie zmiany wartości prawdopodobieństwa krzyżowania

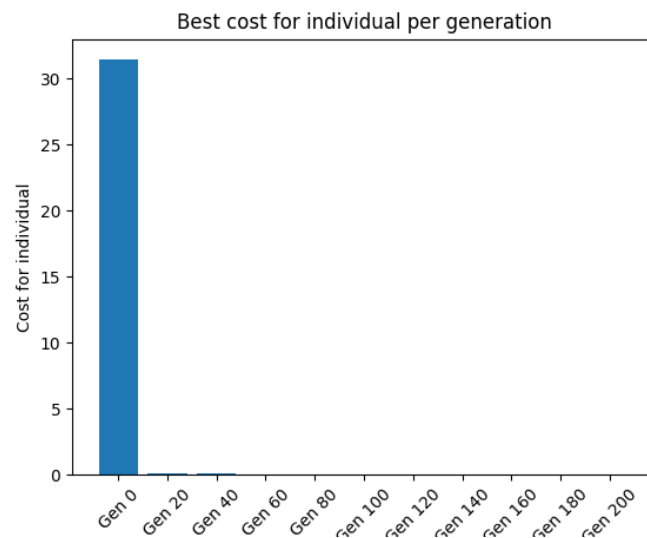
Na podstawie wyżej pokazanych danych można dojść do wniosku, że przy prawdopodobieństwie krzyżowania od 0.8 do 0.95 wartość znalezionej minimum jest bardzo podobna, jednak najlepsza jest dla wartości 0.95

Badania dotyczące poszczególnych iteracji populacji

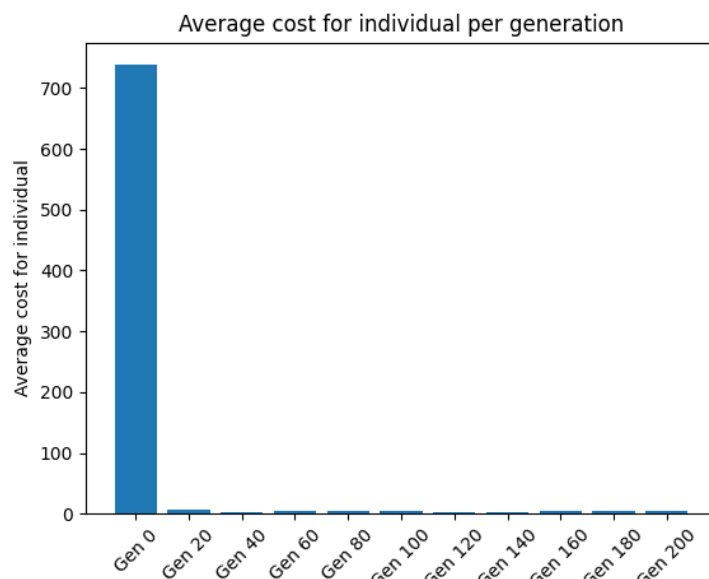
W tym punkcie będę używał najlepszych dotychczas znalezionych hiperparametrów:

- prawdopodobieństwo krzyżowania = 0.95
- prawdopodobieństwo mutacji = 0.2
- siła mutacji = 0.5
- liczba osobników w populacji = 200
- liczba iteracji = 200

Na wykresach słupkowych na rysunkach 16 pokaże wartość kosztu najlepszego osobnika w całej populacji w poszczególnych iteracjach, a także przeciętną wartość kosztu dla całej populacji na rysunku 17.

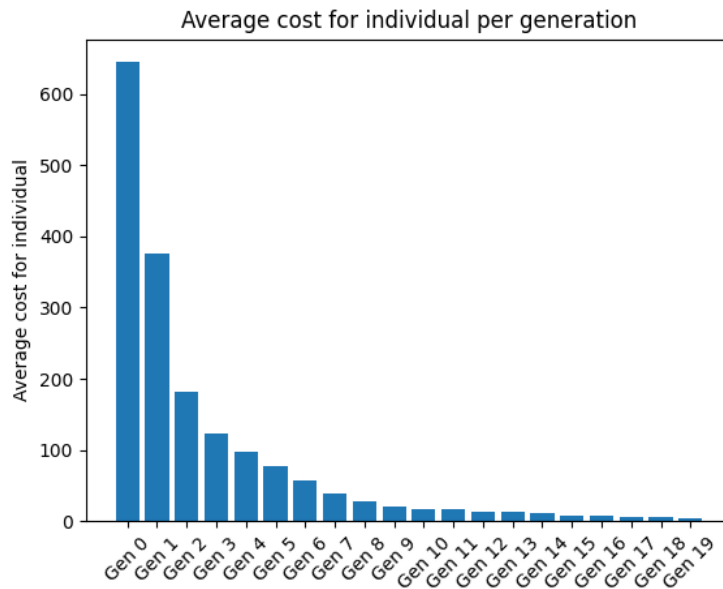


Rysunek 16. Wartości kosztu najlepszych osobników

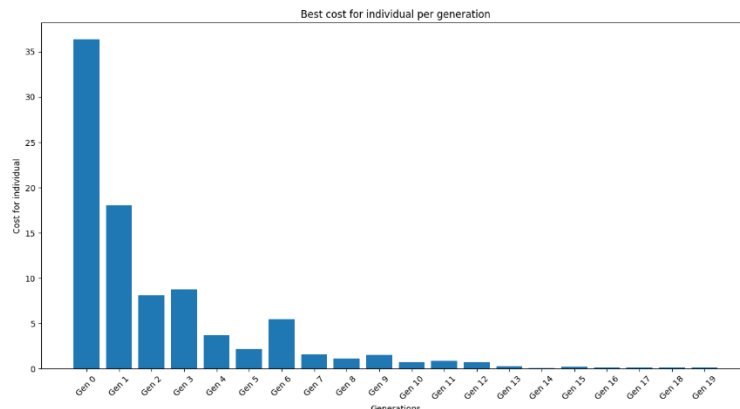


Rysunek 17. Średnie wartości kosztu populacji

Można zauważyć, iż nie ma dużej różnicy dla generacji populacji większej niż 20, więc uruchomię algorytm po raz kolejny, ale tym razem pokażę wartości kosztu najlepszego osobnika oraz średnią wartość kosztu dla poszczególnych iteracji populacji. Wynik tego procesu pokazano na rysunkach 18 oraz 19.



Rysunek 18. Średnie wartości kosztu populacji



Rysunek 19. Wartości kosztu najlepszych osobników

Z powyższych dwóch rysunków można zauważyć, iż średni koszt zmniejsza się bardzo szybko, nawet można powiedzieć potęgowo. wartość kosztu zazwyczaj też się zmniejsza, ale czasami w następnych iteracjach wartość może się zwiększyć, spowodowano to tym, że używamy sukcesji generacyjnej, więc nie zachowujemy najlepszego osobnika w populacji, jednak gdyby w implementacji danego algorytmu

użyto sukcesji elitarniej, to kształt słupków byłby podobny do tego, który jest na rysunku 18 (czyli każda następna wartość byłaby nie mniejszą od poprzedniej).