Problem komiwojażera – Algorytm genetyczny

Sprawozdanie

Marcin Polanowski

1.Wstęp

Problem komiwojażera (ang. Traveling salesman problem) – zagadnienie optymalizacyjne polegające na znalezienieu minimalnego cyklu Hamiltona w pełnym grafie ważonym. Istnieje wiele odmian I podejść do tego problemu, ja zajmę się odmianą symetryczną(graf nieskierowany).

2.Złożoność obliczeniowa

Cykl hamiltona to taki cykl, który zawiera wszystkie wierczołki w grafie i każdy tylko raz. Cykl ten można więc reprezentować jako pewna permutacja liczb od 1 do n gdzie n to liczba wierczhołków grafu. Jeśli będziemy chcieli rozwiązać ten problem przeglądając wszystkie możliwe rozwiązania to jego złożoność wyniesie O(n!), gdzyż n! To liczba wszystkich możliwych permutacji n-elementowych.
Teoria złożności klasyfikuje TSP jako problem NP-trudny.

3.Opis chromosomu

Jak opisałem wyżej:

<x1,x2, … ,xn> gdzie x1x2...xn – pewna permutacja liczb 0 do n-1 (wierczhołków w grafie)

Reprezentacja taka jest nazywana reprezentacją ścieżkową

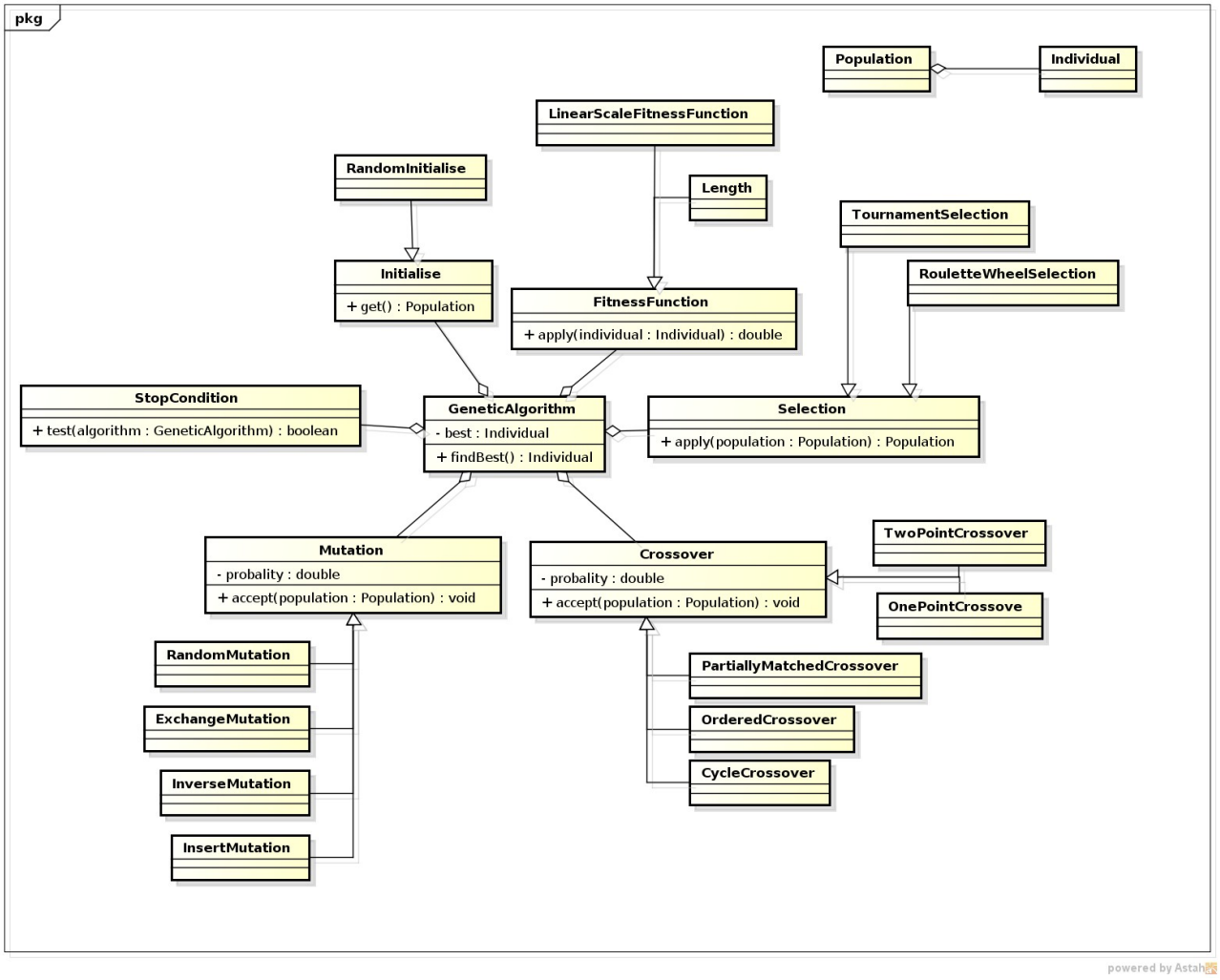
4.Opis funkcji przystosowania

Ponieważ szukamy minimalnego cyklu hamiltona funkcją celu będzie długość cyklu length(<x1,x2, … ,xn>). Funkcję tą musimy minimalizować, więc funkcja przystosowania będzie wyglądała następująco:

f(<x1,x2, … ,xn>) = MAX - length(<x1,x2, … ,xn>) gdzie MAX jest bardz dużą liczbą większą bądź równą maksimum funkcji length. Parametr ten musimy sami dobrać.

5.Opis implementacji

Główny algorytm jest wykonywany w obiekcie klasy GeneticAlgorithm. Składa się on z kilku operacji wykonywanych w pętli. Każda z tych operacji (generowanie populacji początkowej, selekcja, krzyżowanie, mutacja) istnieje w różnych wariantach i podejściach w zależności od użytkownika, lub rozwiązywanego problemu. Oddelegowałem te funkcje do oddzielnych obiektów i po zastosowaniu dziedziczenia oraz polimorfizmu uzyskałem sprawny mechanizm dzięki, któremu można szybko zmienić daną metodę krzyżowania, mutacji itd. Dodatkow dodałem również interfejs graficzny.



6. Opis wykorzystanych zbiorów

a. att48
min length(<x1, x2, … , x48>) = 33523.71
MAX = 180000
max f(<x1, x2, … , x48>) = MAX - 33523.71 = 180000 - 33523.71 = 146476.29

b. tsp225
min length(<x1, x2, … , x225>) = 3859
MAX = 22000
max f(<x1, x2, … , x225>) = MAX – 3859 = 22000 – 3859 = 18141

c. lin105
min length(<x1, x2, … , x105>) = 14382.996
MAX = 100000
max f(<x1, x2, … , x105>) = MAX – 14382.996 = 100000 – 14382.996 = 85617,004

7. Badania

Istnieją różne metody ustalania parametrów. Główny Ich podział to:
-Dostrajanie parametrów – przed uruchomieniem algorytmu
-Sterowanie parametrami – w trakcie pracy algorytmu

Sterowanie parametrami polega na zaimplementowaniu mechanizmu wewnątrz algorytmu który będzie zmieniał parametry w trakcie jego działania. Dostrajanie parametrów zaś to po prostu stworzenie zbiorów elementów dla każdego parametru (np. Krzyżowanie {0.5, 0.6, 0.7, 0.8, 0.9, 1}) i odpalenie algorytmu ze wszystkimi możliwymi kombinacjami. Wadą tego rozwiązania jest mnogość tych kombinacji. Ja zastosowałem skróconą procedurę dostrajania parametrów

a. att48

Główne badania przeprowadziłem właśnie na tym zbiorze. Uproszczena procedura dostrajania polega na tym że na przyjęciu jakiś paraetrów początkowych a potem zmianie pojedynczego parametru przy stałości pozostałych parametrów.

1.Parametry początkowe:
2.Testowanie mutacji 0-0.1 delta=0.01
3. Testowanie krzyżowania 0.5-1 delta=0.05
4. Testowanie selekcji (dla turnieju od 1-20 delta 1)
5. Testowanie wielkości populacji 50 -1000 delta 50
6. Testowanie licby pokolen 500 – 1000 delta=100

Przeprowadziłem dwie takie iteracje:

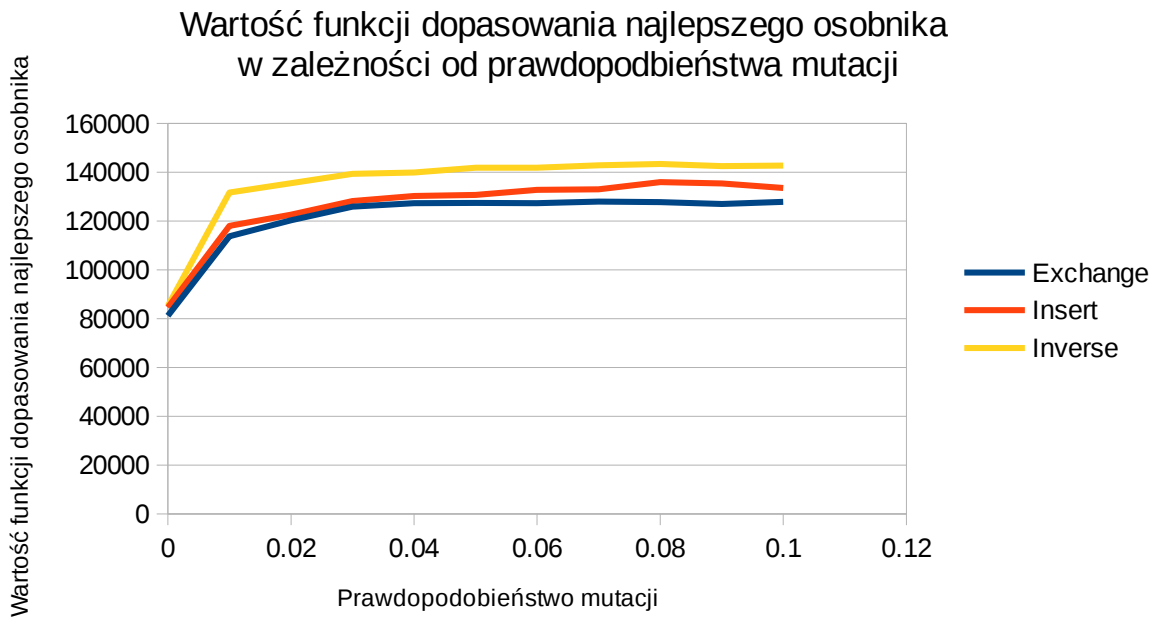
Iteracja 1:

ad 1.Parametry początkowe:

t=500
N=50
Selekcja – Turniej
k=3
Krzyżowanie – OX
pk=0.7

ad 2.Testowanie mutacji 0-0.1 delta=0.01

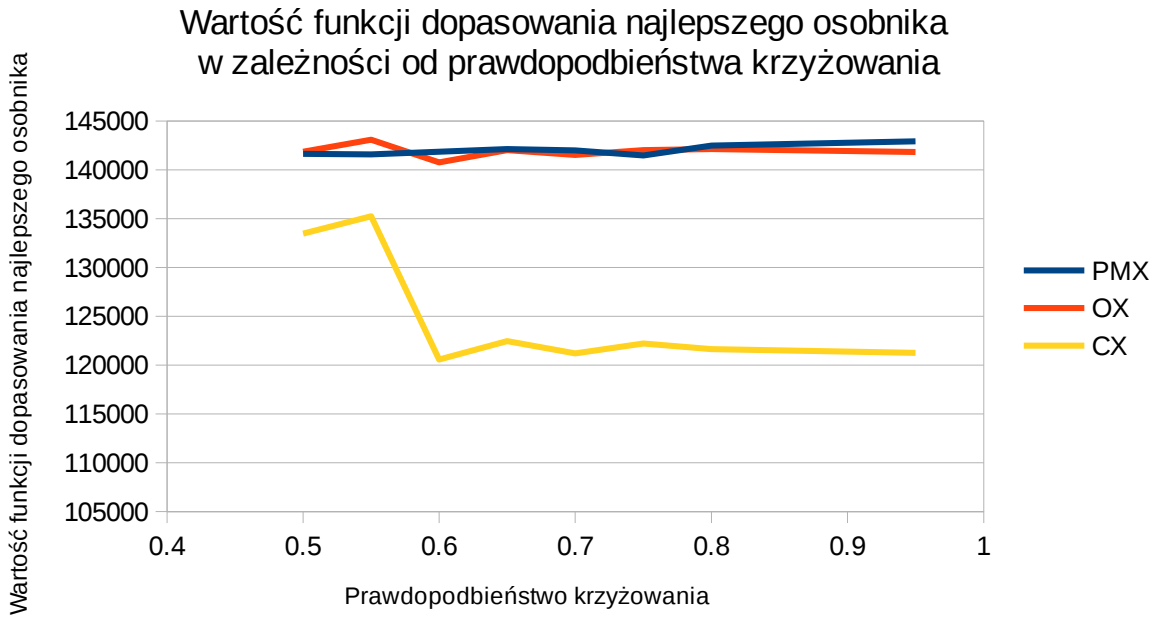
Wynik:
Typ mutacji – Inwersja
pm = 0.08



<u>Obserwacja</u>
Przy wszystkich trzech wariantach mutacji wynik rośnie wraz ze wzrostem pm I osiąga swe maksimum przy 0.7-0.8 po czym maleje

<u>Obserwacja</u>
Inwersja wydaje się osiągać najlepsze wyniki, potem wstawianie, pote wymiana

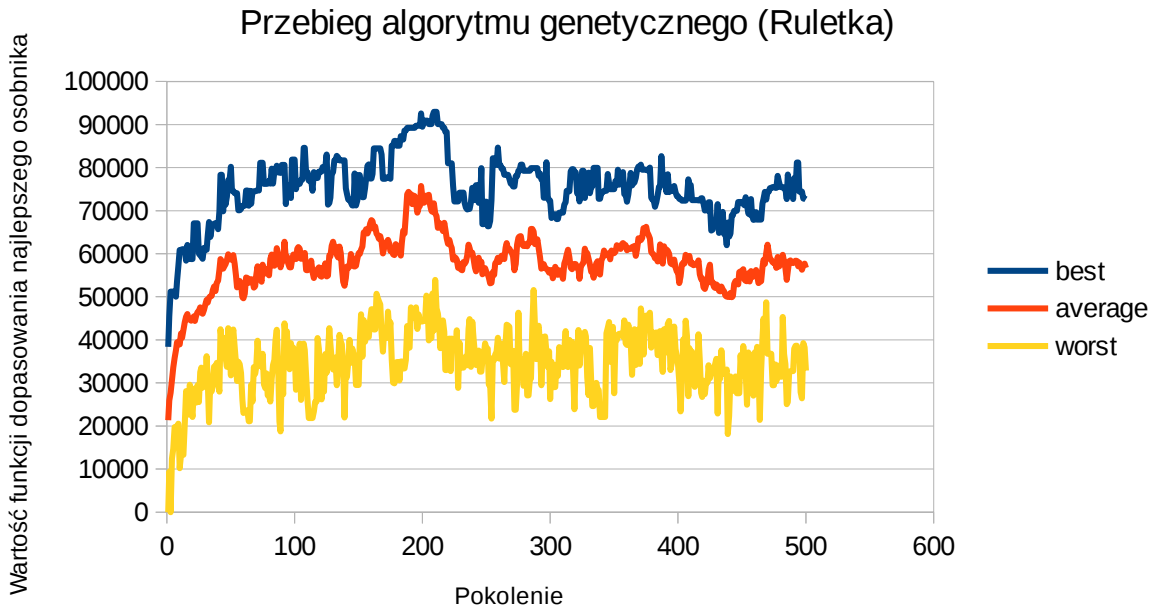
ad 3. Testowanie krzyżowania 0.5-1 delta=0.05



Wynik:
Typ krzyżowania – OX
pk=0.55

<u>Obserwacja</u>
CX znacznie odbiega od pozostałych typów krzyżowania

ad 4. Testowanie selekcji (dla turnieju od 1-20 delta 1)

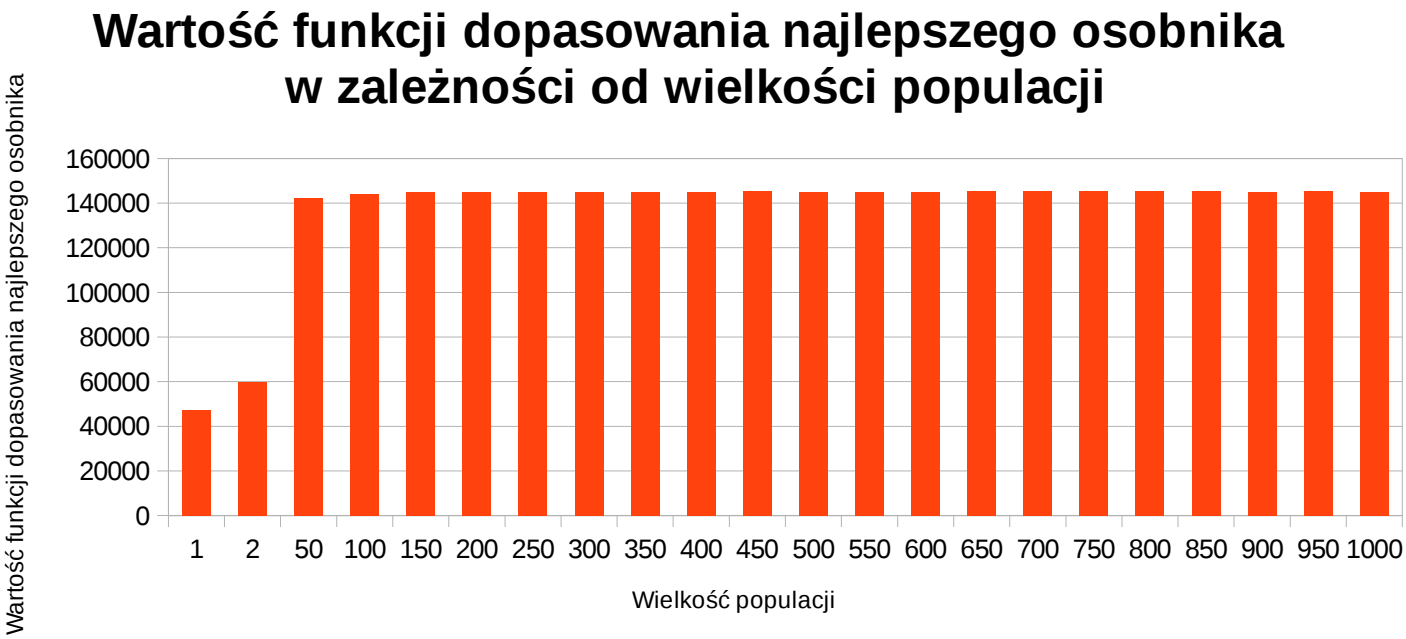


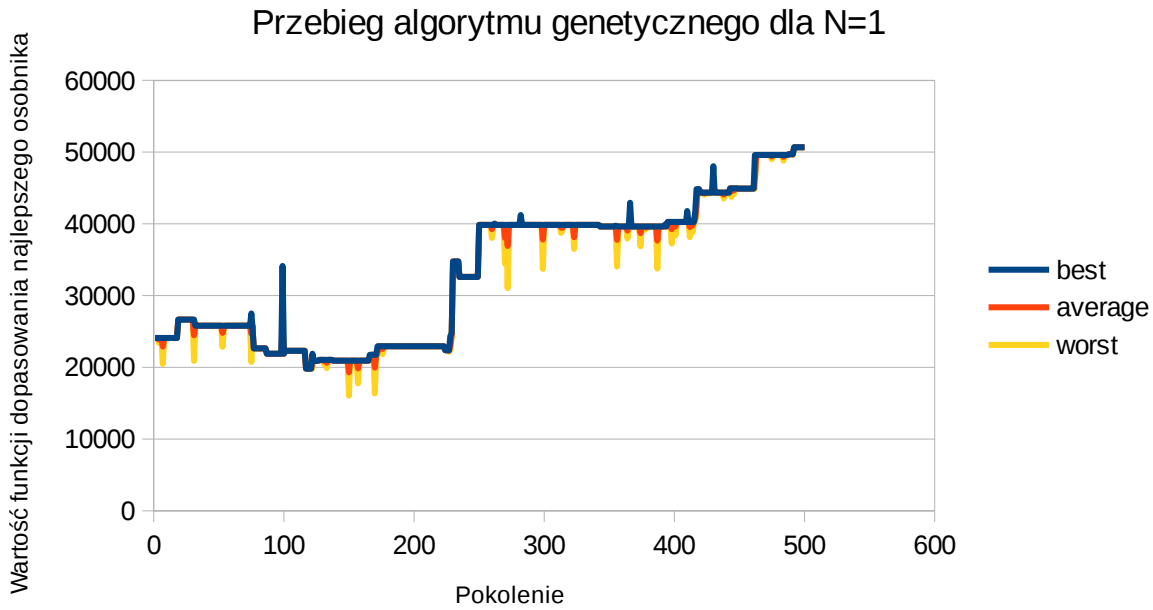
Wynik:
Typ selekcji – Turniej
k=3

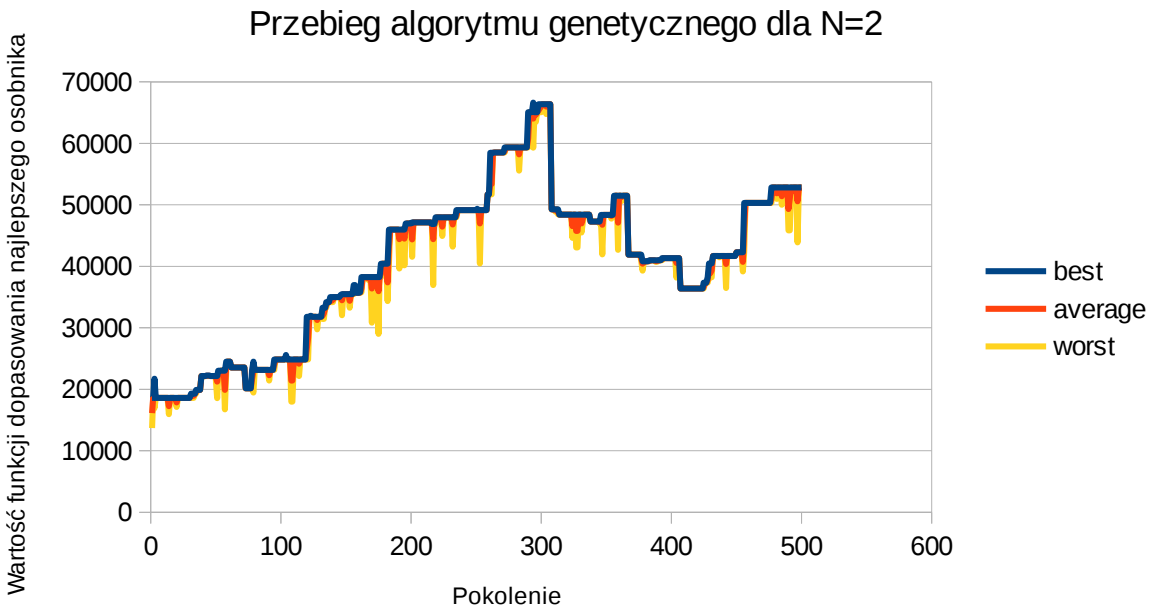
<u>Obserwacja</u>
Ruletka jest bardzo nieefektywną metodą selekcji

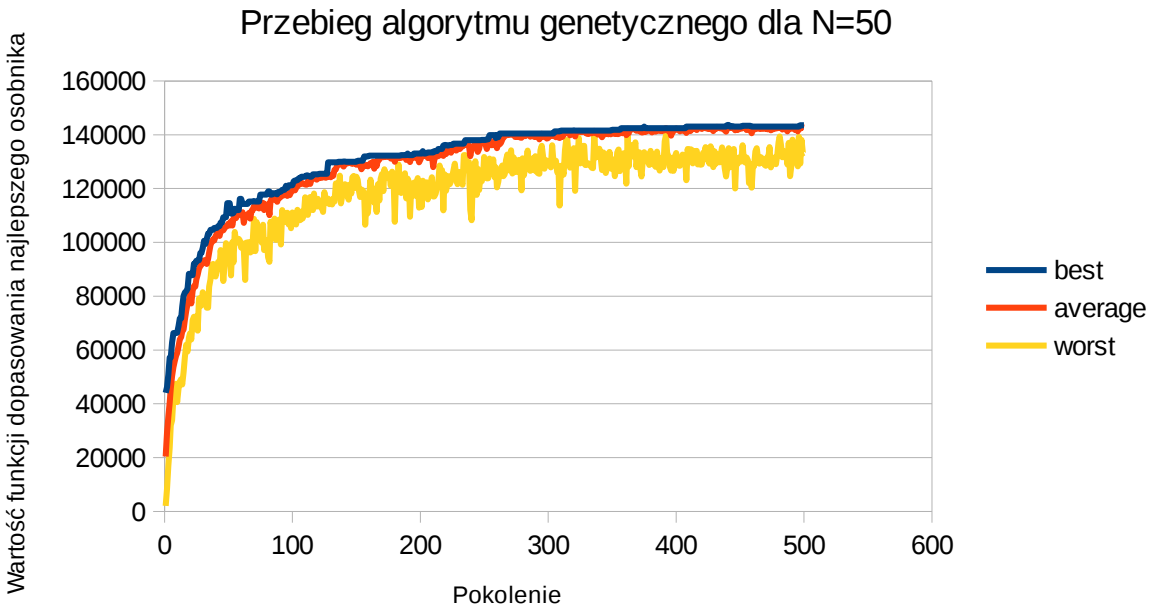
<u>Obserwacja</u>
Rozmiar turnieju mało wpływa na jakość algorytmu

ad 5. Testowanie wielkości populacji 50 -1000 delta 50









Wynik:
N=50

<u>Obserwacja:</u>
Rozmiar populacji mało wpływa na jakość algorytmu

<u>Obserwacja:</u>
N=1 jest przypadkiem skrajnym w którym działa tylko mutacja, jest więc bardzo zbliżony do algorytmu całkowicie losowego

<u>Obserwacja:</u>
N=2 podobnie, lecz działa również krzyżowanie

ad 6. Testowanie licby pokolen 500 – 1000 delta=100

Obserwacja
Złamanie na poziomie 600-700

t=900

Iteracja 2:

ad 5. Testowanie wielkości populacji 50 -1000 delta 50

N=50

ad 4. Testowanie selekcji (dla turnieju od 1-20 delta 1)

Typ selekcji – Turniej
k=3

Obserwacja
Dziwny spadek dla k=20

ad 3. Testowanie krzyżowania 0.5-1 delta=0.05

Typ krzyżowania – OX
pk=0.8

Obserwacja
Mały wpływ krzyżowania

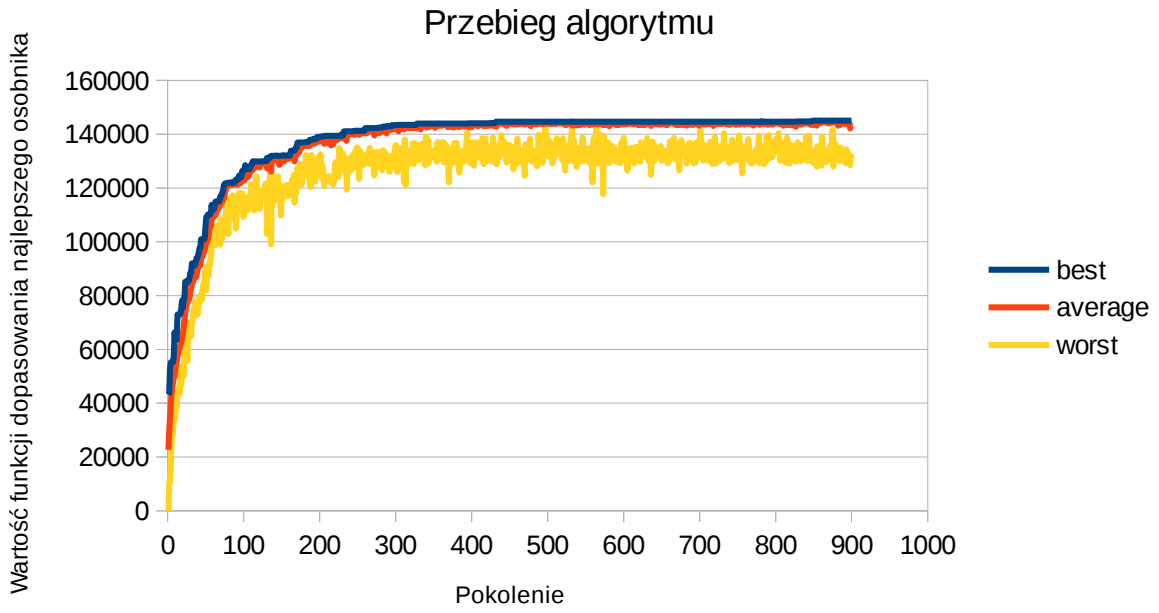
Obserwacja
CX lekko odbiega od pozostałych typów mutacji

Obserwacja
Dziwny spadek dla pk=0.95

ad 2. Testowanie mutacji 0-0.1 delta=0.01

Typ mutacji – Inwersja
pm= 0.09

Obserwacja
Znów dziwny spadek dla pm=1



b. tsp225

Wartość funkcij dopasowania:
-Optymalna ścieżka: 3859
-Najlepszy wynik uzyskany przez algorytm: 35285.52
-Średni wynik uzyskany przez algorytm: 35120.834

c. lin105

Wartość funkcij dopasowania:
-Optymalna ścieżka: 85617,004
-Najlepszy wynik uzyskany przez algorytm: 83964.33
-Średni wynik uzyskany przez algorytm: 83366.243

8. Wnioski:

- Brak wpływu niektórych parametrów na działanie algorytmu może być spowodowane dobrym dobraniem parametrów początkowych lub zbyt małą wielkością zbioru

- Niepokojące spadki przy najwyższych wartościach parametrów (biorąc pod uwagę błędy przy formatowaniu na wyjściu) mogą być spowodowane błędami przy konwersji pomiędzy postacią binarną a dziesiętną

- Prawdopodobnie szybciej byłoby prowadzić badania nie w excelu, a w zaimplementowanym wewnątrz programu mechaniźmie, który automatycznie wykonałby wszystkie obliczenia, przedstawił wykresy i wyniki

- Problem ruletki można rozwiązać wprowadzając współczynnik skalowania:
f(<x1,x2, … ,xn>) = MAX - a*length(<x1,x2, … ,xn>)