WSI ĆWICZENIE 2

Michał Mokrzycki

NOVEMBER 2023

1 Wstęp

Ćwiczenie polegało na implementacji algorytmu ewolucyjnego z

- **selekcją turniejową** (k=2) : czyli wybieranie z populacji dwóch osobników i porównywanie wartości ich funkcji, wygrywający osobnik jest umieszczany w wynikowej populacji
- **krzyżowaniem jednopunktowym** : losujemy punkt przecięcia chromosomu rodziców i przepisujemy początkową część chromosomu jednego rodzica i końcową drugiego do jednego potomka, a drugi potomek powstaje z pozostałego materiału genetycznego
- **mutacja gaussowską** : czyli zaburzanie współrzędnych osobnika rozkładem normalnym przeskalowanym przez siłę mutacji
- **sukcesją generacyjną** : czyli utworzenie nowej populacji z populacji mutantów

Oraz wykorzystanie tej implementacji do znalezienia minimum dla sumy funkcji $f1$ i $f2$ zdefiniowanych następująco:

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

$$f_2(x_2, y_2) = 2x_2^2 + 1.05x_2^4 + \frac{x_2^6}{6} + x_2y_2 + y_2^2$$

. Dla dziedziny

$$D = [-5, 5]x[-5, 5]$$

Celem ćwiczenia było też zbadanie wpływu hiperparametrów i początkowego rozkładu populacji na współrzędne znalezionych minimumów dla funkcji f1.

2 Próby dopasowania najlepszych parametrów

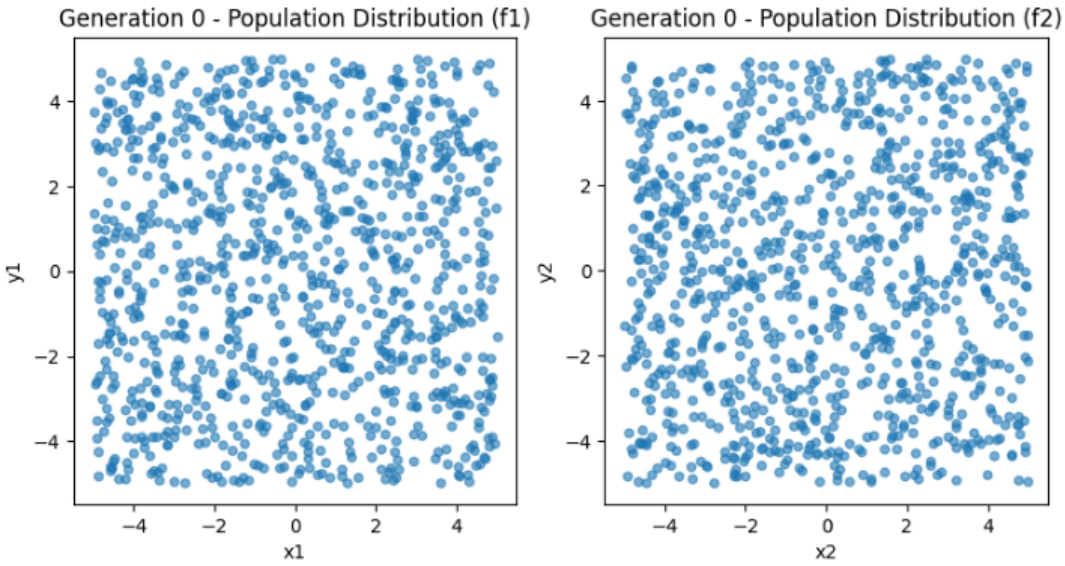
x1	y1	x2	y2	f1(x1,y1)	f2(x2,y2)	f1+f2	size	it	rate	strength	c
-2.8051181	3.1313112	-7.87032e-10	-6.1125e-07	2.61909e-09	2.3441e-11	2.64021e-09	1000	101	0.01	0.01	0
2.999998	1.99999	1.3134e-06	-3.4999e-08	4.40993e-09	1.359679e-10	5.76961e-09	500	201	0.01	0.01	0
-2.805	3.131276	-0.000246	-0.00393	5.18284e-11	2.217e-12	6.0347e-11	500	201	0.01	0.01	0
3.00003	2.000142	-1.0003e-07	-1.2902e-07	1.6003e-13	5.4707e-14	2.14057e-13	1000	101	0.01	0.01	0
2.99999	1.999999	1.161e-07	-1.2902e-07	2.72336e-11	1.3651e-11	4.0885e-11	1000	101	0.05	0.01	0
3.5844	-1.848	1.0158e-06	-2.5988e-06	1.397e-09	1.085e-09	2.482679e-09	100	101	0.05	0.01	0

$x1, y1$ -> znalezione minima funkcji $f1$
$x2, y2$ -> znalezione minima funkcji $f2$
$f1(x1, y1)$ -> wartość funkcji f1 dla x1 i y1
$f2(x2, y2)$ -> wartość funkcji f2 dla x2 i y2
$f1 + f2$ -> minimalna wartość sumy funkcji
size -> rozmiar populacji
it -> liczba iteracji do zatrzymania
rate -> parametr częstotliwości mutacji
strength -> siła mutacji
cross -> parametr częstotliwości krzyżowania

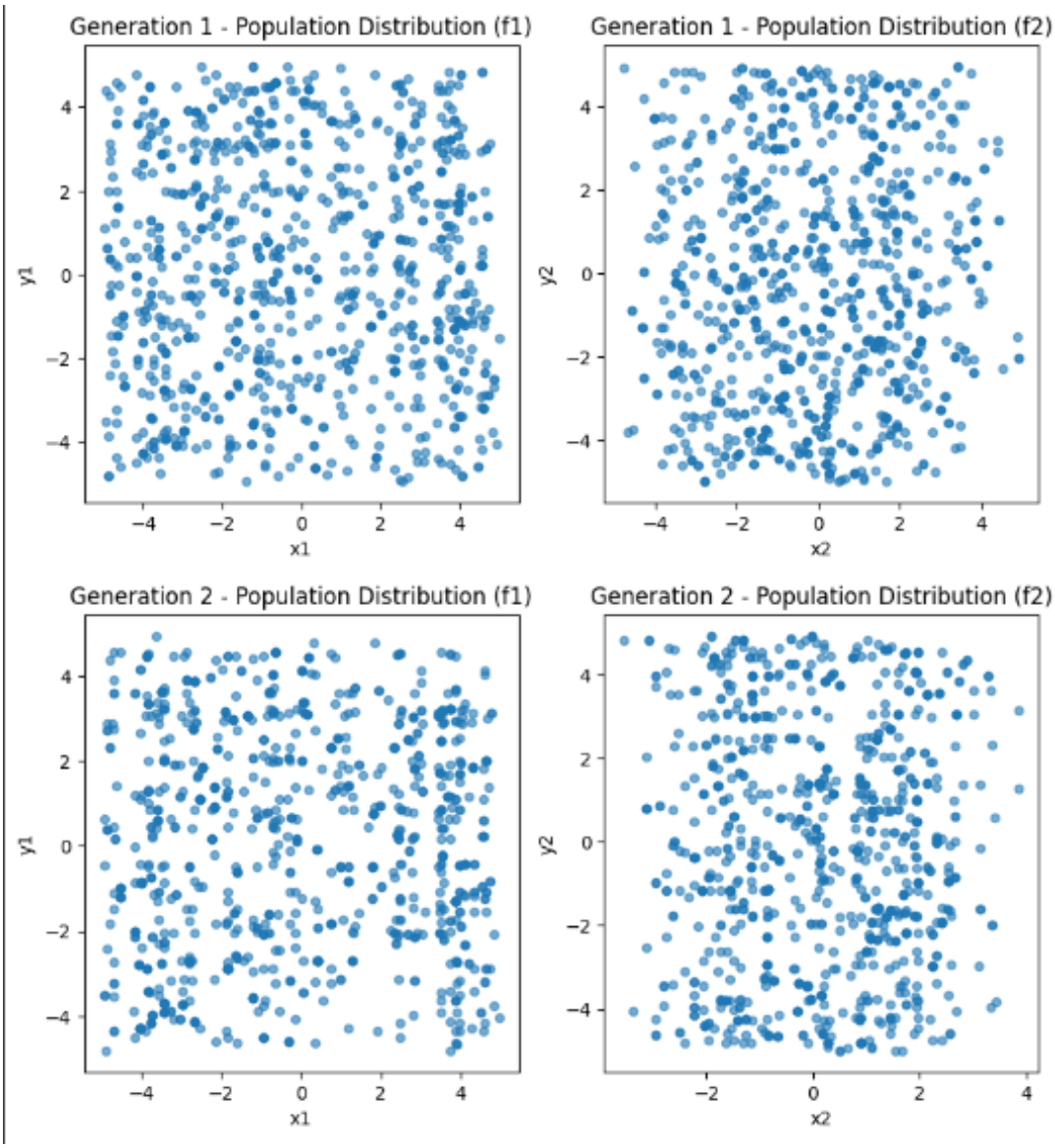
Przeprowadzono kilka eksperymentów z różnymi parametrami tak aby otrzymać najlepszy wynik. Punkty x1,x2,y1,y2 to **najlepiej** dopasowane punkty znalezione w naszym przypadku po 50 próbach. Wartości funkcji dla minimów to **średnie** wyniki dla danej liczby eksperymentów. Jak widać w załączonej tabeli dla każdych parametrów wyniki są bardzo blisko minimum globalnych danych funkcji.Na podstawie danych powyższych prób można jedynie stwierdzić, że znacznie zmniejszenie populacji wpłynęło negatywnie na dokładność wyniku. Przy badaniu wpływu pozostałych parametrów będziemy przyjmować rozmiar populacji: 1000 przy 101 iteracjach.

3 Wizualizacja zmiany populacji w każdej iteracji

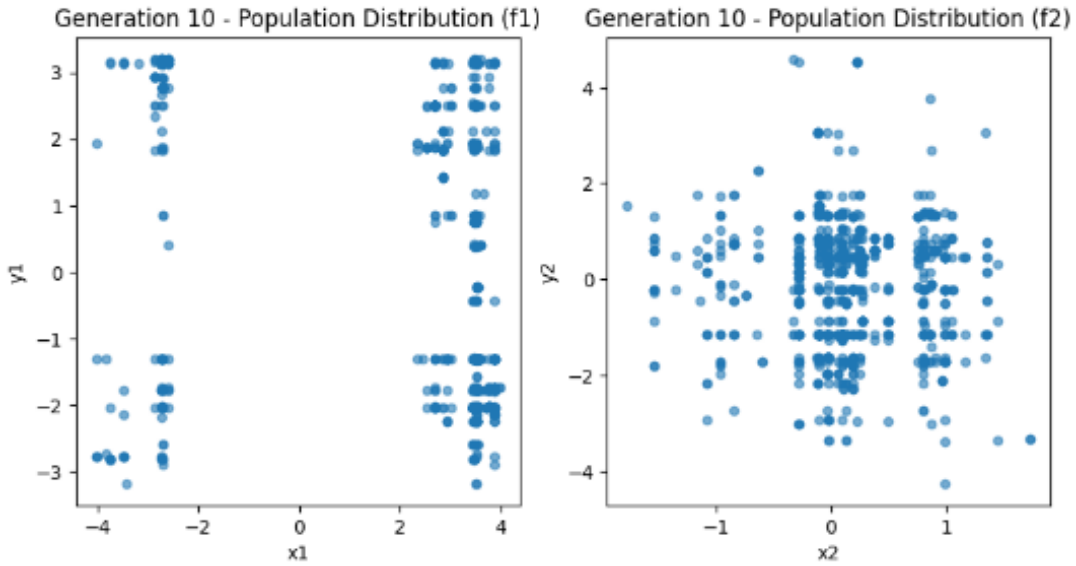
Początkowy, jednorodny rozkład populacji na obszarze prezentuje się następująco dla funkcji f1 i f2:



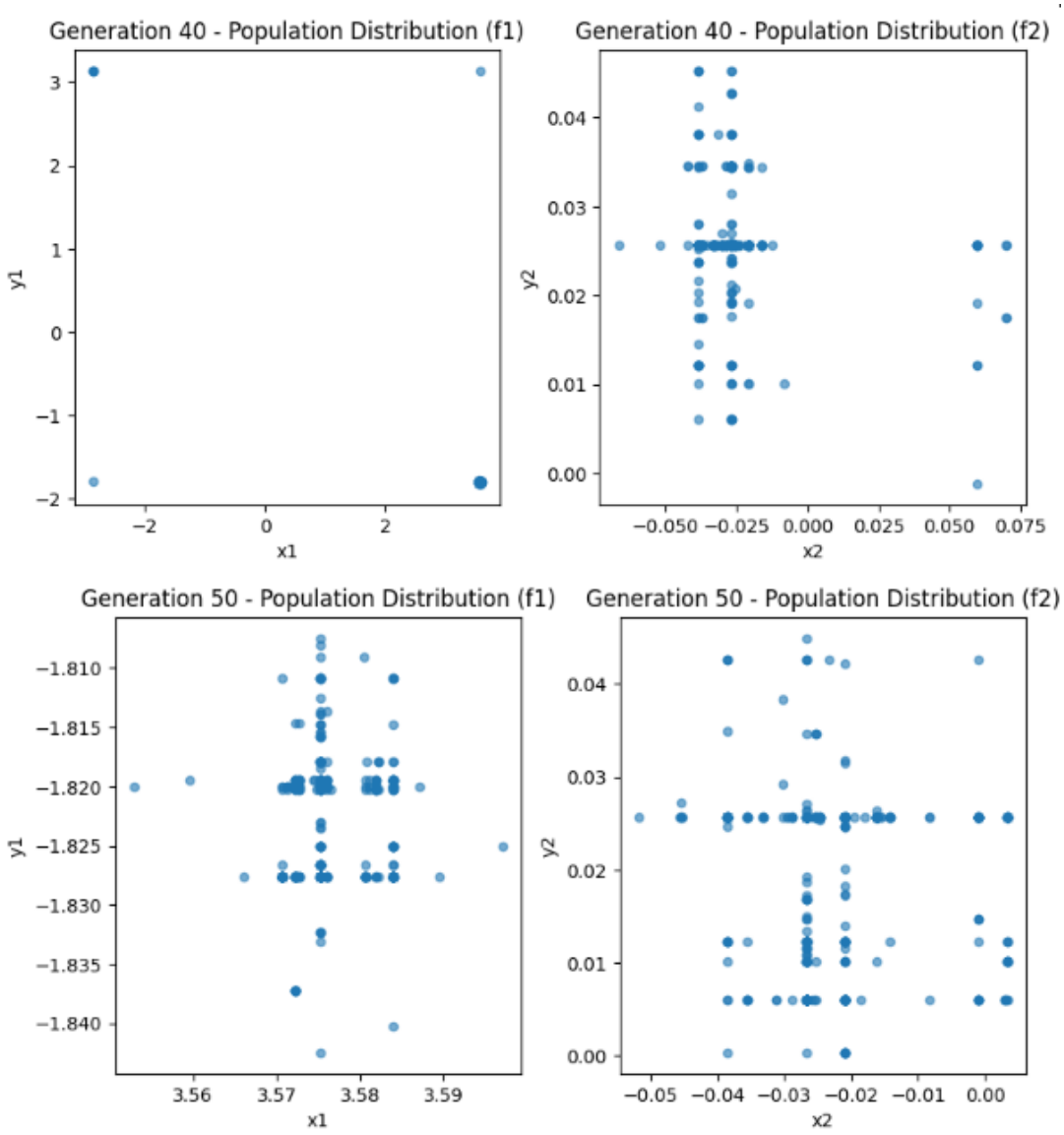
Po wykonaniu wszystkich czterech części algorytmu(selekcja,krzyżowanie,mutacja i sukcesja), po pierwszej oraz drugiej iteracji, populacja zmienia się następująco:



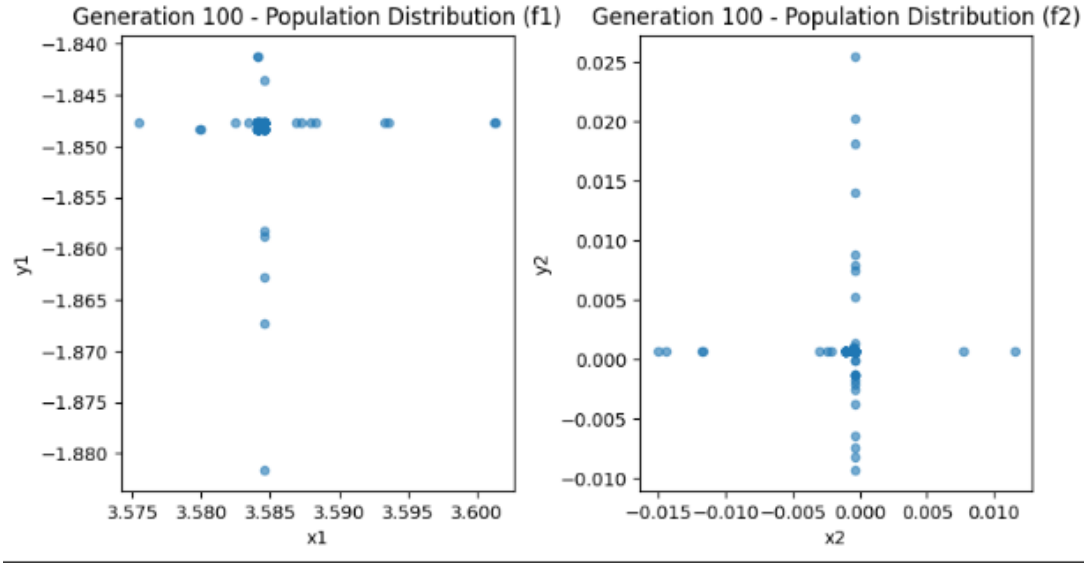
Dla przykładu dziesiąta generacja populacji:



Znaczącą różnice w dystrybucji obserwujemy później dopiero między przejściem z 40 do 50 generacji.



A po 100 iteracjach otrzymujemy następujący rozkład osobników populacji:



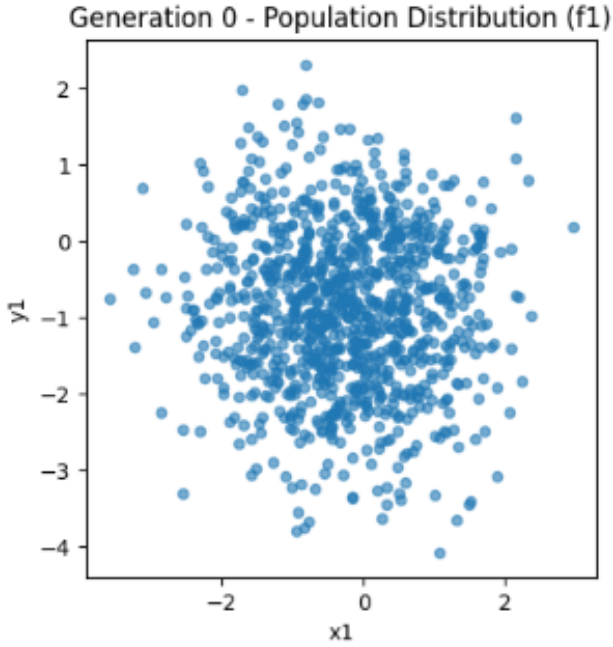
4 Badanie wpływu hiperparametrów

4.1 Dystrybucji populacji początkowej

Powyżej przedstawiona jest dystrybucja populacji początkowej rozkładem jednorodnym po całym obszarze, zbadamy teraz wpływ dystrybucji na wynik znalezionych minimów funkcji f1. Założymy rozkład normalny wokół

$$[x_1 = -0.3, y_1 = -0.9]$$

, a dla funkcji f2 jednorodny po całym obszarze. Wizualizacja przykładowej losowej dystrybucji wygląda następująco:



Wyniki dla paru prób prezentują się następująco:

x1	y1	f(x1,y1)	size	it	rate	strength
3.0001	2.00006	5.376e-07	1000	101	0.01	0.01
-3.6262	-3.16417	1.4072	1000	101	0.01	0.01
-2.80507	3.1316	3.5899	1000	101	0.01	0.01
2.20909	2.0487	16.926899	500	101	0.01	0.01
1.85886	1.795078	36.7507	100	101	0.01	0.01
2.97616	2.0193	0.01805	1000	101	0.01	0.01
2.767	2.15314	1.578782	1000	101	0.01	0.01
3.5844	-1.84812	4.1825e-08	5000	101	0.01	0.01
2.99993	2.000001	1.52689e-07	5000	101	0.01	0.01

Można z tego wywnioskować, że wyniki dla małych wielkości populacji są kompletnie niemiarodajne. Dla populacji rzędu 1000, w niektórych przypadkach udało się uzyskać miarodajne minima. Dla dużych populacji większość wyników była poprawna (podobna do tych gdy rozkład populacji był jednorodny). Błąd wyniku dla małych populacji może wiązać się z tym, że zbyt dużo osobników było zlokalizowanych w jednym punkcie, w związku z czym istnieje ryzyko, że algorytm utknie w minimum lokalnym. Możliwe jest

też, że w przypadku braku punktów w pobliżu minimum globalnych, algorytm może uniemożliwić eksplorowanie obszarów w pobliżu tych punktów.

4.2 Prawdopodobieństwa mutacji

x1	y1	f(x1,y1)	rate	size	strength	cross
3.58445	-1.84809	6.82213e-08	0.1	1000	0.01	0.85
2.9997	1.99998	3.74724e-06	0.2	1000	0.01	0.85
2.9998	2.00101	1.52106e-05	0.3	1000	0.01	0.85
2.99935	2.0011	2.2716e-05	0.5	1000	0.01	0.85
3.58546	-1.8458	0.000153	1	1000	0.01	0.85
3.59172	-1.87462	0.011925	1	1000	0.1	0.85
3.54771	-1.905447	0.133709	1	1000	0.5	0.85
3.02062	1.970761	0.018123	0.5	1000	0.5	0.85
2.99969	2.000351	3.3666e-06	0.1	1000	0.5	0.85
2.9969	1.9698	0.017489	0	1000	0.1	0.85

Z przeprowadzonych eksperymentów w tabeli można wywnioskować, że gdy siła mutacji nie jest duża, wzrost prawdopodobieństwa mutacji tylko nieznacznie obniża dokładność wyniku. Jednak gdy kompletnie zrezygnujemy z mutacji, wynik nie będzie dokładny. Gdy siła mutacji jest większa, wzorst prawdopodobieństwa mutacji już znacząco (negatywnie) wpływa na dokładność wyniku. Najbardziej optymalne wartości tych parametrów to prawdopodobieństwo: 0.1 i siła: 0.01

4.3 Prawdopodobieństwa krzyżowania

x1	y1	f(x1, y1)	cross	size
3.58445	-1.8481	4.4904e-08	0.8	1000
-2.8051	3.1313	3.42794e-08	0.95	1000
3.00009	1.99987	3.7733e-07	0.95	500
3.000003	1.999994	6.02159e-08	0.8	500
3.001812	1.9956876	0.00028	0.95	100
2.978481	2.07696	0.08809	0.8	100

Z danych eksperymentów wynika, że dla takiego przedziału prawdopodobieństwo krzyżowania nie wpływa znacząco na wynik, jednakże

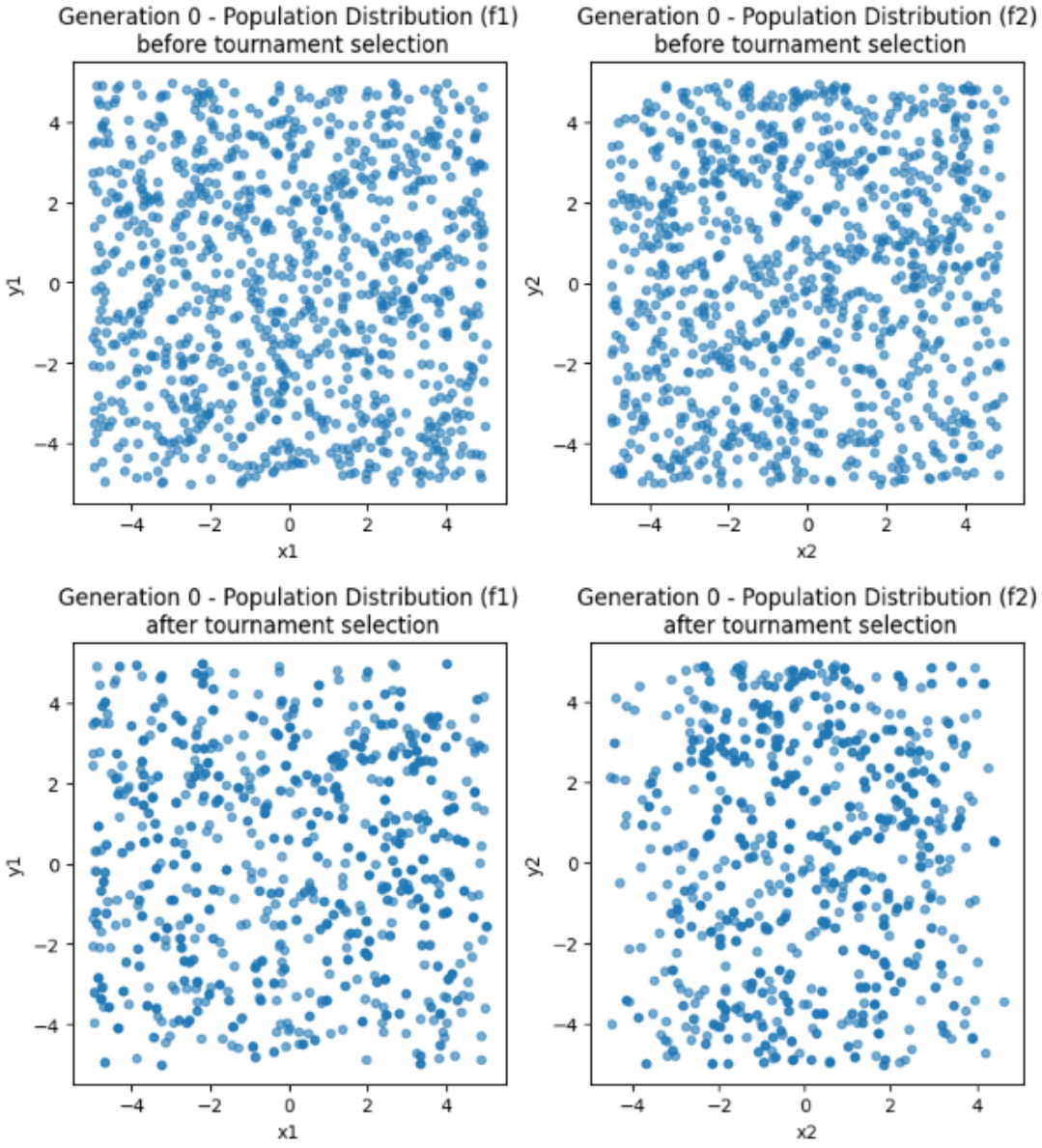
5 Poszczególne etapy optymalizacji

W tym paragrafie przedstawimy wizualizacje poszczególnych etapów optymalizacji z podziałem na odpowiednie funkcje. Dzięki czemu, będziemy mogli zaobserwować czy i jak bardzo dana funkcja wpływa na zbieganie ogółu populacji do minimuów globalnych.

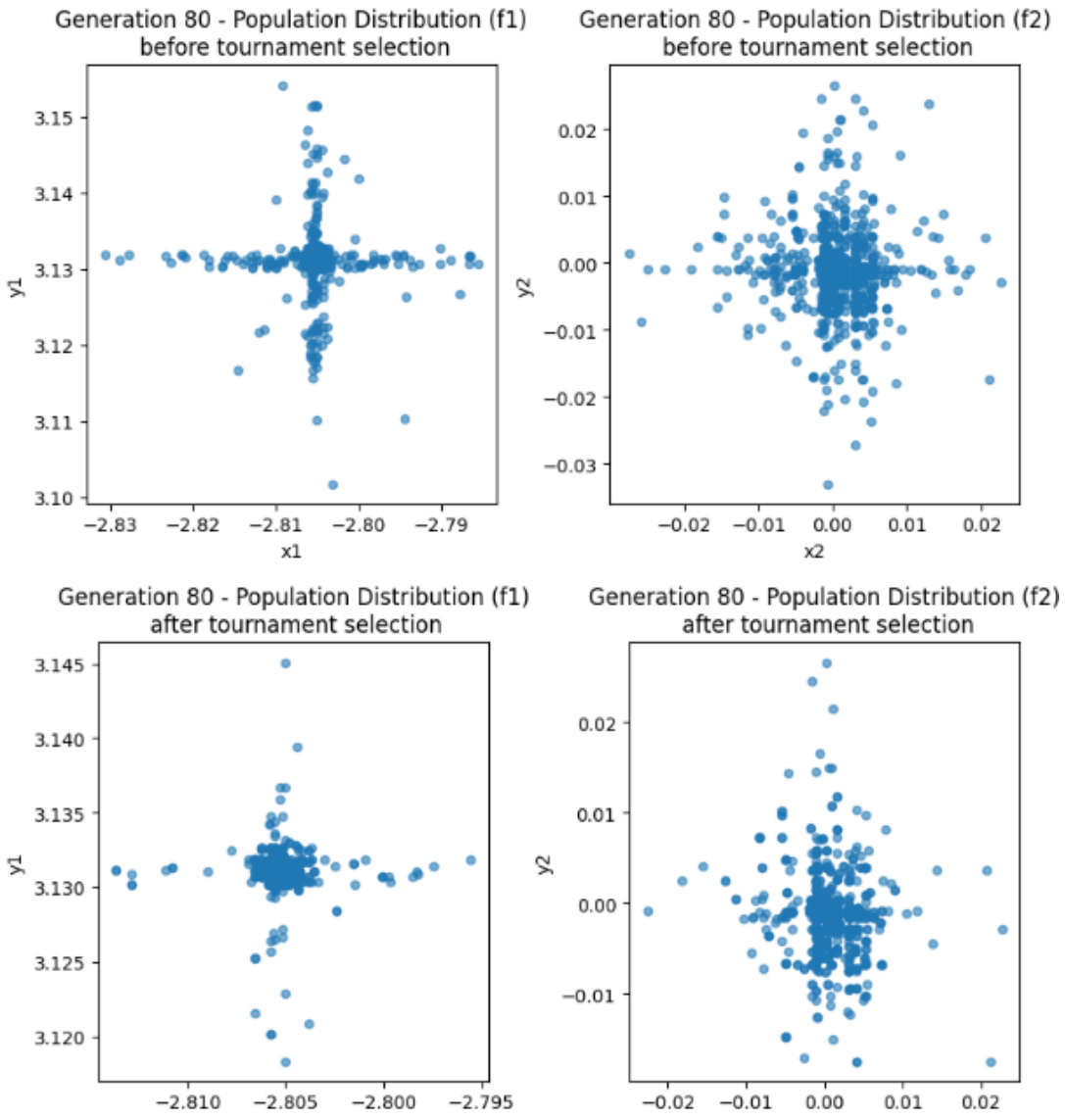
5.1 Wpływ selekcji turniejowej

Na wykresach można zauważyć wpływ selekcji turniejowej na zagęszczenie populacji przy minimach globalnych.

5.1.1 Dla początkowych generacji



5.1.2 Dla późniejszych generacji

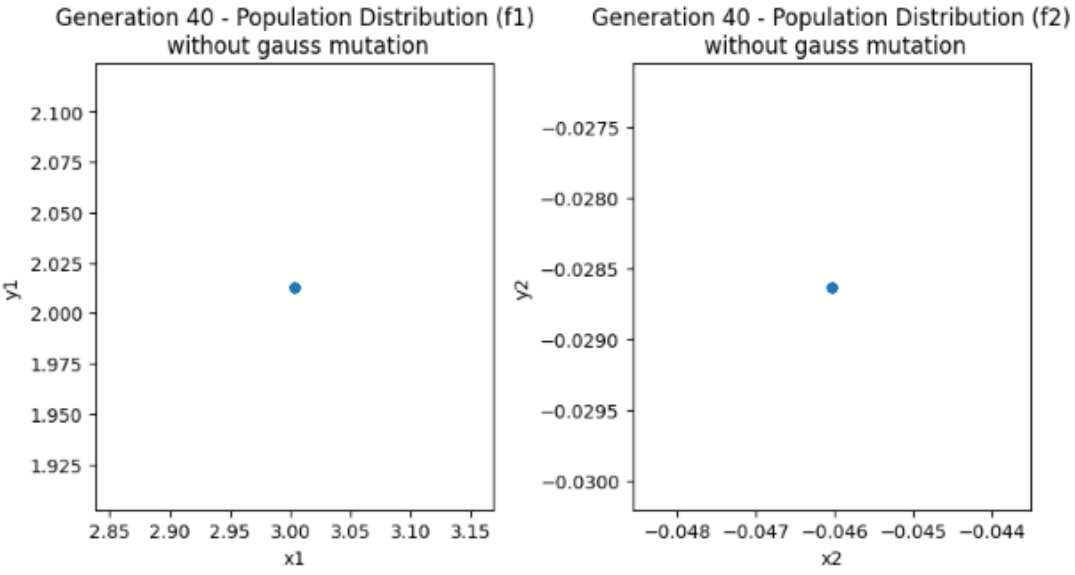


5.1.3 Wnioski

Widzimy, że selekcja turniejowa "przerzedza"naszą populację. Dla początkowych iteracji może wydawać się, że eliminacja pojedynczych osobników jest losowa, jednak dla wyższych iteracji znacząco widać, że wygrywają te osobniki którym bliżej do minimum globalnego. Wnioski są takie, że implementacja tej funkcji znacząco zwiększa działanie algorytmu, a dla odpowiednio dużych populacji szansa że do turnieju nie zostanie wylosowany "dobry"osobnik jest znikoma.

5.2 Wpływ mutacji gaussowskiej

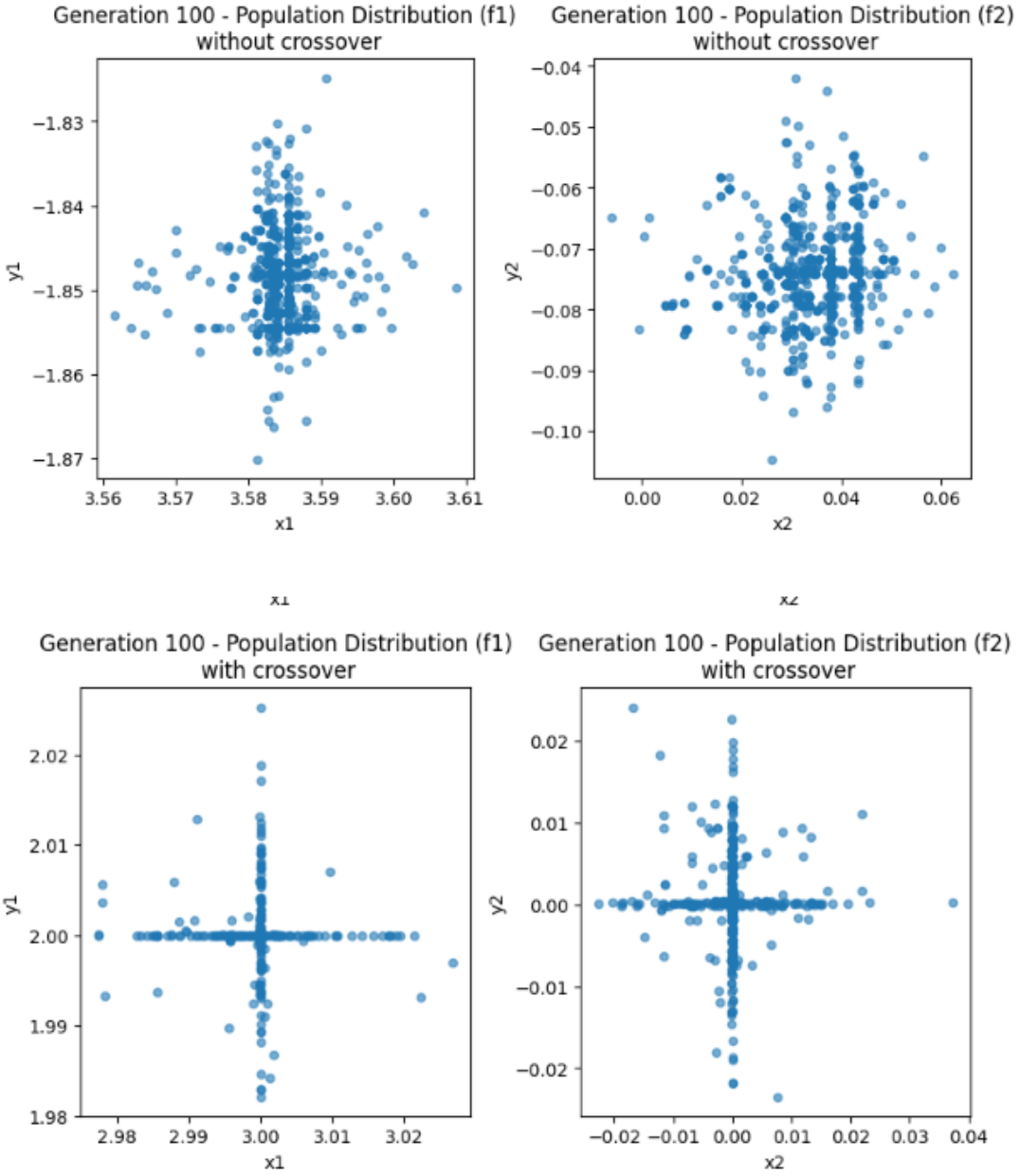
Tutaj łatwiej będzie przedstawić wpływ braku implementacji tej funkcji na wynik.



Po kilkunastu/kilkudziesięciu iteracjach algorytm "gubi"najlepsze rozwiązanie, co nie zdarza się gdy podczas mutacji generujemy punkty z otoczenia punktu bieżącego. Zbyt częsta, lub silna mutacja może także pogorszyć wynik, ale jej całkowity brak doprowadza w naszym wypadku do wyników rzędu $< 0.001; 0.01$ co jest znacznie mniej dokładne niż w poprzednich eksperymentach.

5.3 Wpływ krzyżowania jednopunktowego

Krzyżowanie jednopunktowe może przynosić zarówno plusy jak i minusy. W naszej implementacji, jego brak lub bardzo niska wartość zwykle powodowały obniżenie dokładności algorytmu. Jednak w niektórych przypadkach, może też powodować wydłużoną zbieżność czy utratę pewnych cech osobników. Zwykle jednak, odpowiada za zwiększenie genetyczne w populacji co jest pożądane. Poniżej przedstawione zostały różnice w zbieżności algorytmu:



Gołym okiem widoczne jest jak przy wzroście prawdopodobieństwa krzyżowania, populacja układa się wzdłuż prostych równoległych do osi OX i OY i przecinających się w minimum globalnym. Jest to ciekawe zjawisko, aczkolwiek prowadzi do dokładniejszego znalezienia minima.

5.4 Wpływ sukcesji generacyjnej

Nie będziemy wizualizować wpływu sukcesji generacyjnej na populacje, bo w naszej implementacji jest to po prostu zamienienie obecnej populacji z populacją mutantów utworzoną przez poprzednie funkcje. Dla naszego algorytmu spełnia ona swoje zadanie i pozwala na znalezienie dość dokładnego minima, jednak można by zastanowić się nad implementacja sukcesji elitarnej np.

```
# elite succession
def generational_succession(population, offspring):
    new_population = sorted(population + offspring, key=lambda ind: f1(ind[0], ind[1]) + f2(ind[2]
    return new_population[:population_size]
```

Która do obecnej populacji dodawała by populacje mutantów i dopiero wtedy tworzyła nową. Mogło by to poprawić wyniki, ale przy równoczesnym zwiększeniu pracy algorytmu.