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

```
39
40 dim <- getProblemDimen(funcName)
41 B <- matrix(unlist(getDefaultBounds(funcName)),ncol=dim,byrow=TRUE)
42 f <- function(xx) goTest(par=c(xx, rep(0, dim-length(xx))), fnName=funcName,
       checkDim = TRUE)
43 globalOpt <- getGlobalOpt(funcName)
44
45 if (graphs) {
46   xprobes <- abs(B[2,1] - B[1,1]) / quality
47   yprobes <- abs(B[2,2] - B[1,2]) / quality
48   x <- seq(B[1,1], B[2,1], by = xprobes)
49   y <- seq(B[1,2], B[2,2], by = yprobes)
50   z <- outer(x, y, Vectorize(function(x,y) f(c(x,y))))
51   nbcol = 100
52   color = rev(rainbow(nbcol, start = 0/6, end = 4/6))
53   zcol  = cut(z, nbcol)
54   persp3d(x, y, z, theta=50, phi=25, expand=0.75, col=color[zcol],
55           ticktype="detailed",axes=TRUE)
56   persp3D(x, y, z, theta = -45, phi = 20, color.palette = jet.colors)
57 }
58
59 if (isSingleTest) {
60
61   vector <- rep(NA,n)
62   for (i in 1:n) {
63     GAmin <- ga(type = "real-valued", fitness = function(xx) -f(xx),
64                 min = c(B[1,]), max = c(B[2,]),
65                 popSize = GAPopulation, maxiter = GAIterations,
66                 pmutation = GAMutations, pcrossover = GACrossovers)
67     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
68     vector[i] <- f(solution[1,])
69   }
70   result <- matrix(c(vector),nrow = n,ncol = 1)
71   write.table(result, file = "resultsSingle.csv", row.names=FALSE, na="",
         col.names=FALSE, sep=";")
72
73 } else {
74
75   gMin <- .Machine$integer.max
76   gBest <- NA
77
78   temp <- c()
79   values <- mutationTests
80   averages <- c()
81   for (mutation in values) {
82     sum <- 0
83     vector <- rep(NA,n)
84     for (i in 1:n) {
85       GAmin <- ga(type = "real-valued",
86                   fitness = function(xx) -f(xx),
87                   min = c(B[1,]), max = c(B[2,]),
88                   popSize = GAPopulation, maxiter = GAIterations,
89                   pmutation = mutation, pcrossover = GACrossovers)
90       solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
91       eval <- f(solution[1,])
92       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")
}
```

```
147
148 temp <- c()
149 values <- elitismTests
150 averages <- c()
151 for (elitism in values) {
152   sum <- 0
153   vector <- rep(NA,n)
154   for (i in 1:n) {
155     GAmin <- ga(type = "real-valued",
156                 fitness = function(xx) -f(xx),
157                 min = c(B[1,]), max = c(B[2,]),
158                 popSize = GAPopulation, maxiter = GAIterations,
159                 pmutation = GAMutations, pcrossover = GACrossovers, elitism =
                        elitism)
160     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
161     eval <- f(solution[1,])
162     if (eval < gMin) {
163       gMin <- eval
164       gBest <- GAmin
165     }
166     sum <- sum + eval
167     vector[i] <- eval
168   }
169   temp <- c(temp, vector)
170   averages <- c(averages, (sum / n))
171 }
172 result <- matrix(c(temp),nrow = n,ncol = length(values))
173 write.table(result, file = "resultsElitism.csv", row.names=FALSE, na="",
        col.names=FALSE, sep=";")
174
175 if (graphs) {
176   plot(values, averages,
177        main="Function values for different elitism",
178        ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
179        type="l", col="red", xlab="params", ylab="value")
180   abline(globalOpt,0, col="green")
181 }
182
183 temp <- c()
184 values <- populationTests
185 averages <- c()
186 for (population in values) {
187   sum <- 0
188   vector <- rep(NA,n)
189   for (i in 1:n) {
190     GAmin <- ga(type = "real-valued",
191                 fitness = function(xx) -f(xx),
192                 min = c(B[1,]), max = c(B[2,]),
193                 popSize = population, maxiter = GAIterations,
194                 pmutation = GAMutations, pcrossover = GACrossovers)
195     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
196     eval <- f(solution[1,])
197     if (eval < gMin) {
198       gMin <- eval
199       gBest <- GAmin
200     }
```

```
      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.

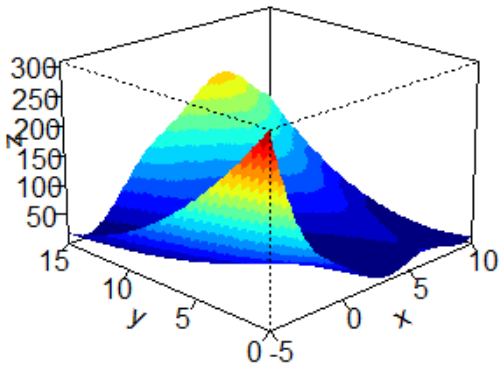
- Branin (2)
- Gulf (3)
- CosMix4 (4)
- EMichalewicz (5)
- Hartman6 (6)
- PriceTransistor (9)
- Schwefel (10)
- Zeldasine20 (20)

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

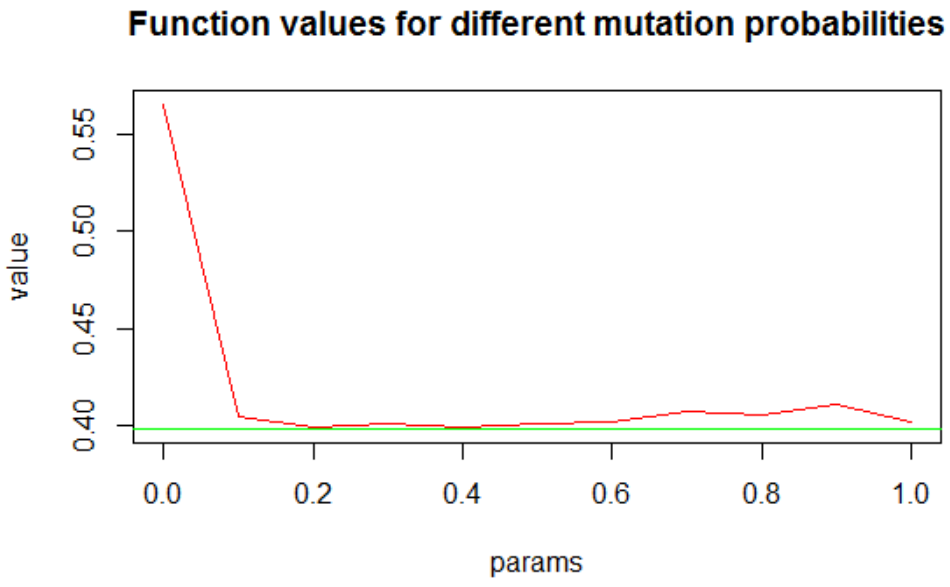
3.1 Branin (2 parametry)

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

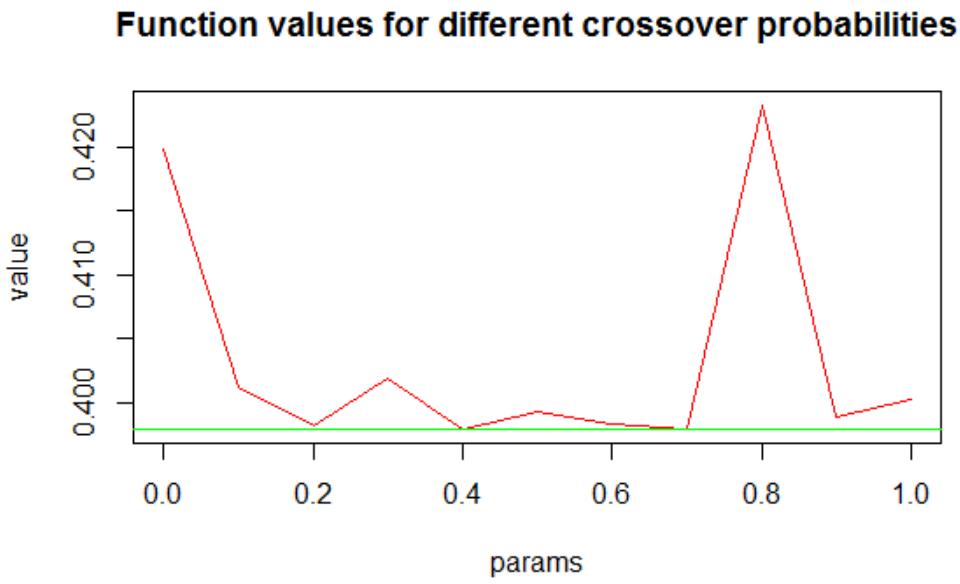


Rysunek 1: Wykres funkcji Branin (d=2)

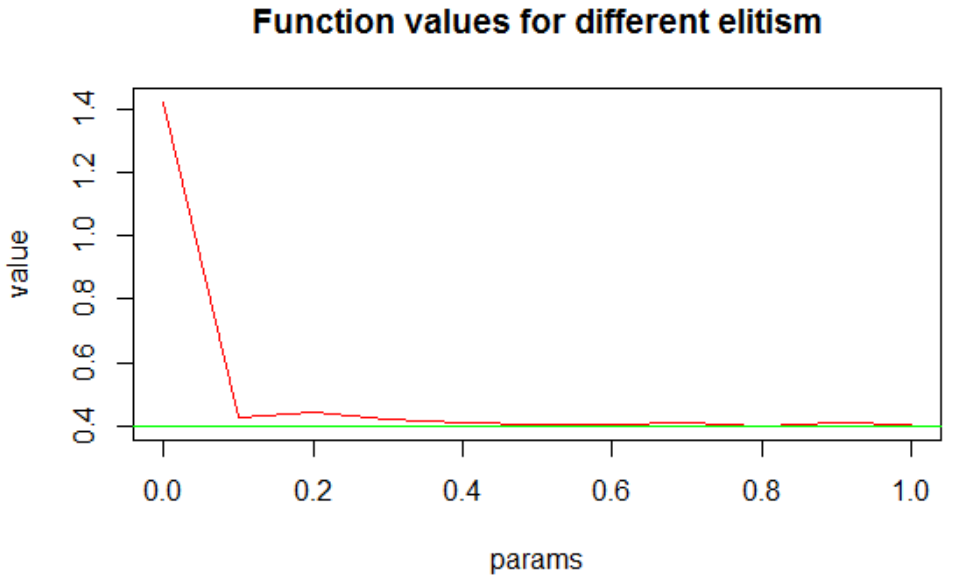
Na kolejnych stronach zamieszczono wyniki pomiarów dla różnych wartości parametrów algorytmu genetycznego.



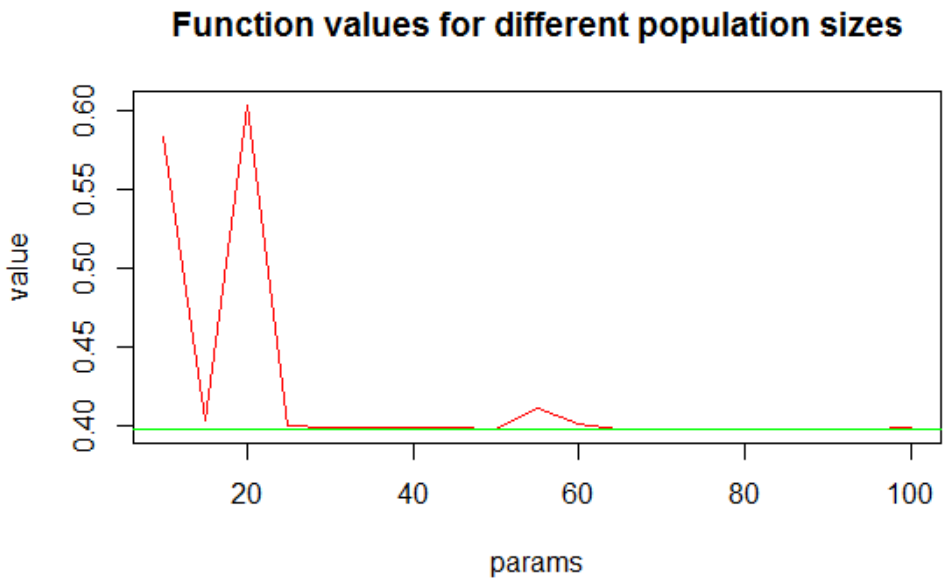
Rysunek 2: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



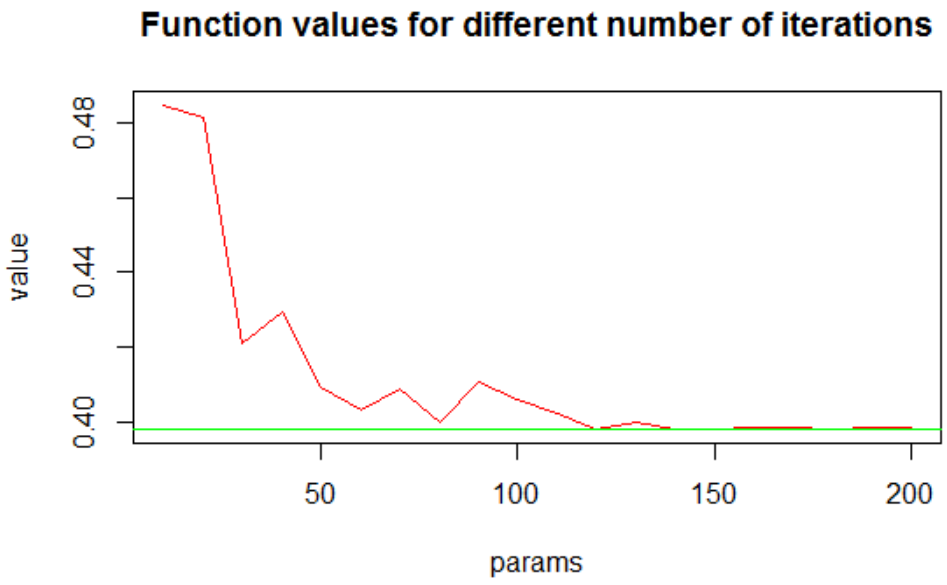
Rysunek 3: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowanie



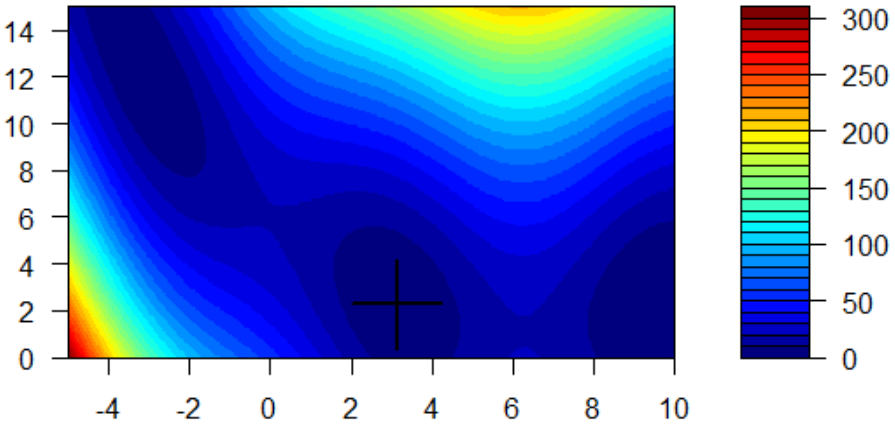
Rysunek 4: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 5: Wartość znalezionego optimum w zależności od rozmiarów populacji



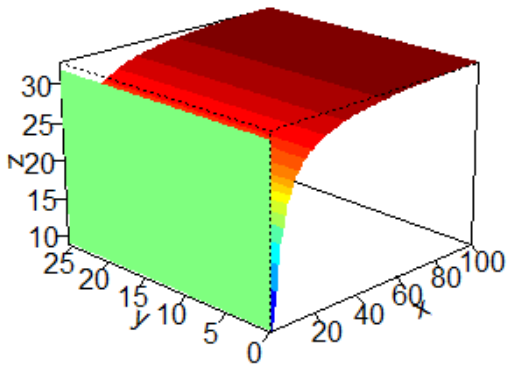
Rysunek 6: Wartość znalezionego optimum w zależności od ilości iteracji



Rysunek 7: Poglądowa lokalizacja najlepszego znalezionego optimum

3.2 Gulf (3 parametry)

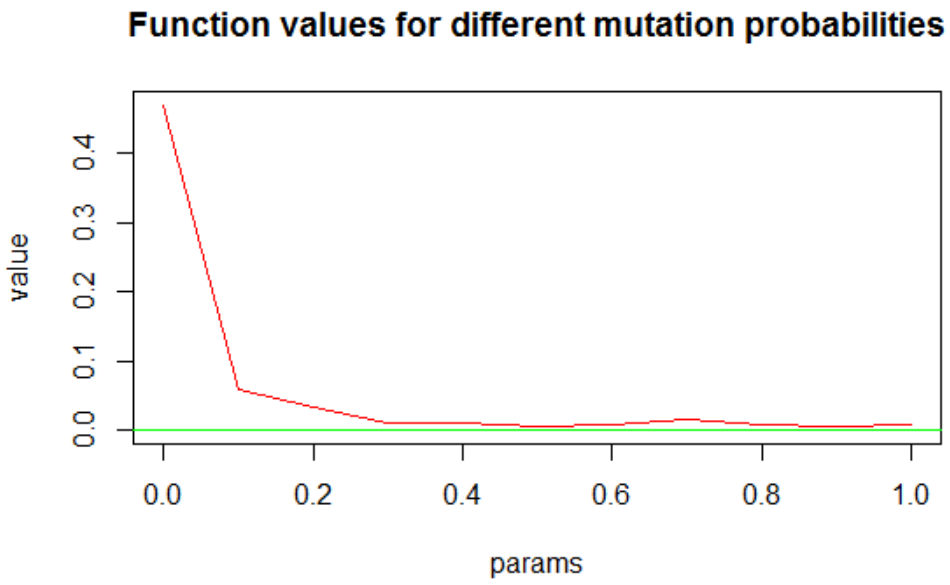
Gulf jest funkcją przyjmującą trzy parametry. Na ilustracji (rys. 8) przedstawiono jej wykres dla pierwszych dwóch wymiarów.



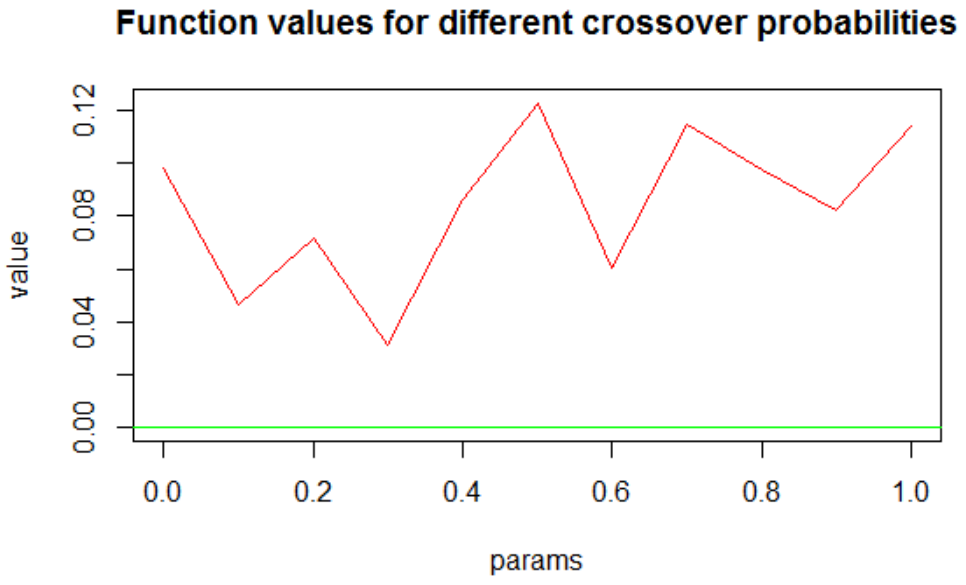
Rysunek 8: Wykres funkcji Gulf (d=3)

Na kolejnych stronach zamieszczono wyniki pomiarów dla różnych wartości parametrów

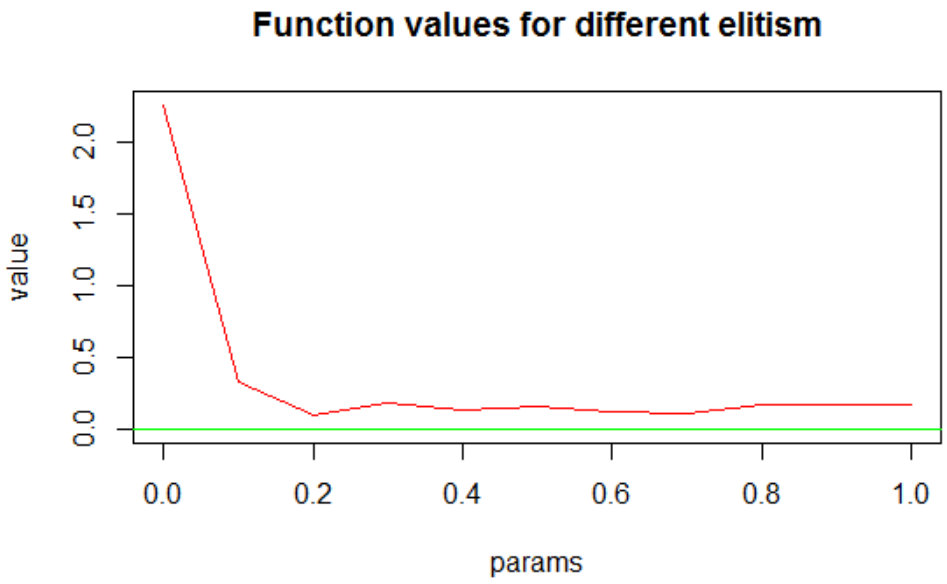
algorytmu genetycznego.



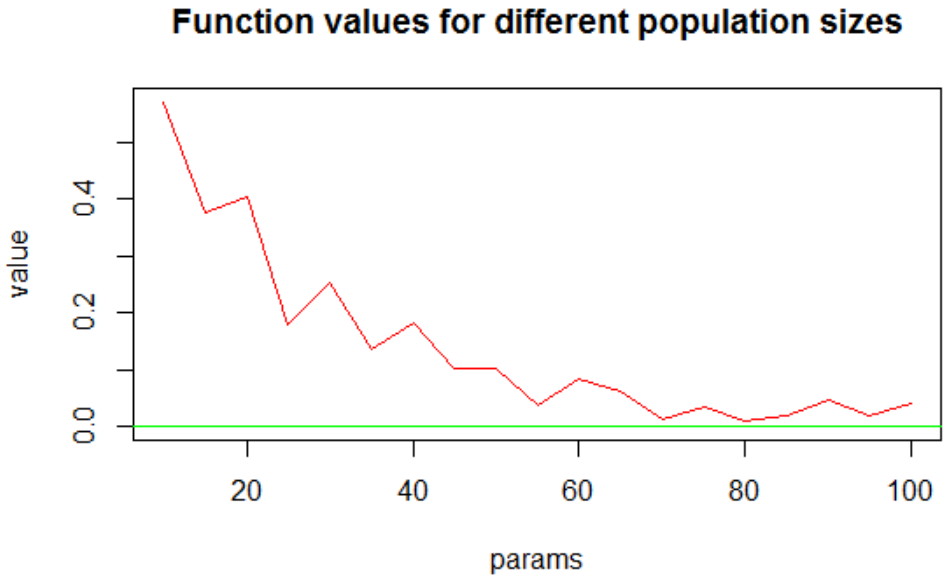
Rysunek 9: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



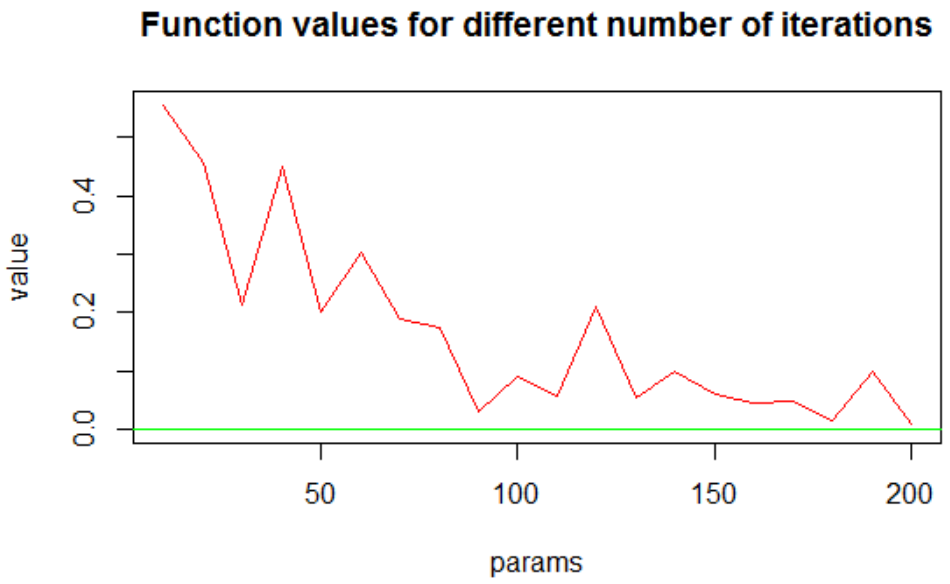
Rysunek 10: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowanie



Rysunek 11: Wartość znalezionego optimum w zależności od przyjętego elityzmu

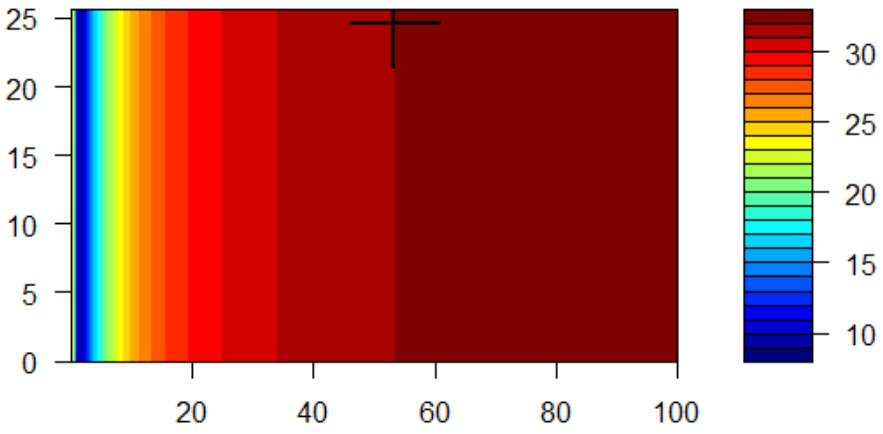


Rysunek 12: Wartość znalezionego optimum w zależności od rozmiarów populacji



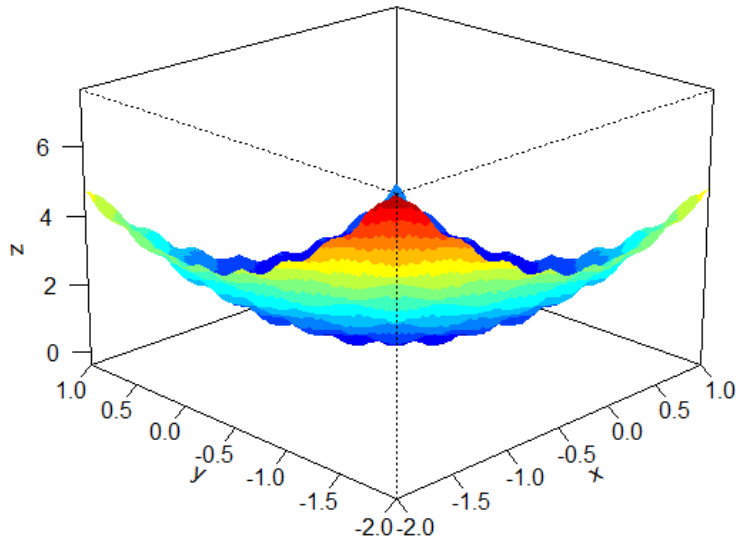
Rysunek 13: Wartość znalezionego optimum w zależności od ilości iteracji

Jak możemy zauważyć na ilustracji poniżej (rys. 42) 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.

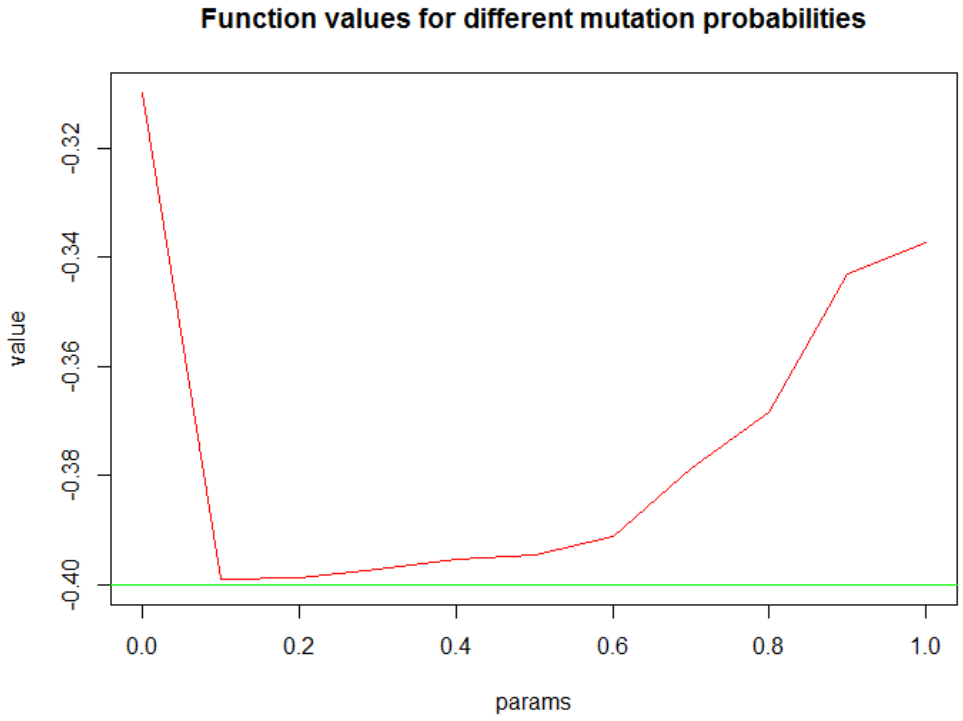


Rysunek 14: Poglądowa lokalizacja najlepszego znalezionego optimum

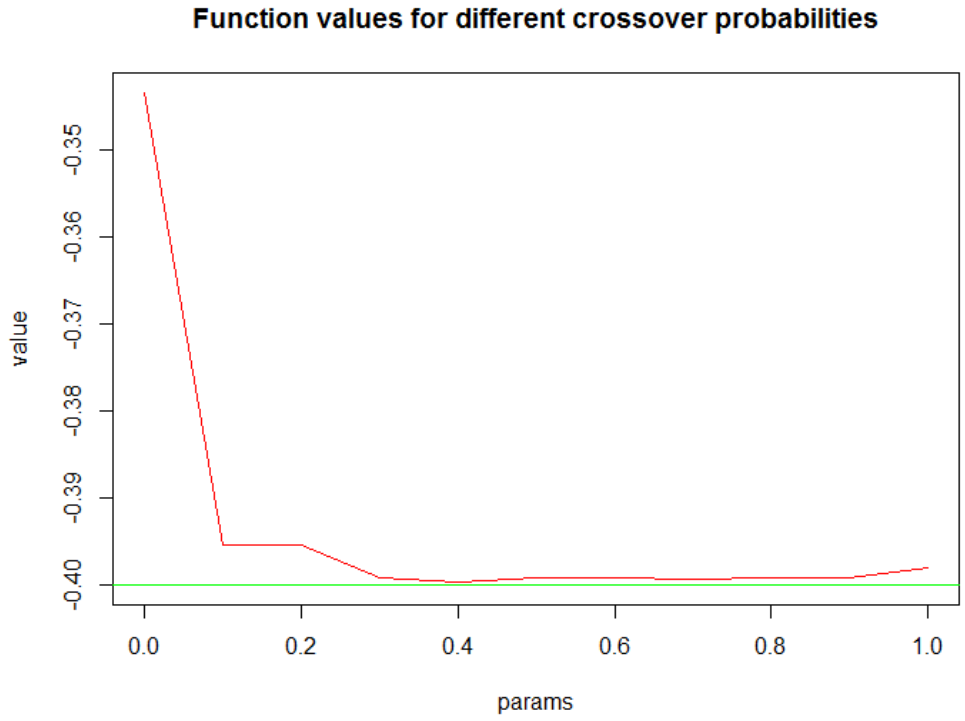
3.3 CosMix4 (4 parametry)



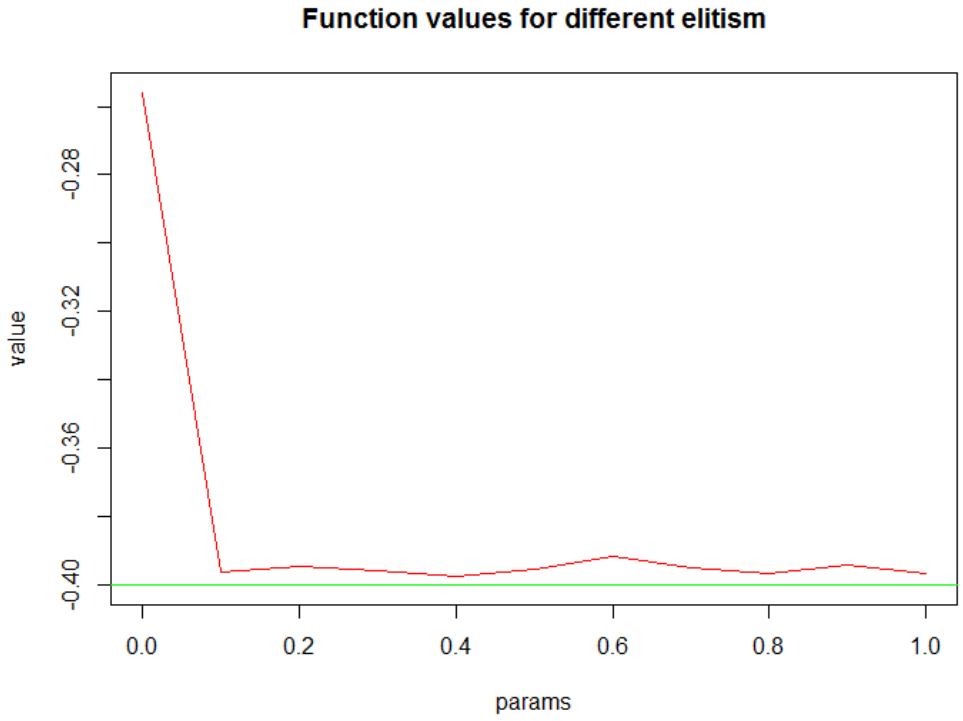
Rysunek 15: Poglądowa lokalizacja najlepszego znalezionego optimum



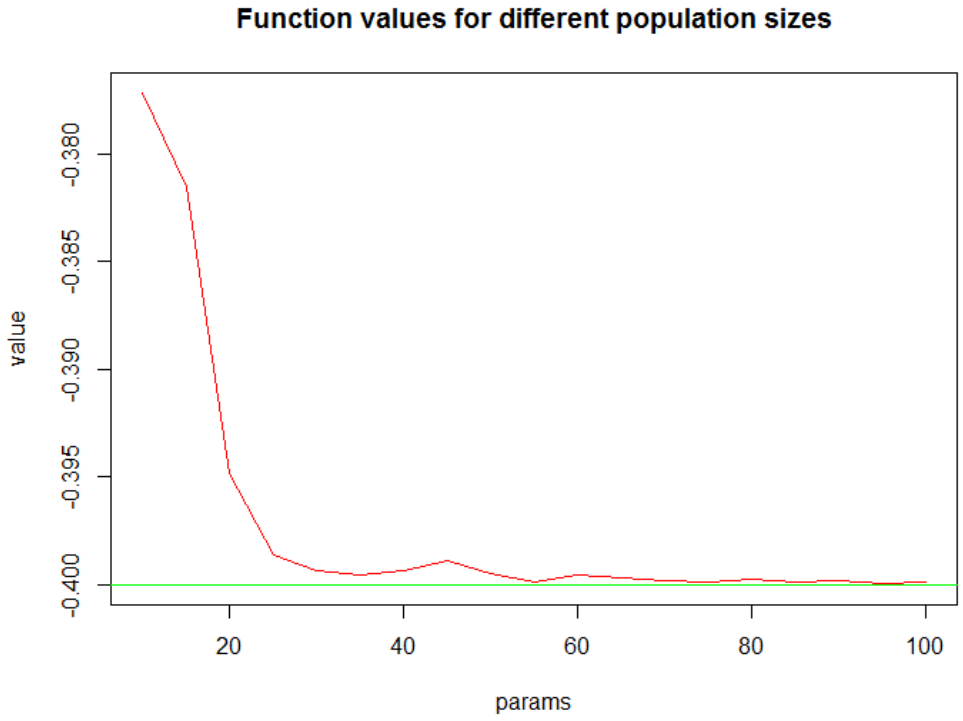
Rysunek 16: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



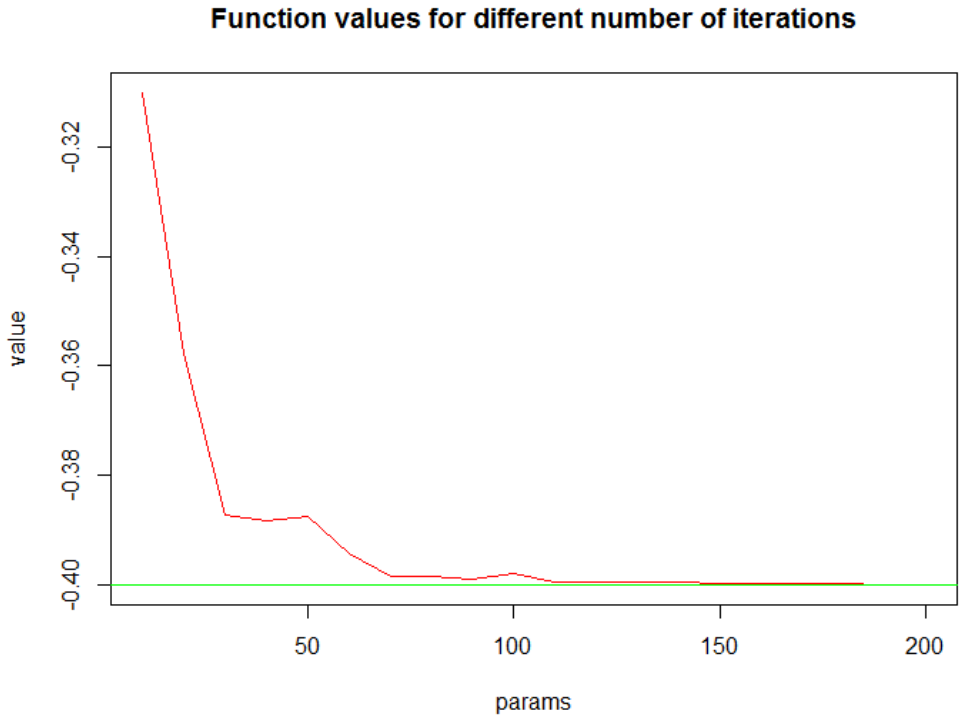
Rysunek 17: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



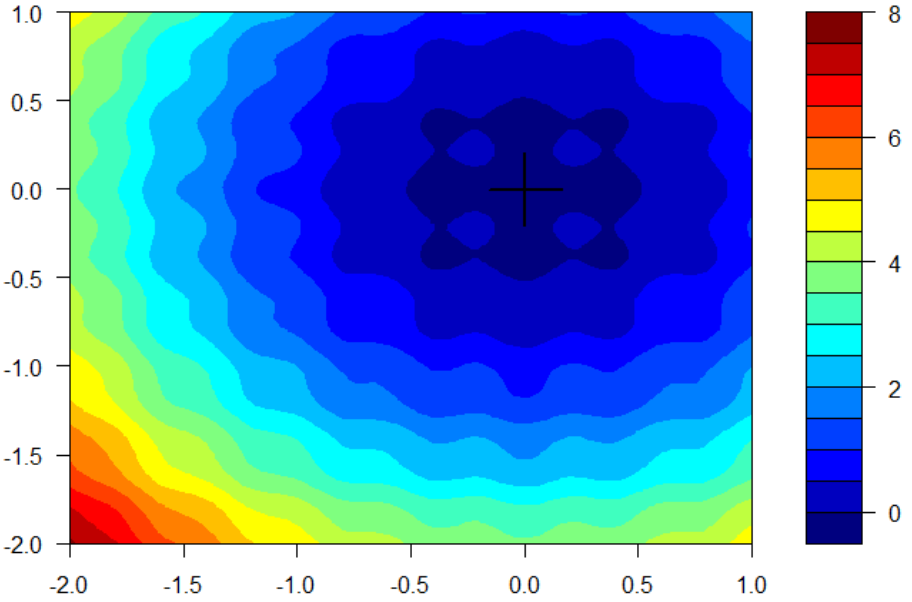
Rysunek 18: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 19: Wartość znalezionego optimum w zależności od rozmiarów populacji

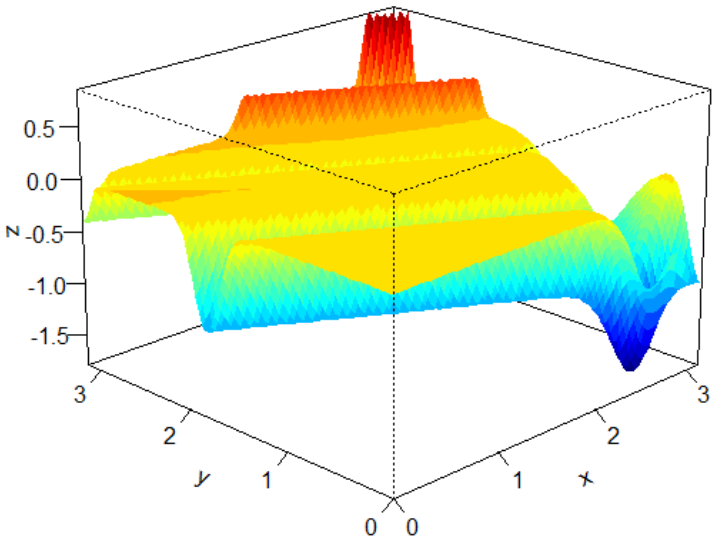


Rysunek 20: Wartość znalezionego optimum w zależności od iteracji

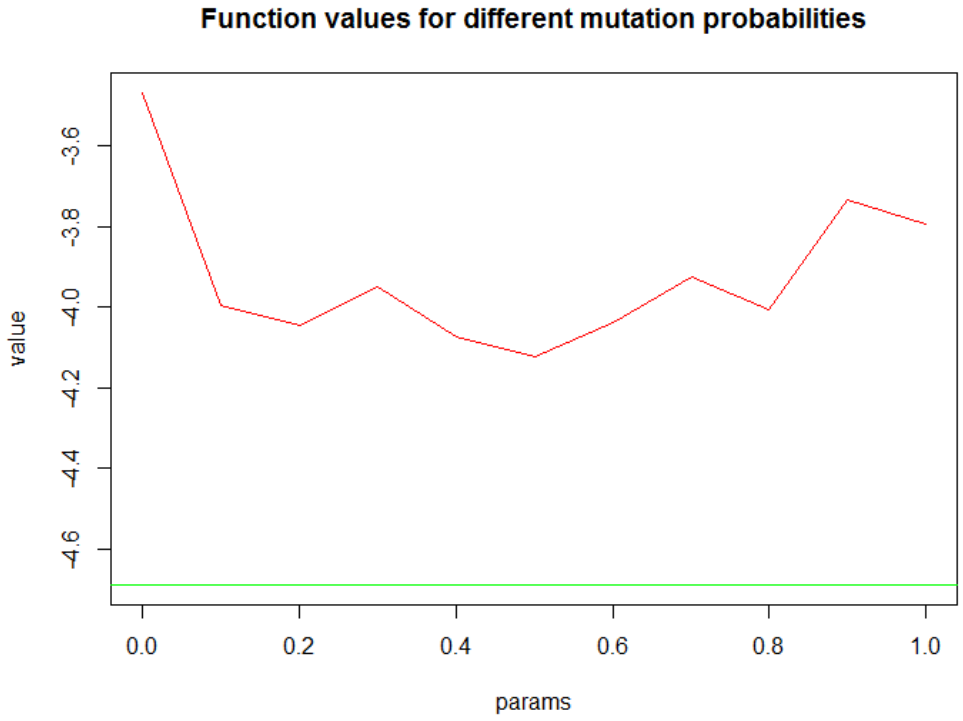


Rysunek 21: Poglądowa lokalizacja najlepszego znalezionego optimum

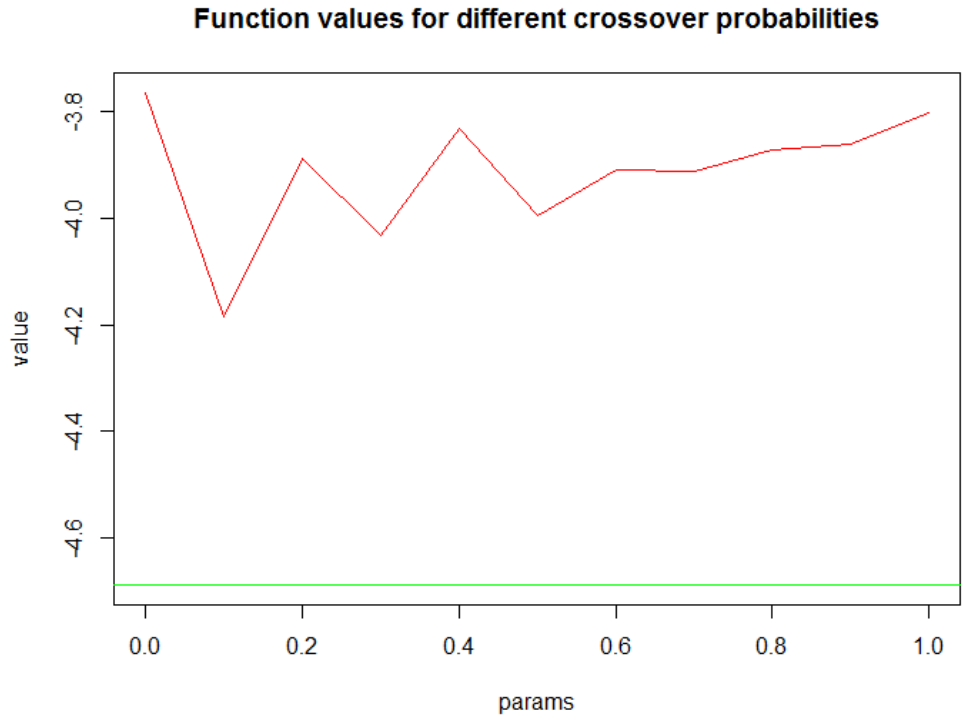
3.4 EMichalewicz (5 parametrów)



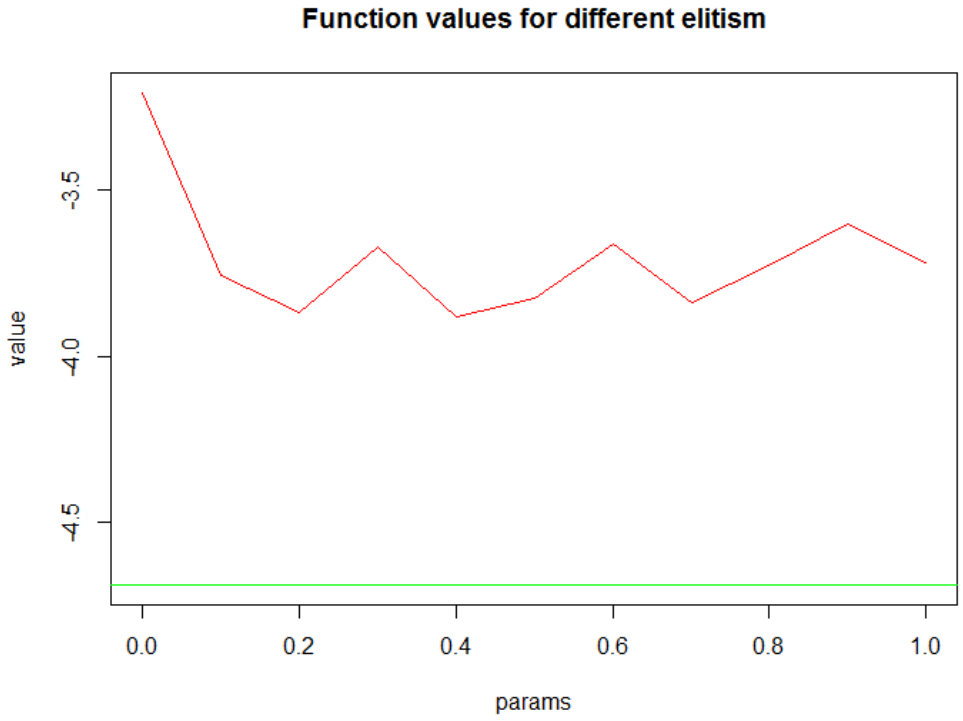
Rysunek 22: Poglądowa lokalizacja najlepszego znalezionego optimum



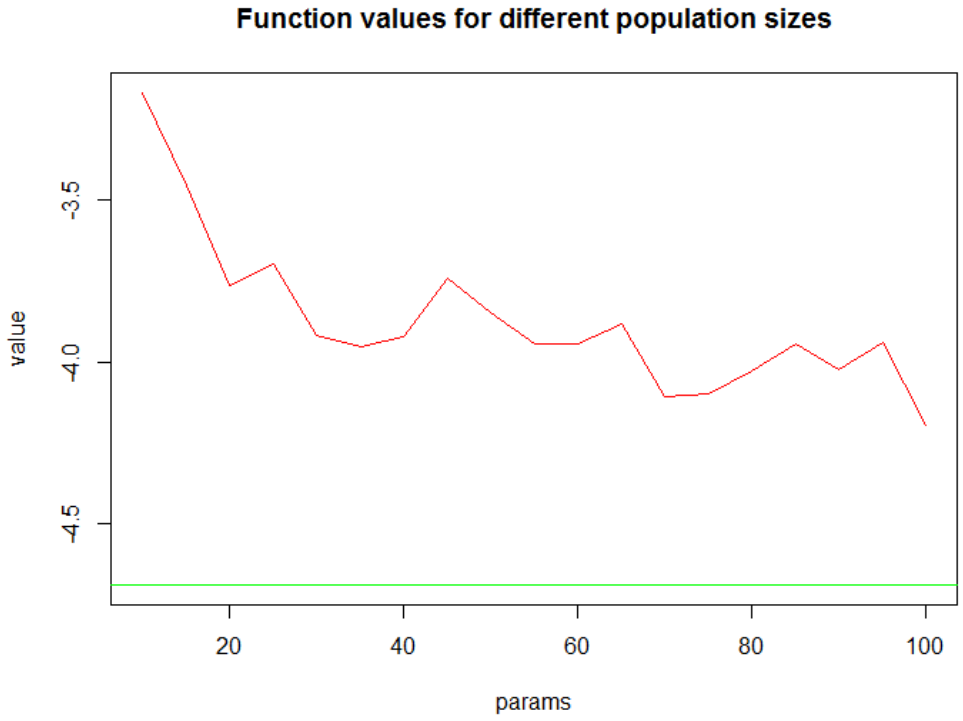
Rysunek 23: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



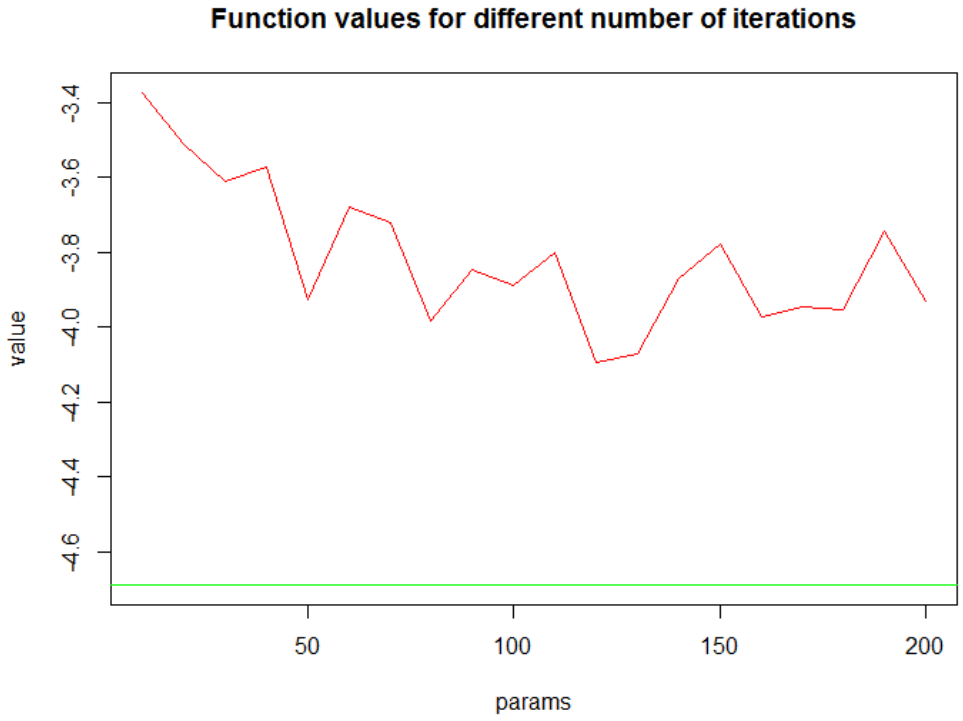
Rysunek 24: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



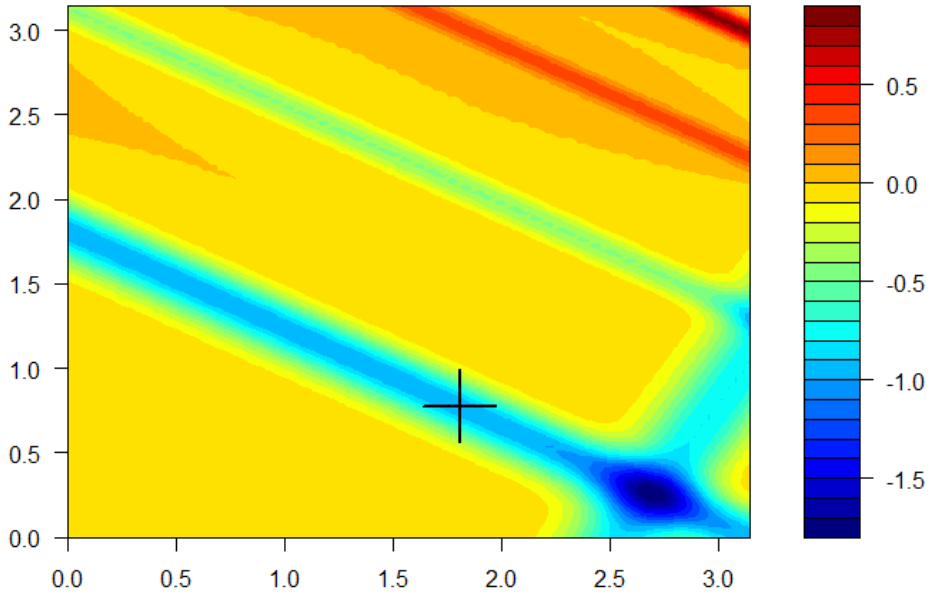
Rysunek 25: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 26: Wartość znalezionego optimum w zależności od rozmiarów populacji

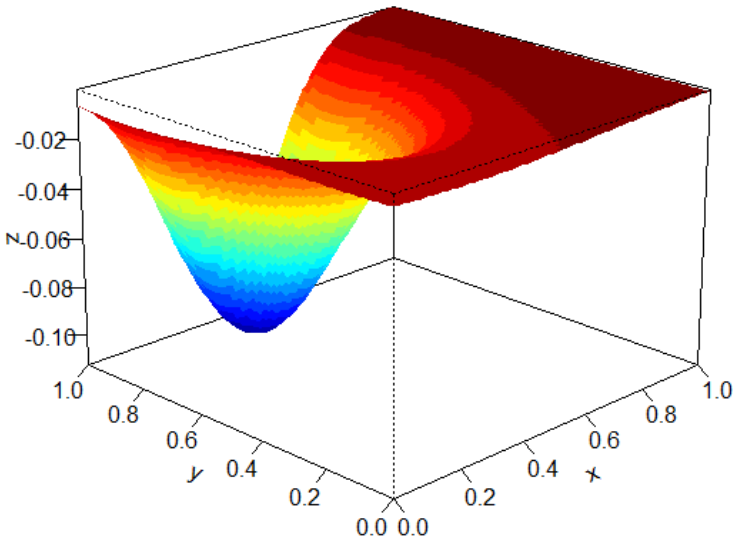


Rysunek 27: Wartość znalezionego optimum w zależności od iteracji

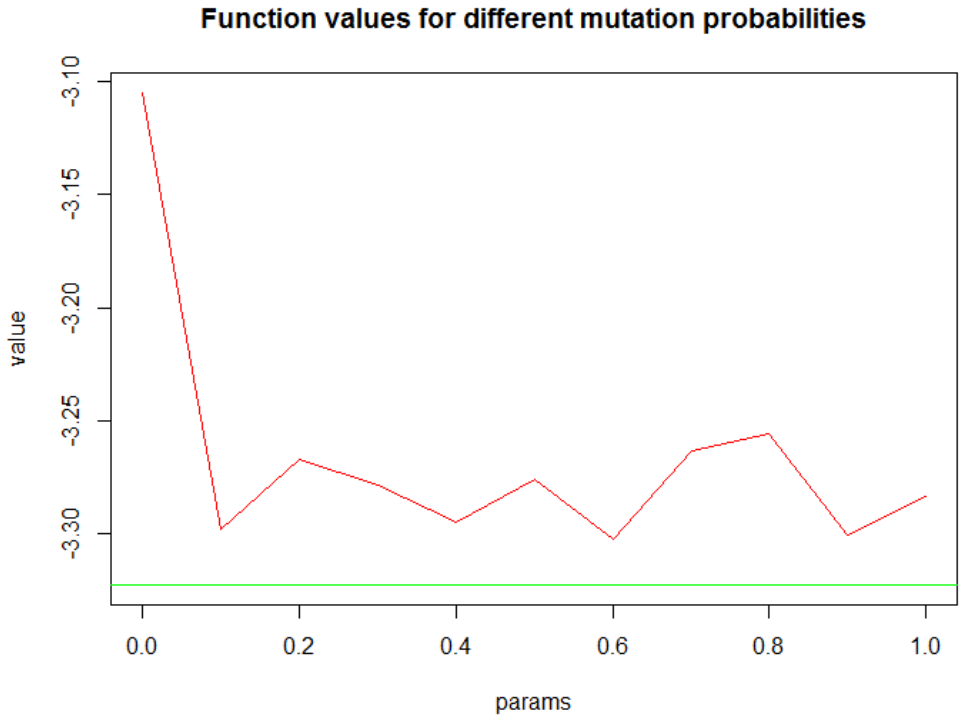


Rysunek 28: Poglądowa lokalizacja najlepszego znalezionego optimum

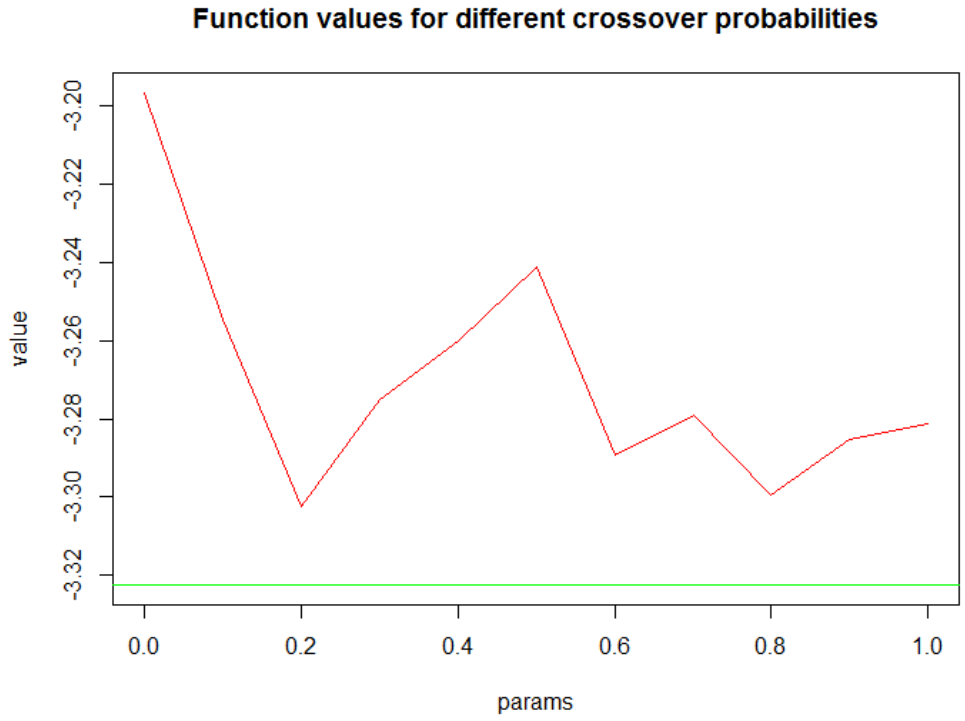
3.5 Hartman6 (6 parametrów)



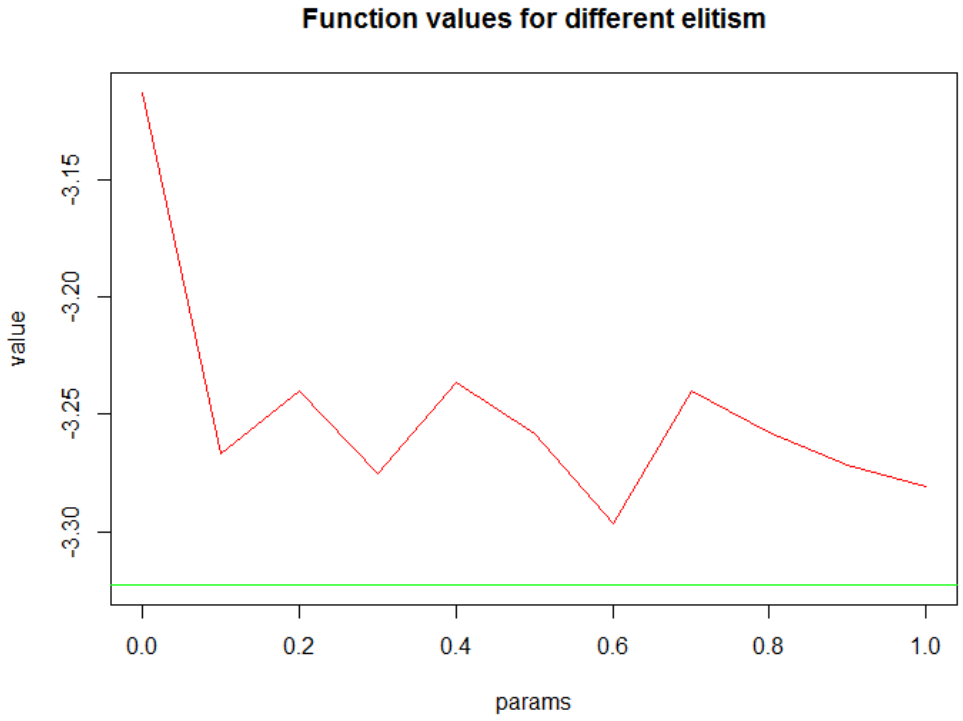
Rysunek 29: Poglądowa lokalizacja najlepszego znalezionego optimum



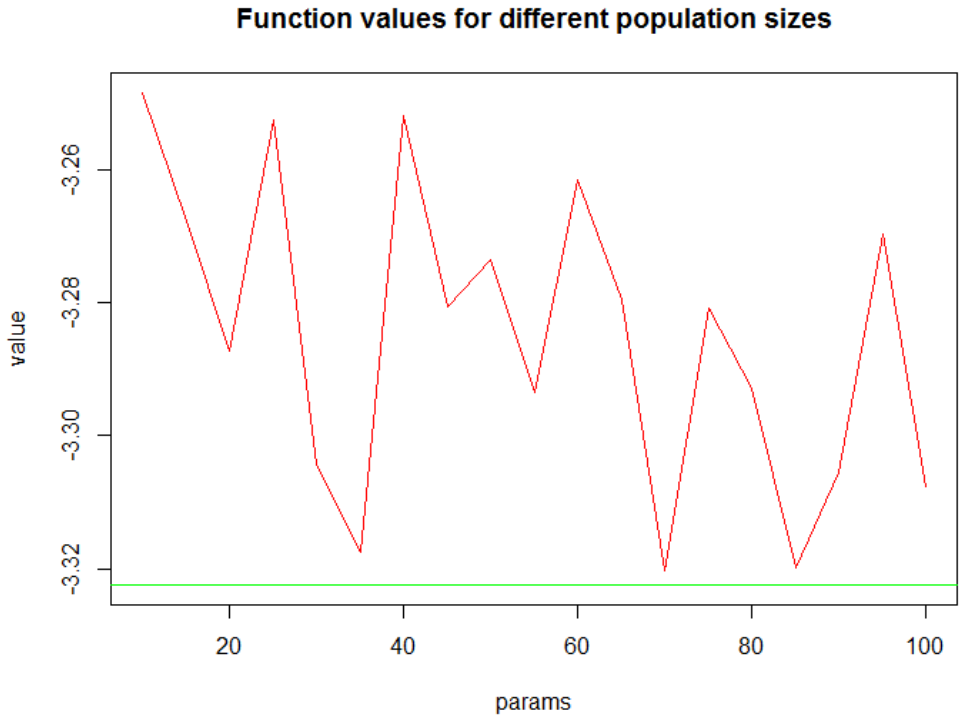
Rysunek 30: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



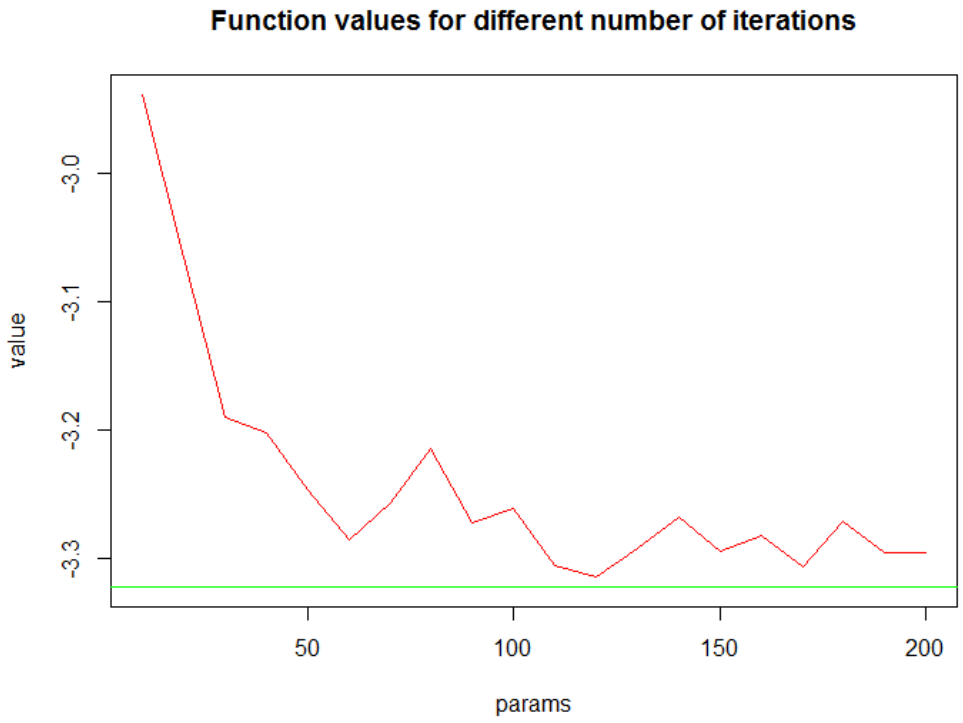
Rysunek 31: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



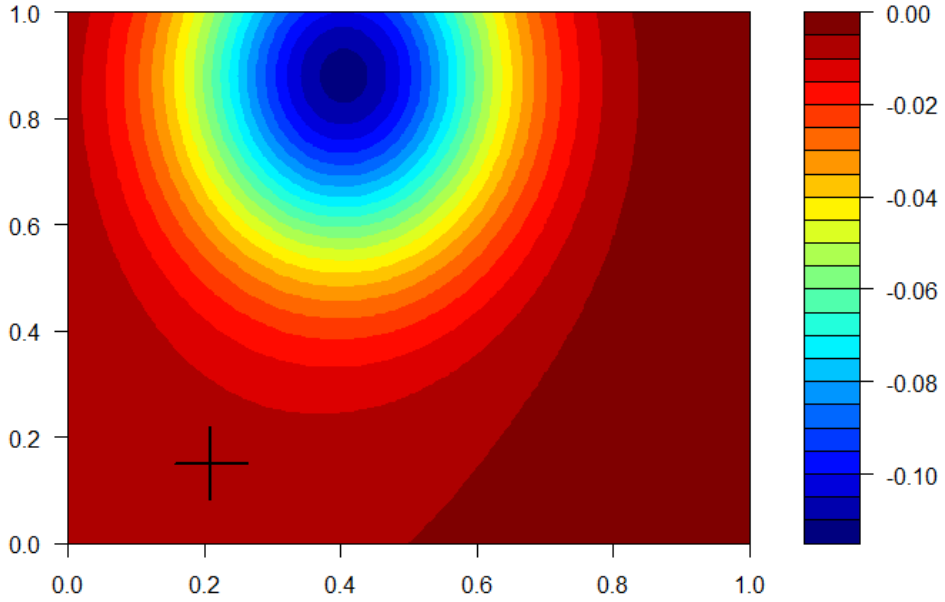
Rysunek 32: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 33: Wartość znalezionego optimum w zależności od rozmiarów populacji

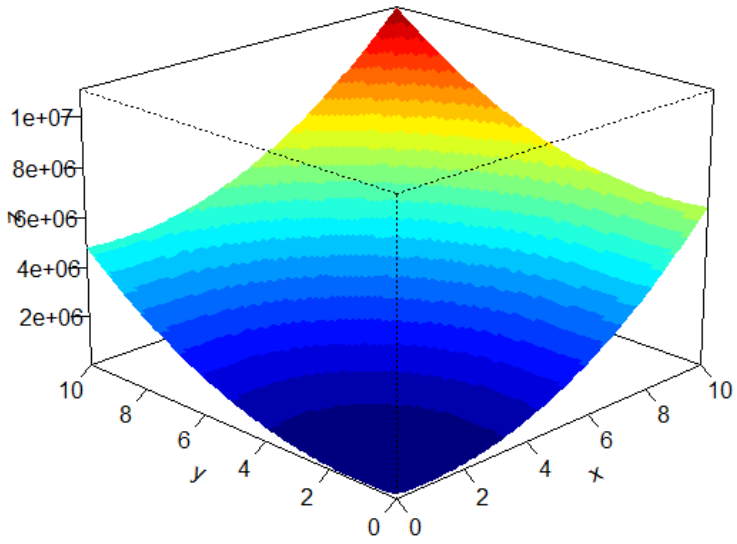


Rysunek 34: Wartość znalezionego optimum w zależności od iteracji

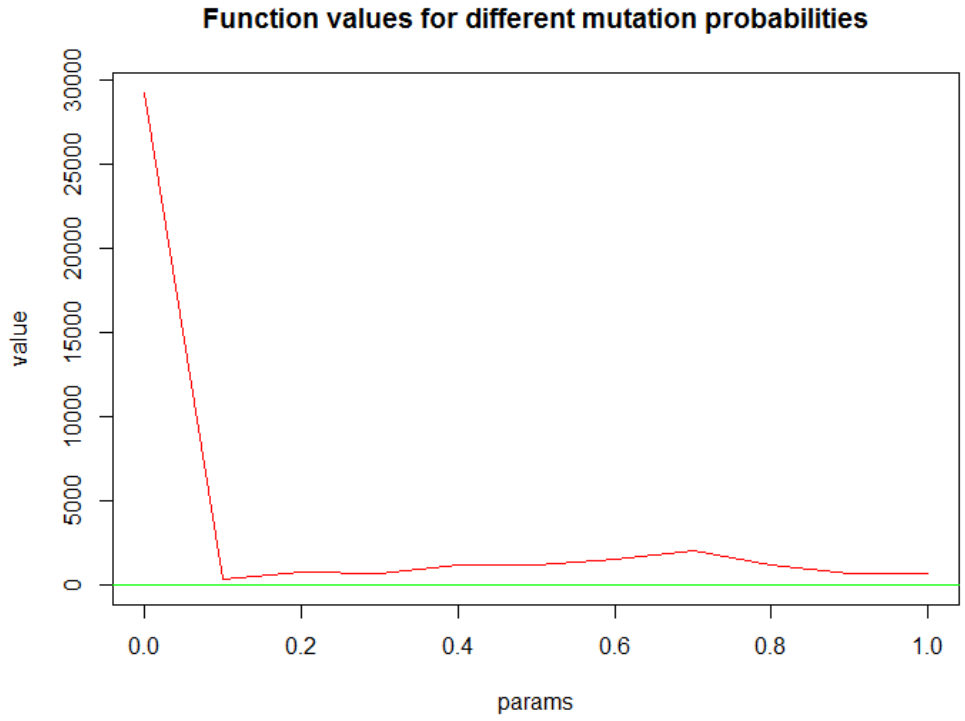


Rysunek 35: Poglądowa lokalizacja najlepszego znalezionego optimum

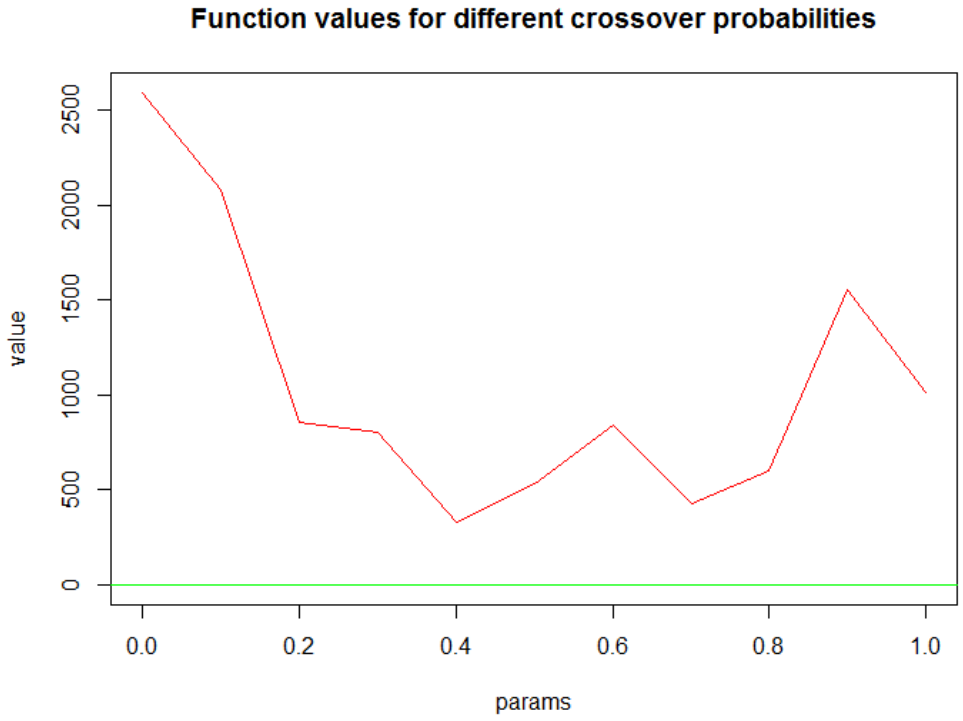
3.6 PriceTransistor (9 parametrów)



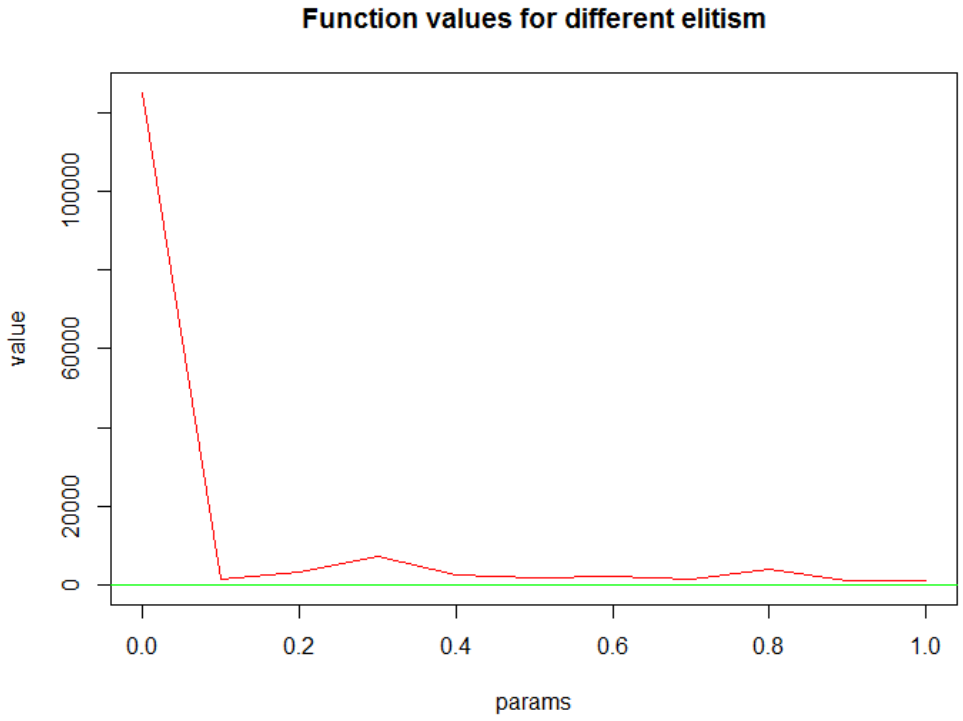
Rysunek 36: Poglądowa lokalizacja najlepszego znalezionego optimum



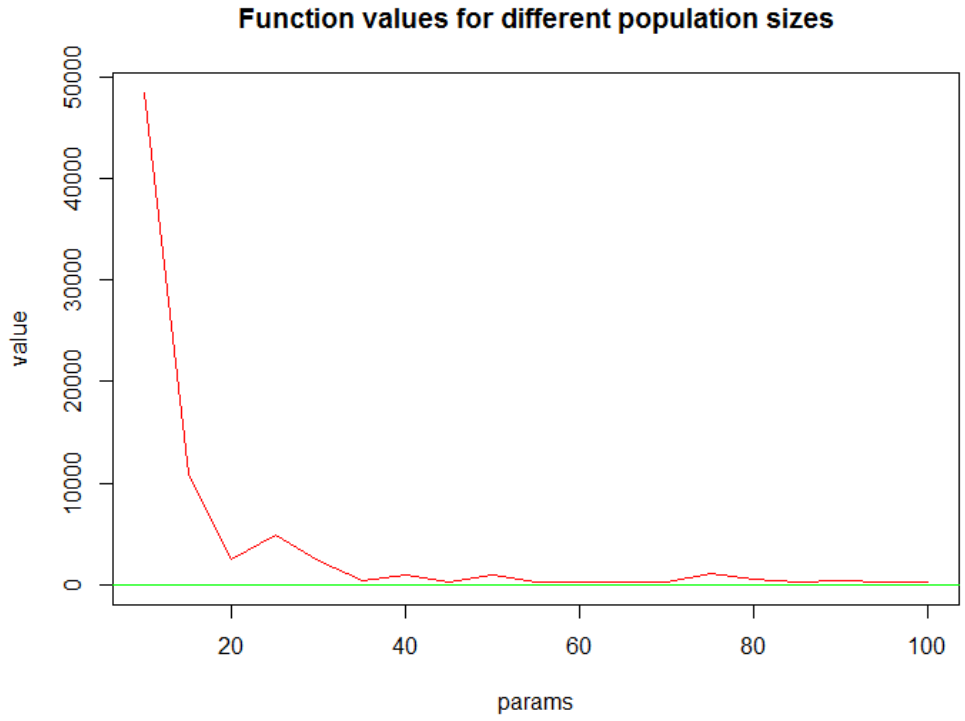
Rysunek 37: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



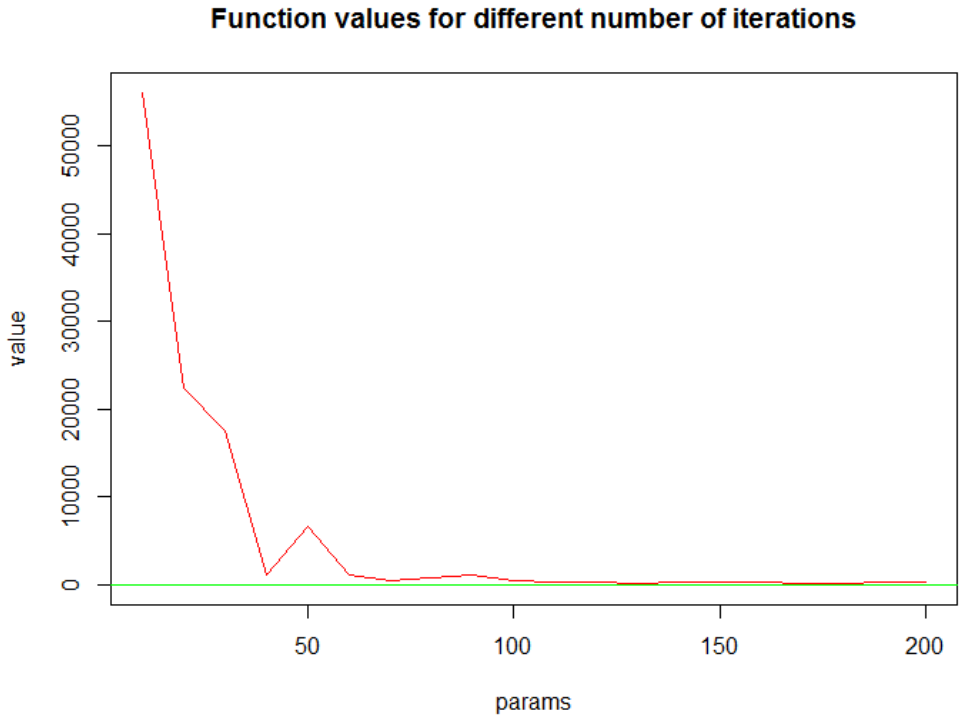
Rysunek 38: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



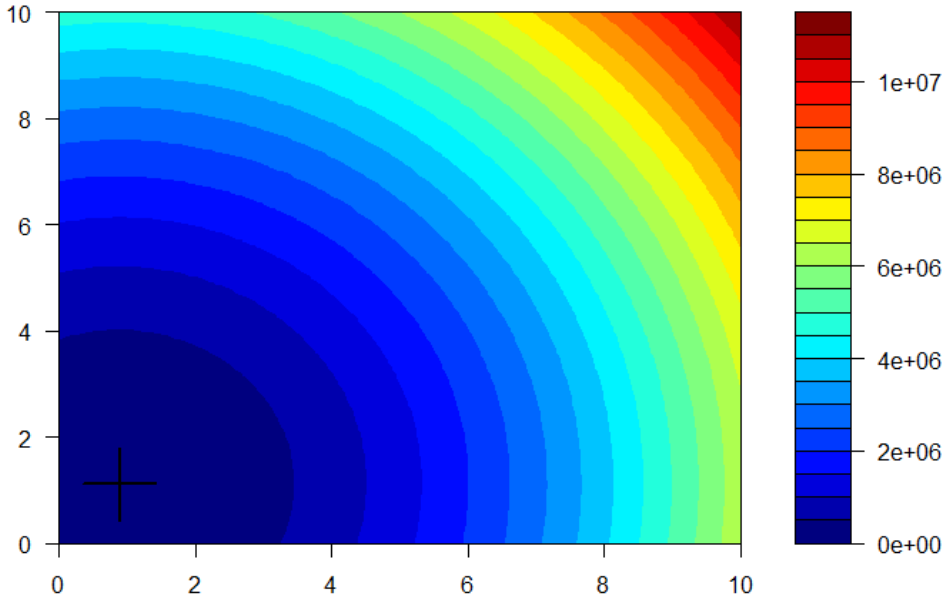
Rysunek 39: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 40: Wartość znalezionego optimum w zależności od rozmiarów populacji

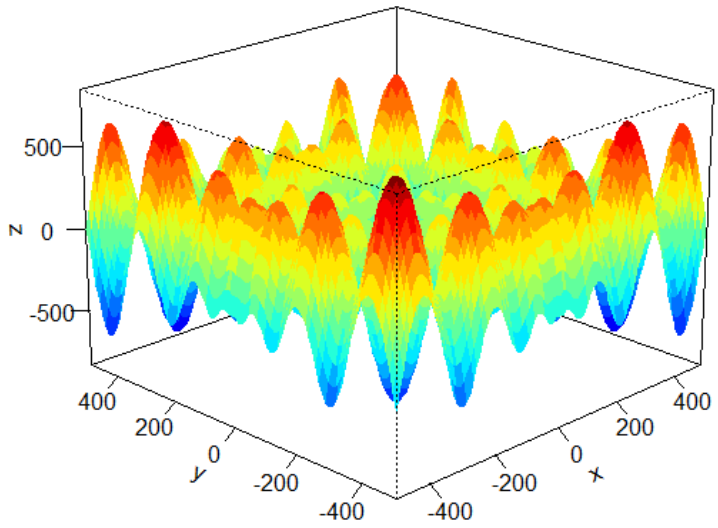


Rysunek 41: Wartość znalezionego optimum w zależności od iteracji

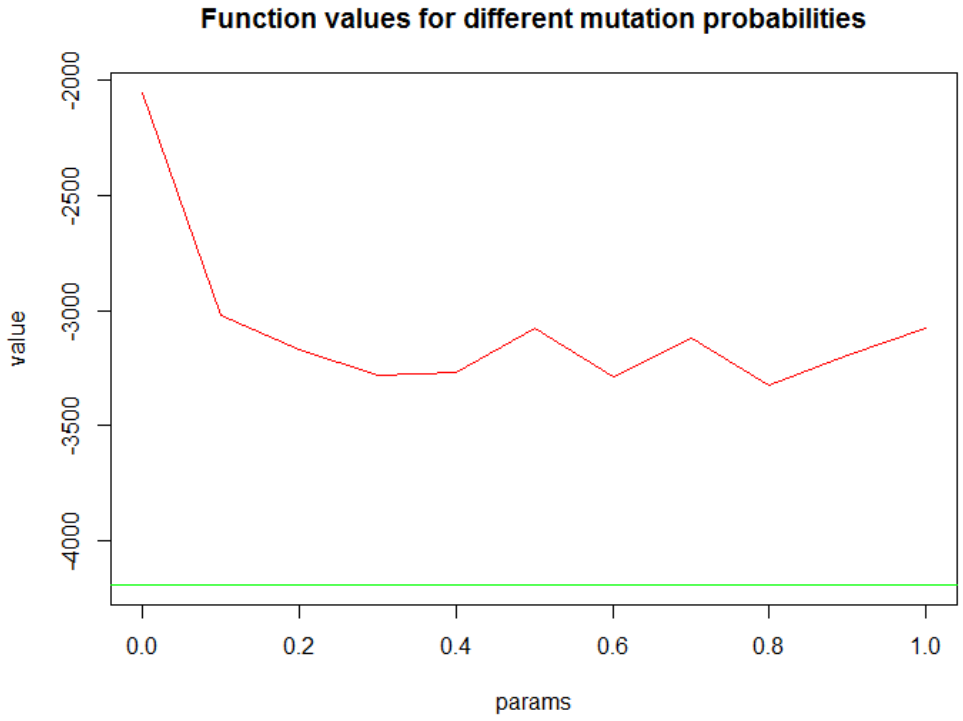


Rysunek 42: Poglądowa lokalizacja najlepszego znalezionego optimum

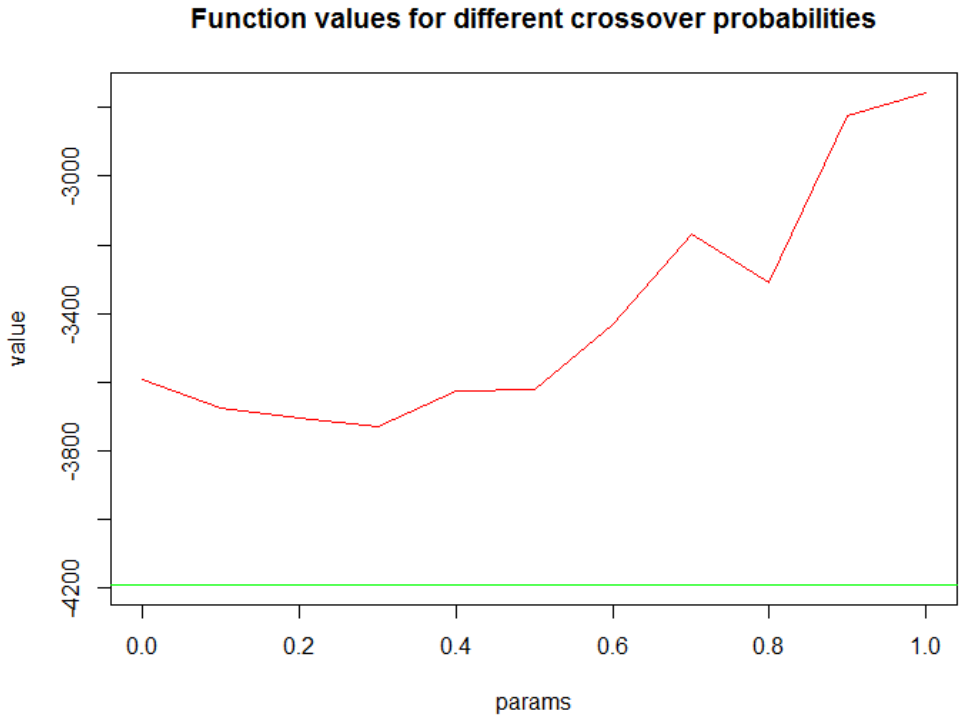
3.7 Schwefel (10 parametrów)



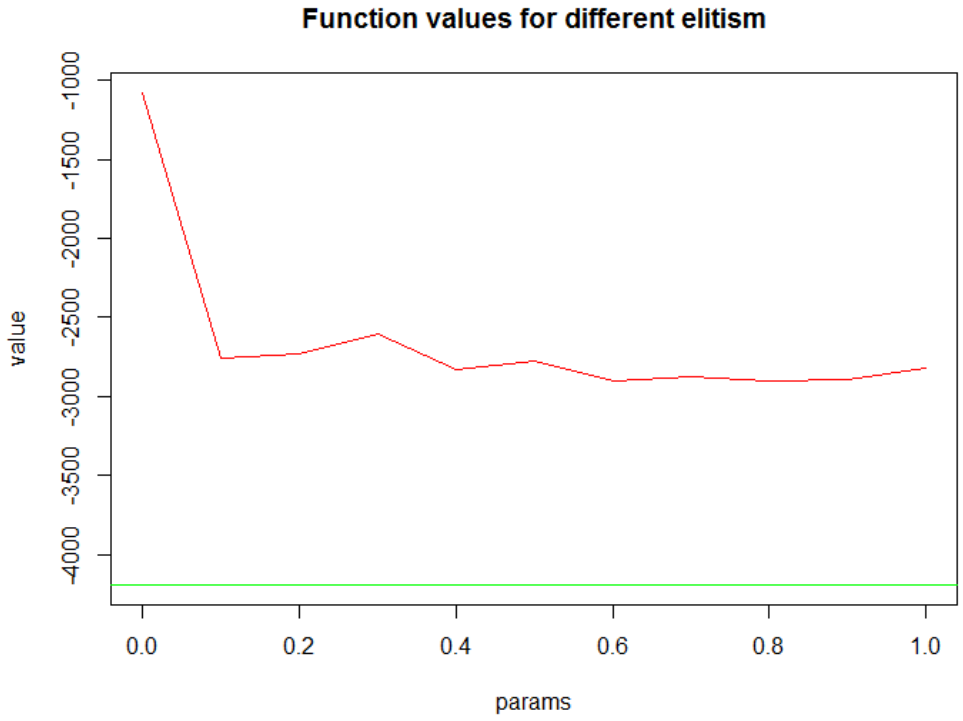
Rysunek 43: Poglądowa lokalizacja najlepszego znalezionego optimum



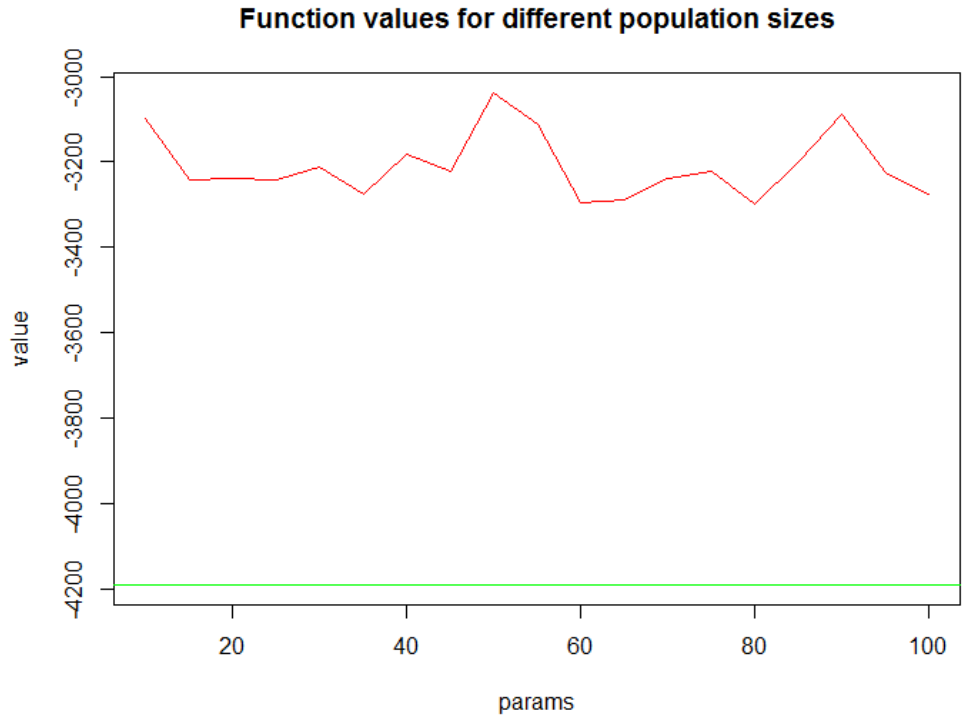
Rysunek 44: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



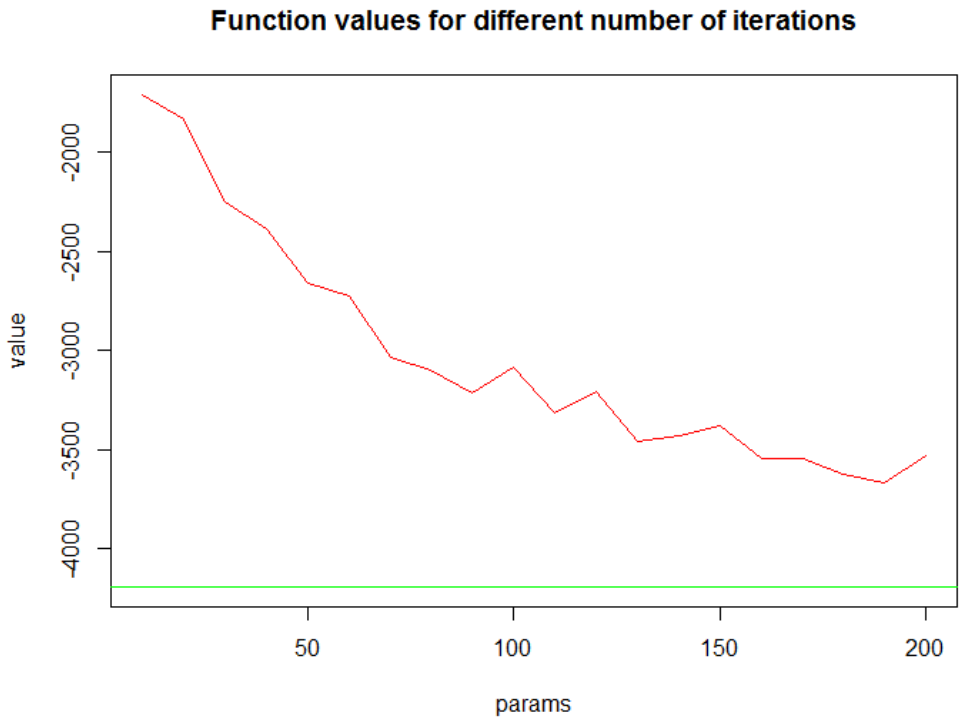
Rysunek 45: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



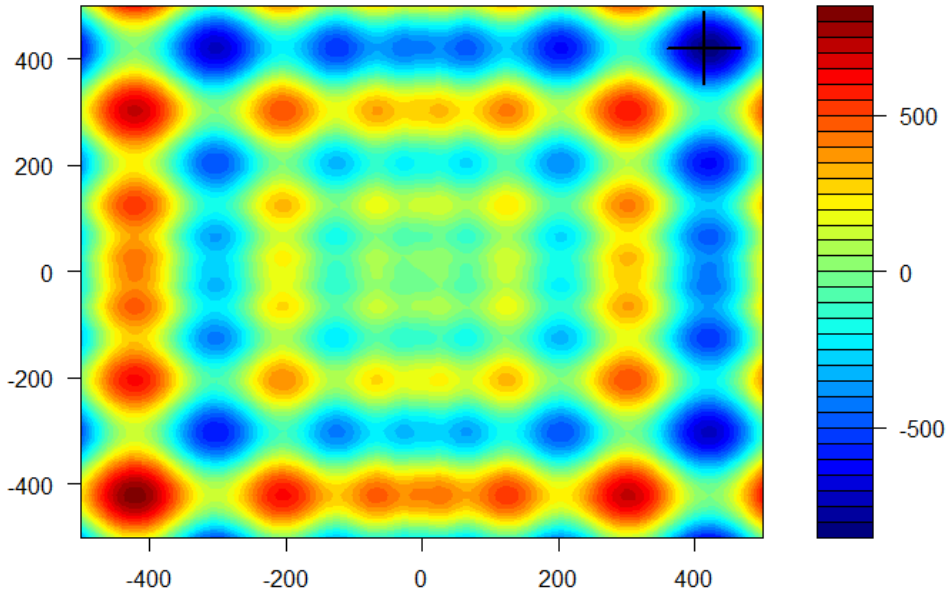
Rysunek 46: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 47: Wartość znalezionego optimum w zależności od rozmiarów populacji

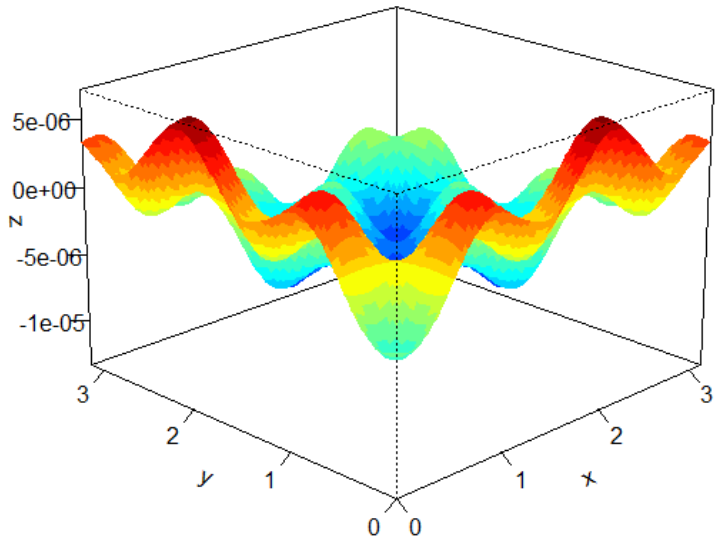


Rysunek 48: Wartość znalezionego optimum w zależności od iteracji

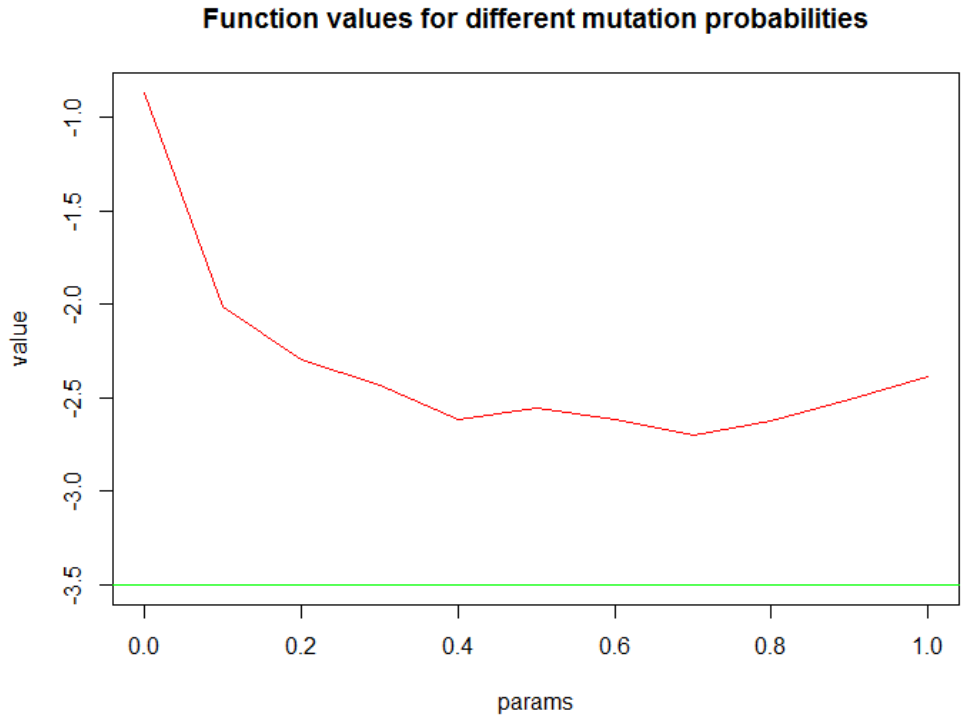


Rysunek 49: Poglądowa lokalizacja najlepszego znalezionego optimum

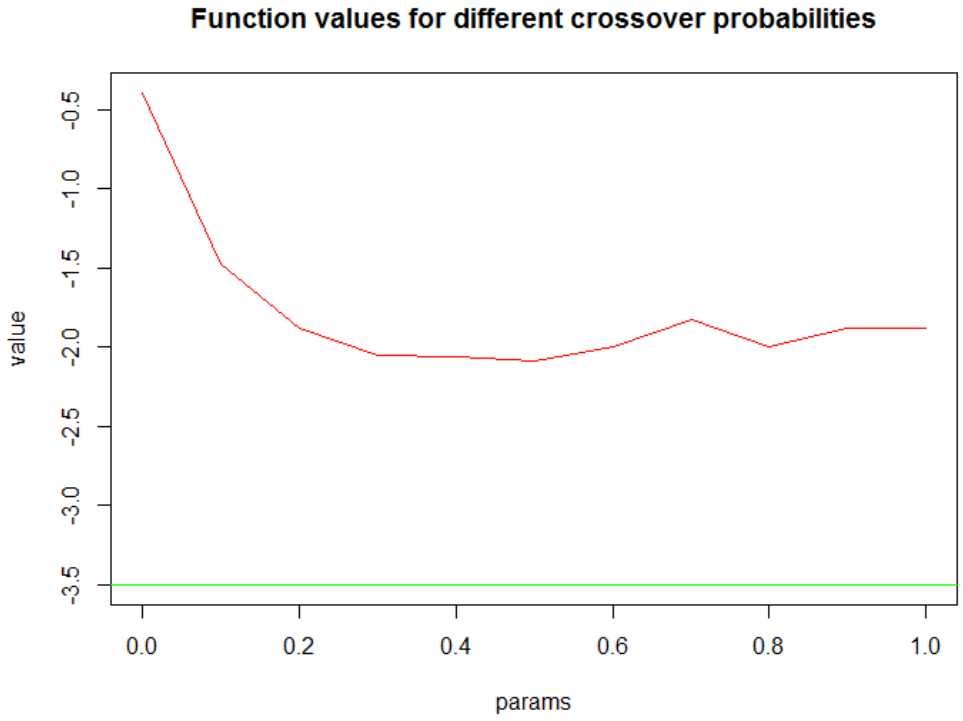
3.8 Zeldasine20 (20 parametrów)



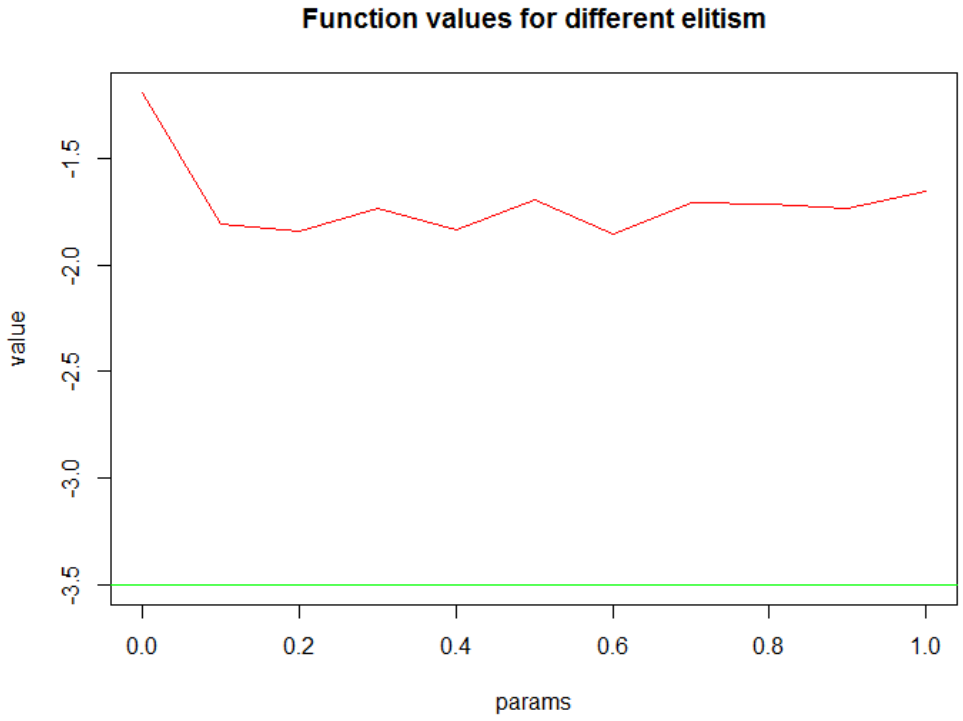
Rysunek 50: Poglądowa lokalizacja najlepszego znalezionego optimum



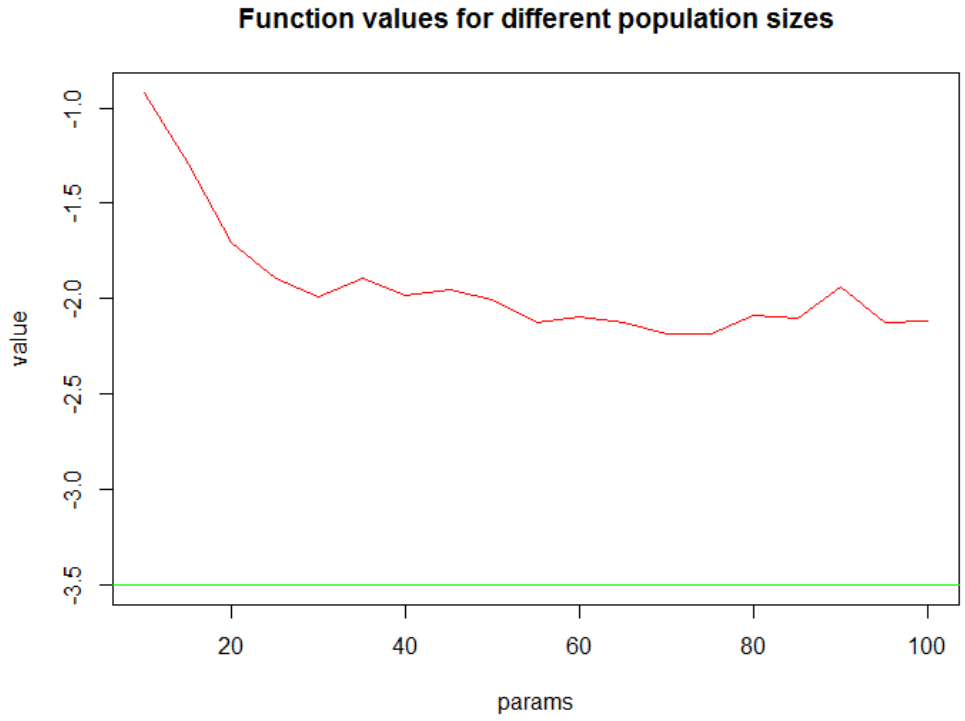
Rysunek 51: Wartość znalezionego optimum w zależności od prawdopodobieństwa mutacji



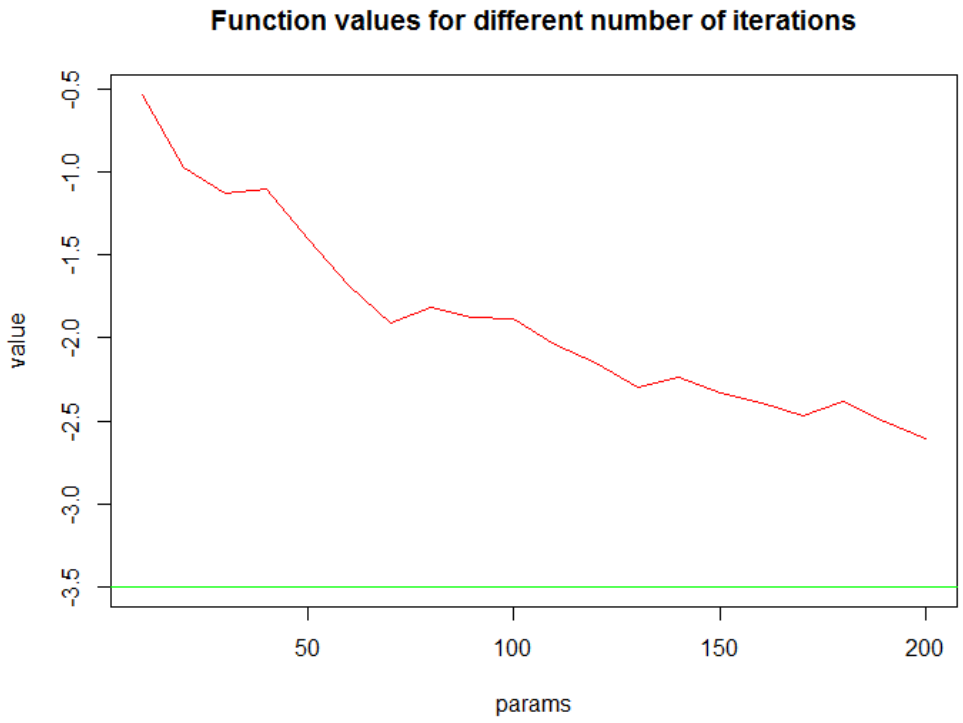
Rysunek 52: Wartość znalezionego optimum w zależności od prawdopodobieństwa krzyżowania



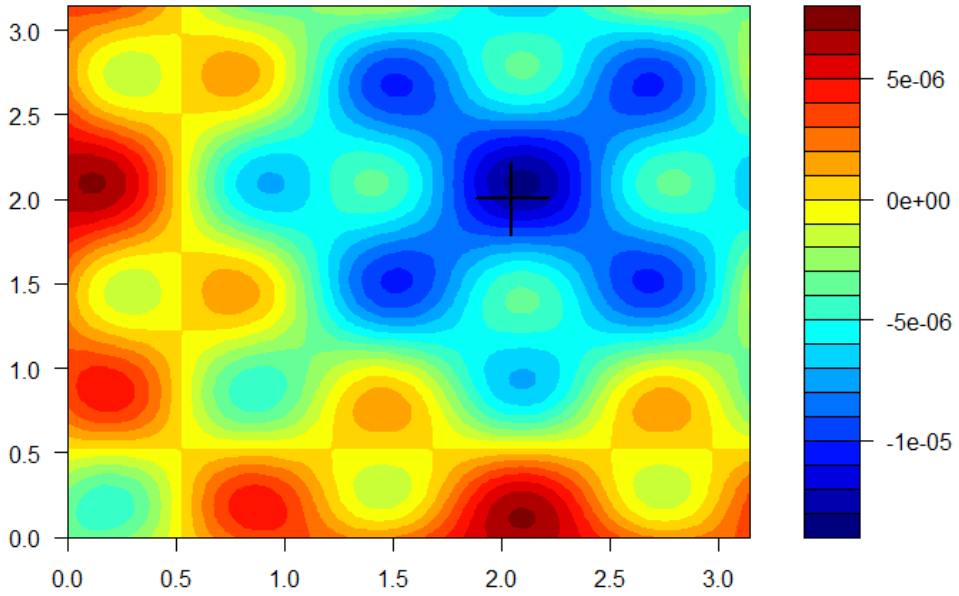
Rysunek 53: Wartość znalezionego optimum w zależności od przyjętego elityzmu



Rysunek 54: Wartość znalezionego optimum w zależności od rozmiarów populacji



Rysunek 55: Wartość znalezionego optimum w zależności od iteracji



Rysunek 56: Poglądowa lokalizacja najlepszego znalezionego optimum

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