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 3](#_Toc41468891)

[Spis tabel 4](#_Toc41468892)

[Spis listingów 4](#_Toc41468893)

[1 Cel ćwiczenia 5](#_Toc41468894)

[2 Badanie efektywności własnych funkcji 6](#_Toc41468895)

[2.1 Funkcja Schuberta 6](#_Toc41468896)

[2.2 Funkcja Hosaki 8](#_Toc41468897)

[3 Problem komiwojażera 10](#_Toc41468898)

[4 Program genetyczny hybrydowy 11](#_Toc41468899)

[5 Tabele i wykresy z doświadczeń 11](#_Toc41468900)

[5.1 Badanie wpływu rozmiaru populacji 13](#_Toc41468901)

[5.1.1 Badanie wpływu rozmiaru populacji na przystosowanie dla funkcji Schuberta 13](#_Toc41468902)

[5.1.2 Badanie wpływu rozmiaru populacji na przystosowanie dla funkcji Hosaki 13](#_Toc41468903)

[5.1.3 Obserwacje nt. wpływu liczności populacji na jej średnie przystosowanie oraz przystosowanie najlepszego osobnika 14](#_Toc41468904)

[5.2 Badanie wpływu liczby pokoleń 15](#_Toc41468905)

[5.2.1 Badanie wpływu liczby pokoleń na przystosowanie dla funkcji Schuberta 15](#_Toc41468906)

[5.2.2 Badanie wpływu liczby pokoleń na przystosowanie dla funkcji Hosaki 16](#_Toc41468907)

[5.2.3 Obserwacje nt. wpływu liczby pokoleń na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika 17](#_Toc41468908)

[5.3 Badanie wpływu prawdopodobieństwa mutacji 18](#_Toc41468909)

[5.3.1 Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Schuberta 18](#_Toc41468910)

[5.3.2 Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Hosaki 19](#_Toc41468911)

[5.3.3 Obserwacje nt. wpływu prawdopodobieństwa mutacji na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika 20](#_Toc41468912)

[5.4 Badanie wpływu prawdopodobieństwa krzyżowania 21](#_Toc41468913)

[5.4.1 Badanie wpływu prawdopodobieństwa krzyżowania na wartość funkcji Schuberta 21](#_Toc41468914)

[5.4.2 Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Hosaki 22](#_Toc41468915)

[5.4.3 Obserwacje nt. wpływu prawdopodobieństwa krzyżowania na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika 23](#_Toc41468916)

[5.5 Badanie wpływu prawdopodobieństwa krzyżowania i mutacji 24](#_Toc41468917)

[5.6 Badanie wpływu poziomu elitaryzmu 26](#_Toc41468918)

[5.6.1 Badanie wpływu elitaryzmu na przystosowanie do funkcji Schuberta 26](#_Toc41468919)

[5.6.2 Badanie wpływu elitaryzmu na przystosowanie do funkcji Hosaki 26](#_Toc41468920)

[5.6.3 Obserwacje nt. wpływu elitaryzmu na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika 27](#_Toc41468921)

[6 Wnioski 28](#_Toc41468922)

[6.1 Wnioski nt. własnych implementacji 28](#_Toc41468923)

[6.2 Wnioski nt. implementacji i języka R 29](#_Toc41468924)

[7 Kod z komentarzem 30](#_Toc41468925)

[7.1 Przegląd najważniejszych fragmentów kodu 30](#_Toc41468926)

[7.2 Pełny kod programu 33](#_Toc41468927)

[8 Literatura 37](#_Toc41468928)

# Spis ilustracji

[Rysunek 6.1Przebieg zmienności wartości funkcji dopasowania najlepszego osobnika i średniej dla całej populacji w zależności od pokolenia dla przykładowego uruchomienia algorytmu genetycznego dla funkcji Hosaki. 12](#_Toc41001908)

[Rysunek 6.2 Przebieg dla innego uruchomienia (też dla funkcji Hosaki). 12](#_Toc41001909)

[Rysunek 6.3 Zależność wartości funkcji celu Schuberta od liczebności populacji 13](#_Toc41001910)

[Rysunek 6.4 Zależność wartości funkcji celu Hosaki od liczebności populacji 13](#_Toc41001911)

[Rysunek 6.5 Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla calej populacji w zależności od liczby pokoleń 15](#_Toc41001912)

[Rysunek 6.6 Powiększenie wykresu dla najlepszego osobnika 15](#_Toc41001913)

[Rysunek 6.7 Powiększenie wykresu dla średniej z całej populacji 15](#_Toc41001914)

[Rysunek 6.8 Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla calej populacji w zależności od liczby pokoleń 16](#_Toc41001915)

[Rysunek 6.9 Powiększenie wykresu dla najlepszego osobnika 16](#_Toc41001916)

[Rysunek 6.10 Powiększenie wykresu dla średniej z całej populacji 16](#_Toc41001917)

[Rysunek 6.11 Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa mutacji. 18](#_Toc41001918)

[Rysunek 6.12 Powiększenie wykresu dla najlepszego osobnika 18](#_Toc41001919)

[Rysunek 6.13 Powiększenie wykresu dla średniej z całej populacji 18](#_Toc41001920)

[Rysunek 6.14 Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa mutacji. 19](#_Toc41001921)

[Rysunek 6.15 Powiększenie wykresu dla najlepszego osobnika 19](#_Toc41001922)

[Rysunek 6.16 Powiększenie wykresu dla średniej z całej populacji 19](#_Toc41001923)

[Rysunek 6.17 Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa krzyżowania 21](#_Toc41001924)

[Rysunek 6.18 Powiększenie wykresu dla najlepszego osobnika 21](#_Toc41001925)

[Rysunek 6.19 Powiększenie wykresu dla średniej z całej populacji 22](#_Toc41001926)

[Rysunek 6.20 Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa krzyżowania 22](#_Toc41001927)

[Rysunek 6.21 Powiększenie wykresu dla najlepszego osobnika 23](#_Toc41001928)

[Rysunek 6.22 Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od wartości parametru „elitism” 26](#_Toc41001929)

[Rysunek 6.23 Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od wartości parametru „elitism” 26](#_Toc41001930)

[Rysunek 6.24 Powiększenie wykresu dla najlepszego osobnika. 27](#_Toc41001931)

# Spis tabel

[Tabela 6.1 Wartości parametrów i zmiennych przyjęte w badaniach 11](#_Toc41004859)

[Tabela 6.2 Poszczególne osobniki populacji w 1. i 60. pokoleniu w zależności od wartości prawdopodobieństw: mutacji i krzyżowania 24](#_Toc41004860)

# Spis listingów

[Listing 8.1 Kod funkcji objective.fun.of 30](#_Toc41004929)

[Listing 8.2 Kod funkcji objective.fun.get 30](#_Toc41004930)

[Listing 8.3 Kod funkcji objective.fun.plot 31](#_Toc41004931)

[Listing 8.4 Kod funkcji GA.run.iterations 31](#_Toc41004932)

[Listing 8.5 Kod funkcji GA.run.experiment 31](#_Toc41004933)

[Listing 8.6 Kod funkcji GA.run.experiment.list 32](#_Toc41004934)

[Listing 8.7 Wywołanie funkcji GA.run.experiment 32](#_Toc41004935)

[Listing 8.8 Funkcja GA.run.once 33](#_Toc41004936)

[Listing 8.9 Skrypt global\_opt.R 33](#_Toc41004937)

[Listing 8.10 Skrypt global\_opt.R 35](#_Toc41004938)

[Listing 8.11 Skrypt main.R 36](#_Toc41004939)

# Cel ćwiczenia

Celem ćwiczenia jest:

* sprawdzenie efektywności funkcji wielomodalnych z zaimplementowanymi własnymi funkcjami krzyżowania, mutacji oraz selekcji 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.

# Badanie efektywności własnych funkcji

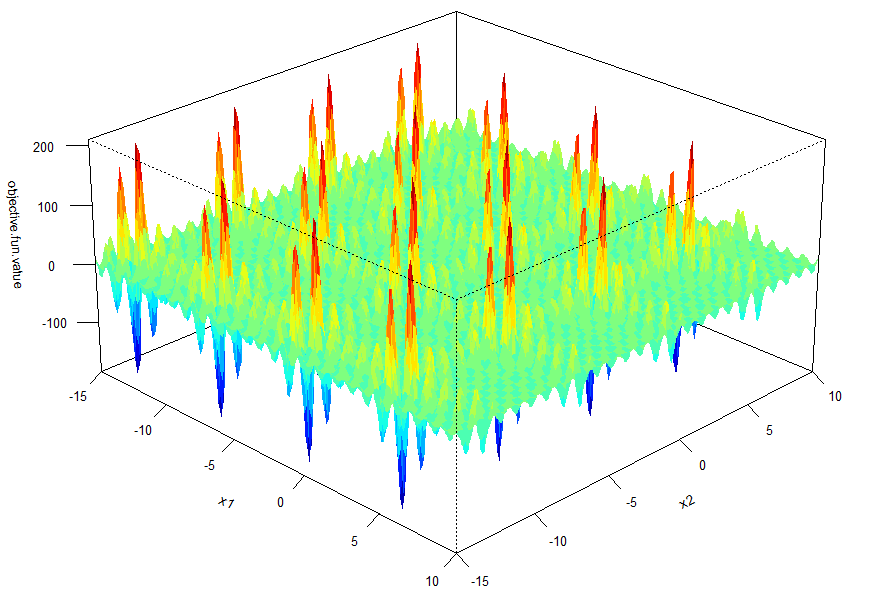
## Funkcja Schuberta

Funkcja Schuberta określona jest poniższym wzorem [1]:

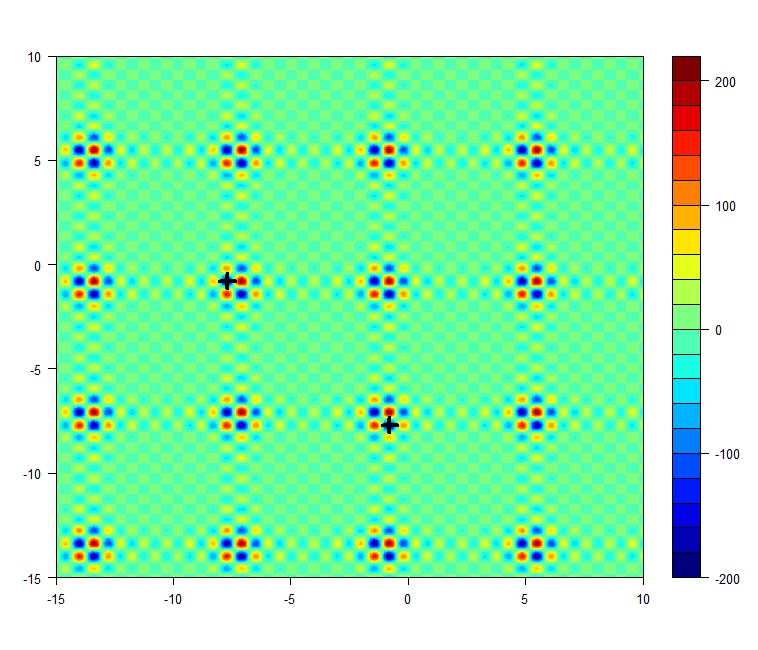
Wykres funkcji Schuberta, wygenerowany dla następujących wartości zmiennych:

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

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



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



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

W rozpatrywanym zakresie znaleziono 2 minima globalne, w których funkcja celu przyjmuje tę wartość. Są to (z dokładnością do 0,1):

Ekstrema globalne oznaczono na rysunku czarnymi krzyżykami.

Ekstrema globalne uzyskano w tym przypadku za pomocą pakietu GlobalOptTest. Podejście takie odpowiada przeglądowi zupełnemu, w którym dla każdej wartości zmiennych obliczana jest wartość funkcji. Dlatego jednym z celów ćwiczenia jest zastosowanie (w późniejszym etapie) algorytmu genetycznego, który pozwoli operować jedynie na populacji o ograniczonym rozmiarze.

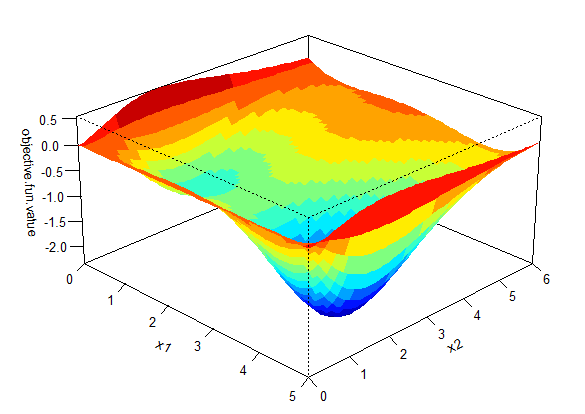
## Funkcja Hosaki

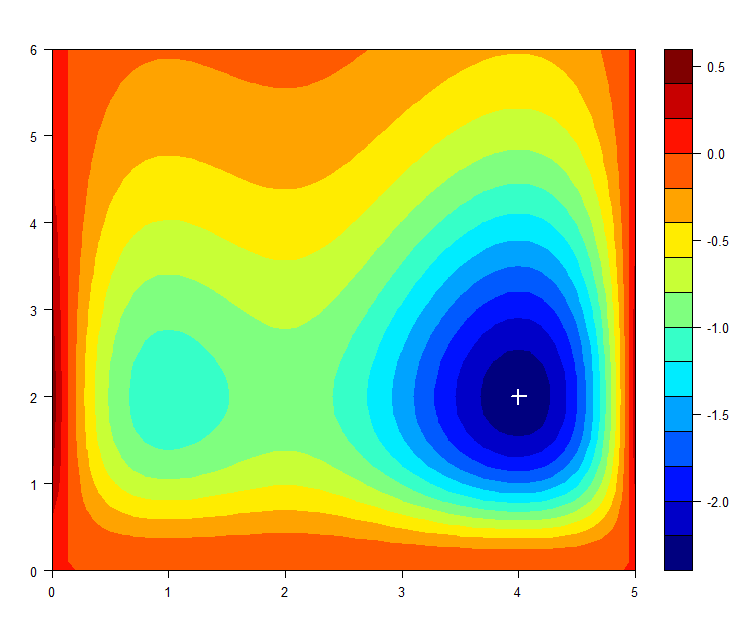
Funkcja Hosaki określona jest poniższym wzorem [2]:

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 globalOptTests dla funkcji Hosaki.





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

**dla** (z dokładnością do 0,1).

Ekstremum globalne oznaczono na rysunku białym krzyżykiem.

# Problem komiwojażera

# Program genetyczny hybrydowy

Wykorzystano następujące paczki języka R:

* GA – zawierająca implementację algorytmu genetycznego,
* GlobalOptTest – zawierająca funkcje celu wraz z wartościami ich ekstremów.
* TSPlib – zawierająca funkcje pozwalające na wczytanie oraz rozwiązanie plików związanych z problemem komiwojażera

Środowisko implementacyjne:

* Interpreter języka R w wersji 3.6.1 z pakietu Anaconda,
* Środowisko VS code z wtyczką do języka R

# Tabele i wykresy z doświadczeń

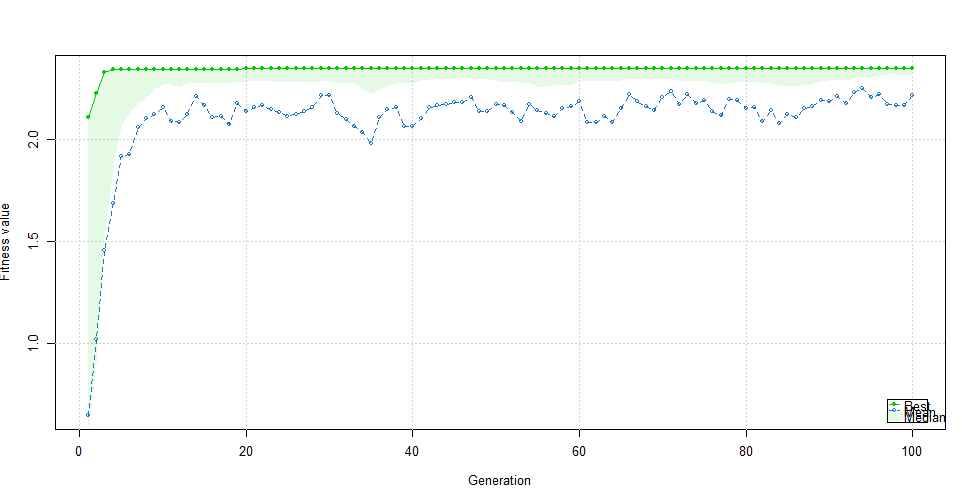
Badania przeprowadzono dla algorytmu genetycznego przy różnych wartościach parametrów. Domyślnie parametry badano niezależnie, tzn. w każdym badaniu zmianom podlegała wartość tylko jednego parametru, inne zaś pozostawały stałe (chyba że w opisie badania zaznaczono inaczej).

Wartości badanych parametrów zamieszczono w tabeli. Kolumna const odpowiada wartości parametru w eksperymentach, gdzie był on stały, a var w eksperymentach gdzie podlegał zmianom (był zmienną).

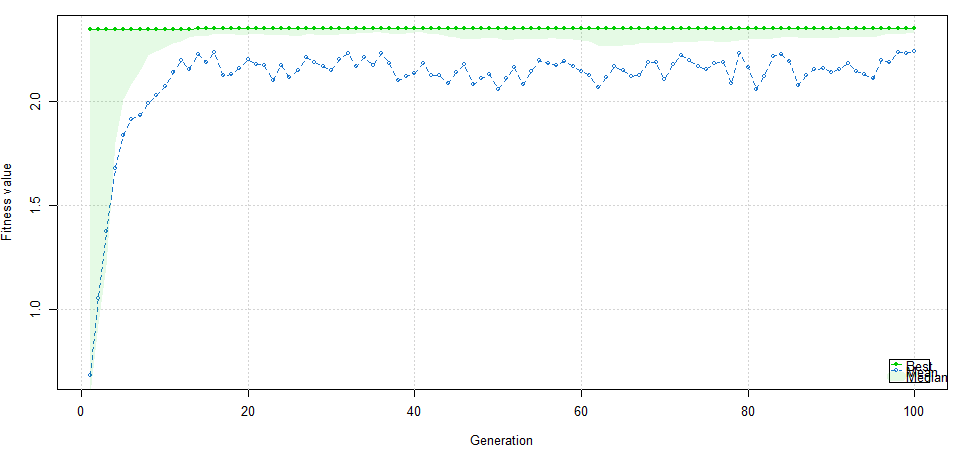
Tabela . Wartości parametrów i zmiennych przyjęte w badaniach

|  |  |  |
| --- | --- | --- |
| Nazwa parametru | Const | Var |
| Liczebność populacji | 100 | 1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160 |
| Liczba pokoleń | 100 | 1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160 |
| Elitaryzm | 1 | 0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 60, 80, 100 |
| Prawdopodobieństwo mutowania | 0,8 | [0,0; 1,0] z krokiem 0,05 |
| Prawdopodobieństwo krzyżowania | 0,2 | [0,0; 1,0] z krokiem 0,05 |

Wyniki uśredniano **dla 10 powtórzeń** przebiegu algorytmu genetycznego dla każdej z badanych wartości parametrów. Działanie to było konieczne ponieważ wyniki mogą się różnić w zależności od uruchomienia algorytmu genetycznego, gdyż występuje w nim nie determinizm – np. przy losowym wyborze populacji początkowej. Dla zilustrowania różnic w 2 przykładowych uruchomieniach algorytmu genetycznego zamieszczono wykresy zmian wartości funkcji celu dla tych samych parametrów dla funkcji „Hosaki”. Różnice widoczne są „gołym okiem”.



Rysunek .Przebieg zmienności wartości funkcji dopasowania najlepszego osobnika i średniej dla całej populacji w zależności od pokolenia dla przykładowego uruchomienia algorytmu genetycznego dla funkcji Hosaki.



Rysunek . Przebieg dla innego uruchomienia (też dla funkcji Hosaki).

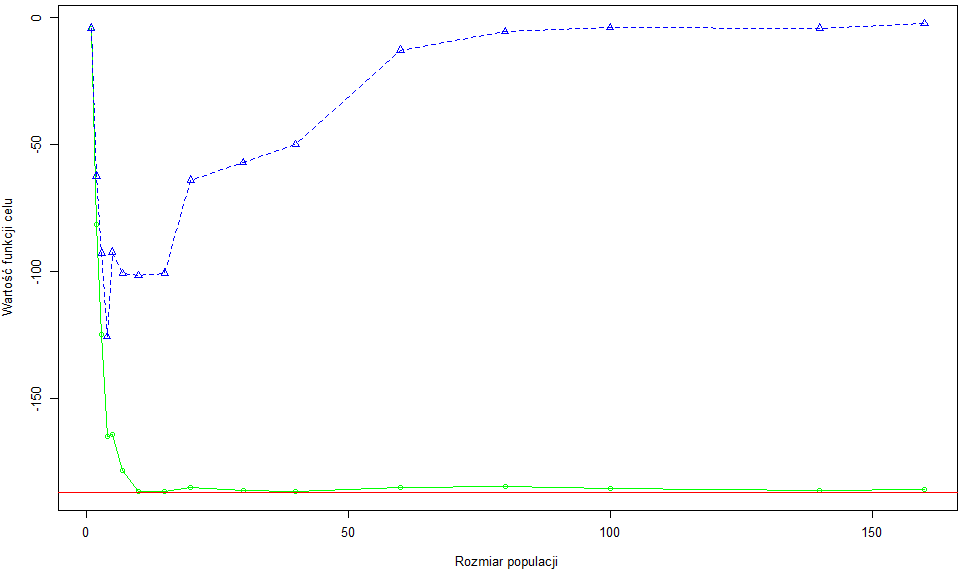
**Uwaga**: Przyjęto dalej następujące oznaczenia na wykresach (legenda):

* **kolor zielony** – wartość najlepsza w danej populacji,
* **kolor niebieski** – wartość średnia w danej populacji,
* **kolor czerwony** – optimum globalne danej funkcji.

## Badanie wpływu rozmiaru populacji

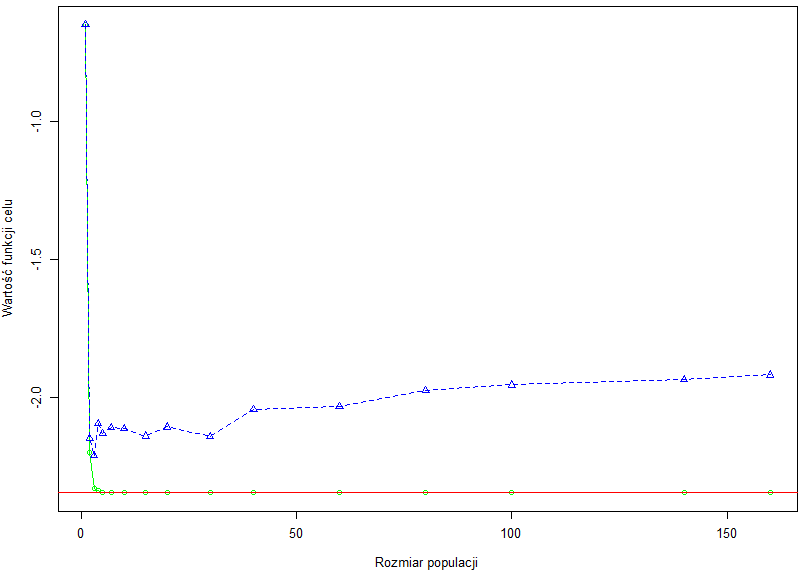
Zmienny rozmiar populacji, pozostałe parametry – stałe. Badania przeprowadzono dla każdej z 2 rozpatrywanych funkcji celu.

### Badanie wpływu rozmiaru populacji na przystosowanie dla funkcji Schuberta



Rysunek . Zależność wartości funkcji celu Schuberta od liczebności populacji

### Badanie wpływu rozmiaru populacji na przystosowanie dla funkcji Hosaki



Rysunek . Zależność wartości funkcji celu Hosaki od liczebności populacji

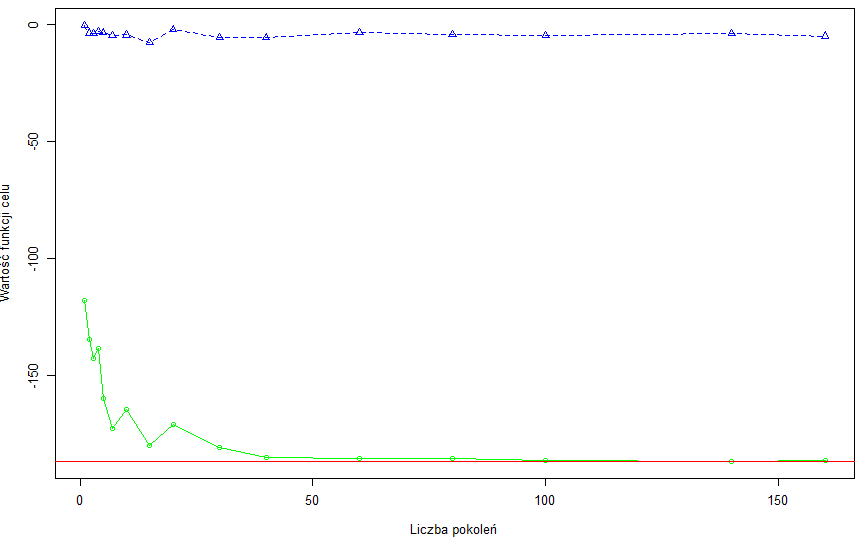
### Obserwacje nt. wpływu liczności populacji na jej średnie przystosowanie oraz przystosowanie najlepszego osobnika

Obserwacje dot. wpływu liczebności populacji na wartość funkcji celu:

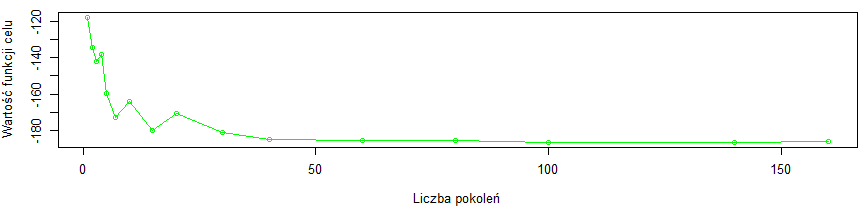
* w przypadku obu funkcji celu dla ustalonych wartości parametru już niewielkie zwiększenie liczebności populacji (do 5 osobników) powoduje odnalezienie ekstremum globalnego,
* średnia jakość osobników w populacji początkowo również bardzo mocno rośnie wraz ze wzrostem liczności populacji, jednak dla większych liczności średnia jakość ulega pogorszeniu,
* pogorszenie wartości średniej dla dużych rozmiarów populacji wynika z faktu, że pozostałe parametry, w tym **liczba pokoleń** pozostają stałe. W związku z tym, przy dużej liczbie osobników, większość populacji **„nie zdąży” wyewoluować** przy zbyt niskiej liczbie pokoleń. Natomiast raz znaleziony najlepszy osobnik jest i tak przekazywany do dalszego pokolenia – nie zmienia się praktycznie zatem jakość najlepszego osobnika.

## Badanie wpływu liczby pokoleń

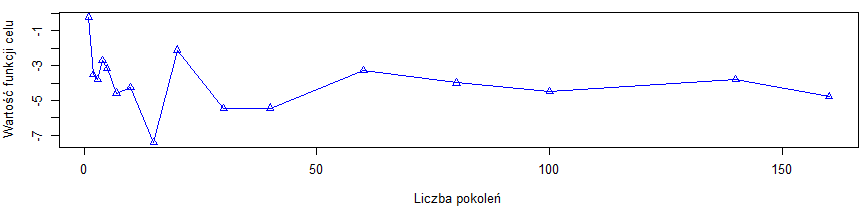
### Badanie wpływu liczby pokoleń na przystosowanie dla funkcji Schuberta



Rysunek . Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla calej populacji w zależności od liczby pokoleń

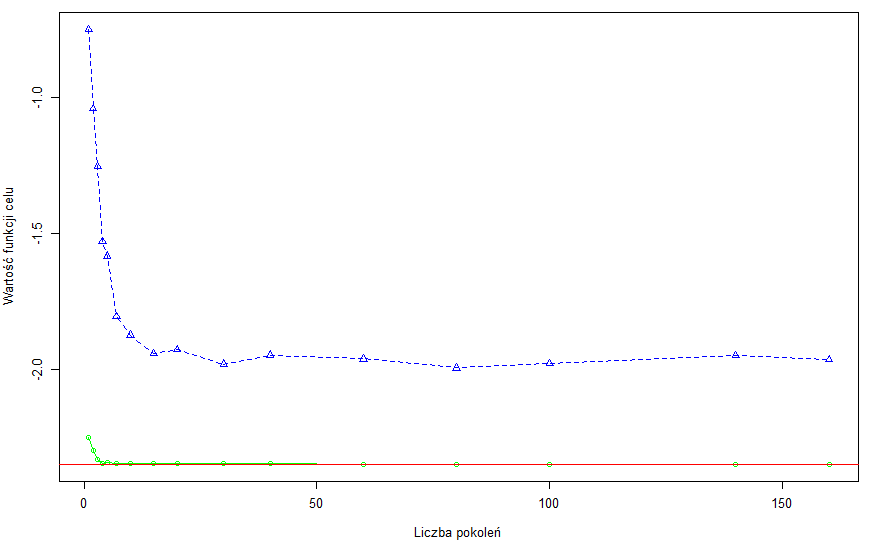


Rysunek . Powiększenie wykresu dla najlepszego osobnika

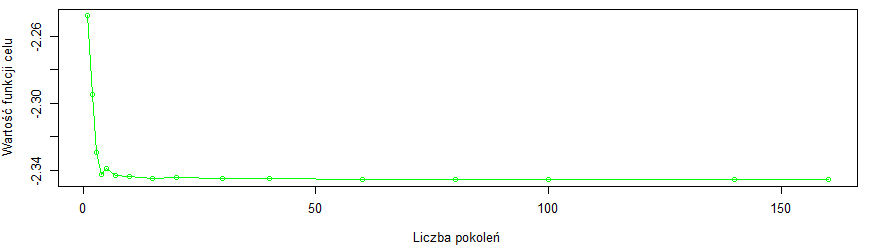


Rysunek . Powiększenie wykresu dla średniej z całej populacji

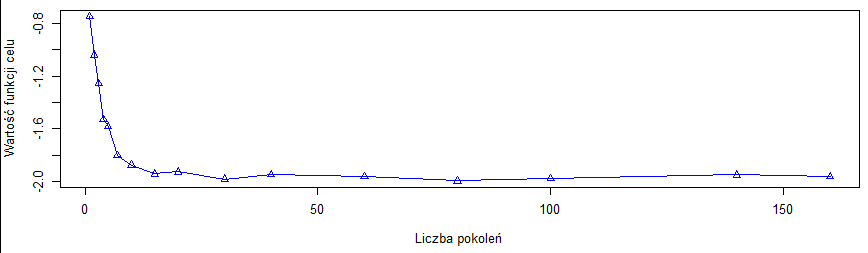
### Badanie wpływu liczby pokoleń na przystosowanie dla funkcji Hosaki



Rysunek . Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla calej populacji w zależności od liczby pokoleń



Rysunek . Powiększenie wykresu dla najlepszego osobnika



Rysunek . Powiększenie wykresu dla średniej z całej populacji

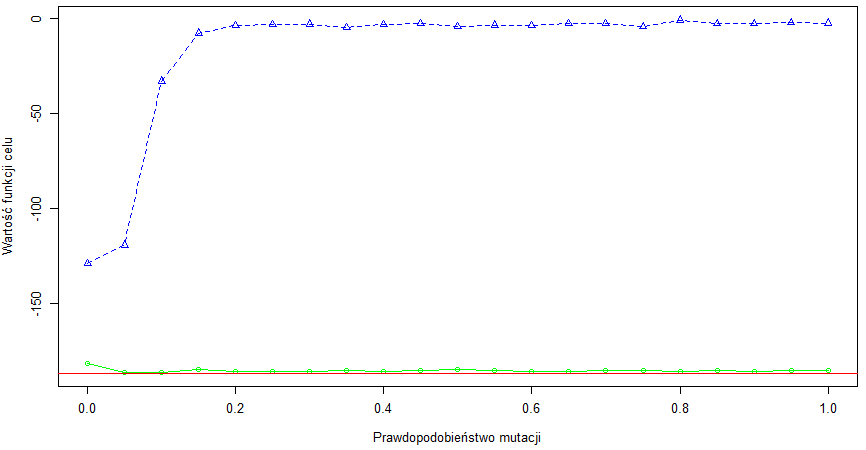
### Obserwacje nt. wpływu liczby pokoleń na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika

Na podstawie badań poczyniono następujące obserwacje:

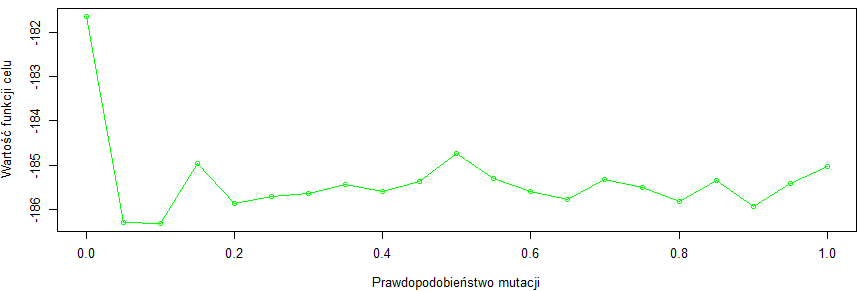
* w przypadku obu funkcji celu dla ustalonych zwiększanie liczby pokoleń powoduje początkowo gwałtowną poprawę średniej jakości populacji jak również jakości najlepszego osobnika. Następnie występuje jednak moment stabilizacji, gdzie w kolejnych pokoleniach nie odnotowuje się już znaczącej poprawy,
* stabilizacja dla najlepszego osobnika może być związana (i w rozpatrywanych przykładach jest) ze znalezieniem ekstremum globalnego funkcji – stąd w kolejnych pokoleniach nie można już poprawić tego wyniku,
* średnia jakość populacji również ulega stabilizacji jednak na niższym (co do wartości bezwzględnej) poziomie aniżeli wartość funkcji przystosowania dla najlepszego osobnika.

## Badanie wpływu prawdopodobieństwa mutacji

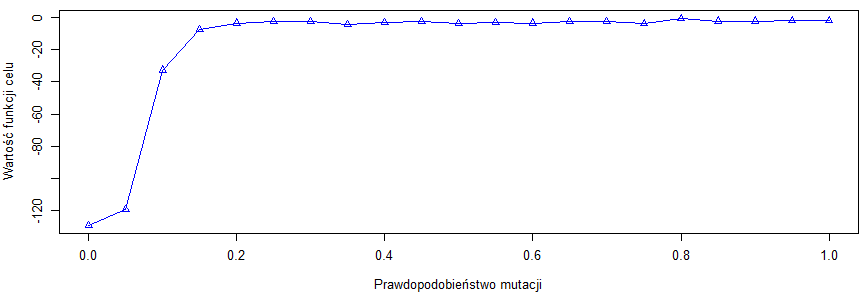
### Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Schuberta



Rysunek . Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa mutacji.

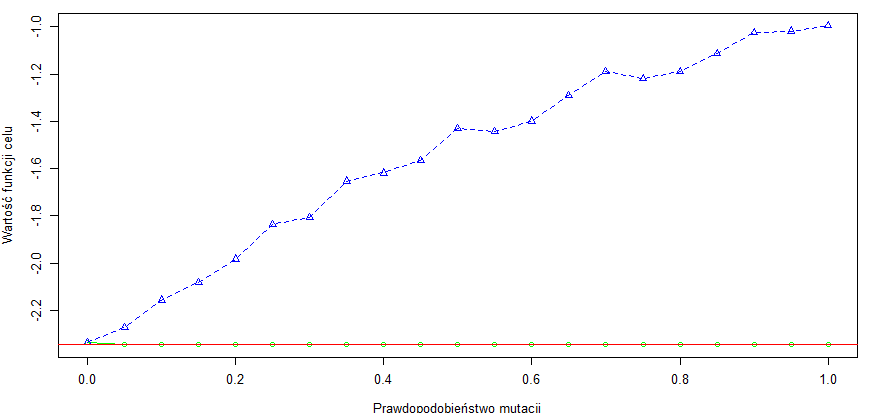


Rysunek . Powiększenie wykresu dla najlepszego osobnika

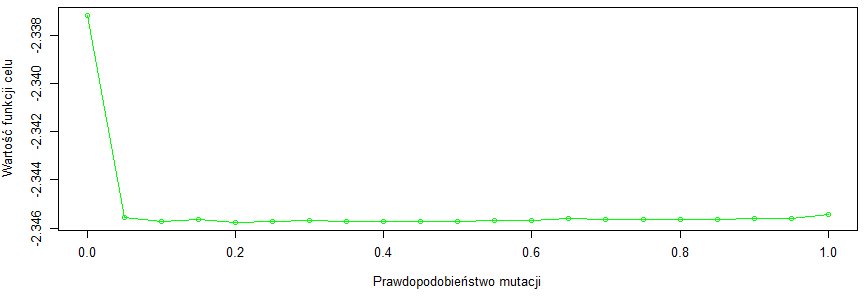


Rysunek . Powiększenie wykresu dla średniej z całej populacji

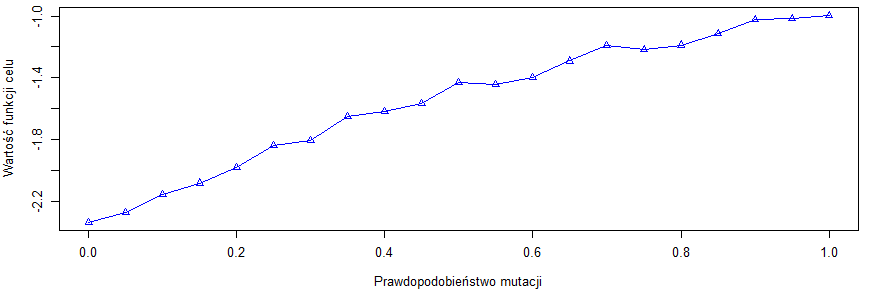
### Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Hosaki



Rysunek . Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa mutacji.



Rysunek . Powiększenie wykresu dla najlepszego osobnika



Rysunek . Powiększenie wykresu dla średniej z całej populacji

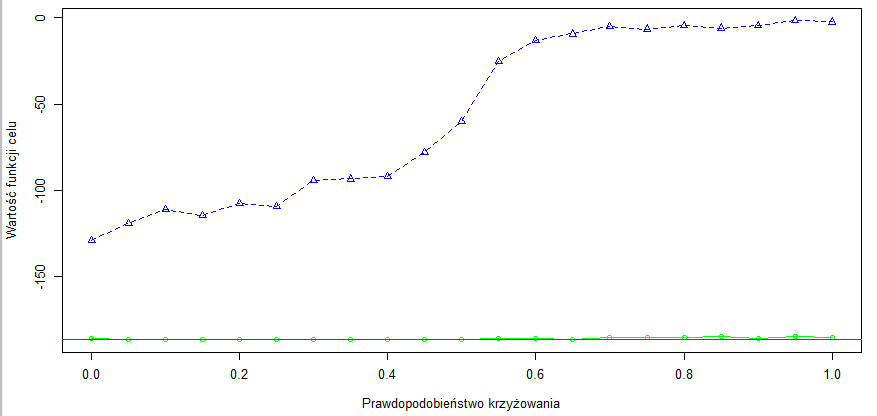
### Obserwacje nt. wpływu prawdopodobieństwa mutacji na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika

Na podstawie badań poczyniono następujące obserwacje:

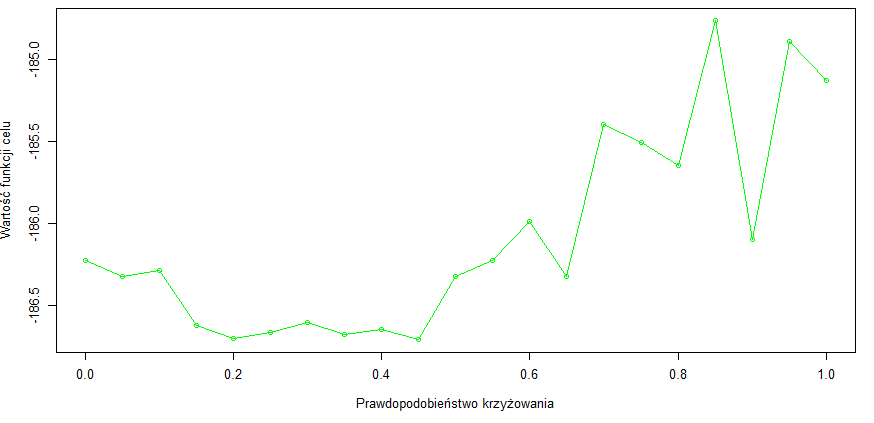
* przystosowanie najlepszego osobnika jest najwyższe w przypadku prawdopodobieństwa mutacji równego ok. 0,1,
* zerowe prawdopodobieństwo mutacji bardzo negatywnie wpływa na przystosowanie najlepszego osobnika – wówczas może on bowiem utknąć w minimum lokalnym – mutacja to znacząca zmiana, która może spowodować „uwolnienie” z minimum lokalnego,
* jednak zwiększanie prawdopodobieństwa mutacji negatywnie wpływa na przystosowanie populacji jako całości (średnie przystosowanie). Pojawiają się bowiem „mutanty” – osobniki po mutacji, których wartość funkcji przystosowania jest znacznie gorsza od pozostałych osobników w populacji,
* rozsądnym wyborem jest zatem prawdopodobieństwo mutacji równe 0,1 – jednocześnie pozwala uniknąć utknięcia w minimum lokalnym, jak również nie psuje zbytnio jakości całej populacji.

## Badanie wpływu prawdopodobieństwa krzyżowania

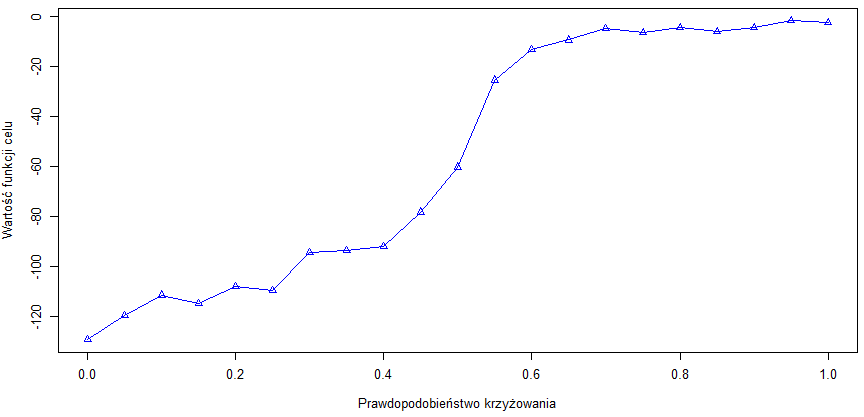
### Badanie wpływu prawdopodobieństwa krzyżowania na wartość funkcji Schuberta



Rysunek . Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa krzyżowania

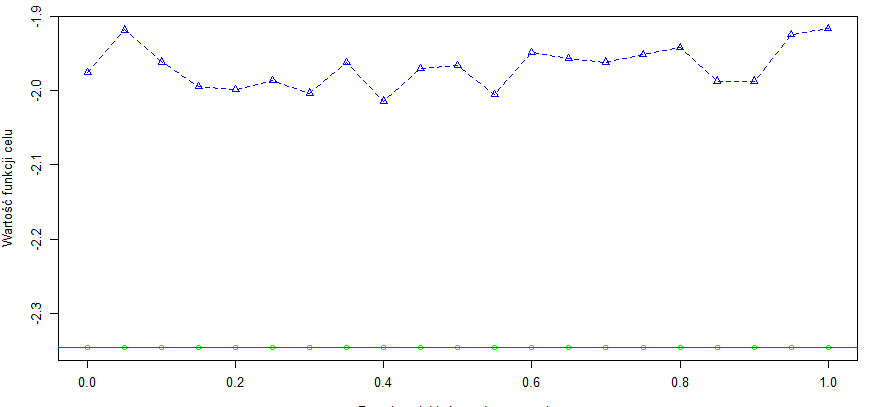


Rysunek . Powiększenie wykresu dla najlepszego osobnika

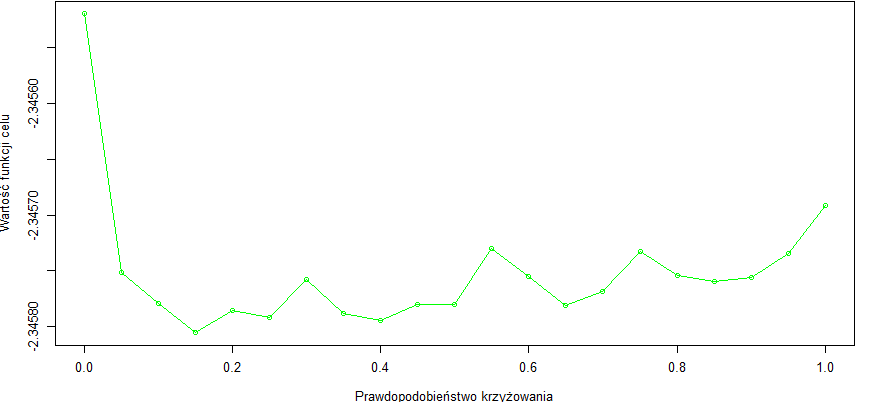


Rysunek . Powiększenie wykresu dla średniej z całej populacji

### Badanie wpływu prawdopodobieństwa mutacji na przystosowanie dla funkcji Hosaki



Rysunek . Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od prawdopodobieństwa krzyżowania



Rysunek . Powiększenie wykresu dla najlepszego osobnika

### Obserwacje nt. wpływu prawdopodobieństwa krzyżowania na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika

Na podstawie badań poczyniono następujące obserwacje:

* w przypadku funkcji Schuberta, zwiększenie prawdopodobieństwa do wartości ok. 0,8 powodowało obniżenie ogólnej jakości populacji, jednak do pewnej wartości (ok. 0,5) poprawie ulegało przystosowanie najlepszego osobnika,
* w przypadku funkcji Hosaki, wpływ ten nie był już tak jednoznaczny – dla średniej jakości całej populacji najkorzystniejsza była wartość ok. 0,9, natomiast ze względu na najlepszego osobnika – ok. 0,15.

## Badanie wpływu prawdopodobieństwa krzyżowania i mutacji

Badanie to ma nieco inny charakter od pozostałych ponieważ miało na celu sprawdzenie, jak zachowa się algorytm przy jednoczesnym wyłączeniu mutacji i krzyżowania, tzn. przy założeniu:

Dla zobrazowania przedstawiono różnice w ewolucji dla populacjach:

* gdzie zachodzi i krzyżowanie i mutacja,
* gdzie zachodzi tylko mutacja,
* gdzie zachodzi tylko krzyżowania,
* gdzie nie zachodzi ani krzyżowanie, ani mutacja.

Tabela . Poszczególne osobniki populacji w 1. i 60. pokoleniu w zależności od wartości prawdopodobieństw: mutacji i krzyżowania dla funkcji Hosaki

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 1. pokolenie | 60. pokolenie |
| 0,8 | 0,2 |  |  |
| 0 | 0,2 |  |  |
| 0,8 | 0 |  |  |
| 0 | 0 |  |  |

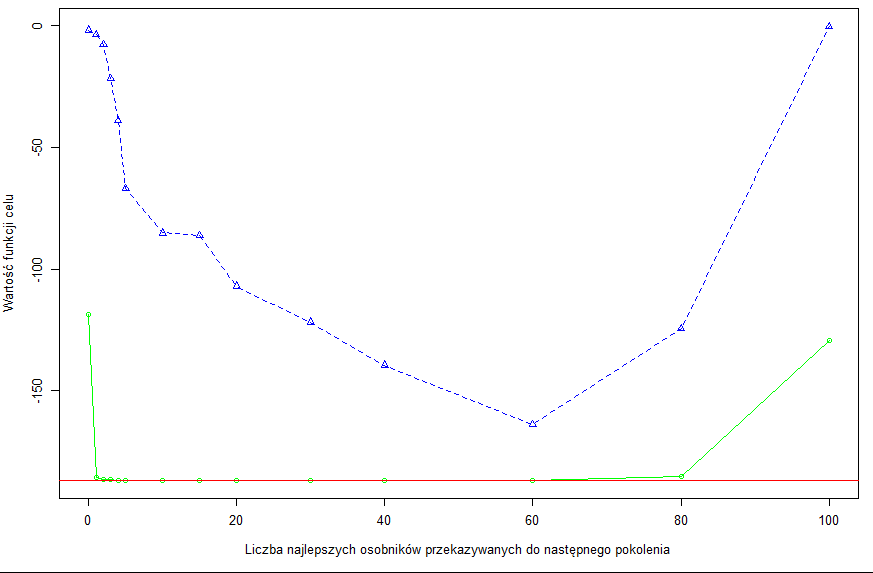
Z eksperymentu można wyciągnąć następujące wnioski:

* przy wyłączeniu operatora krzyżowania i mutacji, zadanie sprowadzało się do wyboru najlepszego spośród dostępnych (wylosowanych) osobników z populacji początkowej. W powyższej tabeli przedstawiono przykłady dla populacji 100 osobników – stąd liczba wylosowanych osobników jest dość duża, co pozwoliło na dość dokładny wybór najlepszego spośród nich.
* przy wyłączeniu krzyżowania i stosowaniu jedynie operatora mutacji zbieżność wartości funkcji celu w kolejnych pokoleniach jest mniejsza. Wynika to z tego, że zmiany w genotypie są przypadkowe. W populacji pojawiają się osobniki zmutowane o wartościach cech odbiegających od pozostałych,
* przy wyłączeniu mutacji i stosowaniu jedynie operatora krzyżowania maksimum globalne na powyższym przykładzie zostało odnalezione. Jednak ogólnie, brak mutacji grozi utknięciem w minimum lokalnym.

## Badanie wpływu poziomu elitaryzmu

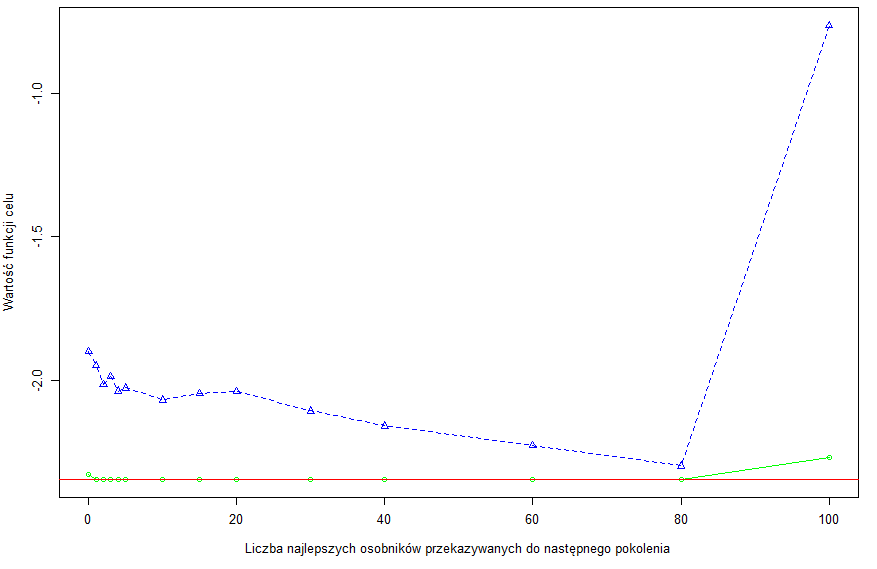
Elitaryzm określa ile najlepszych osobników z pokolenia zostanie przekazanych do następnego pokolenia bez zmian. W skrajnym przypadku może nie występować w ogóle, lub też obejmować całą populację (całe pokolenie).

### Badanie wpływu elitaryzmu na przystosowanie do funkcji Schuberta

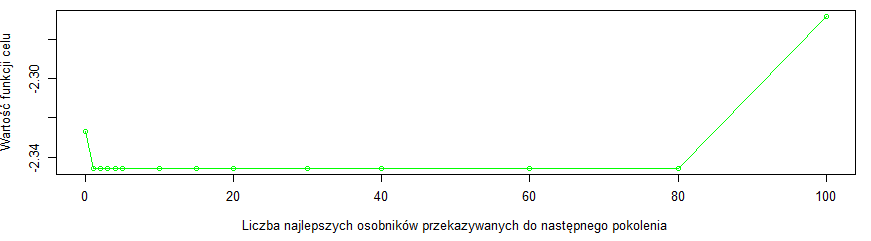


Rysunek . Zależność wartości funkcji celu Schuberta dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od wartości parametru „elitism”

### Badanie wpływu elitaryzmu na przystosowanie do funkcji Hosaki



Rysunek . Zależność wartości funkcji celu Hosaki dla najlepszego osobnika z populacji oraz średniej wartości dla całej populacji w zależności od wartości parametru „elitism”



Rysunek . Powiększenie wykresu dla najlepszego osobnika.

### Obserwacje nt. wpływu elitaryzmu na średnie przystosowanie populacji oraz przystosowanie najlepszego osobnika

Na podstawie badań poczyniono następujące obserwacje:

* zerowa wartość parametru „elitism” powoduje, że najlepszy osobnik z danego pokolenia jest „gubiony”, tzn. może nie zostać przekazany do następnego pokolenia,
* zwiększenie tego parametru już do wartości 1 zapobiega wyżej opisanemu zjawisku – stąd wartość najlepszego osobnika utrzymuje się na stałym poziomie w szerokim zakresie wartości parametru „elitism”,
* mimo, że wartość najlepszego osobnika jest już zachowywana od wartości parametru „elitism” równej 1, to dalsze zwiększanie udziału przekazywanych do następnego pokolenia najlepszych osobników z pokolenia poprzedniego pozytywnie wpływa na średnie przystosowanie populacji,
* jednak zbyt duża liczba przekazywanych osobników – równa liczebności całej populacji – powoduje ponowne, znaczące pogorszenie wartości funkcji celu – przekazywana jest bowiem cała populacja (wówczas określenie „najlepsze” w przypadku osobników z „elity” nie ma większego sensu, gdyż wszystkie osobniki przekazywane są do kolejnego pokolenia).

# Wnioski

## Wnioski nt. własnych implementacji

## Wnioski nt. implementacji i języka R

* język R stanowi bardzo ciekawą alternatywę do innych języków programowania,
* szczególnie przydatne są funkcyjne aspekty tego języka, które umożliwiły implementację skryptów bez zbędnego narzutu programistycznego (np. zwracanie funkcji jako wynik funkcji, która jako argument przyjmuje nazwę funkcji z pakietu globalOptTest),
* jeden z autorów sprawozdania miał okazję wcześniej implementować już podobne zadanie (z własną implementacją algorytmu genetycznego) w języku Kotlin. Ilość kodu potrzebna np. na przeprowadzenie badań (pomijając implementację samego algorytmu, której w tym przypadku nie było – skorzystano z gotowej paczki GA) w przypadku języka R jest znacząco niższa. Prostsze jest również generowanie wykresów,
* język R zawiera jednak kilka „puapek”. Jedną z nich jest problem, na który spowodował opóźnienie w oddaniu listy: podając nazwę elementu listy można bowiem używać składni nazwa.listy$nazwa.elementu lub nazwa.listy[[nazwa.elementu]]. Okazało się że jedynie ten drugi sposób umożliwia przekazanie nazwy elementu jako zmiennej typu string. Początkowo użyta składnia nazwa.listy$nazwa.elementu (gdzie nazwa.elementu to zmienna typu string) nie powodowała błędu, ale tworzyła w liście nowy element o nazwie „nazwa.elementu”. Stąd znalezienie tego błędu, znacząco wydłużyło czas implementacji.

# Kod z komentarzem

Komentarze umieszczono w kodzie w języku angielskim, dla zachowania spójności z nazewnictwem metod.

## Przegląd najważniejszych fragmentów kodu

Celem ułatwienia korzystania z pakietu globalOptTests napisano funkcję, która jako argument pobiera nazwę funkcji zdefiniowanej w pakiecie, i zwraca funkcję opakowującą funkcję pakietową goTest.

Funkcja ma więc sygnaturę ), gdzie ) odpowiada zwracanej funkcji celu. Jest to praktyczne zastosowanie funkcyjnych aspektów języka R.

Listing 7.1 Kod funkcji objective.fun.of

|  |
| --- |
| *# returns objective function given by name* objective.fun.of <- **function**(function.name) {  *# Get the length of the parameter vector expected by a given objective function.* problem.dimen <- *getProblemDimen*(function.name)   **function**(x1, x2) {  x1x2 <- *c*(x1, x2)  *goTest*(par = *c*(x1x2, *rep*(0, problem.dimen - *length*(x1x2))), fnName = function.name, checkDim = TRUE)  } } |

Do obliczenia wartości funkcji celu zadanej jej nazwą, utworzono funkcję objective.fun.get, która oblicza wartość funkcji celu dla wszystkich argumentów z zakresu określonego domyślnymi zakresami argumentów tejże funkcji celu. Funkcja została napisana dla 2-wymiarowych funkcji celu iż zwraca listę (x1, x2, values), gdzie x1 i x2 to wektory (spróbkowanych) wartości argumentu 1. i 2. (z krokiem 0,1), a values to macierz wartości funkcji.

Listing 7.2 Kod funkcji objective.fun.get

|  |
| --- |
| objective.fun.get <- **function**(function.name) {  *# Default lower and upper bounds (box constraints) for the given objective function* default.bounds <- *getDefaultBounds*(function.name)   x1 <- *seq*(default.bounds$**lower**[1], default.bounds$**upper**[1], by = 0.1)  x2 <- *seq*(default.bounds$**lower**[2], default.bounds$**upper**[2], by = 0.1)   *# Acutal objective function of given name* objective.fun <- *objective.fun.of*(function.name)   *# Apply objective.fun to each pair (x1', x2') where x1' belongs to x1 and x2 belongs to x2* objective.fun.value <- *outer*(x1, x2, *Vectorize*(objective.fun))  *list*(x1 = x1, x2 = x2, values = objective.fun.value) } |

Funkcja ta znalazła zastosowanie m.in. w funkcji objective.fun.plot, służącej do wyświetlania funkcji celu na ekranie w postaci wykresu 3-wymiarowego i tzw. mapy cieplnej.

Listing 7.3 Kod funkcji objective.fun.plot

|  |
| --- |
| *# plots function given by name* objective.fun.plot <- **function**(function.name, axes = {*axis*(1); *axis*(2)}) {  objective.fun <- *objective.fun.get*(function.name)  *# filled.contour(objective.fun$x1, objective.fun$x2, objective.fun$values, plot.axes={axis(1); axis(2); points(-0.8,-7.7, pch=3, cex=2, col="black", lwd=4); points(-7.7, -0.8, pch=3, cex=2, col="black", lwd=4)}, color.palette = jet.colors)  filled.contour*(objective.fun$**x1**, objective.fun$**x2**, objective.fun$**values**, plot.axes=axes, color.palette=jet.colors)  *persp3D*(objective.fun$**x1**, objective.fun$**x2**, objective.fun$**value**, theta = 45, phi = 25, expand = 0.5, ticktype = **"detailed"**, axes = TRUE) } |

Do powtórzenia przebiegu algorytmu genetycznego dla zadanych parametrów zdefiniowano funkcję GA.run.iterations, która zwraca listę w postaci (best.mean, mean.mean, iterations.best.scores), gdzie best.mean to średnia z najlepszych wyników w każdej iteracji, a mean.mean to średnia z średnich wyników dla całej populacji dla każdej iteracji.

Listing 7.4 Kod funkcji GA.run.iterations

|  |
| --- |
| *# runs function genetic algorithm for function given by name iterations.count times # params is list that contains params for genetic algorithm* GA.run.iterations <- **function**(function.name, iterations.count, params) {  default.bounds <- *getDefaultBounds*(function.name)  objective.fun <- *objective.fun.of*(function.name)  iteration.mean.scores <- NULL *# accumulator for mean results in each iteration* iteration.best.scores <- NULL *# accumulator for best results in each iteration* **for** (iteration **in** 1:iterations.count) {  GA <- *ga*(type = **"real-valued"**,  fitness = **function**(x) -*objective.fun*(x[1], x[2]),  lower = *c*(default.bounds$**lower**[1], default.bounds$**lower**[2]),  upper = *c*(default.bounds$**upper**[1], default.bounds$**upper**[2]),  popSize = params$**pop.size**,  maxiter = params$**max.iter**,  elitism = params$**elitism**,  pcrossover = params$**pcrossover**,  pmutation = params$**pmutation**)  iteration.best.scores[iteration] <- -GA@**fitnessValue** *# objective.fun(GA@solution[, 1], GA@solution[, 2])* iteration.mean.scores[iteration] <- *mean*(*apply*(GA@**population**, 1, **function**(x1x2) { *objective.fun*(x1x2[1], x1x2[2]) }))  }  *list*(best.mean = *mean*(iteration.best.scores), mean.mean = *mean*(iteration.mean.scores), iteration.best.scores = iteration.best.scores) } |

Funkcję tę wykorzystano w funkcji GA.run.experiment, która jako argumenty przyjmuje: nazwę funkcji, liczbę iteracji pętli uśredniającej wyniki, listę parametrów domyślnych oraz nazwę i zakres wartości parametru, który będzie ulegał zmianie. Zwracana jest lista (best.scores, mean.scores), zawierająca wektory odpowiednio najlepszych i średnich wyników dla każdej z wartości zadanego parametru spośród zadanego zakresu zmienności parametru.

Listing 7.5 Kod funkcji GA.run.experiment

|  |
| --- |
| *# runs experiment for every param.name value in param.range* GA.run.experiment <- **function**(function.name, iterations.count, params, param.name, param.range) {  param.best.scores <- NULL *# accumulator for best mean results in generation* param.mean.scores <- NULL *# accumulator for mean mean results in generation  # GA.best <- NULL # best result* **for** (param.value.index **in** *seq\_along*(param.range)) {  params[[param.name]] <- param.range[param.value.index]  param.score <- *GA.run.iterations*(function.name, iterations.count, params)  param.best.scores[param.value.index] <- param.score$**best.mean** param.mean.scores[param.value.index] <- param.score$**mean.mean** }  *list*(best.scores = param.best.scores, mean.scores = param.mean.scores) } |

Żeby całkowicie zautomatyzować przeprowadzenie badań zdefiniowano funkcję GA.run.experiment.list, która wykonuje zadane eksperymenty w trybie wsadowym, drukując dla każdego z nich odpowiednie wykresy na ekranie.

Listing 7.6 Kod funkcji GA.run.experiment.list

|  |
| --- |
| *# runs experiments as a batch* GA.run.experiment.list <- **function** (function.name, param.names.and.labels, param.value.ranges, iterations.count=10) {  **for** (param.index **in** *seq\_along*(param.names.and.labels$**names**)) {  param.name <- param.names.and.labels$**names**[param.index]  param.value.range <- param.value.ranges[[param.name]]  param.score.range <- *GA.run.experiment*(function.name, iterations.count, params, param.name, param.value.range)  *scores.plot*(  x = param.value.range,  y1 = param.score.range$**best.scores**,  y2 = param.score.range$**mean.scores**,  const = *getGlobalOpt*(function.name),  x.label = param.names.and.ranges$**labels**[param.index]  )  } } |

Wywołanie tej funkcji celem przeprowadzenia pełnego zestawu badań dla funkcji Schuberta przedstawiono w poniższym listingu.

Listing . Wywołanie funkcji GA.run.experiment

|  |
| --- |
| *# experiments for Schuber function* function.name <- **"Schubert"** params <- *list*(pop.size = 100, max.iter = 100, elitism = 0, pcrossover = 0.8, pmutation = 0.2) param.names.and.labels <- *list*(  names = *c*(**"pop.size"**, **"max.iter"**, **"elitism"**, **"pcrossover"**, **"pmutation"**),  labels = *c*(**"Rozmiar populacji"**, **"Liczba pokoleń"**, **"Poziom elityzmu"**, **"Prawdopodobieństwo krzyżowania"**, **"Prawdopodobieństwo mutacji"**) ) param.value.ranges <- *list*(  pop.size = *c*(1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160),  max.iter = *c*(1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160),  elitism = *c*(0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 60, 80, 100),  pcrossover = *seq*(0, 1, by = 0.1),  pmutation = *seq*(0, 1, by = 0.1) )  *GA.run.experiment.list*(function.name, param.names.and.labels, param.value.ranges) |

Do pojedynczego wywołania algorytmu genetycznego dla funkcji celu zadanej jej nazwą oraz dla zadanej listy parametrów i argumentu opcjonalnego – funkcji monitorującej przebieg wykonania algorytmu, napisano funkcję GA.run.once, zwracającą obiekt GA [4]. Oprócz wartości zwracanej, funkcja może powodować efekt uboczny w przypadku przekazania funkcji monitorującej – np. drukowanie osobników z kolejnych pokoleń na ekranie monitora.

Listing . Funkcja GA.run.once

|  |
| --- |
| *# runs GA only once for function given by name with params # monitor function can be attached to show progress in each generation* GA.run.once <- **function**(function.name, params, monitor=NULL) {  default.bounds <- *getDefaultBounds*(function.name)  objective.fun <- *objective.fun.of*(function.name)  *ga*(type = **"real-valued"**,  fitness = **function**(x) -*objective.fun*(x[1], x[2]),  lower = *c*(default.bounds$**lower**[1], default.bounds$**lower**[2]),  upper = *c*(default.bounds$**upper**[1], default.bounds$**upper**[2]),  popSize = params$**pop.size**,  maxiter = params$**max.iter**,  elitism = params$**elitism**,  pcrossover = params$**pcrossover**,  pmutation = params$**pmutation**,  monitor = monitor  ) } |

## Pełny kod programu

Listing 7.9 Skrypt global\_opt.R

|  |
| --- |
| *# dependencies require*(**"GA"**) *require*(**"globalOptTests"**)  function.names <- *c*(**"Schubert"**, **"Hosaki"**)  *# returns objective function given by name* objective.fun.of <- **function**(function.name) {  *# Get the length of the parameter vector expected by a given objective function.* problem.dimen <- *getProblemDimen*(function.name)   **function**(x1, x2) {  x1x2 <- *c*(x1, x2)  *goTest*(par = *c*(x1x2, *rep*(0, problem.dimen - *length*(x1x2))), fnName = function.name, checkDim = TRUE)  } }  objective.fun.get <- **function**(function.name) {  *# Default lower and upper bounds (box constraints) for the given objective function* default.bounds <- *getDefaultBounds*(function.name)   x1 <- *seq*(default.bounds$**lower**[1], default.bounds$**upper**[1], by = 0.1)  x2 <- *seq*(default.bounds$**lower**[2], default.bounds$**upper**[2], by = 0.1)   *# Acutal objective function of given name* objective.fun <- *objective.fun.of*(function.name)   *# Apply objective.fun to each pair (x1', x2') where x1' belongs to x1 and x2 belongs to x2* objective.fun.value <- *outer*(x1, x2, *Vectorize*(objective.fun))  *list*(x1 = x1, x2 = x2, values = objective.fun.value) }  *# plots function given by name* objective.fun.plot <- **function**(function.name, axes = {*axis*(1); *axis*(2)}) {  objective.fun <- *objective.fun.get*(function.name)  *# filled.contour(objective.fun$x1, objective.fun$x2, objective.fun$values, plot.axes={axis(1); axis(2); points(-0.8,-7.7, pch=3, cex=2, col="black", lwd=4); points(-7.7, -0.8, pch=3, cex=2, col="black", lwd=4)}, color.palette = jet.colors)  filled.contour*(objective.fun$**x1**, objective.fun$**x2**, objective.fun$**values**, plot.axes=axes, color.palette=jet.colors)  *persp3D*(objective.fun$**x1**, objective.fun$**x2**, objective.fun$**value**, theta = 45, phi = 25, expand = 0.5, ticktype = **"detailed"**, axes = TRUE) }  *# objective.fun.plot(function.names[1])  # finds global optimums of function given by name* objective.fun.opt.indecies <- **function**(function.name) {  objective.fun.value <- (*objective.fun.get*(function.name))$**values** *which*(objective.fun.value == *min*(objective.fun.value), arr.ind = TRUE) }  *# translates indecies into values for function given by name* objective.fun.opt.indecies.translate <- **function**(matrix.of.idecies, function.name) {  *# Default lower and upper bounds (box constraints) for the given objective function* default.bounds <- *getDefaultBounds*(function.name)   x1 <- *seq*(default.bounds$**lower**[1], default.bounds$**upper**[1], by = 0.1)  x2 <- *seq*(default.bounds$**lower**[2], default.bounds$**upper**[2], by = 0.1)  result <- NULL  add.to.result <- **function**(row.col) {  *append*(result, *c*(x1[row.col[1]], x2[row.col[2]]))  }  *apply*(matrix.of.idecies, 1, add.to.result)  result }  *# translates indecies into values for function given by name* objective.fun.opt.indecies.translate.one <- **function**(indecies, function.name) {  *# Default lower and upper bounds (box constraints) for the given objective function* default.bounds <- *getDefaultBounds*(function.name)   x1 <- *seq*(default.bounds$**lower**[1], default.bounds$**upper**[1], by = 0.1)  x2 <- *seq*(default.bounds$**lower**[2], default.bounds$**upper**[2], by = 0.1)  *c*(x1[indecies[1]], x2[indecies[2]]) } |

Listing 7.10 Skrypt global\_opt.R

|  |
| --- |
| GA.run.iterations <- **function**(function.name, iterations.count, params) {  default.bounds <- *getDefaultBounds*(function.name)  objective.fun <- *objective.fun.of*(function.name)  iteration.mean.scores <- NULL *# accumulator for mean results in each iteration* iteration.best.scores <- NULL *# accumulator for best results in each iteration* **for** (iteration **in** 1:iterations.count) {  GA <- *ga*(type = **"real-valued"**,  fitness = **function**(x) -*objective.fun*(x[1], x[2]),  lower = *c*(default.bounds$**lower**[1], default.bounds$**lower**[2]),  upper = *c*(default.bounds$**upper**[1], default.bounds$**upper**[2]),  popSize = params$**pop.size**,  maxiter = params$**max.iter**,  elitism = params$**elitism**,  pcrossover = params$**pcrossover**,  pmutation = params$**pmutation**)  iteration.best.scores[iteration] <- -GA@**fitnessValue** *# objective.fun(GA@solution[, 1], GA@solution[, 2])* iteration.mean.scores[iteration] <- *mean*(*apply*(GA@**population**, 1, **function**(x1x2) { *objective.fun*(x1x2[1], x1x2[2]) }))  }  *list*(best.mean = *mean*(iteration.best.scores), mean.mean = *mean*(iteration.mean.scores), iteration.best.scores = iteration.best.scores) }  *# runs experiment for every param.name value in param.range* GA.run.experiment <- **function**(function.name, iterations.count, params, param.name, param.range) {  param.best.scores <- NULL *# accumulator for best mean results in generation* param.mean.scores <- NULL *# accumulator for mean mean results in generation  # GA.best <- NULL # best result* **for** (param.value.index **in** *seq\_along*(param.range)) {  params[[param.name]] <- param.range[param.value.index]  param.score <- *GA.run.iterations*(function.name, iterations.count, params)  param.best.scores[param.value.index] <- param.score$**best.mean** param.mean.scores[param.value.index] <- param.score$**mean.mean** }  *list*(best.scores = param.best.scores, mean.scores = param.mean.scores) }  *# runs experiments as a batch* GA.run.experiment.list <- **function** (function.name, param.names.and.labels, param.value.ranges, iterations.count=10) {  **for** (param.index **in** *seq\_along*(param.names.and.labels$**names**)) {  param.name <- param.names.and.labels$**names**[param.index]  param.value.range <- param.value.ranges[[param.name]]  param.score.range <- *GA.run.experiment*(function.name, iterations.count, params, param.name, param.value.range)  *scores.plot*(  x = param.value.range,  y1 = param.score.range$**best.scores**,  y2 = param.score.range$**mean.scores**,  const = *getGlobalOpt*(function.name),  x.label = param.names.and.ranges$**labels**[param.index]  )  } }  *#plots best (y1) and mean (y2) scores for each param value x* scores.plot <- **function**(x, y1, y2, const, x.label, y.label = **"Wartość funkcji celu"**) {  *matplot*(x, *cbind*(y1, y2), col = *c*(**"green"**, **"blue"**), type = *c*(**"o"**, **"o"**), pch = 1:2, xlab = x.label, ylab = y.label)  *abline*(h = const, col = **"red"**) }  *# experiments for Schuber function* function.name <- **"Schubert"** params <- *list*(pop.size = 100, max.iter = 100, elitism = 0, pcrossover = 0.8, pmutation = 0.2) param.names.and.labels <- *list*(  names = *c*(**"pop.size"**, **"max.iter"**, **"elitism"**, **"pcrossover"**, **"pmutation"**),  labels = *c*(**"Rozmiar populacji"**, **"Liczba pokoleń"**, **"Poziom elityzmu"**, **"Prawdopodobieństwo krzyżowania"**, **"Prawdopodobieństwo mutacji"**) ) param.value.ranges <- *list*(  pop.size = *c*(1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160),  max.iter = *c*(1, 2, 3, 4, 5, 7, 10, 15, 20, 30, 40, 60, 80, 100, 140, 160),  elitism = *c*(0, 1, 2, 3, 4, 5, 10, 15, 20, 30, 40, 60, 80, 100),  pcrossover = *seq*(0, 1, by = 0.1),  pmutation = *seq*(0, 1, by = 0.1) )  *GA.run.experiment.list*(function.name, param.names.and.labels, param.value.ranges) |

Listing 7.11 Skrypt main.R

|  |
| --- |
| *# plots each generation of GA* monitor <- **function**(obj)  {  *contour*(objective$**x1**, objective$**x2**, objective$**values**, drawlabels = FALSE, col = *grey*(0.5))  *title*(*paste*(**"iteration ="**, obj@**iter**), font.main = 1)  *points*(obj@**population**, pch = 20, col = 2)  *Sys.sleep*(0.2) }  *# runs GA only once for function given by name with params # monitor function can be attached to show progress in each generation* GA.run.once <- **function**(function.name, params, monitor=NULL) {  default.bounds <- *getDefaultBounds*(function.name)  objective.fun <- *objective.fun.of*(function.name)  *ga*(type = **"real-valued"**,  fitness = **function**(x) -*objective.fun*(x[1], x[2]),  lower = *c*(default.bounds$**lower**[1], default.bounds$**lower**[2]),  upper = *c*(default.bounds$**upper**[1], default.bounds$**upper**[2]),  popSize = params$**pop.size**,  maxiter = params$**max.iter**,  elitism = params$**elitism**,  pcrossover = params$**pcrossover**,  pmutation = params$**pmutation**,  monitor = monitor  ) }  params <- *list*(pop.size = 100, max.iter = 100, elitism = 1, pcrossover = 0.01, pmutation = 0.0) function.name <- **"Hosaki"** ga.result <- *GA.run.once*(function.name, params, monitor) |

# 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>