Ćwiczenie 2 WSI 23Z – Algorytmy ewolucyjne
Dawid Bartosiak 318361

1. Wstęp

Zadaniem, które miałem wykonać było zaimplementowanie algorytmu ewolucyjnego do problemu komiwojażera. Z racji ilości miast (n=50) nie było możliwe sprawdzanie *bruteforce* wyników które otrzymałem, zatem konieczne było wybranie seeda i rozpoczęcie od niższych wartości w celu oszacowania wyniku samemu.

2. Implementacja

W celu implementacji skorzystałem z wzoru pseudo-kodu zamieszczonego przez prowadzącego. Aby otrzymać jak najlepsze wyniki wprowadziłem także poprawkę zmieniającą wynik tylko w przypadku otrzymania lepszej drogi (nie ma to znaczenia w kontekście kolejnych ewolucji bo one dzieją się w sposób klasyczny, ale wynik końcowy jest najlepszy z tych przez które przeszedł algorytm). Zdecydowałem się na ten ruch z uwagi na ogromne różnice w wynikach które zaobserwowałem (także je przedstawię w celu uzasadnienia). Przygotowałem parametry którymi możemy sterować takie jak:

- Sposób rozmieszczenia miast
- Rozmiar populacji
- Prawdopodobieństwo mutacji
- Rozmiar turnieju
- Ilość generacji

Dodatkowo zaimplementowałem rysowanie przebiegu trasy czego nie mam jak zamieścić w sprawozdaniu (tzn. Animacja zgodnie z przebiegiem wyznaczonym przez algorytm).

Mutacja odbywa się na zasadzie:

- Sprawdź czy mutacja się odbędzie
- Wylosuj dowolną ilość zamian miast (0 – (ilość miast - 1)) (ale tutaj okazało się to rozwiązaniem nieoptymalnym)
- Zamieniaj losowe miasta między sobą

Przy implementacji w celu szybszego testowania użyłem biblioteki do obliczeń wielowątkowych i skorzystałem z liczenia równolegle 12-stu wyników pojedynczego wywołania i w taki sposób będę prezentować wyniki (wielokrotności 12 [ilość wątków procesora]).

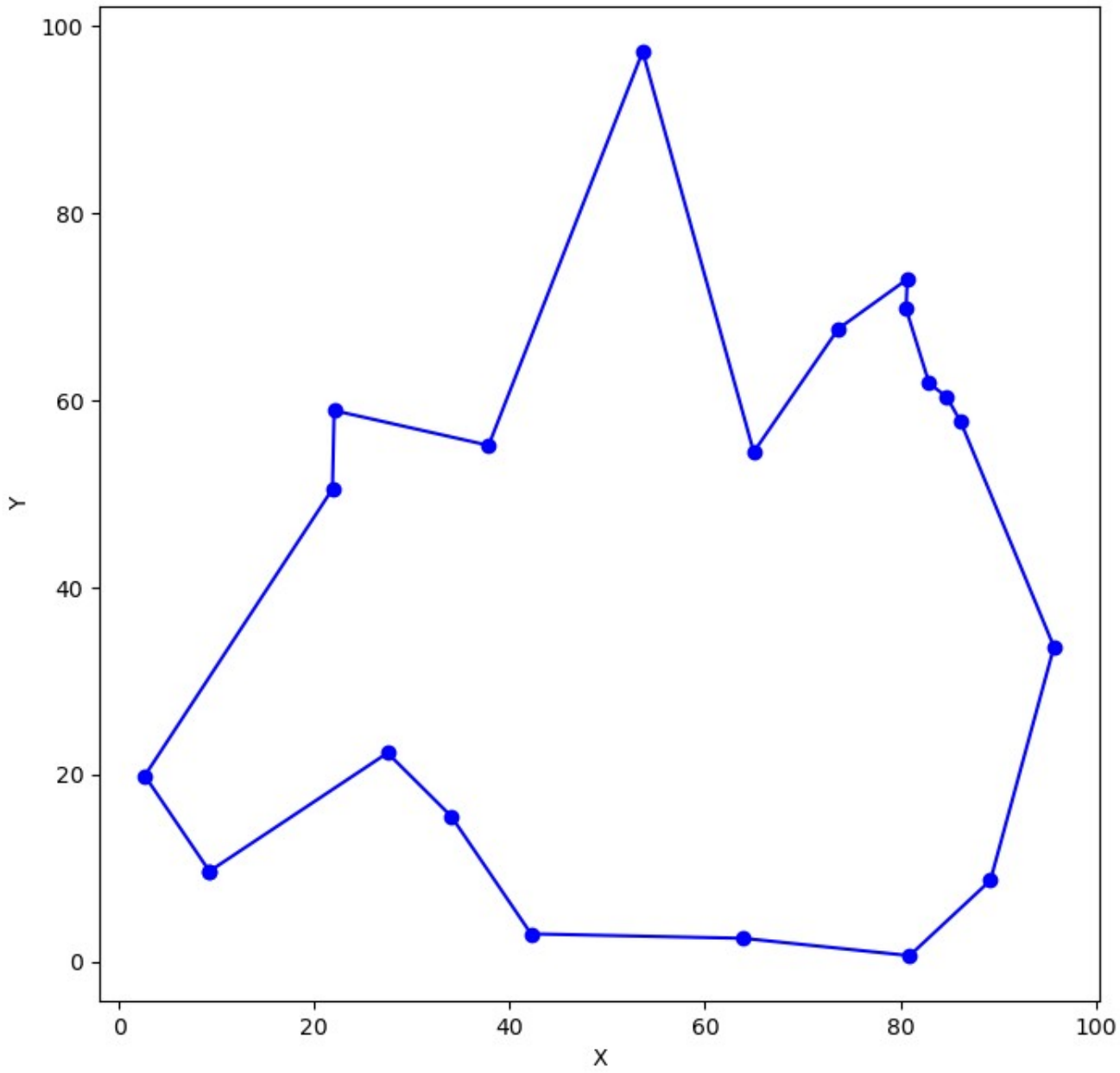
Mojego seeda w przypadku randa używałem tylko w przypadku generacji miast, nie uznaję za odpowiednie generowanie wartości pseudo-losowych algorytmu korzystając zawsze z tego samego seeda.

3. Testy

W celu liczenia w sensownym czasie, jedynie ostatnie porównania (gdy będzie pewność jaka jest zależność od poszczególnych parametrów) będę przedstawiał dla wartości 50 – oszczędzi to kilkukrotnie czas potrzebny na oczekiwanie na wynik, a nie ma większego znaczenia dla poprawności implementacji.

1) N = 20 miast rozłożone losowo, seed=42, ilość prób = 24

Najlepszy możliwy wynik jaki można otrzymać to dystans około 352,318. Prezentuje się on następująco:



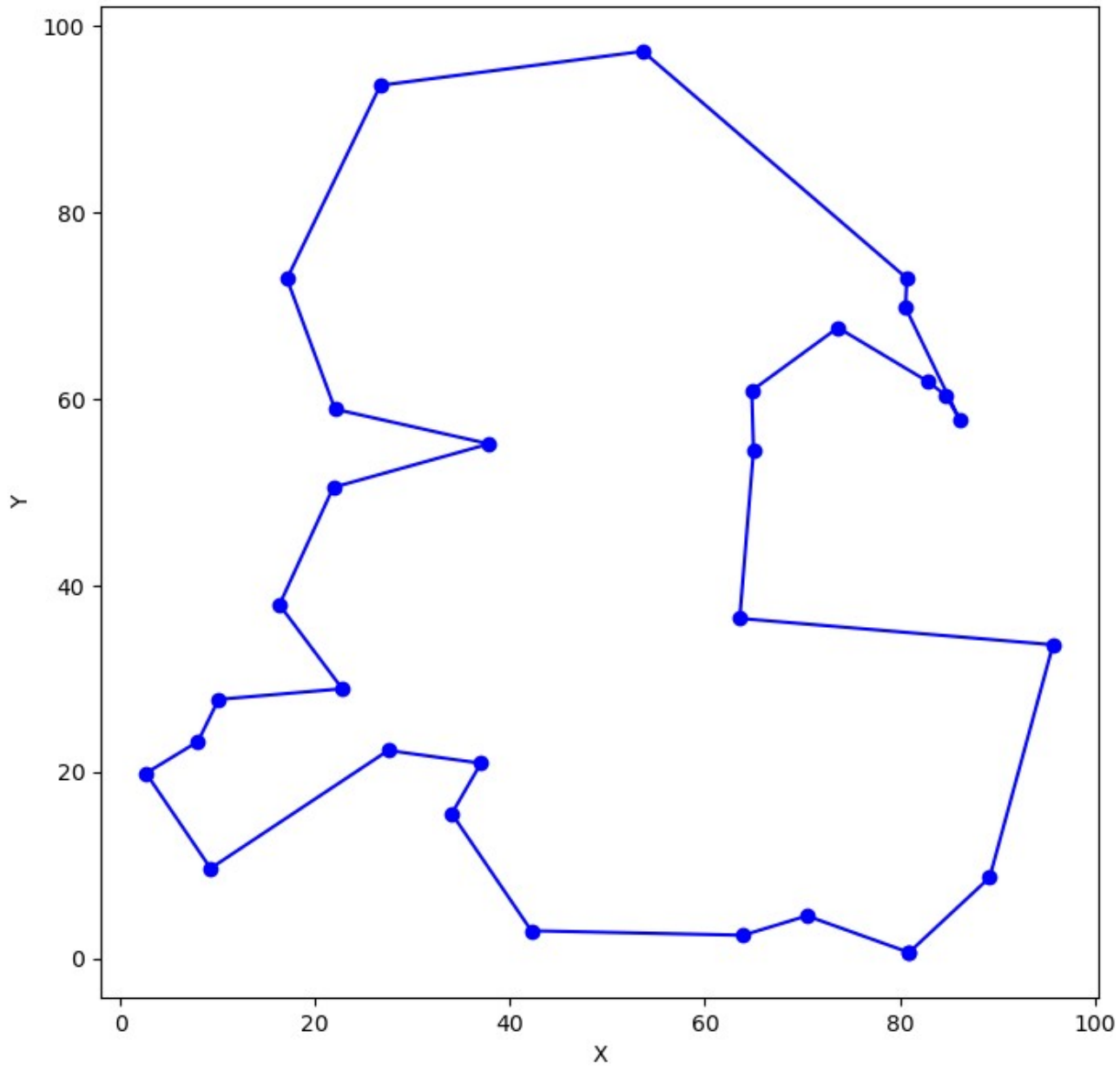
Ilość mutacji	Rozmiar Populacji	Szansa mutacji	Rozmiar turnieju	Ilość generacji	Najlepsza trasa	Średnia trasa	Odchyl. Standard.	Czas(s)
rand	N	0,1	2	N	713,80	812,02	48,52	0,066
rand	10N	0,1	2	N	614,72	691,52	36,98	0,414
rand	N	0,1	2	10N	600,28	675,75	36,74	0,379
rand	N	0,2	2	N	697,58	791,98	41,34	0,066
rand	50N	0,1	2	N	548,18	632,55	34,74	1,870
rand	N	0,1	2	50N	486,18	544,24	29,55	2,030
rand	N	0,4	2	N	702,21	777,86	41,02	0,066
rand	N	0,8	2	N	750,42	833,33	48,52	0,070
rand	10N	0,1	2	10N	396,32	520,93	48,63	3,615
rand	10N	0,2	2	10N	431,06	498,99	28,66	3,907
rand	10N	0,4	2	10N	450,87	516,62	38,52	4,232
rand	50N	0,1	2	50N	352,32	402,62	46,69	121,905
rand	50N	0,2	2	50N	352,32	389,12	43,44	135,290
rand	30N	0,1	2	20N	352,32	417,50	43,77	29,545
rand	30N	0,2	2	20N	352,32	416,18	43,66	26,826
2	10N	0,1	2	10N	352,32	442,13	39,61	3,650
2	10N	0,2	2	10N	352,32	420,05	44,89	3,522
2	10N	0,4	2	10N	352,32	408,63	46,72	3,667
2	50N	0,1	2	50N	352,32	412,98	49,87	117,309
2	50N	0,2	2	50N	352,32	378,92	36,51	118,738
2	50N	0,4	2	50N	352,32	363,12	26,32	131,769
2	30N	0,1	2	20N	352,32	402,39	45,53	24,373
2	30N	0,2	2	20N	352,32	396,67	53,68	25,093
2	30N	0,4	2	20N	352,32	371,59	36,82	23,328
Zmodyfikowany algorytm – bierzemy najlepsze wyniki dla najlepszych z powyższych								
rand	50N	0,2	2	50N	352,32	387,01	43,20	175,895
rand	30N	0,2	2	20N	352,32	413,89	41,14	33,365
2	10N	0,4	2	10N	352,32	396,87	40,56	4,463
2	50N	0,4	2	50N	352,32	363,08	26,28	170,02

2	30N	0,4	2	20N	352,32	373,22	36,39	36,40

Wnioski są już wstępnie widoczne, ale próba dalej jest zbyt mała, dlatego przed przedstawieniem ich przetestuję najpierw kolejną (większą) ilość miast, co pozwoli mi na dokładne wnioski, mniej obarczone błędem statystycznym.

2) N = 30 miast rozłożone losowo, seed=42, ilość prób = 24

Najlepszy możliwy wynik jaki można otrzymać to dystans około 425,98. Prezentuje się on następująco:

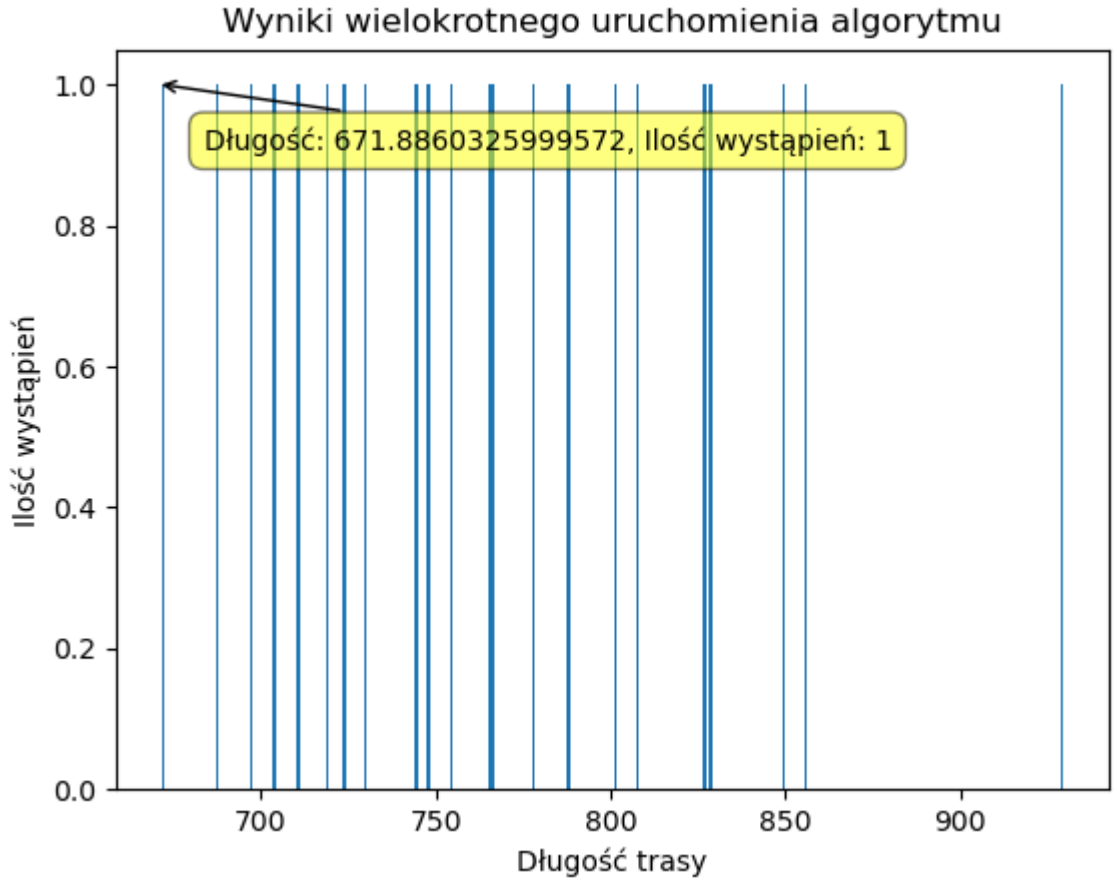


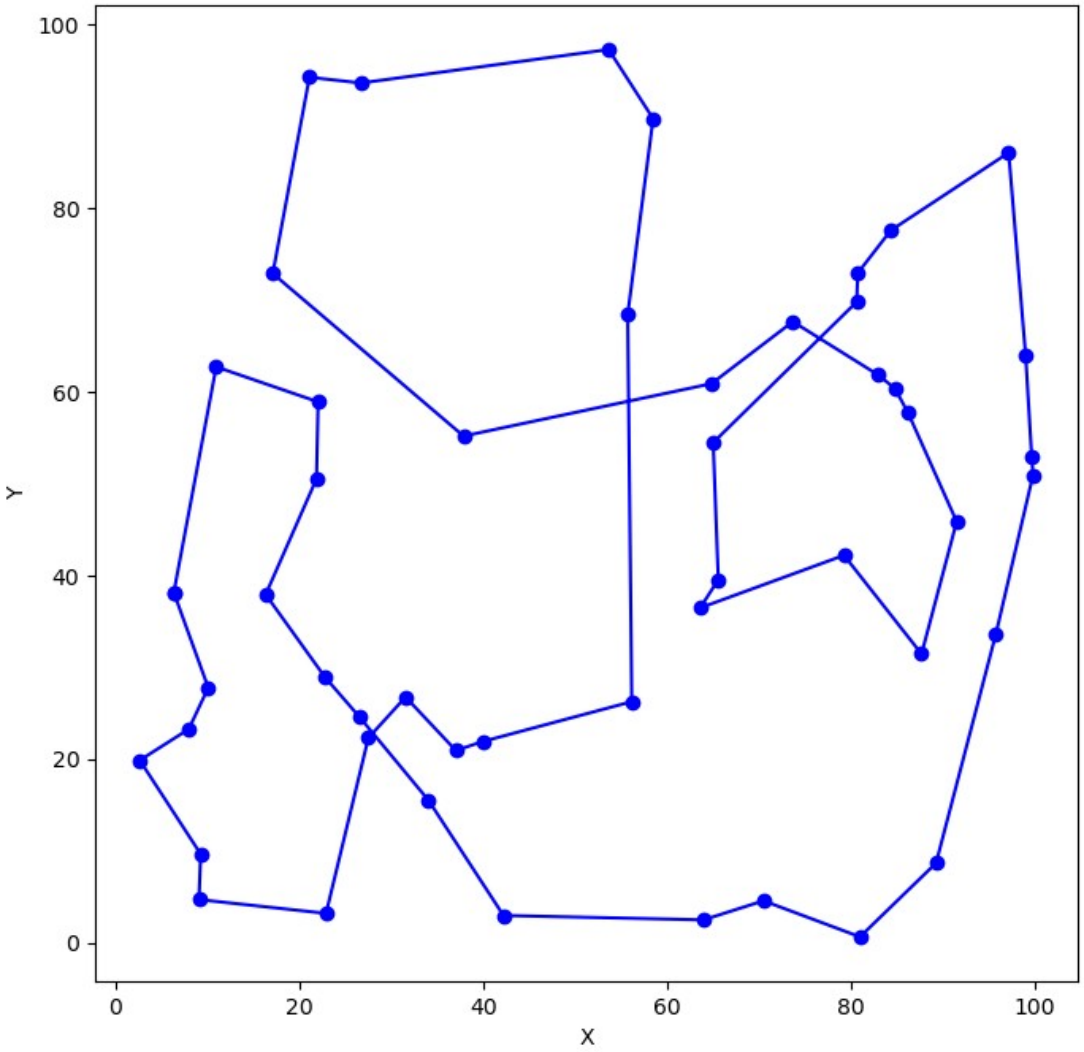
Ilość mutacji	Rozmiar Populacji	Szansa mutacji	Rozmiar turnieju	Ilość generacji	Najlepsza trasa	Średnia trasa	Odchyl. Standard.	Czas(s)
rand	30N	0.2	2	20N	470,49	564,50	58,12	80,200
rand	50N	0.2	2	50N	446,70	525,43	45,63	399,526
2	10N	0.4	2	10N	432,93	530,10	43,09	11,225
2	30N	0.4	2	20N	433,34	482,12	29,35	71,677
2	50N	0.4	2	50N	426,24	478,13	29,65	336,148
2	50N	0.4	4	50N	426,24	494,12	33,80	1192,430
Zmodyfikowany algorytm – bierzemy najlepsze wyniki dla najlepszych z powyższych								
2	30N	0.4	2	20N	425,98	487,62	33,35	104,266
2	50N	0.4	2	50N	433,34	487,17	28,24	404,010

3) N = 50 miast rozłożone losowo, seed=42, ilość prób = 12

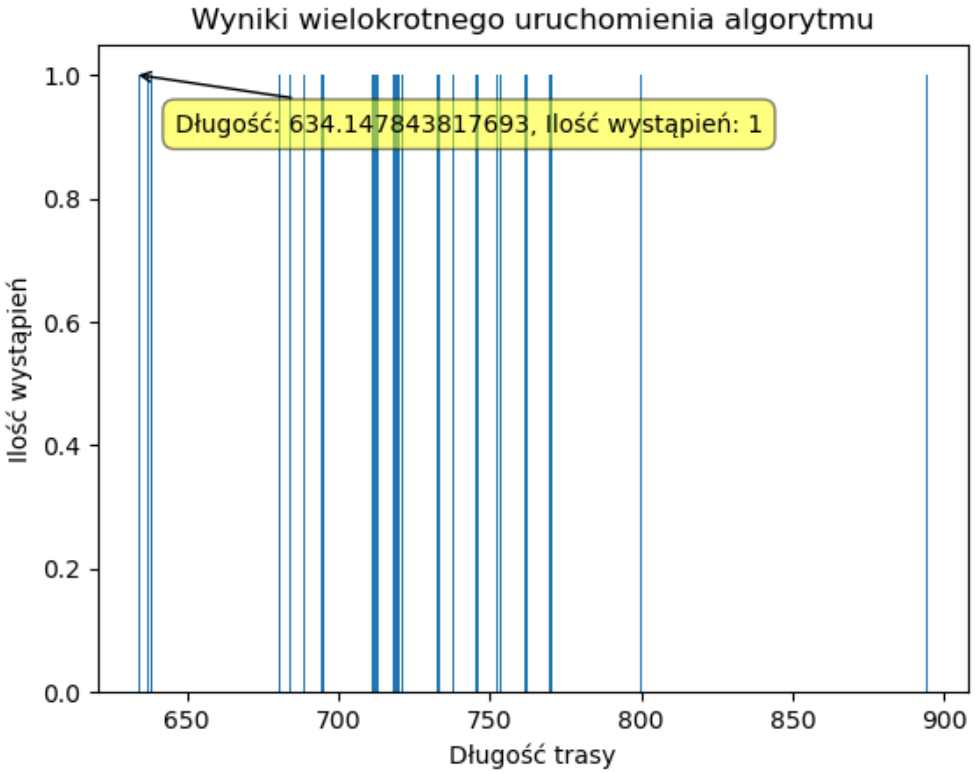
Dla parametrów (kolejność jak w tabeli) 2, 30N, 0,4, 2, 20N:

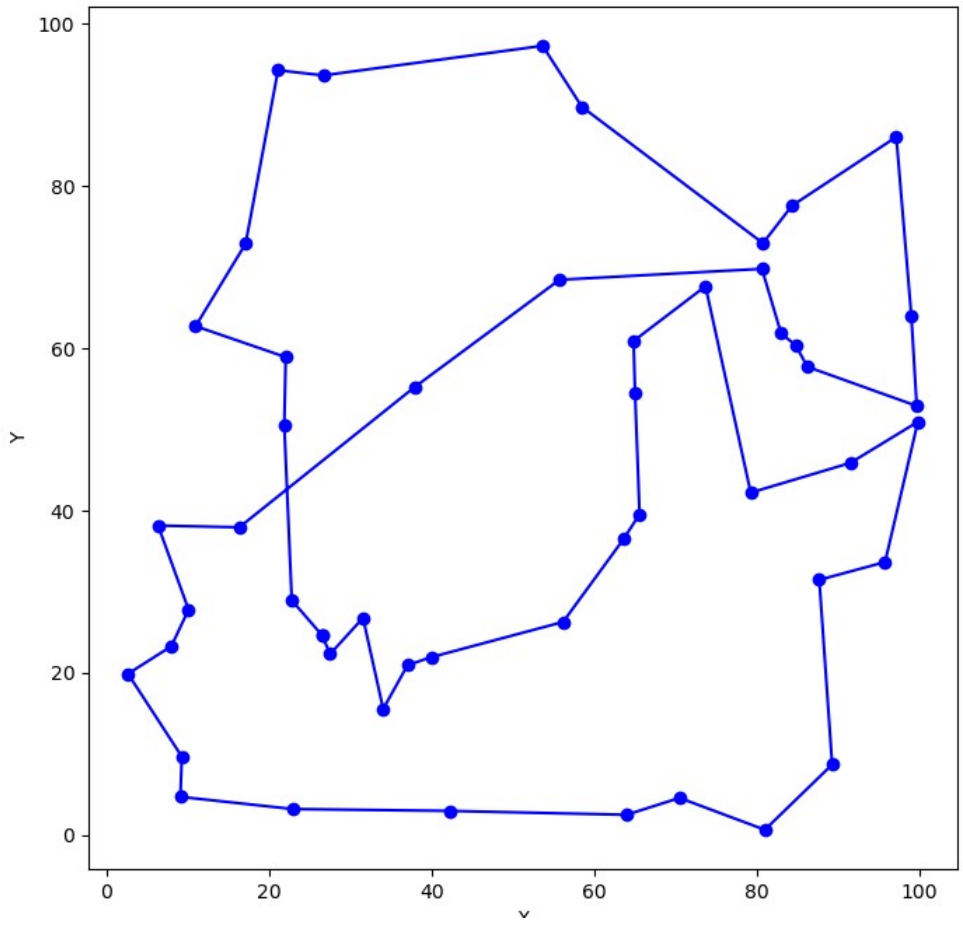
- Najlepsza trasa 671,89
- Średnia 767,21
- Odchylenie Standardowe 61,26
- Czas 472,501s



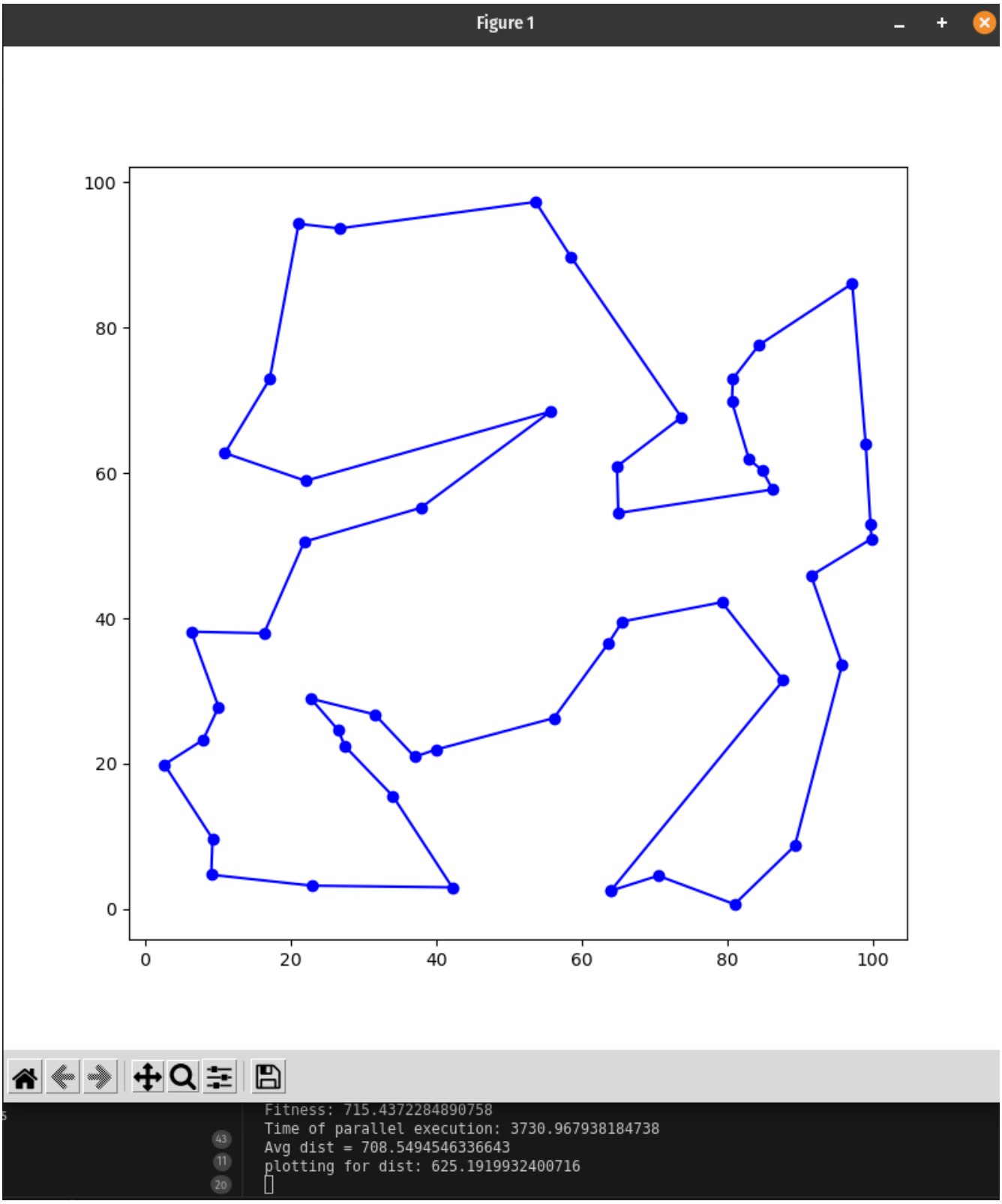


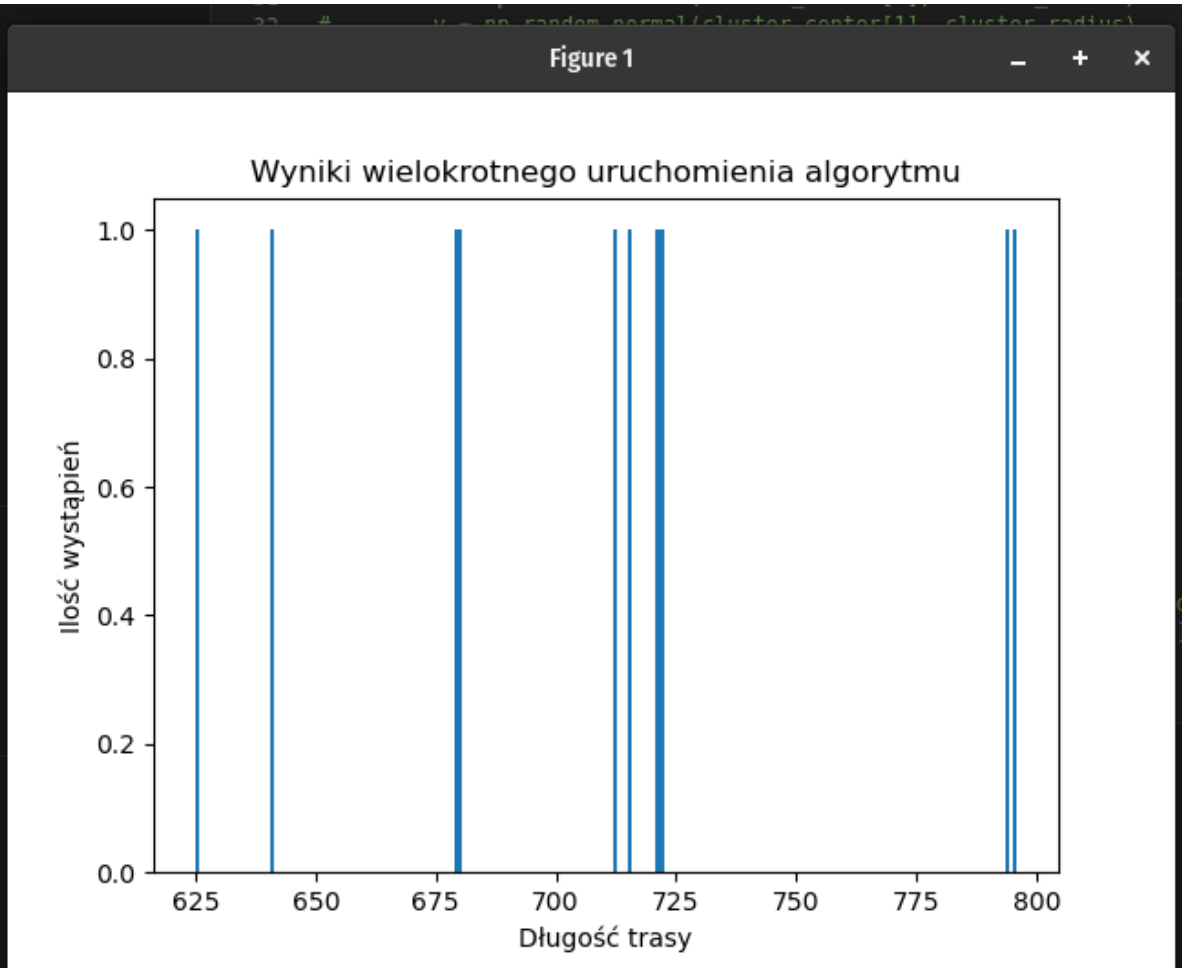
Dla parametrów (kolejność jak w tabeli) 2, 50N, 0,4, 2, 50N:
- Najlepsza trasa 634,15
- Średnia 725,19
- Odchylenie Standardowe 55,61
- Czas 1915,517s


Wyniki wielokrotnego uruchomienia algorytmu
Długość: 634.1478438176 93, Ilość wystąpień: 1
Ilość wystąpień
Długość trasy
0.0
0.2
0.4
0.6
0.8
1.0
650
700
750
800
850
900


Y
0
20
40
60
80
100

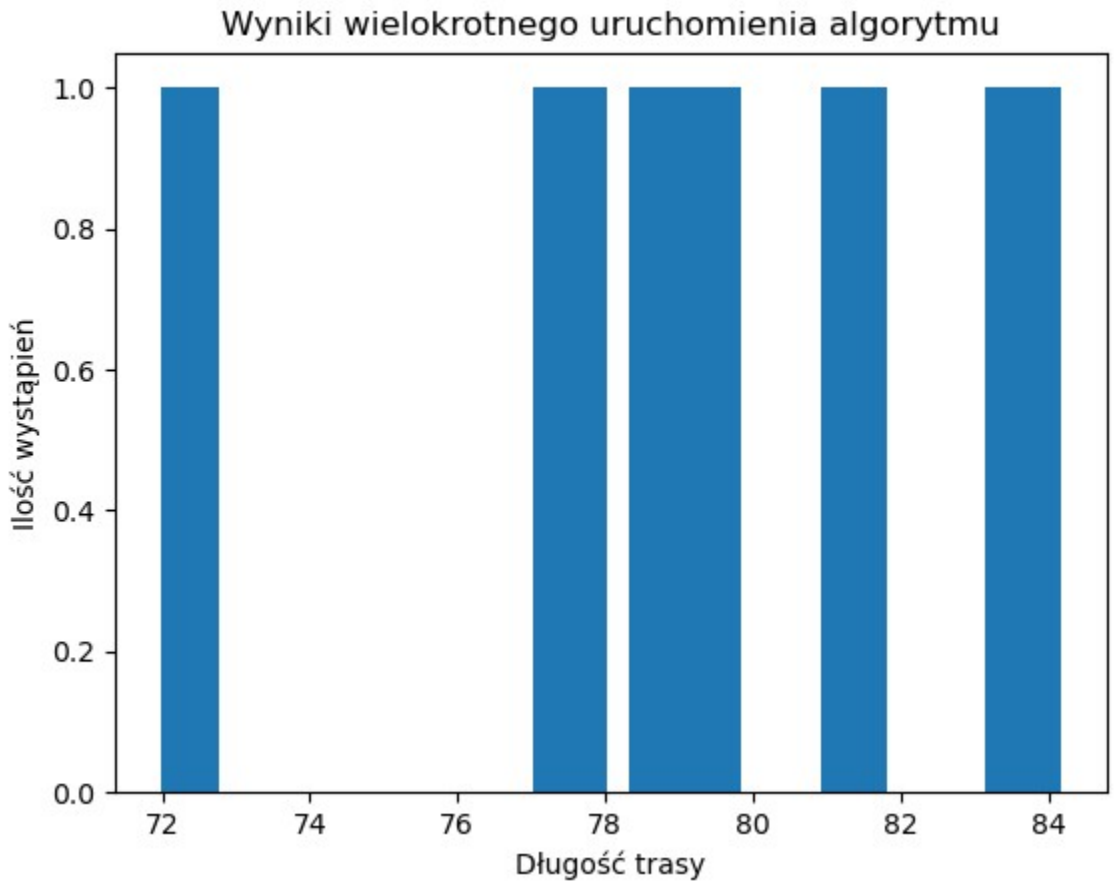
Najlepszy, prawdopodobnie optymalny wynik dostałem zostawiając laptopa na dość długi czas na liczenie z bardzo dużymi wartościami wszystkich parametrów i moją modyfikacją. Wynik wtedy wyglądał następująco:

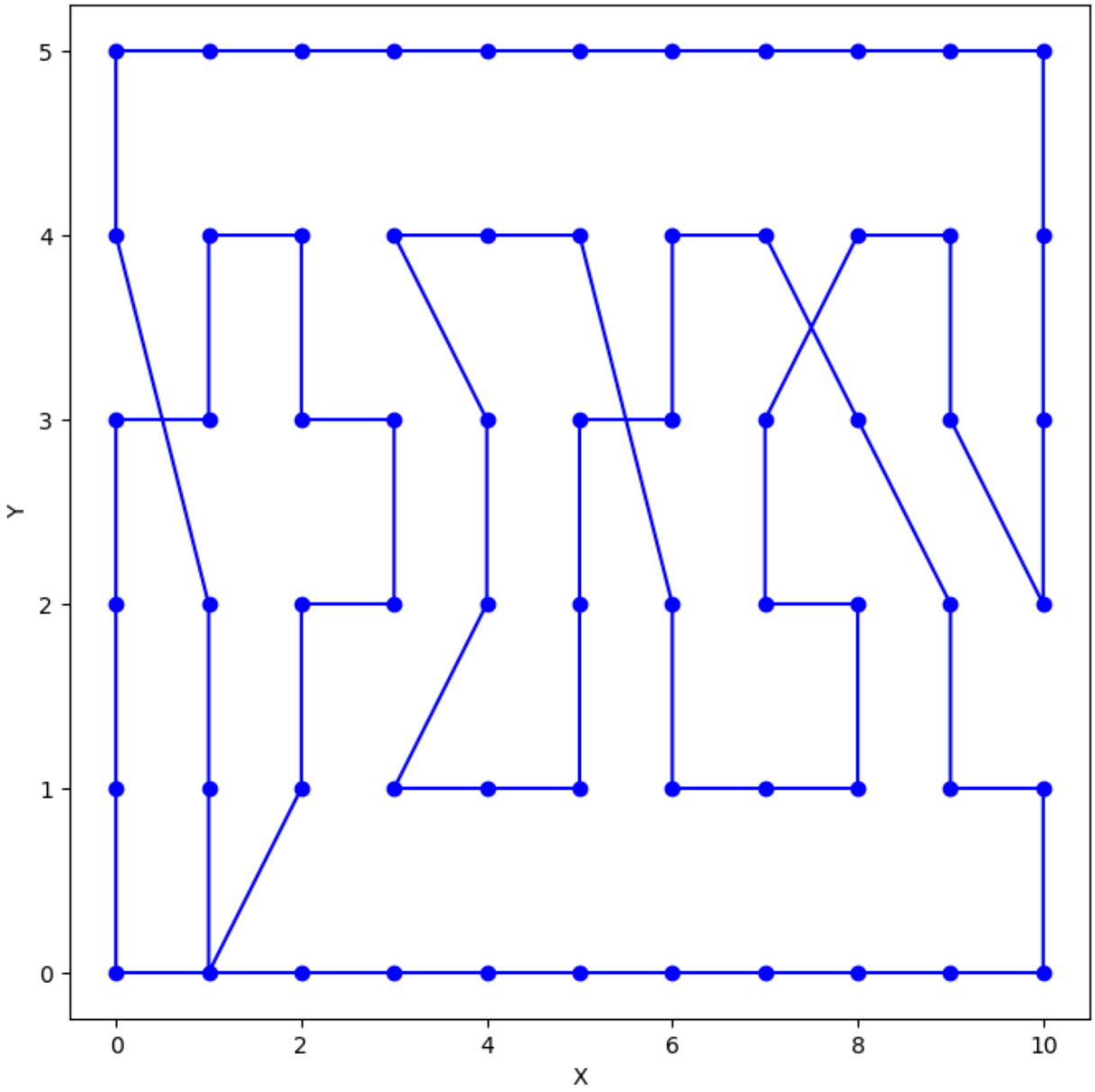





```
dbartosiak@pop-os:~/uczelnia/wsi$  cd /home/dbartosiak/uczelnia/wsi ; /usr/bin/env /
/../../debugpy/launcher 51647 -- /home/dbartosiak/uczelnia/wsi/lab2/main4.py
Fitness: 679.1009062271605
Fitness: 721.0169504705894
Fitness: 680.0203490406302
Fitness: 712.2334723198449
Fitness: 795.6168055719976
Fitness: 625.1919932400716
Fitness: 721.9920500812093
Fitness: 793.9847468083623
Fitness: 640.9000440876999
Fitness: 715.4372284890758
Time of parallel execution: 3730.967938184738
Avg dist = 708.5494546336643
```

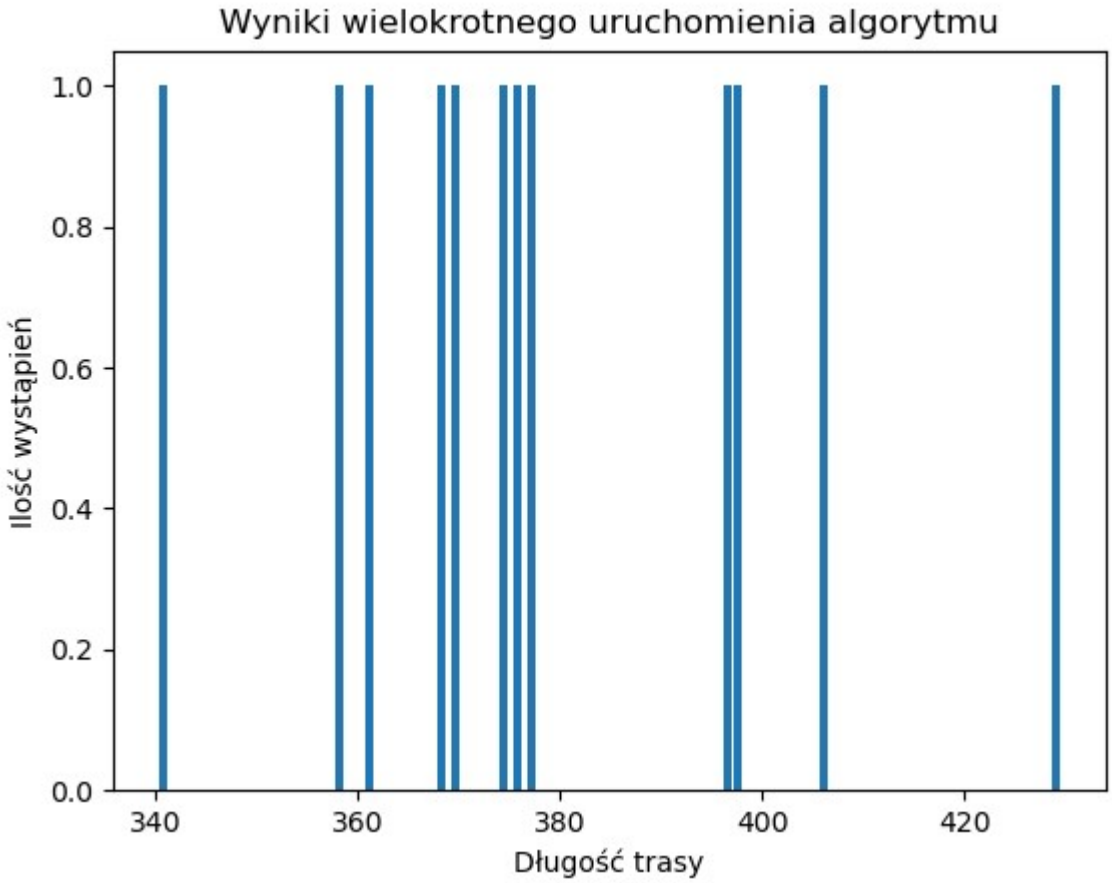
4) Pozostałe rozłożenia miast, bardziej jako ciekawostka dla najlepszych wartości
a) Szachownica: 2, 100N, 0.4, 2, 100N:

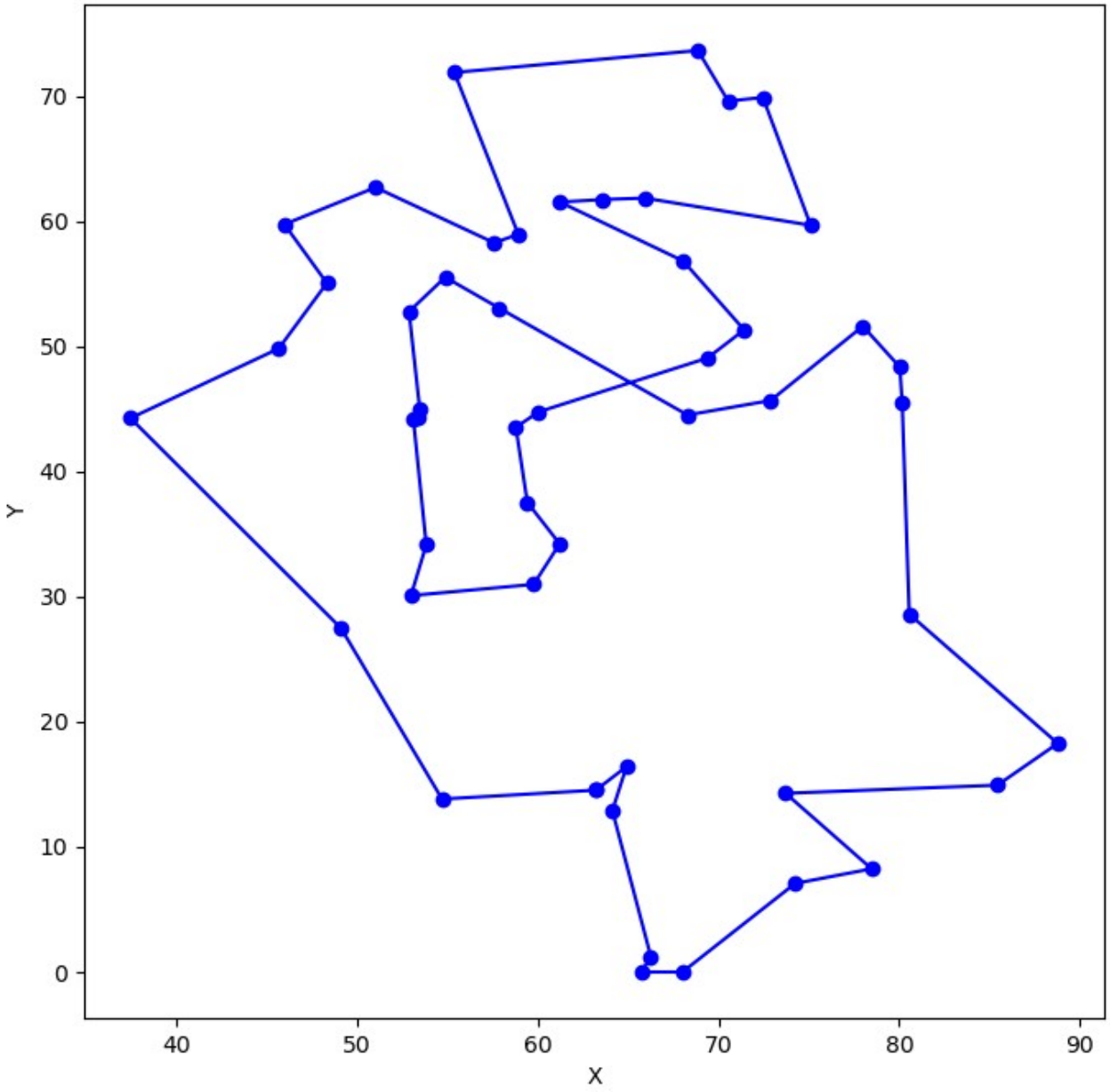




Tutaj czas wynosił 8008 sekund, średnia odległość 79,39, odchylenie standardowe 3,04 a najlepszy wynik 72,37. Tutaj mimo poświęcenia dużej ilości czasu i zasobów do obliczeń nie udało się otrzymać wyniku optymalnego (widać przejście do miasta przez które zasadzie przejeżdzaliśmy co nie ma prawa wystąpić w takim charakterze w rozwiązaniu optymalnym).

b) Kilka skupisk o rozkładzie Gaussa: 2, 100N, 0.4, 2, 100N:





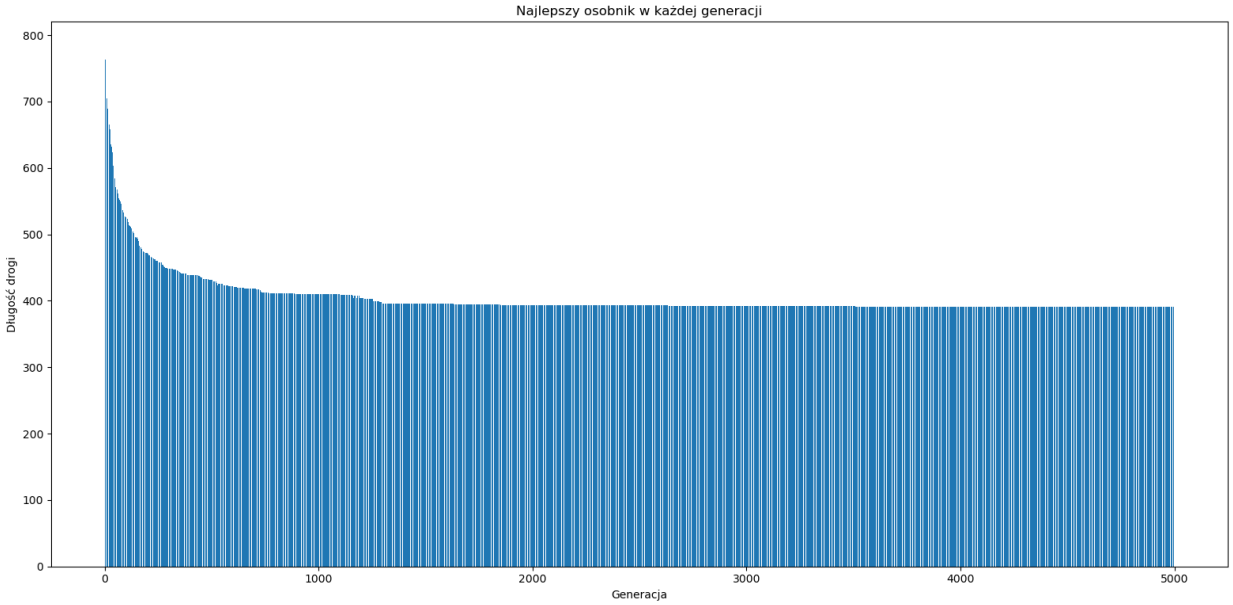
W przypadku kilku skupisk czas wynosi 3800 sekund, średnia odległość 379,53, odchylenie standardowe 24,03 a najlepszy wynik 340,64. Ponownie niestety nie udało się otrzymać rozwiązania optymalnego, co widać ponownie po przebiegu.

4. Przebieg ewolucji w czasie iteracji

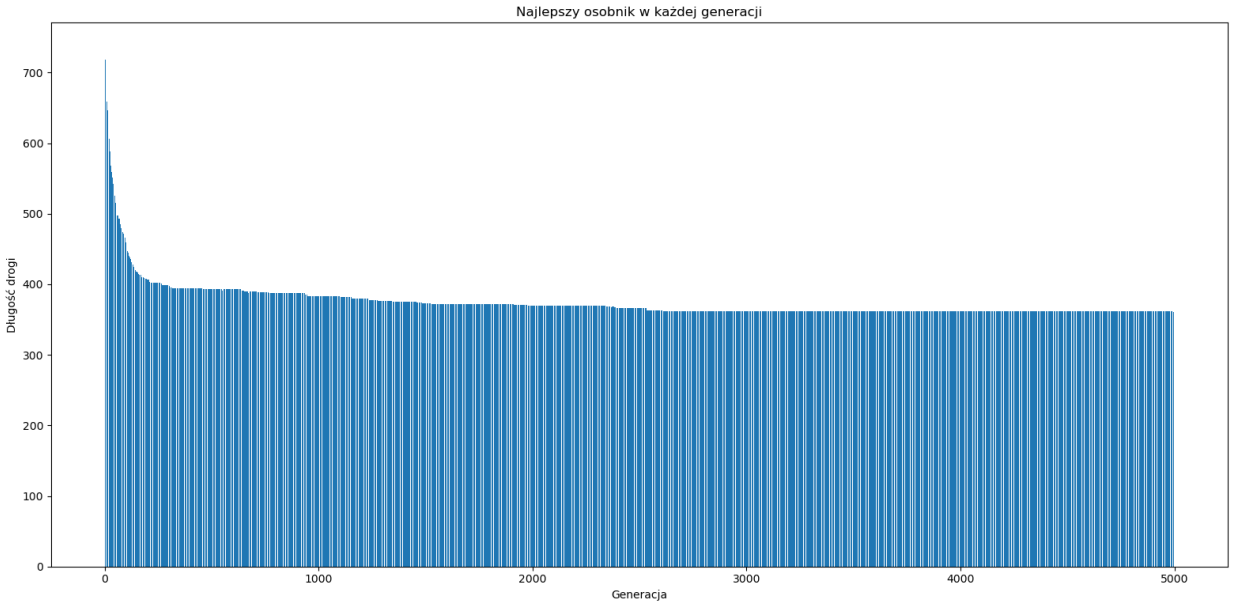
Dzięki obserwacji zmian najlepszej trasy w zależności od iteracji możemy dostać dodatkowe dane odnośnie działania algorytmu. Kolejno wykresy będę zamieszczał dla (uśrednianie z 12 prób) stałej długości 5000 generacji o następujących parametrach:

Zdjęcie zależności długości trasy od iteracji nr.

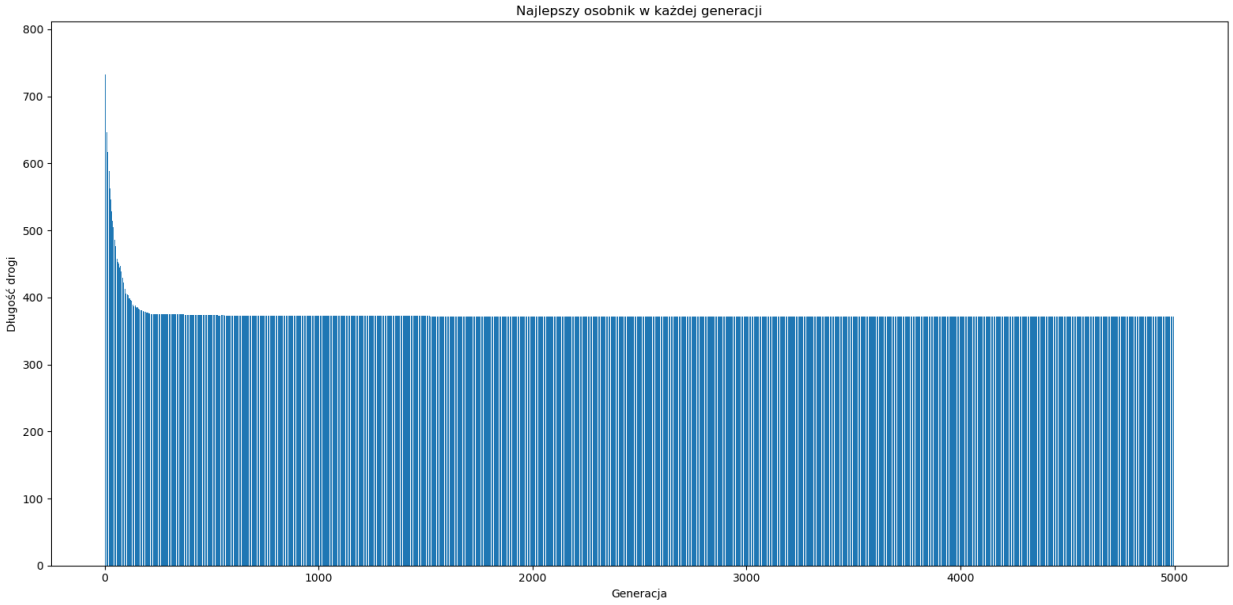
Numer porządkowy	Ilość miast	Rozmiar Populacji	Ilość mutacji	Szansa mutacji	Rozmiar turnieju
1	20	10N	rand	0.2	2
2	20	50N	rand	0.2	2
3	20	10N	2	0.4	2
4	20	50N	2	0.4	2



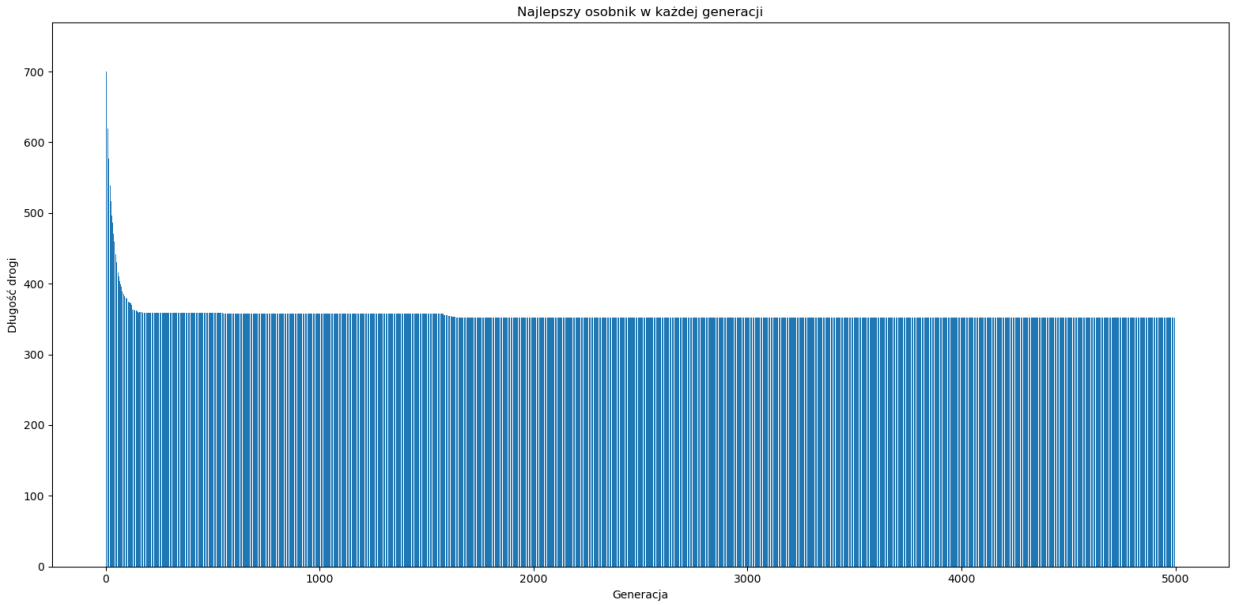
Zdjęcie zależności długości trasy od iteracji nr. 1



Zdjęcie zależności długości trasy od iteracji nr. 2



Zdjęcie zależności długości trasy od iteracji nr. 3



Zdjęcie zależności długości trasy od iteracji nr. 4

W wyniku powyższych eksperymentów zauważyć można że wzrost rozmiaru populacji gwarantuje szybsze osiągnięcie optymalnych wartości. Dodatkowo losowa ilość mutacji zdaje się w tym przeszkadzać, prawdopodobnie z uwagi na ‘psucie’ w wyniku ewolucji tych najlepszych dróg. Dodatkowo przy mniejszej ilości generacji pomysł z zachowaniem najlepszego wyniku faktycznie ma sens z uwagi na tymczasowe wzrosty odległości spowodowane wspomnianym działaniem ewolucji.

5. Optymalizacja

W celu umożliwienia przeprowadzania takiej ilości doświadczeń przy ograniczonym czasie (wiadomym jest, że obliczenia trwające na przykład 20godzin dla pojedynczego przypadku nie mają sensu), zrównolegliłem obliczenia wykorzystując bibliotekę multiprocessing w pythonie, przez co wykonywałem 12 równoległych przypadków, co w najlepszym przypadku oszczędziło mi około 24 godzin (a to wywołanie tylko jednego z przypadków). Załączam zrzut ekranu z wykorzystania procesora (glances):

CPU%	MEM%	VIRT	RES	PID	USER	TIME+	THR	NI	S	R/s	W/s	Command ('k' to kill)
>98.8	0.3	500M	42.0M	44899	dbartosia	1:00	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
98.6	0.3	500M	42.0M	44901	dbartosia	1:03	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
96.7	0.3	500M	42.1M	44891	dbartosia	1:04	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
95.8	0.3	500M	42.1M	44897	dbartosia	1:04	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
95.3	0.3	500M	42.2M	44902	dbartosia	1:01	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
94.1	0.3	500M	42.1M	44892	dbartosia	1:02	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
93.6	0.3	500M	42.1M	44896	dbartosia	1:03	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
92.7	0.3	500M	42.1M	44893	dbartosia	1:01	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
90.5	0.3	500M	42.1M	44894	dbartosia	1:03	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
88.6	0.3	500M	42.2M	44898	dbartosia	1:01	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
88.1	0.3	500M	42.2M	44900	dbartosia	1:03	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py
87.4	0.3	500M	42.1M	44895	dbartosia	1:03	1	6	R	0	0	python3 /home/dbartosiak/uczelnia/wsi/lab2/main.py

6. Wnioski

Moje badania dotyczyły cały czas implementacji algorytmu ewolucyjnego do rozwiązania problemu komiwojażera. Wnioski i obserwacje wynikające z przeprowadzonych eksperymentów:

a) Wpływ mutacji: Mutacja miast była jednym z kluczowych elementów algorytmu ewolucyjnego. Eksperymenty pokazały, że zbyt duża liczba mutacji, oraz zbyt wysoka częstotliwość, szczególnie w algorytmie niezmodyfikowanym może prowadzić do gorszych wyników, przez gubienie rozwiązania optymalnego. Wydaje się, że potrzebne jest znalezienie odpowiedniej dla naszego problemu ilości mutacji.

b) Wpływ rozmiaru populacji: Zwiększenie rozmiaru populacji miało tendencję do poprawy wyników. Większa populacja może zwiększyć szansę na odkrycie lepszych rozwiązań, ale wiąże się to z większym obciążeniem obliczeniowym.

c) Wpływ rozmiaru turnieju: Rozmiar turnieju wpływa na to, jakie rozwiązania są wybierane do reprodukcji, niestety daje to również duży nakład obliczeń (chcemy wybierać najlepszych osobników w ilości: rozmiar turnieju * wielkość populacji). Dla różnych problemów i danych wejściowych, różne rozmiary turnieju mogą działać lepiej, aczkolwiek jeden przedstawiony przeze mnie przypadek nie był odpowiednim przedstawieniem całej ilości wykonanych doświadczeń, gdy nie zmieniamy ilości turniejów, a zmieniamy jego wielkość, z pewnością działa on na korzyść do znajdowania optymalnego rozwiązania, ale zarazem tracimy informację o części nierozpatrzonych przypadków.

d) Wpływ ilości generacji: W przypadku większej liczby generacji algorytm miał więcej czasu na znalezienie optymalnych rozwiązań. Jednak trzeba zwracać uwagę na czas obliczeń, działa analogicznie do rozmiaru populacji, szczególnie dla dowolnej ilości mutacji.

e) Zmodyfikowany algorytm: Wydaje się, że modyfikacja, w której biorę najlepsze wyniki z poprzednich generacji, może być skuteczną strategią poprawy wyników. To podejście pozwala na wykorzystanie najlepszych rozwiązań osiągniętych przez algorytm w całym czasie wykonania, które czasem potrafią zostać zgubione.

f) Trudność znalezienia optymalnego rozwiązania: W wynikach eksperymentów można zauważyć, że znalezienie optymalnego rozwiązania dla problemu komiwojażera jest trudne, szczególnie dla większej ilości miast. Może to wymagać długich obliczeń i/lub bardziej zaawansowanych technik optymalizacji.

g) Optymalizacja obliczeń: Użycie wielowątkowości i równoległego przetwarzania znacząco przyspieszyło obliczenia, co jest istotne przy tak dużym zbiorze danych i różnych konfiguracjach parametrów.

Podsumowując, moje eksperymenty przyniosły cenne informacje na temat wpływu różnych parametrów na jakość rozwiązania problemu komiwojażera. Po przeprowadzonych doświadczeniach nie uważam jednak algorytmu ewolucyjnego za optymalny do rozwiązania tego problemu, z uwagi na duże uzależenienie od losowości, oraz wykonywania dużej ilości obliczeń.