Politechnika Wrocławska

Inteligencja Obliczeniowa i jej zastosowania

Badanie algorytmu genetycznego z zakresu optymalizacji globalnej dla wybranych funkcji testowych

Autorzy:
Paweł Andziul 200648
Marcin Słowiński 200638

Prowadzący:
dr hab. inż. Olgierd Unold,
prof. nadzw. PWr

29 marca 2017

Spis treści

1 Wprowadzenie

Algorytmy genetyczne to

W ramach laboratorium należało przeprowadzić testy algorytmu genetycznego dla różnych parametrów. Jako benchmark oceny należało użyć pakietu „getGlobalOpts" oraz języka R.

Pomiary wykonywano na 2 różnych jednostkach roboczych. Ich parametry nie są istotne z punktu widzenia analizy i możliwości porównania rezultatów.

2 Implementacja

Poniżej (listing 1) zamieszczono kod napisany w języku R przygotowany w celu umożliwienia przeprowadzenia pomiarów.

Listing 1: Skrypt w języku R wykorzystany do badań

```
1
2  rm(list=ls())
3
4  require("GA")
5  require("globalOptTests")
6  require("rgl")
7
8  # Params ----
9
10 n <- 10            # minimum 5
11 GAPopulation <- 50 # default 50
12 GAIterations <- 100 # default 100
13 GAMutations <- 0.1 # default 0.1
14 GACrossovers <- 0.8 # default 0.8
15
16 isSingleTest <- FALSE
17 graphs <- TRUE
18 quality <- 100 #graph resolutions
19
20 mutationTests <- seq(0, 1, 0.1)
21 crossoverTests <- seq(0, 1, 0.1)
22 elitismTests <- seq(0, 1, 0.1)
23 populationTests <- seq(10, 100, 5)
24 iterationTests <- seq(10, 200, 10)
25
26 # Functions ----
27
28 #funcName <- "Branin" #2d
29 funcName <- "Gulf" #3d
30 #funcName <- "CosMix4" #4d
31 #funcName <- "EMichalewicz" #5d
32 #funcName <- "Hartman6" #6d
33 #funcName <- "PriceTransistor" #9d
34 #funcName <- "Schwefel" #10d
35 #funcName <- "Zeldasine20" #20d
36
37 # Processing ----
38
```

```
dim <- getProblemDimen(funcName)
B <- matrix(unlist(getDefaultBounds(funcName)),ncol=dim,byrow=TRUE)
f <- function(xx) goTest(par=c(xx, rep(0, dim-length(xx))), fnName=funcName,
    checkDim = TRUE)
globalOpt <- getGlobalOpt(funcName)

if (graphs) {
  xprobes <- abs(B[2,1] - B[1,1]) / quality
  yprobes <- abs(B[2,2] - B[1,2]) / quality
  x <- seq(B[1,1], B[2,1], by = xprobes)
  y <- seq(B[1,2], B[2,2], by = yprobes)
  z <- outer(x, y, Vectorize(function(x,y) f(c(x,y))))
  nbcol = 100
  color = rev(rainbow(nbcol, start = 0/6, end = 4/6))
  zcol  = cut(z, nbcol)
  persp3d(x, y, z, theta=50, phi=25, expand=0.75, col=color[zcol],
          ticktype="detailed",axes=TRUE)
  persp3D(x, y, z, theta = -45, phi = 20, color.palette = jet.colors)
}

if (isSingleTest) {

  vector <- rep(NA,n)
  for (i in 1:n) {
    GAmin <- ga(type = "real-valued", fitness = function(xx) -f(xx),
                min = c(B[1,]), max = c(B[2,]),
                popSize = GAPopulation, maxiter = GAIterations,
                pmutation = GAMutations, pcrossover = GACrossovers)
    solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
    vector[i] <- f(solution[1,])
  }
  result <- matrix(c(vector),nrow = n,ncol = 1)
  write.table(result, file = "resultsSingle.csv", row.names=FALSE, na="",
      col.names=FALSE, sep=";")

} else {

  gMin <- .Machine$integer.max
  gBest <- NA

  temp <- c()
  values <- mutationTests
  averages <- c()
  for (mutation in values) {
    sum <- 0
    vector <- rep(NA,n)
    for (i in 1:n) {
      GAmin <- ga(type = "real-valued",
                  fitness = function(xx) -f(xx),
                  min = c(B[1,]), max = c(B[2,]),
                  popSize = GAPopulation, maxiter = GAIterations,
                  pmutation = mutation, pcrossover = GACrossovers)
      solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
      eval <- f(solution[1,])
      if (eval < gMin) {
        gMin <- eval
```

```
      gBest <- GAmin
    }
    sum <- sum + eval
    vector[i] <- eval
  }
  temp <- c(temp, vector)
  averages <- c(averages, (sum / n))
}
result <- matrix(c(temp),nrow = n,ncol = length(values))
write.table(result, file = "resultsMutations.csv", row.names=FALSE, na="",
    col.names=FALSE, sep=";")

if (graphs) {
  plot(values, averages,
       main="Function values for different mutation probabilities",
       ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
       type="l", col="red", xlab="params", ylab="value")
  abline(globalOpt,0, col="green")
}

temp <- c()
values <- crossoverTests
averages <- c()
for (crossover in values) {
  sum <- 0
  vector <- rep(NA,n)
  for (i in 1:n) {
    GAmin <- ga(type = "real-valued",
                fitness = function(xx) -f(xx),
                min = c(B[1,]), max = c(B[2,]),
                popSize = GAPopulation, maxiter = GAIterations,
                pmutation = GAMutations, pcrossover = crossover)
    solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
    eval <- f(solution[1,])
    if (eval < gMin) {
      gMin <- eval
      gBest <- GAmin
    }
    sum <- sum + eval
    vector[i] <- eval
  }
  temp <- c(temp, vector)
  averages <- c(averages, (sum / n))
}
result <- matrix(c(temp),nrow = n,ncol = length(values))
write.table(result, file = "resultsCrossover.csv", row.names=FALSE, na="",
    col.names=FALSE, sep=";")

if (graphs) {
  plot(values, averages,
       main="Function values for different crossover probabilities",
       ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
       type="l", col="red", xlab="params", ylab="value")
  abline(globalOpt,0, col="green")
}

```

```
temp <- c()
values <- elitismTests
averages <- c()
for (elitism in values) {
  sum <- 0
  vector <- rep(NA,n)
  for (i in 1:n) {
    GAmin <- ga(type = "real-valued",
                fitness = function(xx) -f(xx),
                min = c(B[1,]), max = c(B[2,]),
                popSize = GAPopulation, maxiter = GAIterations,
                pmutation = GAMutations, pcrossover = GACrossovers, elitism =
                    elitism)
    solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
    eval <- f(solution[1,])
    if (eval < gMin) {
      gMin <- eval
      gBest <- GAmin
    }
    sum <- sum + eval
    vector[i] <- eval
  }
  temp <- c(temp, vector)
  averages <- c(averages, (sum / n))
}
result <- matrix(c(temp),nrow = n,ncol = length(values))
write.table(result, file = "resultsElitism.csv", row.names=FALSE, na="",
    col.names=FALSE, sep=";")

if (graphs) {
  plot(values, averages,
       main="Function values for different elitism",
       ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
       type="l", col="red", xlab="params", ylab="value")
  abline(globalOpt,0, col="green")
}

temp <- c()
values <- populationTests
averages <- c()
for (population in values) {
  sum <- 0
  vector <- rep(NA,n)
  for (i in 1:n) {
    GAmin <- ga(type = "real-valued",
                fitness = function(xx) -f(xx),
                min = c(B[1,]), max = c(B[2,]),
                popSize = population, maxiter = GAIterations,
                pmutation = GAMutations, pcrossover = GACrossovers)
    solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
    eval <- f(solution[1,])
    if (eval < gMin) {
      gMin <- eval
      gBest <- GAmin
    }
    sum <- sum + eval
```

```
      vector[i] <- eval
    }
    temp <- c(temp, vector)
    averages <- c(averages, (sum / n))
  }
  result <- matrix(c(temp),nrow = n,ncol = length(values))
  write.table(result, file = "resultsPopulation.csv", row.names=FALSE, na="",
      col.names=FALSE, sep=";")

  if (graphs) {
    plot(values, averages,
         main="Function values for different population sizes",
         ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
         type="l", col="red", xlab="params", ylab="value")
    abline(globalOpt,0, col="green")
  }

  temp <- c()
  values <- iterationTests
  averages <- c()
  for (iterations in values) {
    sum <- 0
    vector <- rep(NA,n)
    for (i in 1:n) {
      GAmin <- ga(type = "real-valued",
                  fitness = function(xx) -f(xx),
                  min = c(B[1,]), max = c(B[2,]),
                  popSize = GAPopulation, maxiter = iterations,
                  pmutation = GAMutations, pcrossover = GACrossovers)
      solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
      eval <- f(solution[1,])
      if (eval < gMin) {
        gMin <- eval
        gBest <- GAmin
      }
      sum <- sum + eval
      vector[i] <- eval
    }
    temp <- c(temp, vector)
    averages <- c(averages, (sum / n))
  }
  result <- matrix(c(temp),nrow = n,ncol = length(values))
  write.table(result, file = "resultsIterations.csv", row.names=FALSE, na="",
      col.names=FALSE, sep=";")

  if (graphs) {
    plot(values, averages,
         main="Function values for different number of iterations",
         ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
         type="l", col="red", xlab="params", ylab="value")
    abline(globalOpt,0, col="green")
  }

}

if (graphs) {
```

```
  summary(GAmin)
  filled.contour(x, y, z, color.palette = jet.colors, nlevels = 24,
     plot.axes = {
       axis(1);
       axis(2);
       points(solution[1,1], solution[1,2], pch = 3, cex = 5, col = "black", lwd
           = 2)
     }
  )
  plot(GAmin)
}
```

2.1 Parametryzacja skryptu

Parametryzacji podlega jedynie algorytm genetyczny. Wybór funkcji do optymalizacji odbywa się przez podanie jej nazwy. Pozostałe dane są odczytywane z pakietu „globalOptTests".

3 Przebieg badań

Do badań zostały wybrane funkcje o różnych wymiarach zaczynając na 2 kończąc na 20. Poniżej wymieniono te funkcje wraz z ilością wymiarów podaną w nawiasie.

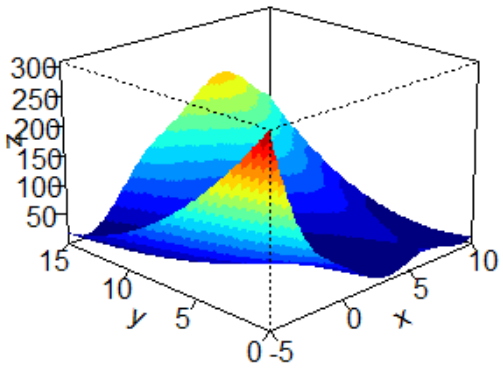
- Branin (2)
- Gulf (3)
- CosMix4 (4)
- EMichalewicz (5)
- Hartman6 (6)
- PriceTransistor (9)
- Schwefel (10)
- Zeldasine20 (20)

Każdy pomiar przeprowadzano 10-krotnie wyniki uśredniając. Domyślne parametry wynosiły kolejno:

- rozmiar populacji - 50
- liczba iteracji - 100
- p. mutacji - 0,1
- p. krzyżowania - 0,8

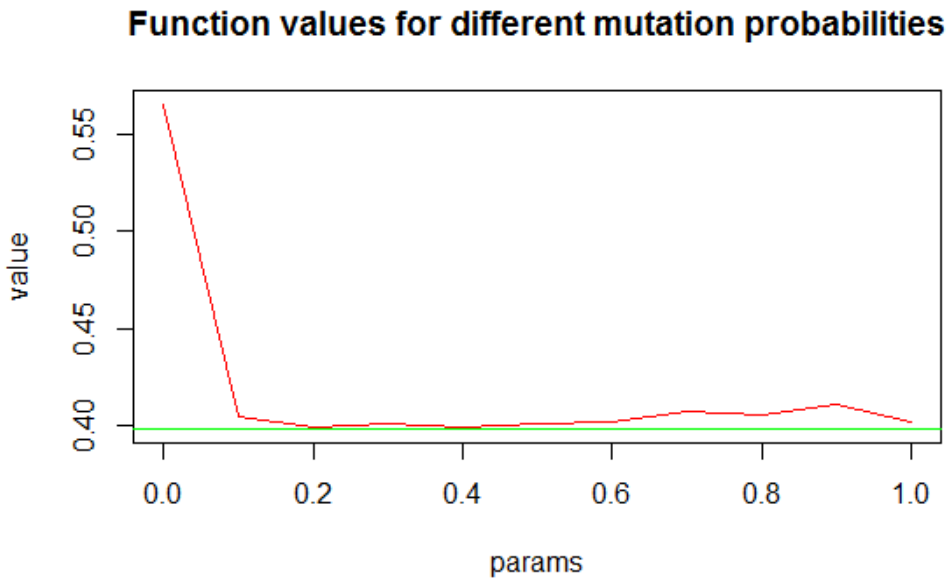
3.1 Branin (2 parametry)

Branin jest funkcją z dwoma parametrami. Na ilustracji (rys. 1) przedstawiono jej wykres.

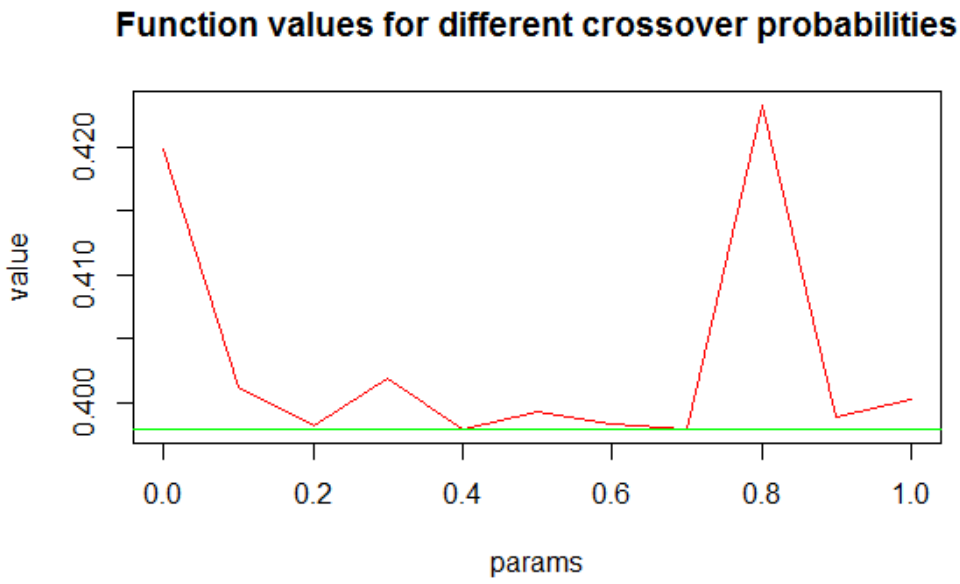


Rysunek 1: Wykres funkcji Branin (d=2)

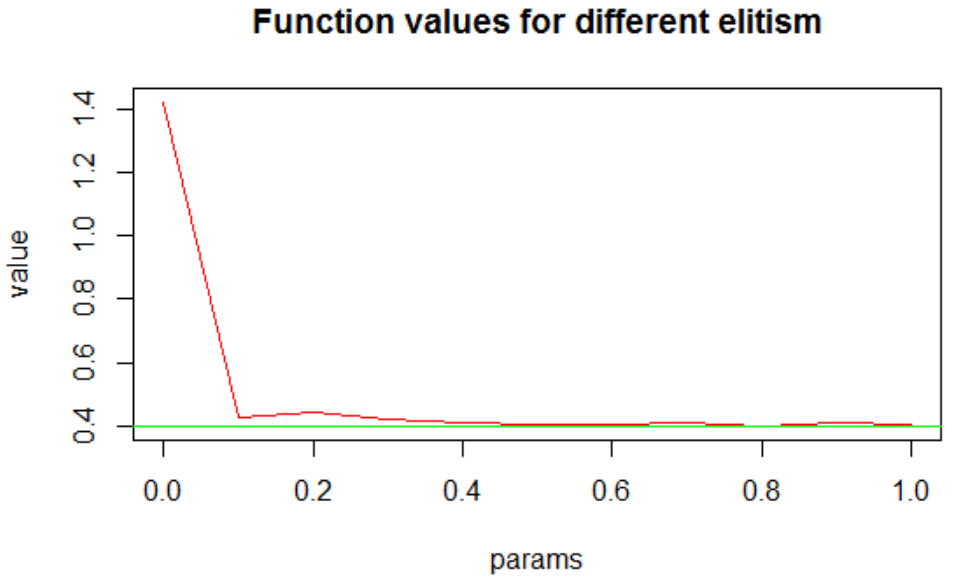
Na kolejnych stronach zamieszczono wyniki pomiarów dla różnych wartości parametrów algorytmu genetycznego.



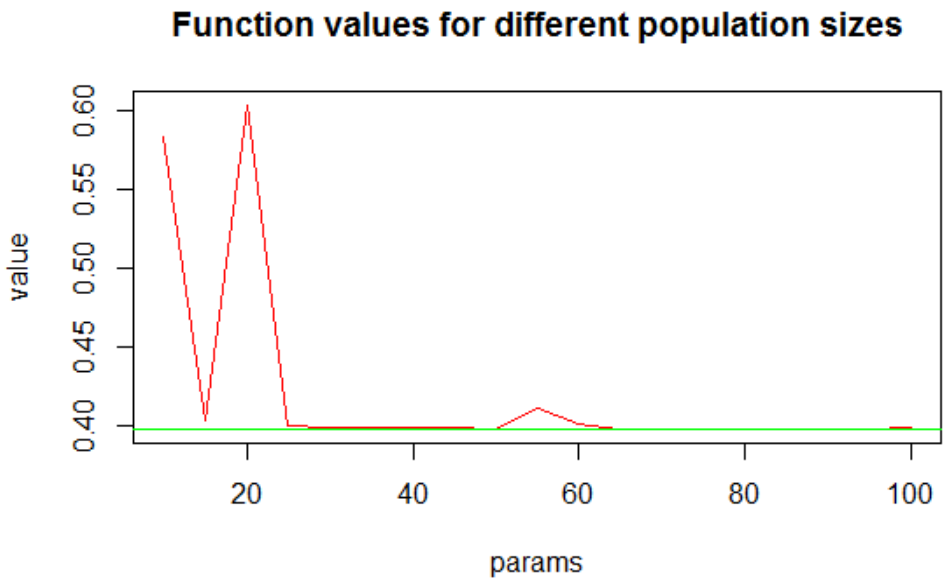
Rysunek 2: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



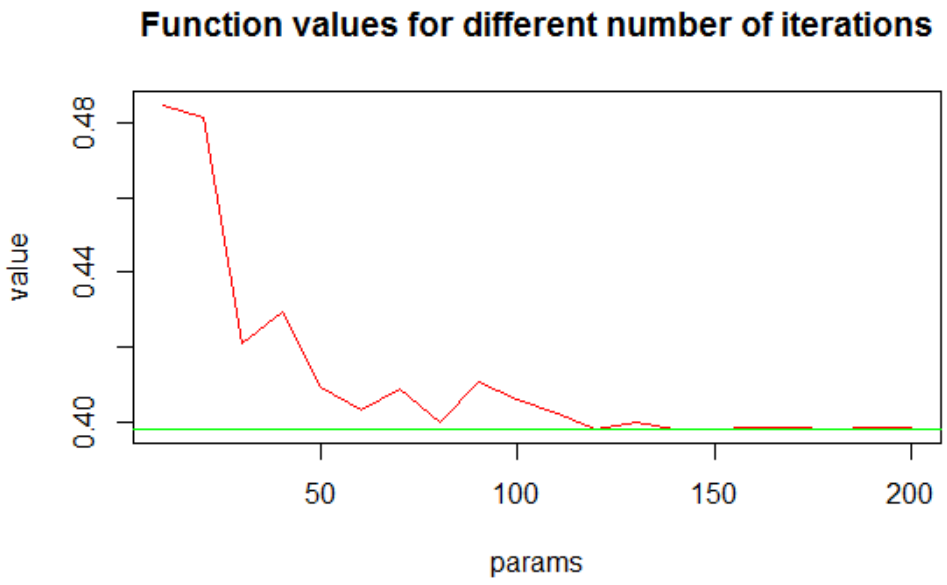
Rysunek 3: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowanie



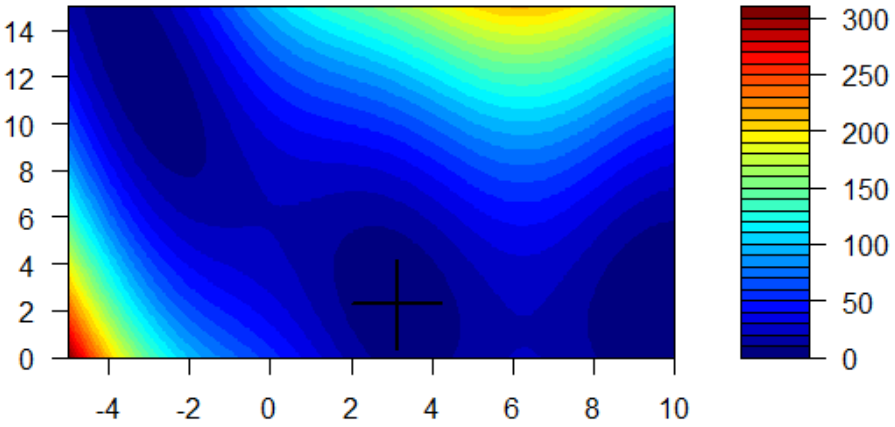
Rysunek 4: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 5: Wartość znalezionego optimum w zależności od rozmiarów populacji



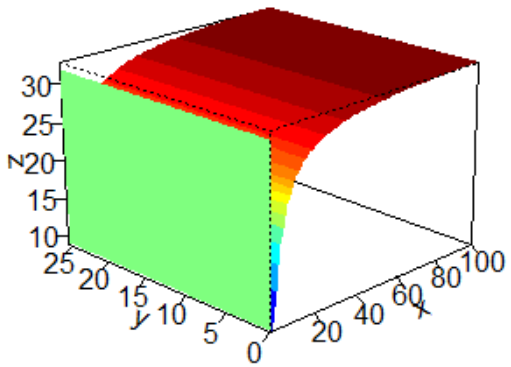
Rysunek 6: Wartość znalezionego optimum w zależności od ilości iteracji



Rysunek 7: Poglądowa lokalizacja najlepszego znalezionego optimum

3.2 Gulf (3 parametry)

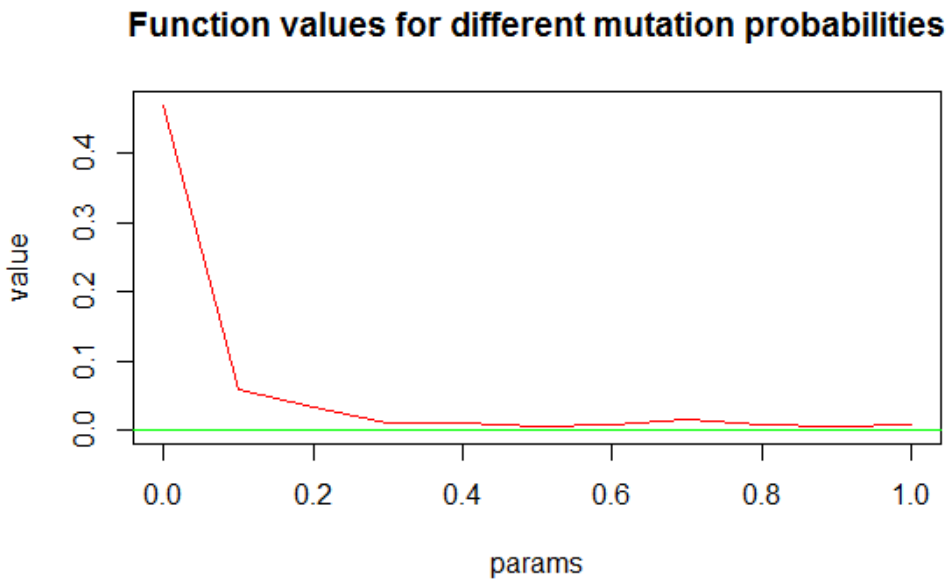
Gulf jest funkcją przyjmującą trzy parametry. Na ilustracji (rys. 8) przedstawiono jej wykres dla pierwszych dwóch wymiarów.



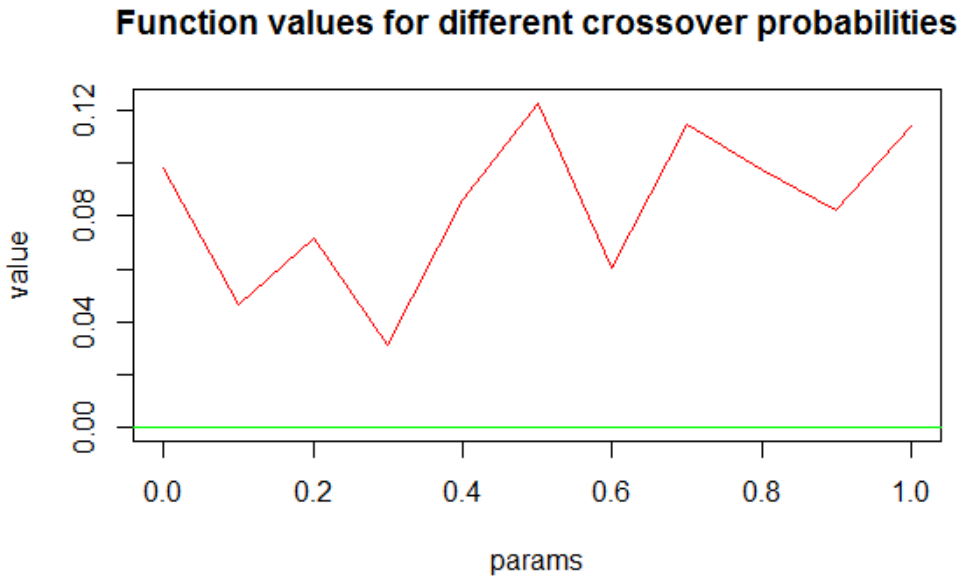
Rysunek 8: Wykres funkcji Gulf (d=3)

Na kolejnych stronach zamieszczono wyniki pomiarów dla różnych wartości parametrów

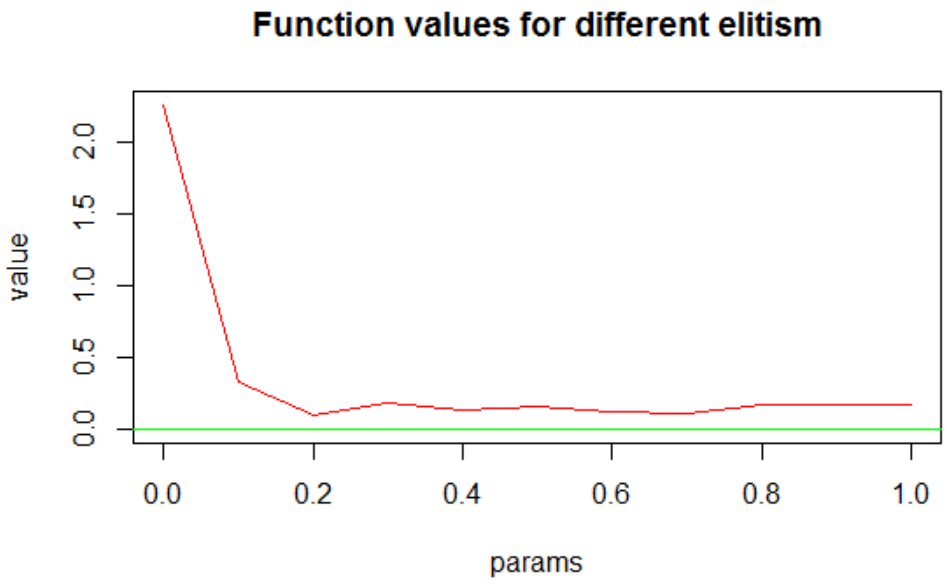
algorytmu genetycznego.



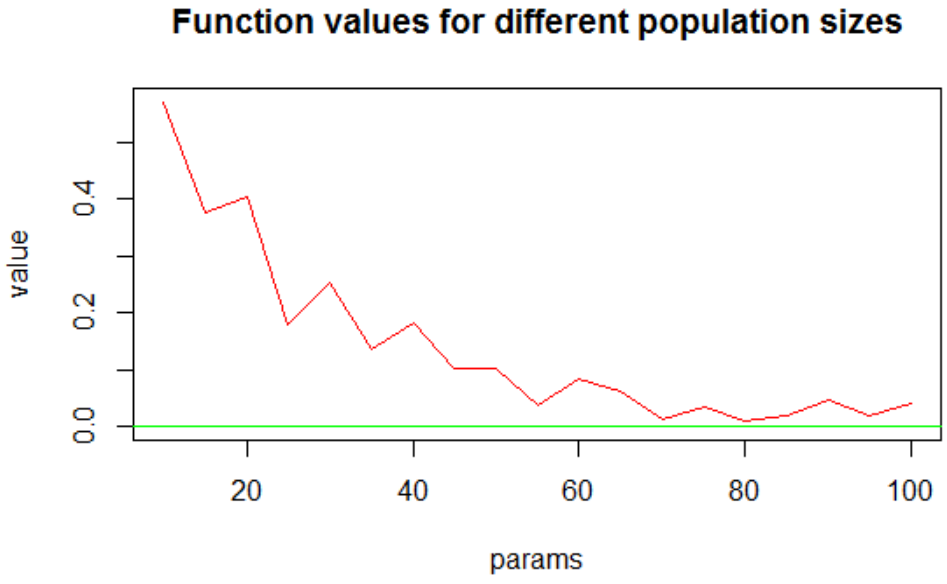
Rysunek 9: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



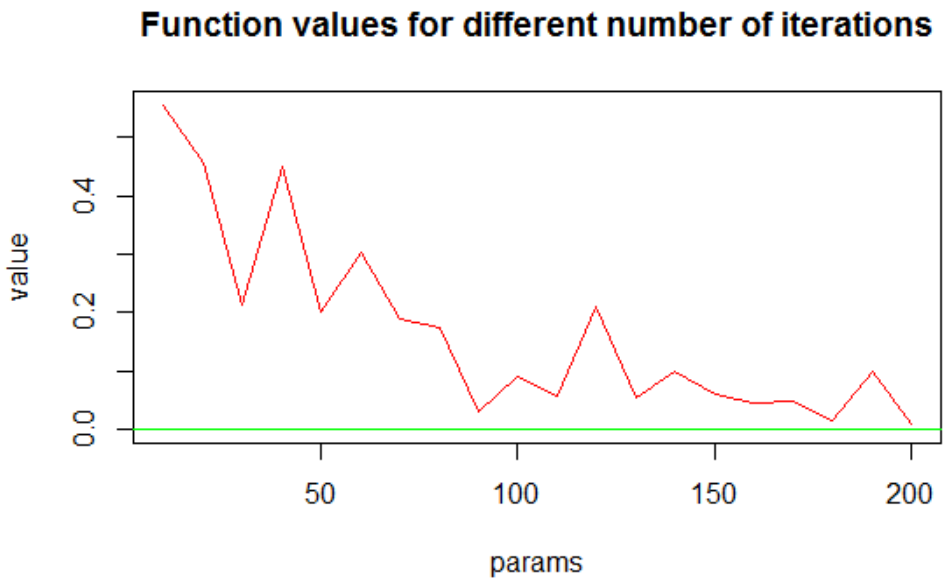
Rysunek 10: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowanie



Rysunek 11: Wartość znalezionego optimum w zależności od przyjętego elityzmu

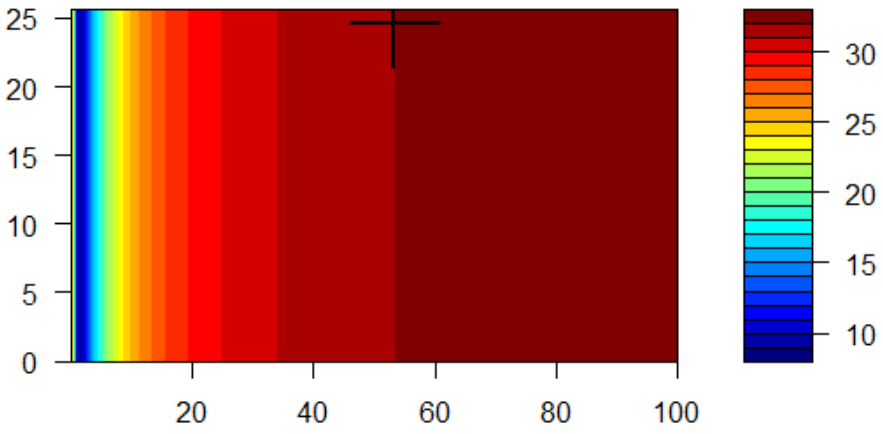


Rysunek 12: Wartość znalezionego optimum w zależności od rozmiarów populacji



Rysunek 13: Wartość znalezionego optimum w zależności od ilości iteracji

Jak możemy zauważyć na ilustracji poniżej (rys. 14) przedstawiona lokalizacja optimum nie jest poprawna, gdyż optymalizacji poddano wersję z 3 parametrami. Ogólnie rzecz biorąc gdyby 3 wymiar przedstawić w postaci gradientu kolorystycznego wtedy byłaby to poprawna lokalizacja niemniej trudna dla intuicyjnego sprawdzenia.



Rysunek 14: Poglądowa lokalizacja najlepszego znalezionego optimum

3.3 CosMix4 (4 parametry)

Test

3.4 EMichalewicz (5 parametrów)

Test

3.5 Hartman6 (6 parametrów)

Test

3.6 PriceTransistor (9 parametrów)

Test

3.7 Schwefel (10 parametrów)

Test

3.8 Zeldasine20 (20 parametrów)

Test

4 Podsumowanie

Test

Akapit

Literatura

[1] Artur Suchwałko "Wprowadzenie do R dla programistów innych języków" https://cran.r-project.org/doc/contrib/R-dla-programistow-innych-jezykow.pdf