POLITECHNIKA WROCŁAWSKA

WYDZIAŁ ELEKTRONIKI

KIERUNEK: Informatyka

SPECJANOŚĆ: Inżynieria Systemów Informatycznych (INS)

Inteligencja Obliczeniowa i jej Zastosowania

Sprawozdanie z laboratorium 3 - 5

AUTORZY:

Piotr Chorościn

Dawid Mikowski

PROWADZĄCY:

Dr hab. inż. Olgierd Unold,

prof. uczelni

OCENA PRACY:

WROCŁAW 2020

Spis treści

[Spis ilustracji 2](#_Toc41858790)

[Cel ćwiczenia 3](#_Toc41858791)

[1 Badanie efektywności własnych funkcji 4](#_Toc41858792)

[1.1 badana funkcja 4](#_Toc41858793)

[1.2 Sposób przeprowadzenia badań 5](#_Toc41858794)

[1.3 Funkcja mutacji 6](#_Toc41858795)

[1.4 Funkcja krzyżowania 8](#_Toc41858796)

[1.5 Wnioski 10](#_Toc41858797)

[2 Problem komiwojażera 10](#_Toc41858798)

[2.1 Instancje 10](#_Toc41858799)

[2.2 Wpływ poziomu elityzmu 11](#_Toc41858800)

[2.3 Wpływ rozmiaru populacji 12](#_Toc41858801)

[2.4 Wpływ prawdopodobieństwa krzyżowania 13](#_Toc41858802)

[2.5 Pozostałe parametry 14](#_Toc41858803)

[3 Program genetyczny hybrydowy 16](#_Toc41858804)

[3.1 Wyniki metod optymalizacji 16](#_Toc41858805)

[3.2 Wnioski 18](#_Toc41858806)

[4 Literatura 19](#_Toc41858807)

# Spis ilustracji

[Rysunek 1 Wykres funkcji Schuberta 4](#_Toc41858764)

[Rysunek 2 Temperaturowy wykres funkcji Schuberta 5](#_Toc41858765)

[Rysunek 3 Kod własnej implementacji funkcji mutacji 6](#_Toc41858766)

[Rysunek 4 Implementacja listy badanych operatorów mutacji 6](#_Toc41858767)

[Rysunek 5 Różnice między najlepszą wartością funkcji celu a wartością globalnego optimum dla operatorów mutacji 7](#_Toc41858768)

[Rysunek 6 Różnice między średnią wartością funkcji celu a wartością globalnego optimum dla operatorów mutacji 7](#_Toc41858769)

[Rysunek 7 Kod własnej funkcji krzyżowania 8](#_Toc41858770)

[Rysunek 8 Implementacja listy badanych operatorów krzyżowania 8](#_Toc41858771)

[Rysunek 9 Różnice między najlepszą wartością funkcji celu a wartością globalnego optimum dla operatorów krzyżowania 9](#_Toc41858772)

[Rysunek 10 Różnice między średnią wartością funkcji celu a wartością globalnego optimum dla operatorów krzyżowania 9](#_Toc41858773)

[Rysunek 11 Parametr poziomu elityzmu dla problemu Dantzig42.tsp 11](#_Toc41858774)

[Rysunek 12 Parametr poziomu elityzmu dla problemu Brazil58.tsp 11](#_Toc41858775)

[Rysunek 13 Parametr rozmiaru populacji dla problemu Dantzig42.tsp 12](#_Toc41858776)

[Rysunek 14 Parametr rozmiaru populacji dla problemu Brazil58.tsp 12](#_Toc41858777)

[Rysunek 15 Parametr prawdopodobieństwa krzyżowania dla problemu Dantzig42.tsp 13](#_Toc41858778)

[Rysunek 16 Parametr prawdopodobieństwa krzyżowania dla problemu Brazil58.tsp 13](#_Toc41858779)

[Rysunek 17 Parametr liczby pokoleń dla problemu Dantzig42.tsp 14](#_Toc41858780)

[Rysunek 18 Parametr prawdopodobieństwa mutacji dla problemu Dantzig42.tsp 14](#_Toc41858781)

[Rysunek 19 Parametr liczby pokoleń dla problemu Brazil58.tsp 15](#_Toc41858782)

[Rysunek 20 Parametr prawdopodobieństwa mutacji dla problemu Brazil58.tsp 15](#_Toc41858783)

[Rysunek 21 Algorytm genetyczny niehybrydowy 16](#_Toc41858784)

[Rysunek 22Metoda BFGS 16](#_Toc41858785)

[Rysunek 23Metoda CG 17](#_Toc41858786)

[Rysunek 24Metoda L-BFGS-B 17](#_Toc41858787)

[Rysunek 25Metoda Nelder-Mead 18](#_Toc41858788)

[Rysunek 26Metoda SANN 18](#_Toc41858789)

# Cel ćwiczenia

Celem ćwiczenia jest:

* sprawdzenie efektywności funkcji wielomodalnych z zaimplementowanymi własnymi funkcjami krzyżowania oraz mutacji a następnie porównanie jakości działania algorytmu z funkcjami domyślnymi,
* wykonanie wcześniejszych zadań dla 3 wybranych parametrów dla problemu komiwojażera,
* wykonanie z wykorzystaniem programu genetycznego hybrydowego zadania rozwiązania problemu komiwojażera,
* analiza i ocena wyników badań

# Badanie efektywności własnych funkcji

## badana funkcja

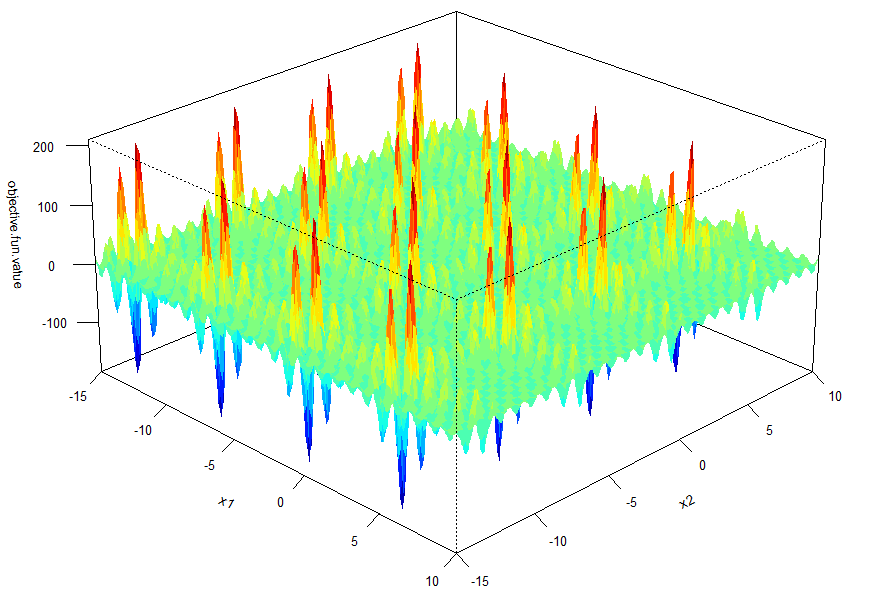
Do testowania efektywności własnych funkcji krzyżowania i mutacji potrzebne było wybranie jednej funkcji wielomodalnej na której sprawdzane będą wyniki. Została do tego wyznaczona funkcja, która była testowana także w poprzednim laboratorium a mianowicie funkcja Schuberta.

Funkcja Schuberta określona jest poniższym wzorem [1]:

Wykres funkcji Schuberta, wygenerowany dla następujących wartości zmiennych:

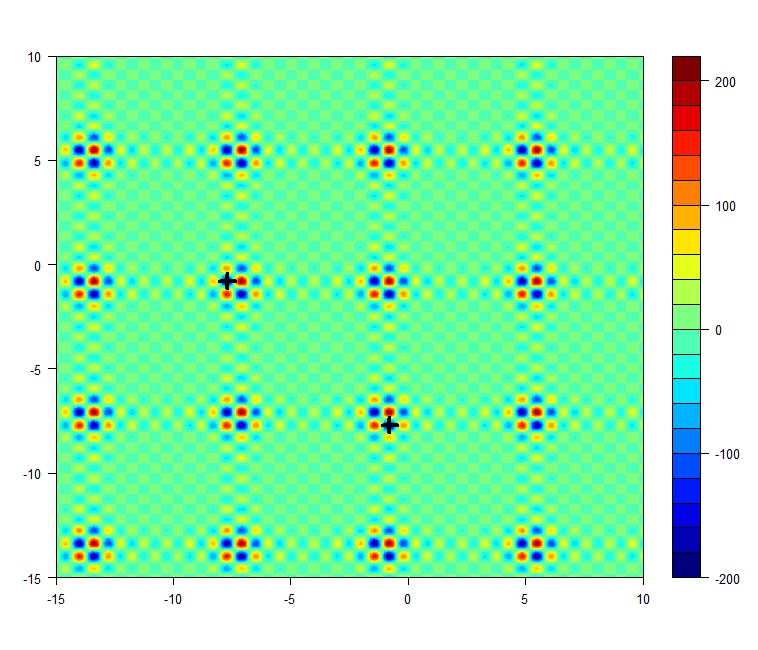
* z krokiem próbkowania 0,1,
* z krokiem próbkowania 0,1,

Są to domyślne zakresy zmiennych z pakietu globalOptTest dla funkcji Schuberta.



Rysunek Wykres funkcji Schuberta

Powyższy trójwymiarowy wykres przedstawiony na płaszczyźnie poprzez rzut z góry przedstawiono na poniższym rysunku.



Rysunek Temperaturowy wykres funkcji Schuberta

Zgodnie z informacjami z pakietu GlobalOptTest, funkcja Schuberta w minimum globalnym przyjmuje wartość **.**

## Sposób przeprowadzenia badań

Dla funkcji Schuberta przyjęto zakres wartości zmiennych odczytany za pomocą funkcji *getDefaultBounds* z pakietu globalOptTests oraz wartość ekstremum globalnego odczytaną za pomocą funkcji getGlobalOpt. Funkcję poddano optymalizacji globalnej stosując gotowy algorytm genetyczny zaimplementowany w pakiecie do języka R o nazwie „GA”, który umożliwia również hybrydyzację. Podczas badania algorytmu hybrydowego zostały ustawione domyślne wartości wszystkich parametrów. Aby zbadać wpływ danego operatora, wszystkie operatory poza badanym miały ustawioną stałą wartość domyślną. W poniższej tabeli zawarte są wszystkie badane parametry oraz wartości domyślne.

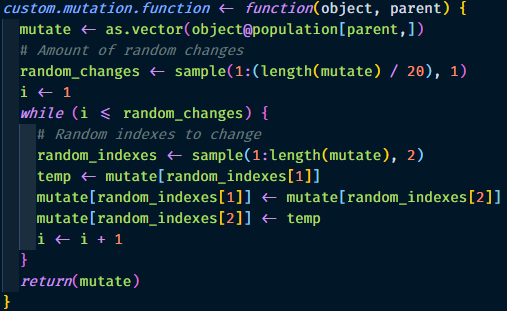
|  |  |  |
| --- | --- | --- |
| Operator | Badane parametry | Wartość domyślna |
| Mutacji | ra, nra, rs, pow | ra |
| Krzyżowania | sp, wa, la, blx, laplace | la |

Domyślne wartości dla parametrów algorytmu genetycznego prezentują się następująco:

|  |  |
| --- | --- |
| Type | Real-valued |
| Population size | 50 |
| Number of generations | 100 |
| Elitism | 2 |
| Crossover possibility | 0.8 |
| Mutation propability | 0.1 |

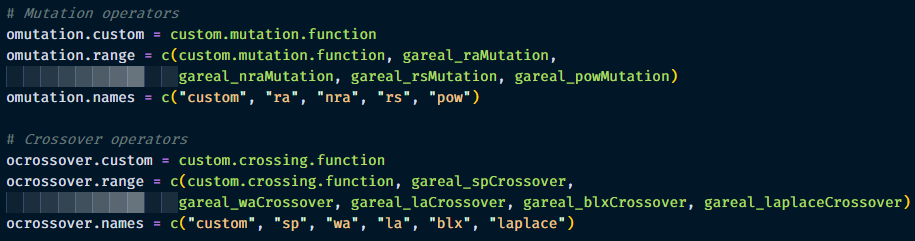
## Funkcja mutacji

Własna implementacja funkcji krzyżowania:

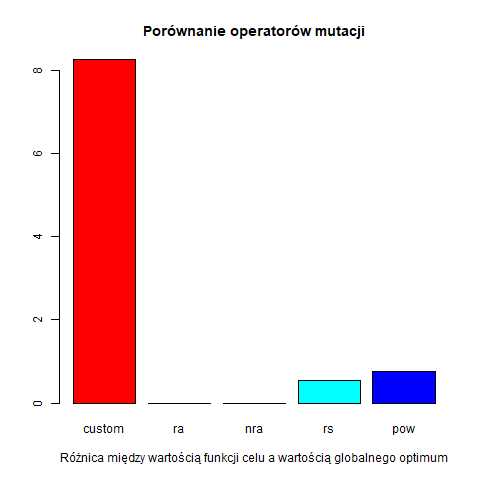


Rysunek Kod własnej implementacji funkcji mutacji

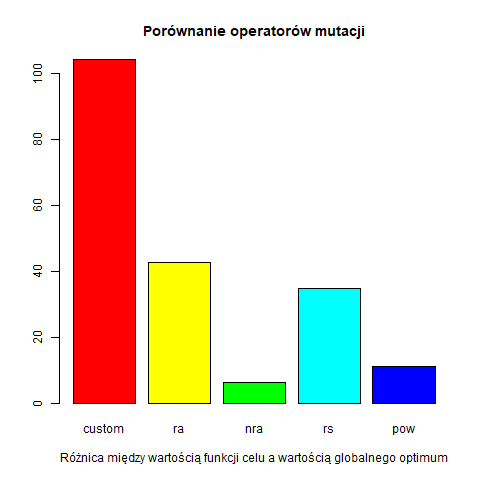
Aby poprawnie wykonać badania należało także przekazać własną funkcję z innymi jako parametr mutation funkcji ga(). Wszystkie potrzebne operatory umieszczono w liście w następujący sposób:



Rysunek Implementacja listy badanych operatorów mutacji



Rysunek Różnice między najlepszą wartością funkcji celu a wartością globalnego optimum dla operatorów mutacji

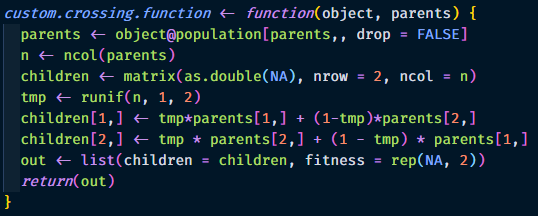


Rysunek Różnice między średnią wartością funkcji celu a wartością globalnego optimum dla operatorów mutacji

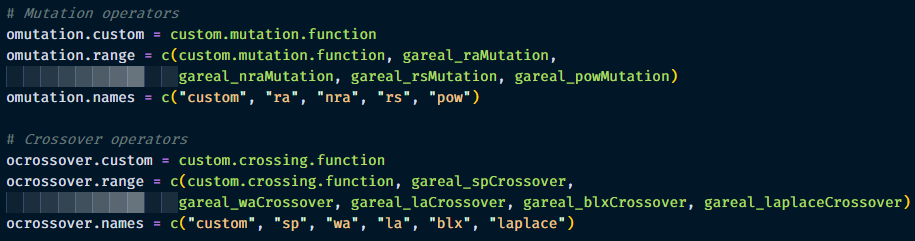
Jak pokazują powyższe wyniki własny operator znacznie odbiega efektywnością od pozostałych wbudowanych operatorów zarówno w przypadku najlepszego wyniku jak i średnich wyników. Najlepsze okazały się operatory „ra” i „nra”, z których operator ra jest domyślnie używanym operatorem mutacji w przypadku typu „Real value” osiągnęły one prawidłową wartość poszukiwanego optimum globalnego. Wyniki średnich wartości pokazują, że ogółem najlepszym operatorem okazał się „nra”, dla którego średnia wartość całej populacji nieznacznie odbiegała od prawidłowego wyniku. Nie uwzględniając własnej funkcji mutacji można zauważyć, że najgorszym operatorem w tym przypadku okazał się operator „pow”, jednak różnica między jego najlepszym wynikiem a „rs” jest nieznaczna.

## Funkcja krzyżowania

Własna implementacja funkcji mutacji wygląda następująco:

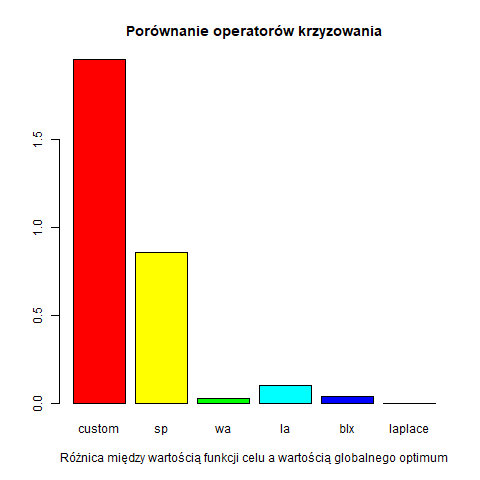


Rysunek Kod własnej funkcji krzyżowania

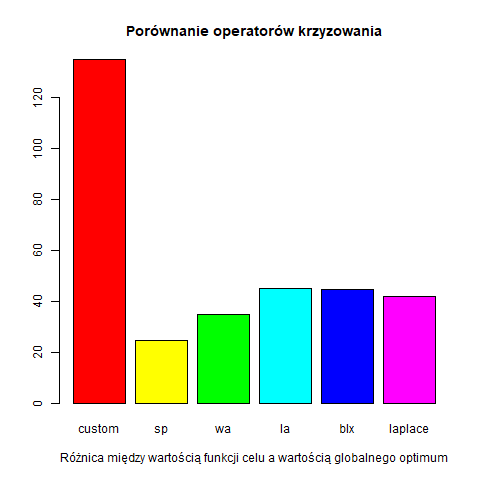


Rysunek Implementacja listy badanych operatorów krzyżowania

Wykonane zostało porównanie jakości wyników dla poszczególnych parametrów mutacji przy zachowaniu domyślnych parametrów algorytmu genetycznego:



Rysunek Różnice między najlepszą wartością funkcji celu a wartością globalnego optimum dla operatorów krzyżowania



Rysunek Różnice między średnią wartością funkcji celu a wartością globalnego optimum dla operatorów krzyżowania

Wyniki operatorów krzyżowania również pokazują, że funkcja własna „custom” nie sprawdza się dobrze w porównaniu z innymi operatorami. Z pozostałych funkcji najgorzej wypadła „sp” a najlepiej „Laplace” której udało się odnaleźć prawidłowe optimum. Jeżeli chodzi o wyniki średnie to operator „sp”, który był najdalej od znalezienia najlepszego rozwiązania sprawdził się najlepiej, jednak nawet on nie zbliżył się do osiągnięcia wyniku operatora mutacji „nra”, może to być spowodowane wykorzystywanym domyślnie operatorem mutacji „ra” który także nie posiadał dobrej efektywności, jeżeli chodzi o wartości średnie.

## Wnioski

Wyniki samodzielnie napisanych operatorów pokazują, że nie nadają się one do wykorzystania, jeżeli chodzi nam o dokładne znalezienie rozwiązania. W najlepszym przypadku dają one wyniki nawet ośmiokrotnie bardziej oddalone od optimum niż najmniej efektywna z funkcji znajdujących się już w pakiecie GA. Z pozostałych badanych operatorów przynajmniej jednemu zawsze udawało się odnaleźć prawidłowe rozwiązanie dzięki czemu można wywnioskować, że testując przynajmniej je uda nam się odnaleźć poszukiwany wynik.

# Problem komiwojażera

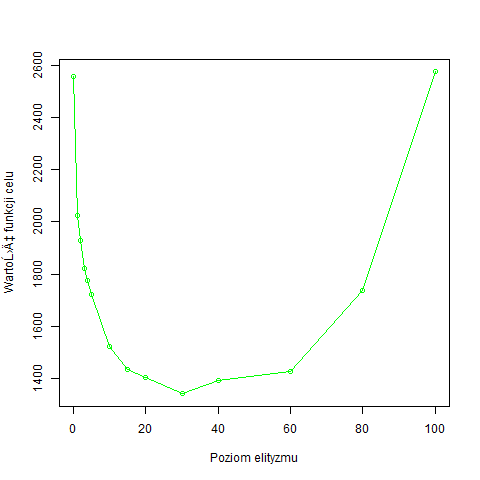
## Instancje

Do badania algorytmu genetycznego dla problemu komiwojażera zdecydowano się wykorzystać dwie instancje ze zbioru TSPlib.

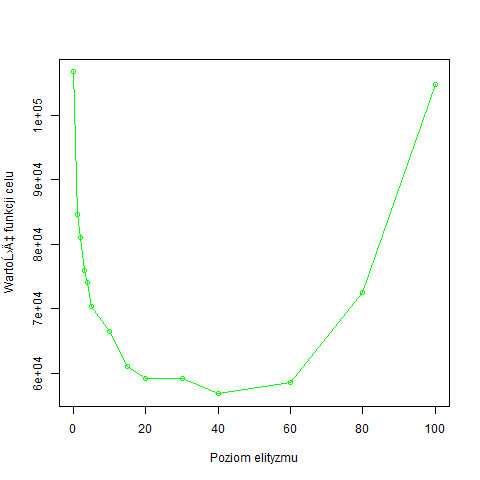
|  |  |  |
| --- | --- | --- |
| Nazwa instancji | Liczba miast | Długość najkrótszej trasy |
| Brazil58.tsp | 58 | 25395 |
| Dantzig42.tsp | 42 | 699 |

Wyniki algorytmu genetycznego zostały przeprowadzone na dwóch instancjach z których każda ma inną liczbę wierzchołków i przedstawia wersję symetryczną badanego problemu. Instancje zostały wybrane biorąc pod uwagę czasochłonność wykonywanych obliczeń. Na wykresy naniesiono najkrótsze trasy dla każdej instancji, aby można było w prosty graficzny sposób uzyskać odniesienie co do jakości uzyskanych rozwiązań. Parametry które zostaną szczególnie opisane zostały wybrane na podstawie zauważonych zależności między ich wartością a jakością rozwiązania.

## Wpływ poziomu elityzmu



Rysunek Parametr poziomu elityzmu dla problemu Dantzig42.tsp



Rysunek Parametr poziomu elityzmu dla problemu Brazil58.tsp

Badania parametru poziomu elityzmu podczas rozwiązywania problemu komiwojażera na obu instancjach pokazują bardzo podobne wyniki. Wartości skrajne dla poziomu elityzmu dają znacznie gorsze wyniki niż wartości znajdujące się w środku. Dla instancji Dantzig.tsp najlepiej sprawdziła się wartość 30 a dla Brazil58.tsp wartość 40. Dzięki wykresom można wywnioskować, że przy wartościach równych 0 występuje zjawisko gubienia najlepszego osobnika. Przy wartościach zbyt dużych natomiast powoduje ciągłe znaczące pogorszenie wartości funkcji celu co skutkuje gorszymi wynikami.

## Wpływ rozmiaru populacji



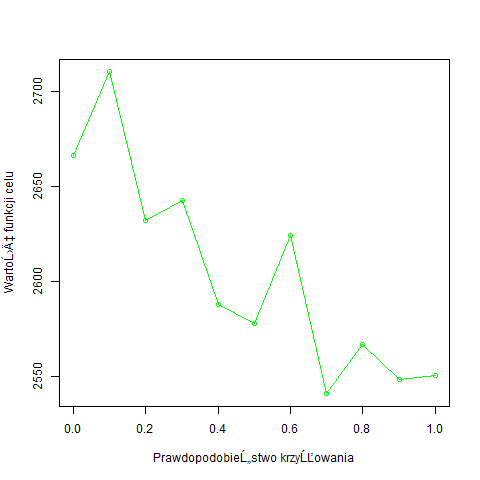
Rysunek Parametr rozmiaru populacji dla problemu Dantzig42.tsp



Rysunek Parametr rozmiaru populacji dla problemu Brazil58.tsp

Zależność rozmiaru populacji co do efektywności algorytmu widać na obu wykresach wynikowych, pokazują one, że im większy jest parametr tym krótszą ścieżkę potrafi on znaleźć. Może to być spowodowane tym że z większej populacji istnieje większe prawdopodobieństwo na znalezienie najlepszego osobnika, który w późniejszych iteracjach będzie przekazywany aż do ostatniej generacji.

## Wpływ prawdopodobieństwa krzyżowania



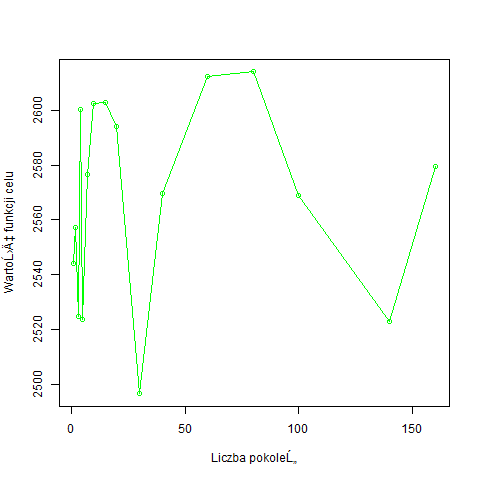
Rysunek Parametr prawdopodobieństwa krzyżowania dla problemu Dantzig42.tsp



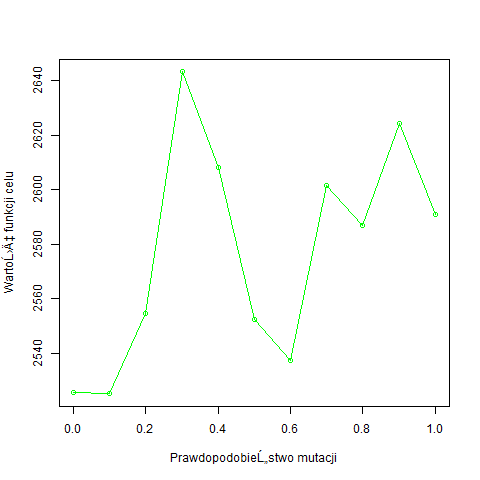
Rysunek Parametr prawdopodobieństwa krzyżowania dla problemu Brazil58.tsp

Parametr prawdopodobieństwa krzyżowania powoduje zwiększenie jakości najlepszego osobnika od wartości 0 do wartości około 0.7 następnie następuje odwrócona sytuacja gdzie zwiększanie parametru pogarsza wynik funkcji celu.

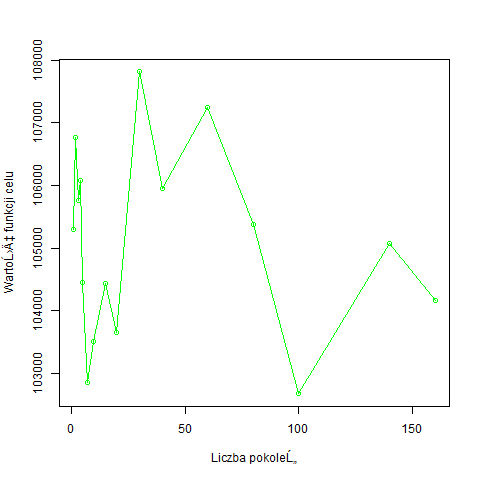
## Pozostałe parametry



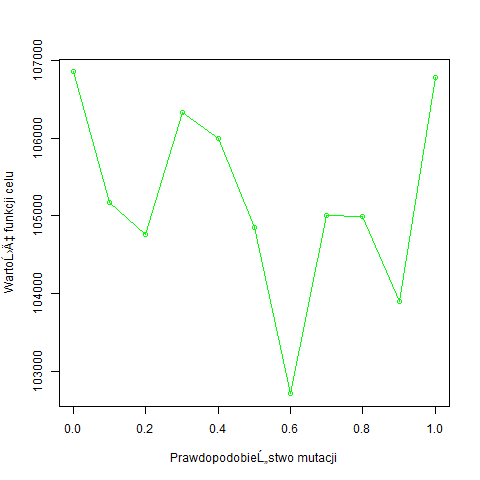
Rysunek Parametr liczby pokoleń dla problemu Dantzig42.tsp



Rysunek Parametr prawdopodobieństwa mutacji dla problemu Dantzig42.tsp



Rysunek Parametr liczby pokoleń dla problemu Brazil58.tsp

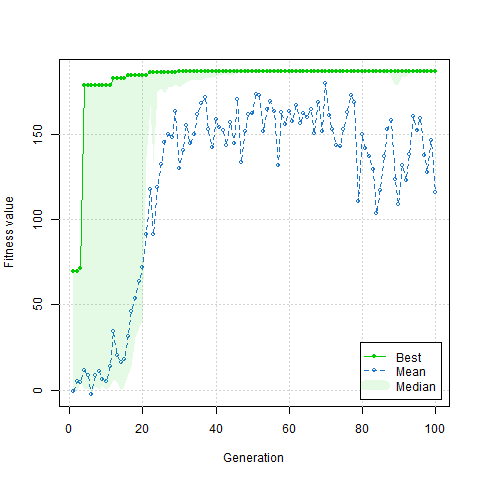


Rysunek Parametr prawdopodobieństwa mutacji dla problemu Brazil58.tsp

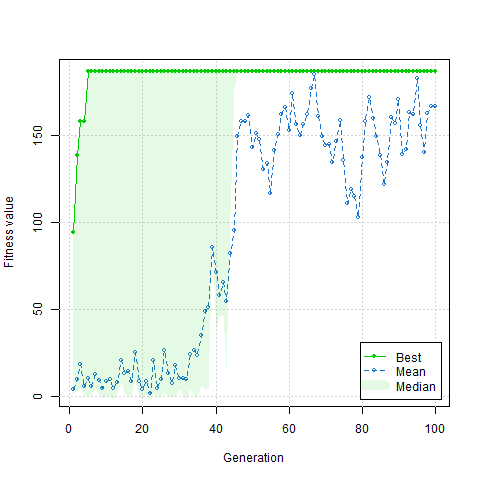
Pozostałe parametry wykazywały wyniki, na których nie dało się zauważyć zależności. Wygląd wykresów pozwala na wnioskowanie, że wyniki zależą bardziej od losowości a nie od wartości tychże parametrów.

# Program genetyczny hybrydowy

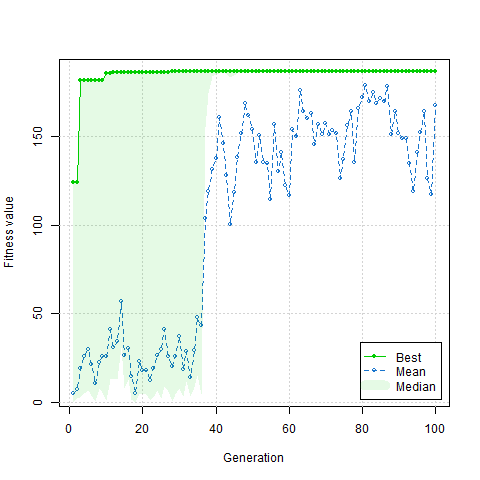
## Wyniki metod optymalizacji



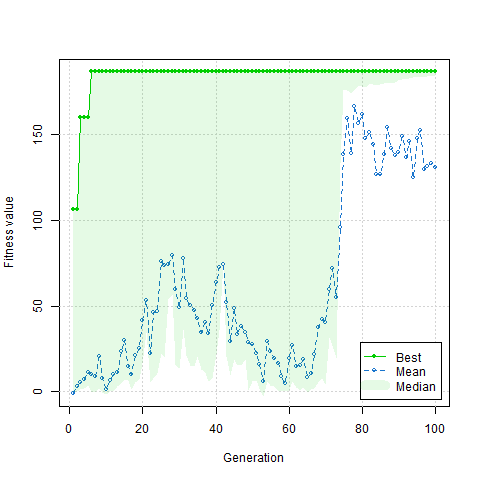
Rysunek Algorytm genetyczny niehybrydowy



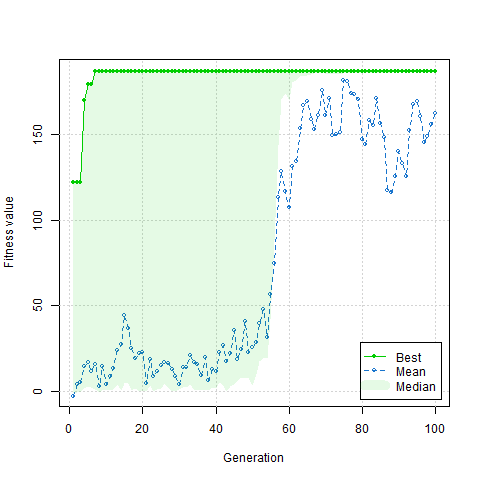
Rysunek Metoda BFGS



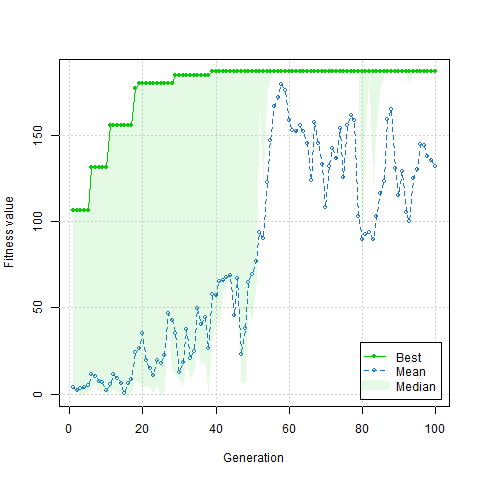
Rysunek Metoda CG



Rysunek Metoda L-BFGS-B



Rysunek Metoda Nelder-Mead



Rysunek Metoda SANN

## Wnioski

Porównując wyniki wszystkich metod hybrydowych algorytmów genetycznych widać, że poza metodą SANN przyśpieszają one znalezienie optimum globalnego dla najlepszej jednostki z populacji. Pomimo tego powodują one dużo wolniejsze zbliżanie się do poprawnego wyniku reszty populacji co można zobaczyć patrząc na linie „Mean” zawierającą wyniki średnie dla całej populacji. Najlepiej sprawdziła się metoda BFGS która potrafiła znaleźć optimum globalne już w 5 generacji. Najgorsze wyniki wykazała metoda SANN, która jak już wcześniej wspomniano nie przyśpieszyła a spowolniła znalezienie optimum o około 9 generacji.

# Literatura

[1] <http://infinity77.net/global_optimization/test_functions_nd_H.html>

[2] <https://www.sfu.ca/~ssurjano/shubert.html>

[3] <https://pl.wikipedia.org/wiki/Algorytm_genetyczny>

[4] <https://cran.r-project.org/web/packages/GA/vignettes/GA.html>

[5] <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html>

[6] <https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)>

[7] <https://sites.google.com/site/gotestfunctions/multimodal-function-list>

[8] <https://arxiv.org/pdf/1605.01931.pdf>