# SPRAWOZDANIE 1

## Algorytmy genetyczne

# Kwadratowy problem przydziału

Kwadratowy problem przydziału polega na wskazaniu lokalizacji dla N fabryk, tak aby zminimalizować koszt transportu (flow) pomiędzy nimi. Znając macierz odległości pomiędzy lokalizacjami i macierz wymaganego przepływu pomiędzy fabrykami A i B możemy obliczyć koszt transportu, czyli sumę iloczynów odległości między obiektami i strumieniami towarów przepływających między tymi obiektami.

# Algorytmy genetyczne

Rodzaj heurystyki przeszukującej przestrzeń alternatywnych rozwiązań w celu wyszukiwania rozwiązań najlepszych.

Sposób działania algorytmów genetycznych nieprzypadkowo przypomina zjawisko ewolucji biologicznej, ponieważ ich twórca John Henry Holland właśnie z biologii czerpał inspiracje do swoich prac. Obecnie zalicza się go do grupy algorytmów ewolucyjnych.

**Losowanie populacji:** Po ustaleniu wielkości populacji należy stworzyć wszystkie osobniki. Ze względu na fakt, że początkowa populacja powinna być jak najbardziej różnorodna każdy osobnik powinien być tworzony całkowicie losowo.

**Krzyżowanie:** W moim przypadku polega na korzystaniu metody RMX. Po selekcji wybieram liczbę osobników równej połowie ustalonej populacji i uzupełniam ją dziećmi utworzonymi właśnie z krzyżowania tą metodą.

**Selekcja:** Krok ten jest esencją całej genetyki. W tym miejscu tworzona jest nowa populacja na podstawie już istniejącej. W zależności od wartości funkcji dany osobnik ma większe (gdy jest 'dobry') lub mniejsze (gdy jest 'słaby') szanse na znalezienie się w kolejnym pokoleniu.

**Mutacja:** W moim przypadku mutacja polegała na zamianie w indeksów listy miejscami, co powoduje zmianę osobnika.

W moim rozwiązaniu skorzystałem z dwóch metod selekcji:

**Ruletka:** Selekcja w moim przypadku polega na przypisaniu przedziału wartości każdemu osobnikowi w zależności od wartości jego funkcji celu. Korzystałem ze wzoru : (Wartość funkcji najgorszego osobnika – wartość funkcji aktualnego osobnika) + 1, ponieważ w naszym przypadku zadanie polega na minimalizacji funkcji celu. Każdy osobnik przyjmował minimalną i maksymalną wartość na podstawie tego wzoru i losowałem liczbę i sprawdzałem dla którego osobnika ona pasuje. Niestety wartości te bardzo się od siebie nie różnią, więc selekcja działa prawie jak losowa.

**Turniej:** Metoda jest zupełnie różna od powyższej i polega na losowym wyborze z całej populacji kilku osobników (jest to tzw. grupa turniejowa), a później z tej grupy wybierany jest osobnik najlepiej przystosowany i on przepisywany jest do nowo tworzonej populacji. Losowanie grup turniejowych oraz wybieranie z nich najlepszego osobnika należy powtórzyć aż do utworzenia całej nowej populacji.

# Wyniki algorytmu GA

## SELEKCJA TURNIEJOWA

Parametry:

Liczba populacji – 100

Rozmiar populacji – 100

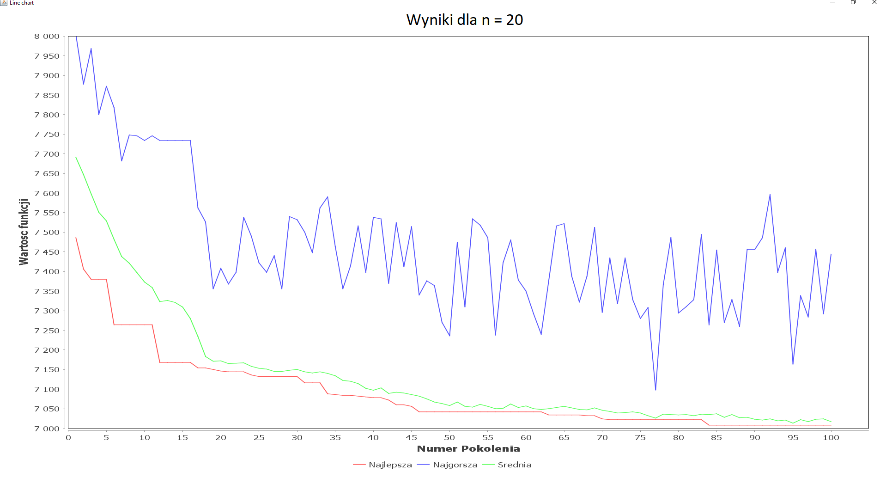
Prawdopodobieństwo mutacji – 1%

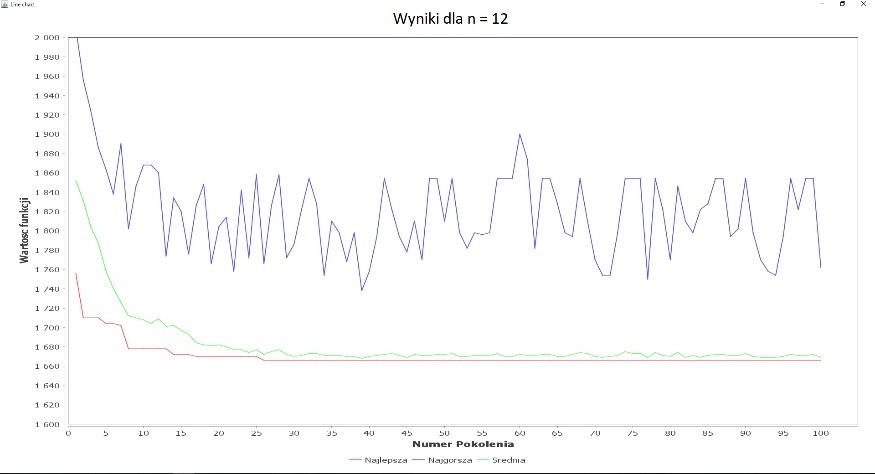
Początkowy indeks do krzyżowania algorytmem PMX – 3

Końcowy indeks do krzyżowania algorytmem PMX – 6

Liczba rund w turnieju – 5

Optimum dla n = 12 - 1652

Optimum dla n = 20 - 6922



Widzimy, że metoda selekcji daje o wiele większe prawdopodobieństwo osobnikom o małej wartości funkcji celu ( w przypadku naszym, czyli dla minimalizacji ten funkcji) . Kolejne pokolenia są coraz lepsze co widać z wyników ze średniej funkcji celi każdego pokolenia. W każdym kolejnym pokoleniu spada jednak różnorodność osobników i każdy z nich staje się prawie identyczny. Dla danych parametrów algorytmu które przestawiłem, algorytm nie znajduję ani jednego osobnika „optymalnego”.

## SELEKCJA TURNIEJOWA

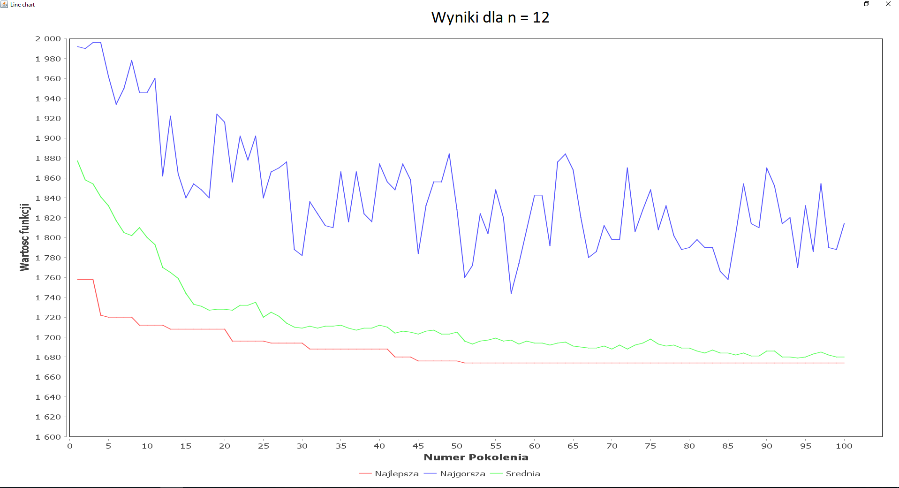
Parametry:

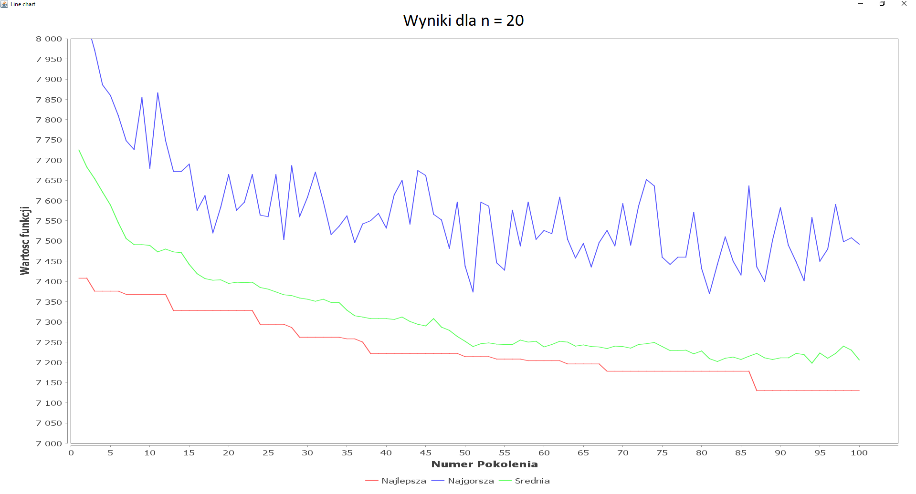
Liczba populacji – 100

Rozmiar populacji – 100

Prawdopodobieństwo mutacji – 1%

Początkowy indeks do krzyżowania algorytmem PMX – 3

Końcowy indeks do krzyżowania algorytmem PMX – 6



Możemy zauważyć, że funkcja celu zbiega wolniej niż w selekcji turniejowej. Tak jak opisywałem wcześniej, ta selekcja jest bardzo zbliżona do losowania osobników. Dla rozwiązań w których nie ma dużej rozbieżności wartości funkcji celu lepsza jest selekcja turniejowa niż ruletki.

# Badanie wpływu parametrów na działanie algorytmu w selekcji turniejowej

## Wpływ liczby pokoleń

Parametry:

Liczba populacji – na wykresie

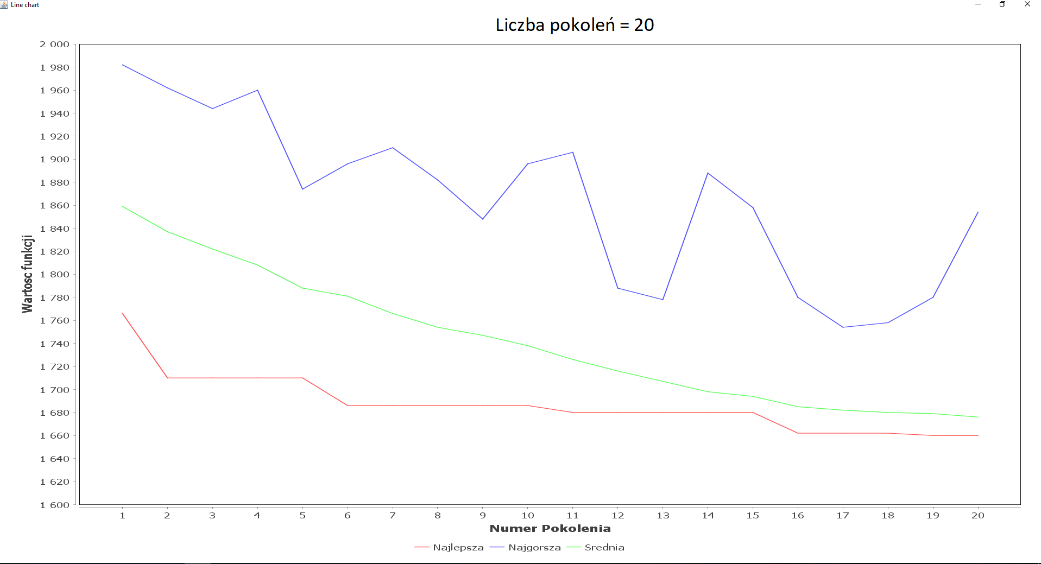
Rozmiar populacji – 100

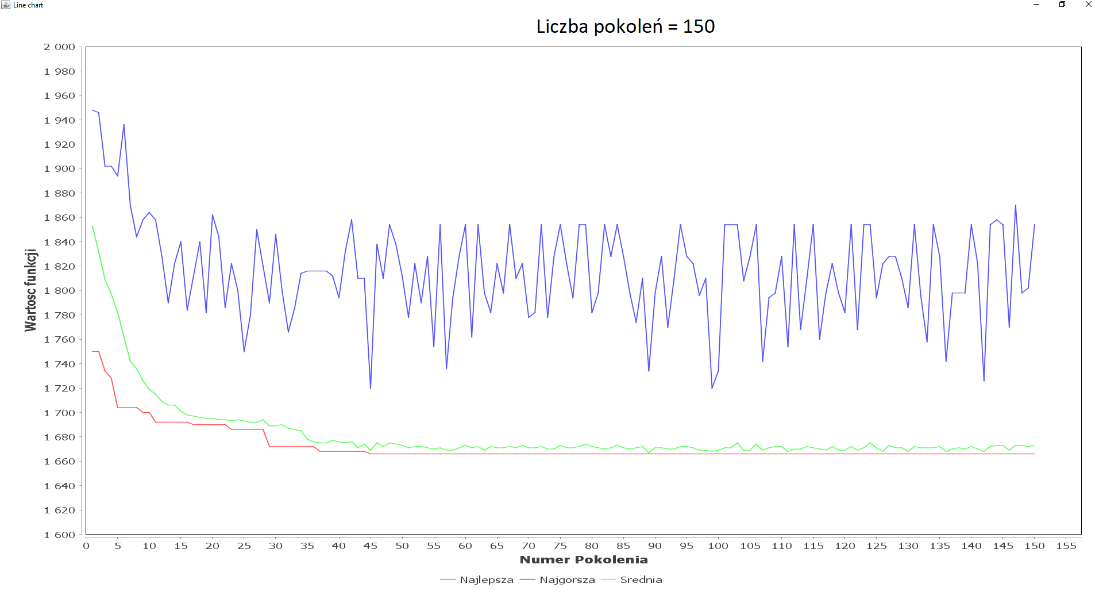
Prawdopodobieństwo mutacji – 1%

Początkowy indeks do krzyżowania algorytmem PMX – 3

Końcowy indeks do krzyżowania algorytmem PMX – 6

N – liczba „fabryk” - 12





## 

Możemy zauważyć, że liczba pokoleń wpływa na zbieżność funkcji celu. Ustawianie zbyt dużej liczby pokoleń nie ma sensu, ponieważ po pewnej wartości i tak większość osobników ma bardzo podobne wartości funkcji celu.

## Wpływ liczby populacji

Parametry:

Liczba populacji – 100

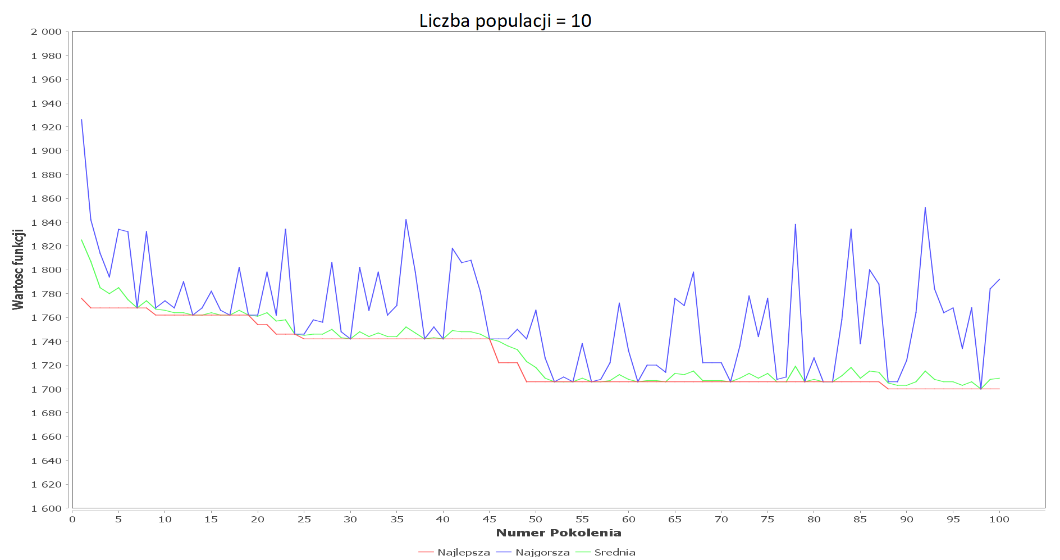
Rozmiar populacji – na wykresie

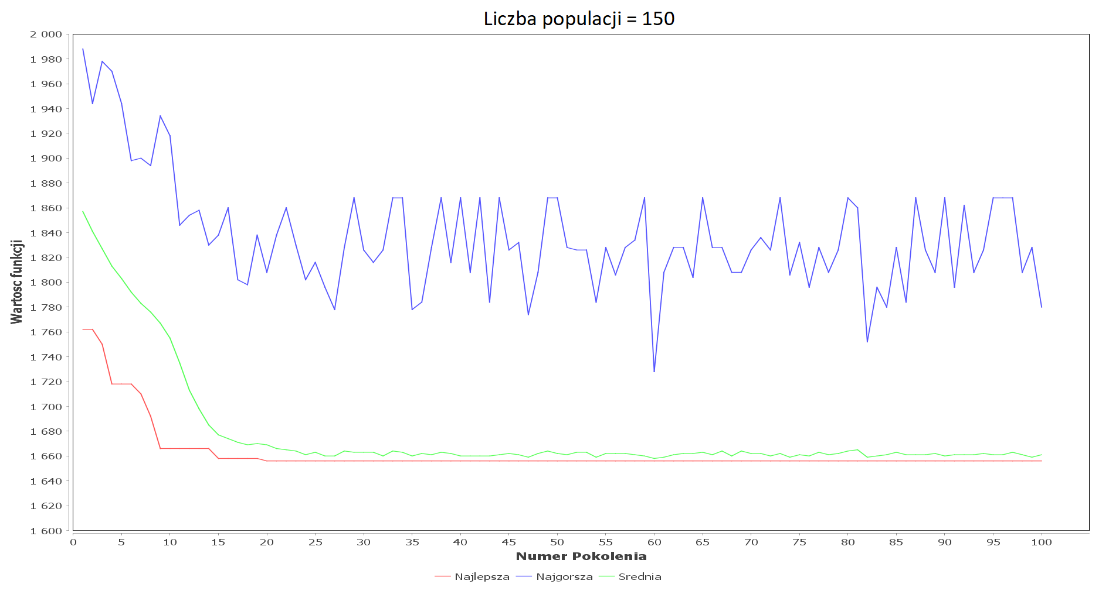
Prawdopodobieństwo mutacji – 1%

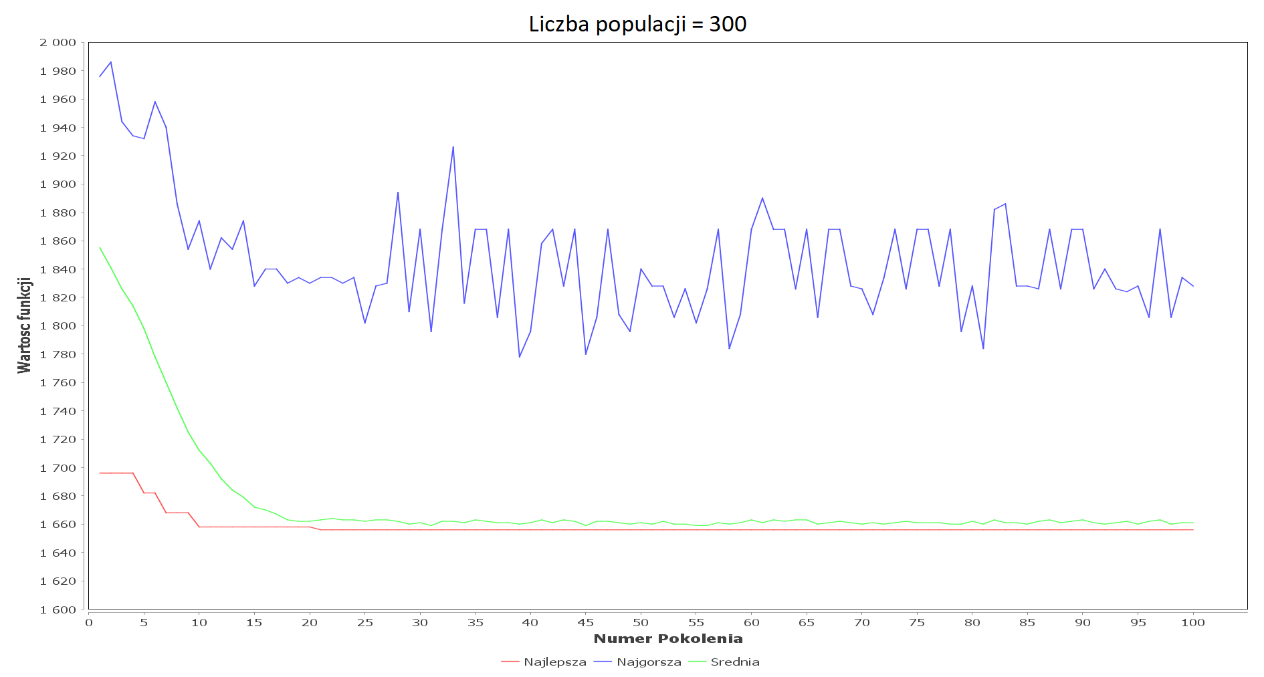
Początkowy indeks do krzyżowania algorytmem PMX – 3

Końcowy indeks do krzyżowania algorytmem PMX – 6

N – liczba „fabryk” - 12







Na podstawie wykresów można zauważyć, że liczba populacji wpływa znacznie na długość procesu zbiegania średniej wartości funkcji celu osobników w stronę „optimum”. Zbyt mała liczba populacji może nie wygenerować nam wystarczająco dużo różnych osobników, co może wpłynąć na kiepskie znajdywanie „optimum”. Większa liczba populacji pozwala nam na utworzenie wystarczająco dużo osobników, dzięki czemu mamy większe prawdopodobieństwo znalezienia rozwiązania „optimum”.

## Wpływ prawdopodobieństwa mutacji

Parametry:

Liczba populacji – 100

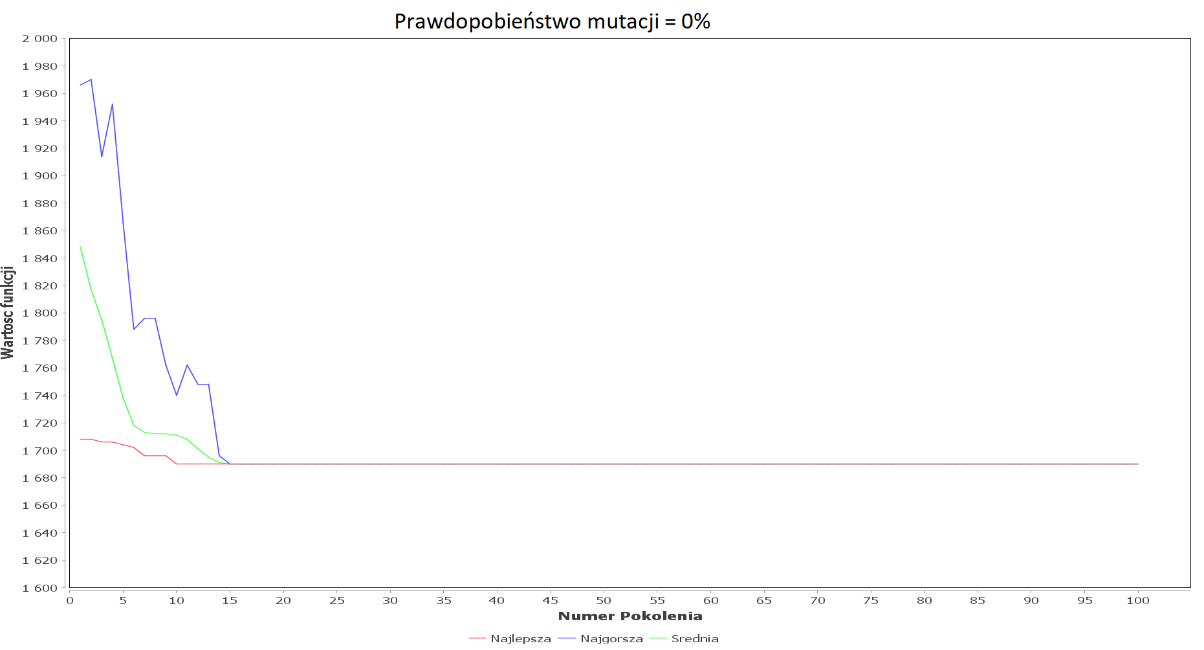
Rozmiar populacji – 100

Prawdopodobieństwo mutacji – na wykresie

Początkowy indeks do krzyżowania algorytmem PMX – 3

Końcowy indeks do krzyżowania algorytmem PMX – 6

N – liczba „fabryk” - 12

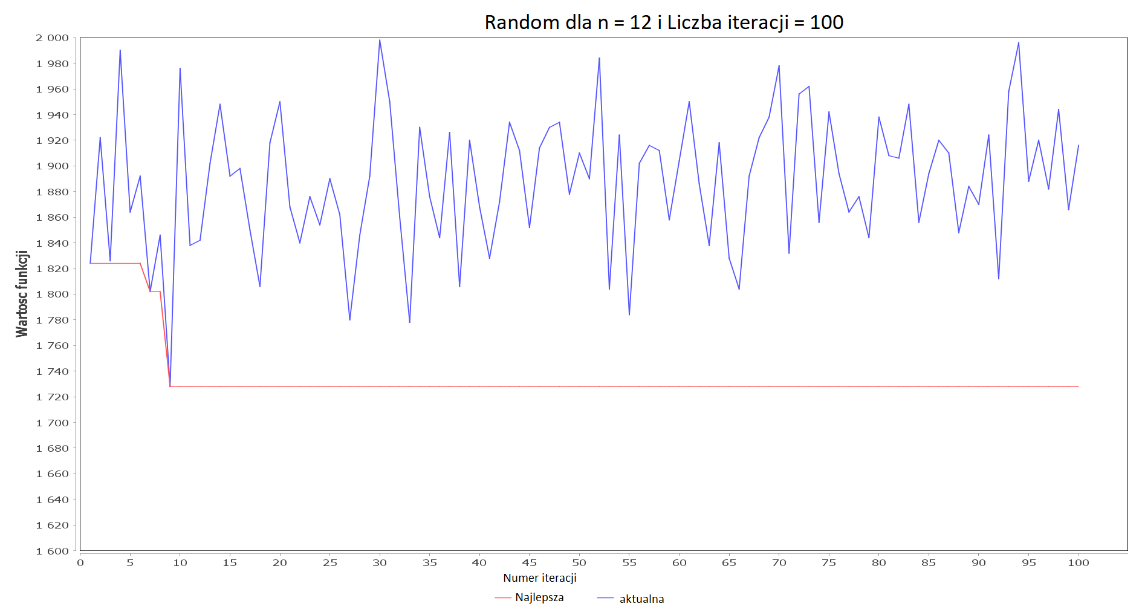


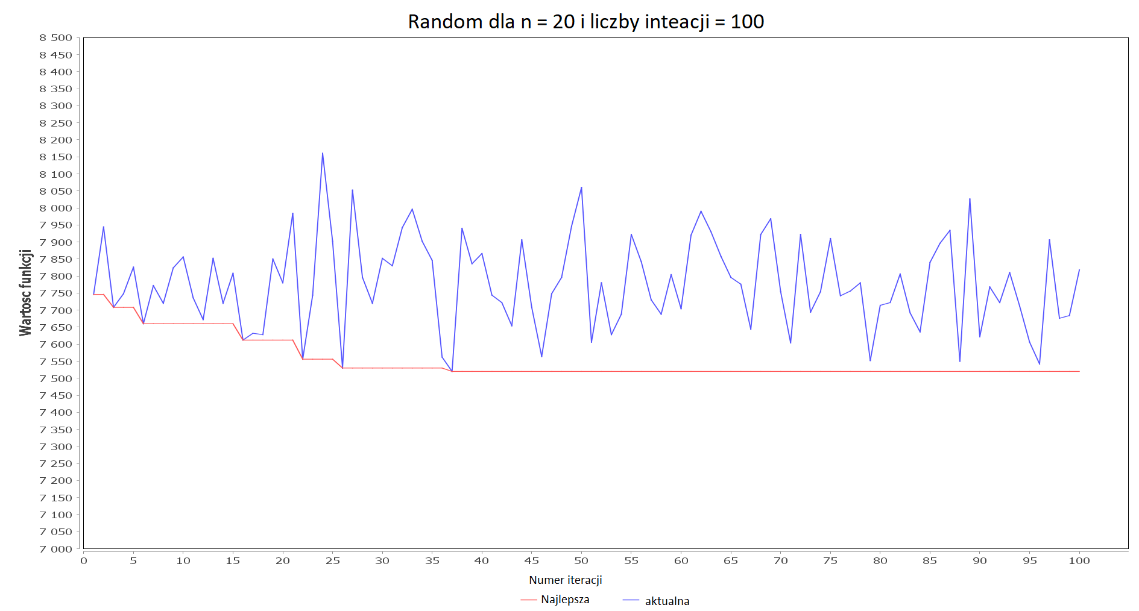
# 

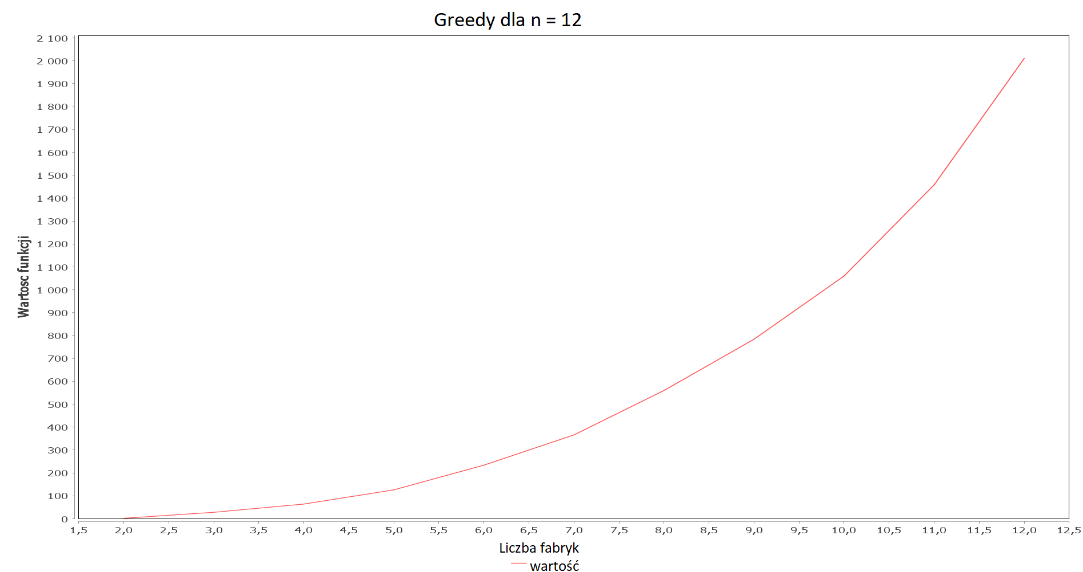
# 

Ustawiając prawdopodobieństwo mutacji na 0% powoduje, że po pewnym pokoleniu wszyscy osobnicy są tacy sami. Natomiast ustawienie mutacji coraz większej sprawia, że osobniki w populacji są generowani coraz bardziej losowo. Dla mutacji równej 100% każdy kolejny osobnik jest losowy ( oprócz tego z najlepszą wartością funkcji celu, ponieważ tak mam zaimplementowany algorytm ).

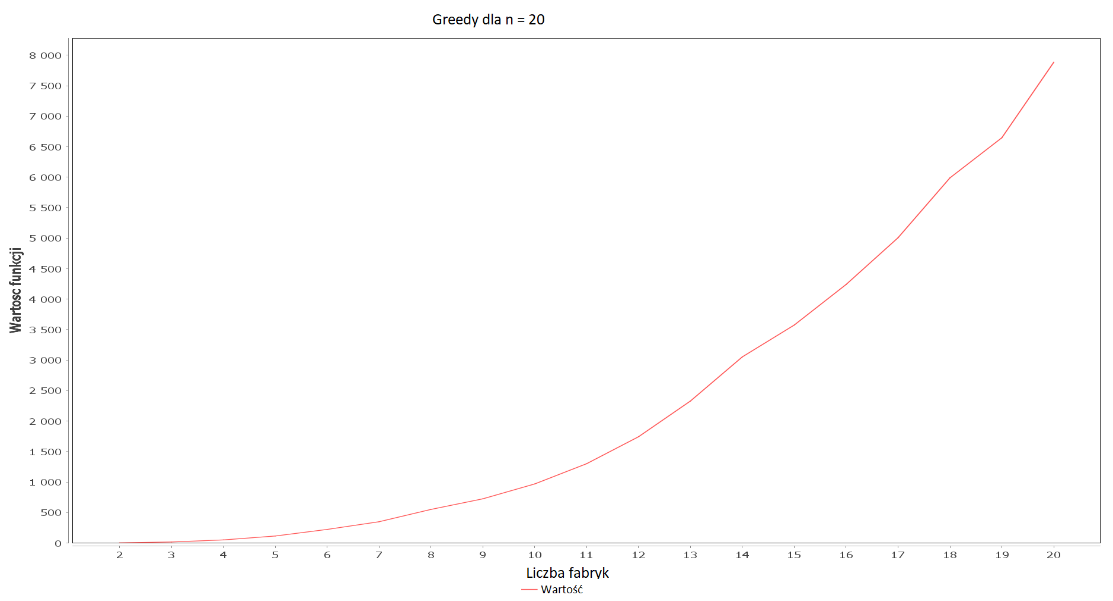
# Wyniki dla algorytmów Greedy oraz RandomSearch







Końcowy wynik = 2010



Końcowy wynik = 7884

Algorytm pierwszy ( Random ) generuje osobnika w sposób totalnie losowy, a drugi generuję na podstawie aktualnej wartości funkcji celu i dobiera po kolei fabryki aby koszt był jak najmniejszy. Jak widać algorytm losowy generuję lepsze wyniki niż greedy. Algorytm zachłanny zaimplementowałem w ten sposób, że znalazłem parę fabryk z najmniejszą wartością funkcji celu i dokładałem do nich kolejne fabryki w sposób analogiczny, dlatego mamy tylko 1 wynik.

# Wnioski

Algorytm genetyczny radzi sobie o wiele lepiej niż te 2 nieewolucyjne ( Random i Greedy ). Wyniki otrzymany przez algorytm losowy uzyskiwał zadowalające mnie wyniki i były one zdecydowanie lepsze, niż wyniki uzyskane przez algorytm zachłanny. Parametr który mogliśmy regulować w losowym był liczba prób a algorytm greedy był u mnie bezparametrowy.

W algorytmie GA można dojść do kilku wniosków na podstawie sprawdzonych parametrów.

Liczba Pokoleń powinna być ustawiona na odpowiednią ilość, ponieważ zbyt mała powoduje otrzymanie nie zadawalającego wyniku a zbyt duża powoduje, że po pewnym pokoleniu wyniki oscylują w podobnych wartościach.

Rozmiar populacji musi być na tyle wielki, żeby zagwarantować różnorodność osobników, przy czym nie może być on za duży, aby nie wydłużał nam naszego procesu znajdywania optimum.

Prawdopodobieństwo mutacji powinno być na takim poziomie aby nie osobniki nie utknęły w pewnej wartości funkcji celu, ale też aby ciągle ono nie występowało, żeby nie zachowywało się jak ciągłe losowanie wartości.