Politechnika Wrocławska

Laboratorium

Inteligencja Obliczeniowa i jej Zastosowania

Algorytmy ewolucyjne i hybrydowe

Authors:
Rafał Pieniążek
Jakub Pomykała

Supervisor:
prof. dr inż. Olgierd Unold

15 maja 2018

Spis treści

1 Wstęp **3**

2 Zastosowany algorytm optymalizacji **3**
2.1 Zastosowane narzędzia implementacji 3
2.1.1 Język R 3
2.1.2 Pakiet GA 3
2.1.3 Pakiet globalOpts 3

3 Własne operatory krzyżowania i mutacji **4**
3.1 Funkcja wielomodalna - Funkcja Shuberta 4
3.1.1 Wzór analityczny 4
3.1.2 Wykres w ustalonym przedziale zmiennych 4
3.1.3 Ekstremum globalne 4
3.2 Zmiana funkcji mutowania 6
3.3 Zmiana funkcji krzyżowania 6

4 Wnioski **7**

5 Literatura **7**

Spis rysunków

1 Wzór analityczny funkcji Schuberta 4
2 Wykres funkcji Schuberta 4
3 Minimum globalne dla funkcji Schuberta 4
4 Minimum globalne dla funkcji Schuberta 5
5 Porównanie domyślnej funkcji mutowania z własną 6
6 Porównanie domyślnej funkcji mutowania z własną - znalezione ekstrema 7
7 Porównanie domyślnej funkcji krzyżowania z własną 8
8 Porównanie domyślnej funkcji krzyżowania z własną - znalezione ekstrema 8

1 Wstęp

Celem laboratorium było przeprowadzenie optymalizacji globalnej dla wybranych funkcji z pakietu globalOptTests.

2 Zastosowany algorytm optymalizacji

W laboratorium zastosowano algorytmy genetyczne będące klasą algorytmów ewolucyjnych. Algorytmy ewolucyjne stanowią kierunek sztucznej inteligencji, która wykorzystuje i symuluje ewolucję biologiczną. Wszystkie algorytmy tej klasy symulują podstawowe zachowania w teorii ewolucji biologicznej - procesy selekcji, mutacji i reprodukcji. Zachowanie jednostek zależy od środowiska. Zbiór jednostek nazywa się populacją. Taka populacja ewoluuje zgodnie z regułami selekcji zgodnie z funkcją celu przypisaną do środowiska. Propagowane do kolejnych pokoleń są tylko najbardziej dopasowane osobniki.

2.1 Zastosowane narzędzia implementacji

2.1.1 Język R

R jest językiem programowania i środowiskiem programistycznym, używanym głównie do obliczeń statystycznych i wizualizacji danych, do sztucznej inteligencji a także do ekonomii i innych zagadnień wykorzystujących obliczenia numeryczne. Został stworzony przez Rossa Ihakę i Roberta Gentlemana na Uniwersytecie w Auckland w Nowej Zelandii.

2.1.2 Pakiet GA

Pakiet GA zawiera zestaw funkcji ogólnego przeznaczenia do optymalizacji z wykorzystaniem algorytmów genetycznych. Dostępnych jest kilka operatorów genetycznych, których można łączyć w celu zbadania najlepszych ustawień dla bieżącego zadania.

2.1.3 Pakiet globalOpts

Pakiet zawierający implementację funkcji przydatnych do przeprowadzania testów wydajnościowych algorytmów optymalizacji globalnej.

3 Własne operatory krzyżowania i mutacji

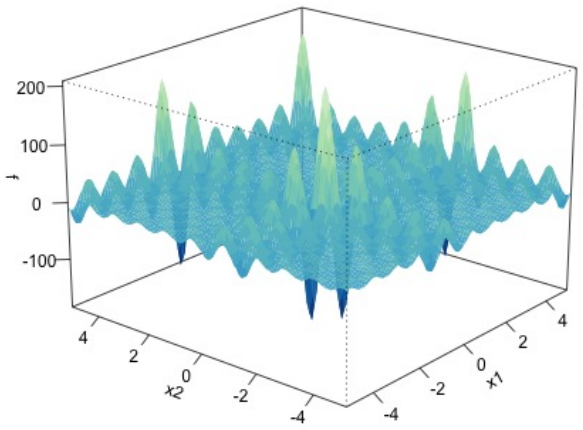
3.1 Funkcja wielomodalna - Funkcja Shuberta

3.1.1 Wzór analityczny

$$f(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$$

Rysunek 1: Wzór analityczny funkcji Schuberta

3.1.2 Wykres w ustalonym przedziale zmiennych

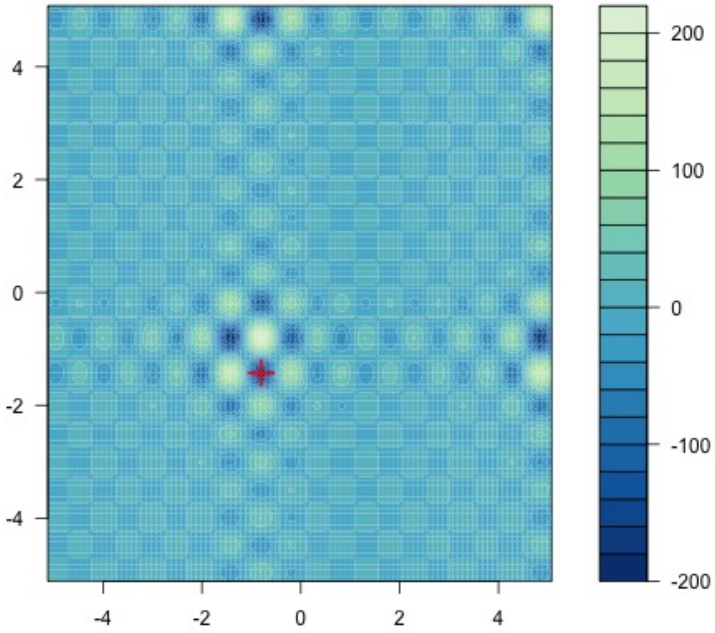


Rysunek 2: Wykres funkcji Schuberta

3.1.3 Ekstremum globalne

$$f(\mathbf{x}^*) = -186.7309$$

Rysunek 3: Minimum globalne dla funkcji Schuberta



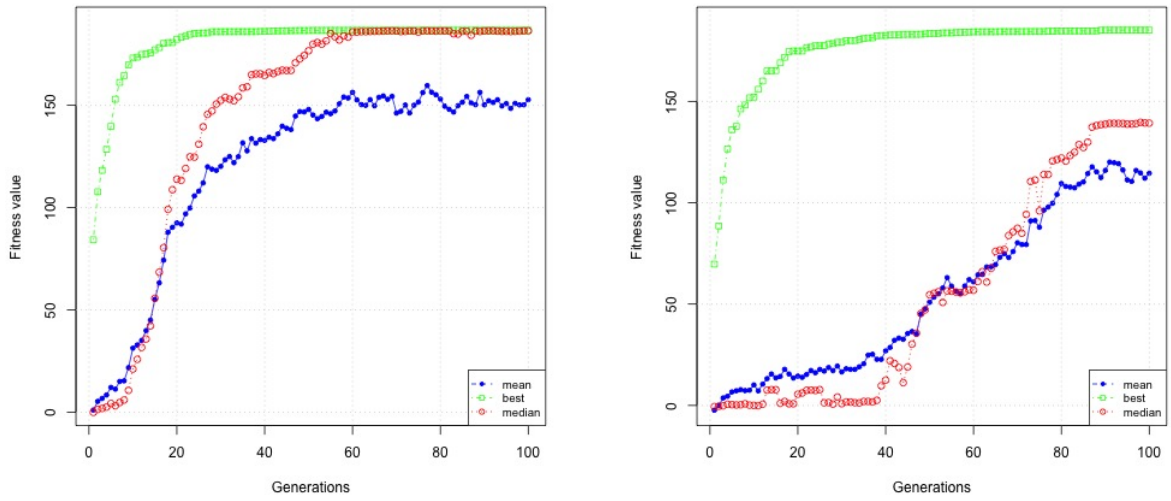
Rysunek 4: Minimum globalne dla funkcji Schuberta

3.2 Zmiana funkcji mutowania

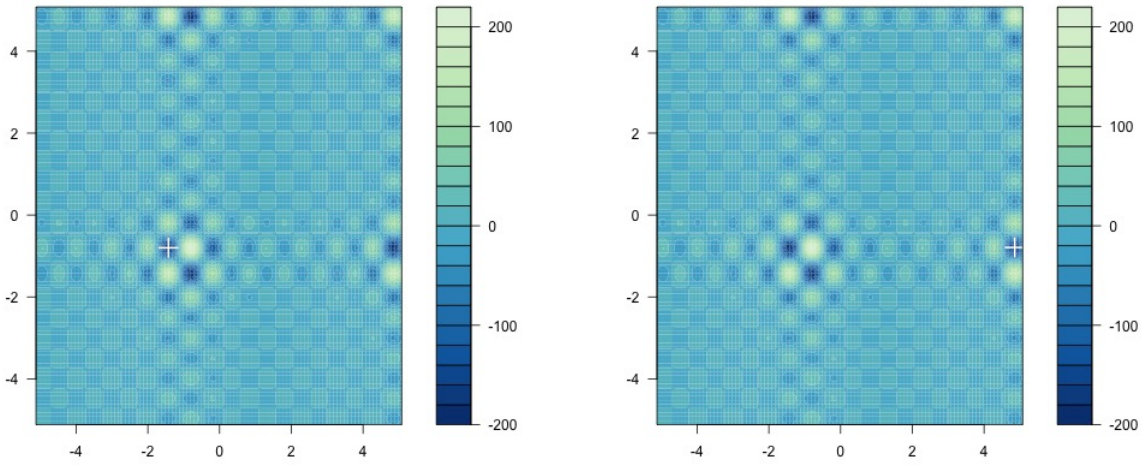
Poniżej przedstawiono własną propozycję implementacji funkcji mutowania.

```
customMutation <- function(object, parent)
    {
      mod <- parent %% 2
      if(mod == 0){
        return (parent * 2)
      } else {
        return (parent / 2) + 1
      }
    }
```

Na poniższych wykresach przedstawiono zestawienie rezultatów działania w przypadku domyślnej i własnej, zaimplementowanej funkcji mutacji. Wyniki dla funkcji domyślnej znajdują się po lewej stronie.



Rysunek 5: Porównanie domyślnej funkcji mutowania z własną



Rysunek 6: Porównanie domyślnej funkcji mutowania z własną - znalezione ekstrema

Zmiana funkcji mutującej nie wpłynęła znacząco na wyniki końcowy. Mimo, iż w początkowej fazie wynik funkcji zbiegał do ekstremum wolniej niż w przypadku funkcji domyślnej, po 100 pokoleniach wynik jest na podobnym poziomie w obu przypadkach. Finalnie jednak algorytm z własną funkcją mutującą nie znalazł ekstremum globalnego. Widoczne jest to na wykresie temperaturowym, na którym zaznaczono rozwiązanie znalezione przez algorytm.

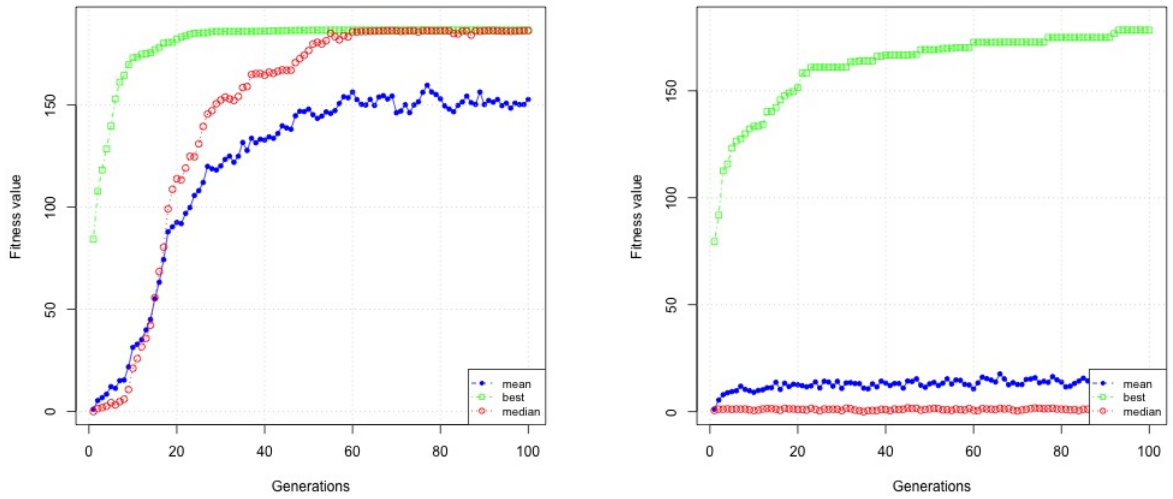
W przypadku funkcji z własną funkcją mutowania można zauważyć spadek średniej i mediany wyników.

3.3 Zmiana funkcji krzyżowania

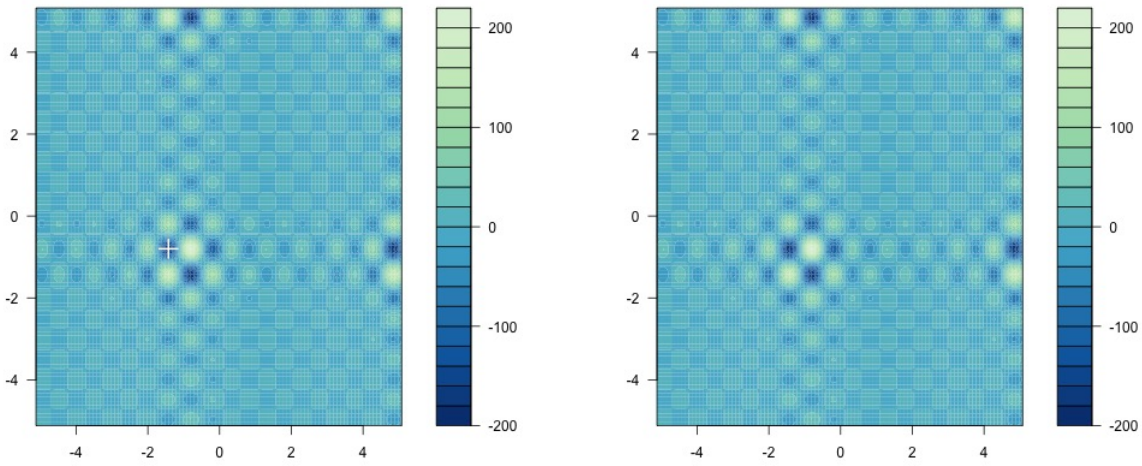
Poniżej przedstawiono własną propozycję implementacji funkcji krzyżowania.

```
customCrossover <- function(object, parents)
    {
      parent_1 <- parents[[1]]
      parent_2 <- parents[[2]]
      wektor_1 <- c(parent_1, parent_2)
      fitness = testFunctionWrapper(parent_1, parent_2)

      tmp_parent_1 <- parents[[1]] + runif(1, -1, 1)
      tmp_parent_2 <- parents[[2]] + runif(1, 1, -1)
      tmp_wektor_1 <- c(parent_1, parent_2)
      tmp_fitness = testFunctionWrapper(parent_1, parent_2)

      if(tmp_fitness > fitness){
        return (list(children=matrix(tmp_wektor_1), fitness=tmp_fitness))
      }
      return (list(children=matrix(wektor_1), fitness=fitness))
    }
```

Na poniższych wykresach przedstawiono porównanie rezultatu działania algorytmu w przypadku zmiany funkcji krzyżowania z domyślnej na własną implementację.



Rysunek 7: Porównanie domyślnej funkcji krzyżowania z własną



Rysunek 8: Porównanie domyślnej funkcji krzyżowania z własną - znalezione ekstrema

W przypadku własnej funkcji krzyżującej algorytm nie znalazł ekstremum globalnego. Wartości średniej i mediany spadły znaczącą w stosunku do domyślnej implementacji. Mediana jest praktycznie równa zero.

4 Wnioski

Implementacja własnych funkcji mutowania i krzyżowania jest zadaniem nietrywialnym, ale możliwym do wykonania. Jakoś wyników uzyskanych przy własnych implementacjach znacząco odbiega

od domyślnych funkcji.

5 Literatura

1. Artur Suchwałko, "Wprowadzenie do R dla programistow innych jezykow", https://cran.r-project.org/doc/contrib/R-dla-programistow-innych-jezykow.pdf, 2014-02-23
2. Luca Scrucca, "On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution", https://arxiv.org/pdf/1605.01931.pdf, 2016-05-09
3. dr inż. Julian Sienkiewicz, "Pakiet R w analizie układów złożonych", http://www.if.pw.edu.pl/~julas/CSAR/csar11.html, 2017
4. W. N. Venables, D. M. Smith, R Core Team, "An Introduction to R", https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf, 2018-04-23
5. Luca Scrucca, "Package 'GA'", ftp://cran.r-project.org/pub/R/web/packages/GA/GA.pdf, 2016-09-29
6. Katharine Mullen, "Package 'globalOptTests'", https://cran.r-project.org/web/packages/globalOptTests/globalOptTests.pdf, 2015-02-15
7. Abdal-Rahman Hedar, "Global Optimization Test Problems", http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm, dostęp online: 2018-05-04