

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	2
2	Implementacja	2
2.1	Parametryzacja skryptu	5
3	Przebieg badań	6
3.1	Branin (2 parametry)	7
3.2	Gulf (3 parametry)	11
3.3	CosMix4 (4 parametry)	16
3.4	EMichalewicz (5 parametrów)	20
3.5	Hartman6 (6 parametrów)	24
3.6	PriceTransistor (9 parametrów)	28
3.7	Schwefel (10 parametrów)	32
3.8	Zeldasine20 (20 parametrów)	36
4	Podsumowanie	40

1 Wprowadzenie

Algorytm genetyczny to algorytm heurystyczny...[]

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 # Settings ----
10
11 nOfRuns <- 20 # number of runs to calc average
12
13 colors <- c("red", "blue", "purple", "black")
14 series <- c("Seria 1", "Seria 2", "Seria 3", "Seria 4")
15
16 # [mutations,crossovers,populations,iterations,color]
17 params = matrix(
18   c(0, 0, 50, 100, 1,
19     0, 0.8, 50, 100, 2,
20     0.1, 0, 50, 100, 3,
21     0.1, 0.8, 50, 100, 4),
22   nrow=4, ncol=5, byrow = TRUE)
23
24 functions <- c("Branin", "Gulf", "CosMix4", "EMichalewicz",
25               "Hartman6", "PriceTransistor", "Schwefel", "Zeldasine20")
26
27 graphs <- TRUE
28 quality <- 100 #graph resolutions
29
30 mutationTests <- seq(0, 1, 0.1)
31 crossoverTests <- seq(0, 1, 0.1)
32 populationTests <- seq(10, 100, 5)
33 iterationTests <- seq(10, 200, 10)
34 elitismTests <- seq(0, 1, 0.1)
35
36 # Processing ----
37
38 customMeasure <- function(fileName, graphName, values, mType, xlab, main) {
```

```

39
40 gMin <- .Machine$integer.max
41 gBest <- NA
42
43 temp <- c()
44 for (defRow in 1:nrow(params)) {
45   averages <- c()
46   for (value in values) {
47     sum <- 0
48     for (i in 1:nOfRuns) {
49       GAmin <- ga(type = "real-valued",
50         fitness = function(xx) -f(xx),
51         min = c(B[1,]), max = c(B[2,]),
52         popSize = if (mType == "pop") value else params[defRow,3],
53         maxiter = if (mType == "itr") value else params[defRow,4],
54         pmutation = if (mType == "mut") value else params[defRow,1],
55         pcrossover = if (mType == "crs") value else params[defRow,2],
56         elitism = if (mType == "elt") value else max(1,
57           round(params[defRow,3] * 0.05)))
58       solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
59       eval <- f(solution[1,])
60       if (eval < gMin) {
61         gMin <- eval
62         gBest <- GAmin
63       }
64       sum <- sum + eval
65     }
66     averages <- c(averages, (sum / nOfRuns))
67   }
68   temp <- c(temp, averages)
69 }
70 result <- matrix(c(temp),nrow = nrow(params),ncol = length(values))
71 write.table(result, file = paste(funcName, fileName, sep=""), row.names=FALSE,
72   na="", col.names=FALSE, sep=";")
73
74 if (graphs) {
75   png(file = paste(funcName, graphName, ".png", sep=""), width=600,
76     height=400, units="px")
77   plot(0, 0, main=main,
78     ylim=c(min(c(temp,globalOpt)),max(c(temp,globalOpt))),
79     xlim=c(min(values),max(values)),
80     type="n", xlab=xlab, ylab="wartosc")
81   abline(globalOpt,0, col="green")
82   colorNames <- c()
83   seriesNames <- c()
84   for (i in 1:nrow(params)) {
85     color <- colors[params[i,5]]
86     colorNames <- c(colorNames, color)
87     seriesNames <- c(seriesNames, series[params[i,5]])
88     lines(values, result[i,], col = color, type = 'l')
89   }
90   legend("topright", seriesNames, lwd=rep(2,nrow(params)),
91     lty=rep(1,nrow(params)), col=colorNames)
92   dev.off()
93   summary(gBest)
94   png(file = paste(funcName, graphName, mType, ".png", sep=""), width=600,

```

```

    height=400, units="px")
192 filled.contour(x, y, z, color.palette = jet.colors, nlevels = 24,
193   plot.axes = { axis(1); axis(2);
194     points(solution[1,1], solution[1,2],
195       pch = 3, cex = 5, col = "black", lwd = 2)
196   }
197 )
198 dev.off()
199 png(file = paste(funcName, graphName, mType, "fitness", ".png", sep=""),
    width=600, height=400, units="px")
200 plot(gBest)
201 dev.off()
202 }
203
204 }
205
206
207 for (funcName in functions) {
208
209   dim <- getProblemDimen(funcName)
210   B <- matrix(unlist(getDefaultBounds(funcName)), ncol=dim, byrow=TRUE)
211   f <- function(xx) goTest(par=c(xx, rep(0, dim-length(xx))),
212     fnName=funcName, checkDim = TRUE)
213   globalOpt <- getGlobalOpt(funcName)
214
215   if (graphs) {
216
217     xprobes <- abs(B[2,1] - B[1,1]) / quality
218     yprobes <- abs(B[2,2] - B[1,2]) / quality
219     x <- seq(B[1,1], B[2,1], by = xprobes)
220     y <- seq(B[1,2], B[2,2], by = yprobes)
221     z <- outer(x, y, Vectorize(function(x,y) f(c(x,y))))
222     nbcol = 100
223     color = rev(rainbow(nbcol, start = 0/6, end = 4/6))
224     zcol = cut(z, nbcol)
225     persp3d(x, y, z, theta=50, phi=25, expand=0.75, col=color[zcol],
226       ticktype="detailed", axes=TRUE)
227
228     png(file = paste(funcName, "1.png", sep=""), width=600, height=400,
229       units="px")
230     persp3d(x, y, z, theta = -45, phi = 20, color.palette = jet.colors)
231     dev.off()
232   }
233
234   customMeasure("resultsMutations.csv", "2", mutationTests, "mut",
235     "p. mutacji", "Znalezienie minimum dla roznych prawdopodobienstw mutacji")
236
237   customMeasure("resultsCrossover.csv", "3", crossoverTests, "crs",
238     "p. krzyzowania", "Znalezienie minimum dla roznych prawdopodobienstw
239     krzyzowania")
240
241   customMeasure("resultsPopulation.csv", "4", populationTests, "pop",
242     "rozmiar populacji", "Znalezienie minimum dla roznych rozmiarow populacji")
243
244   customMeasure("resultsIterations.csv", "5", iterationTests, "itr",

```

```
144     "ilosc iteracji", "Znalezione minimum dla roznym ilosci iteracji")
145
146     customMeasure("resultsElitism.csv", "6", elitismTests, "elt",
147         "elityzm", "Znalezione minimum dla roznym wartosci elityzmu")
148
149 }
```

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 „globalOpt-Tests”. [todo: dopisać o pętli przechodzącej po wszystkich funkcjach oraz po wszystkich parametrach domyślnych]

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 20-krotnie wyniki uśredniając. Domyślne parametry przedstawiono poniżej (tabela 1). Zmianie ulegają wartości prawdopodobieństwa mutacji i krzyżowania.

Tabela 1: Parametry domyślne poszczególnych serii pomiarowych

-	Seria 1	Seria 2	Seria 3	Seria 4
Rozmiar populacji	50	50	50	50
Rozmiar iteracji	100	100	100	100
Prawdopodobieństwo mutacji	0	0	0.1	0.1
Prawdopodobieństwo krzyżowania	0	0.8	0	0.8

Zielone linie na wykresach oznaczają optima zawarte w pakiecie „globalOptTests” dla danej funkcji przy domyślnych ograniczeniach (tych samych dla których wykonywana jest optymalizacja podczas niniejszych pomiarów).

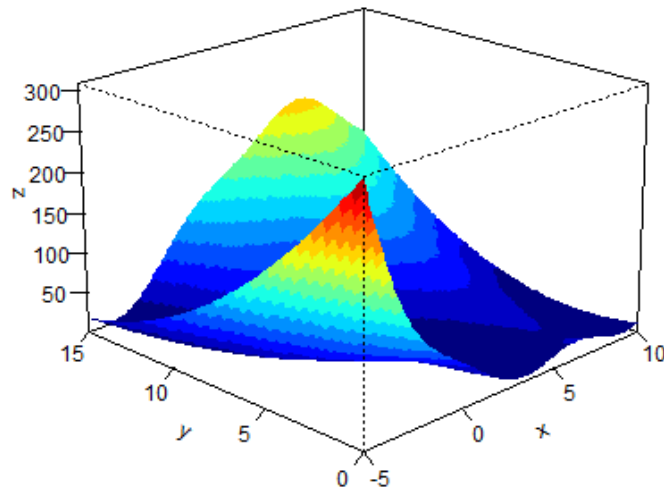
Dla funkcji o ilości parametrów większej niż 2 pominięto ilustracje graficzne znalezionych optimów gdyż optymalizacji podlegają wszystkie wymiary a przedstawienie dwóch pierwszych nie niesie ze sobą przydatnej informacji.

3.1 Branin (2 parametry)

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

$$f(\mathbf{x}) = a(x_2 - bx_1^2 + cx_1 - r)^2 + s(1 - t) \cos(x_1) + s \quad (1)$$

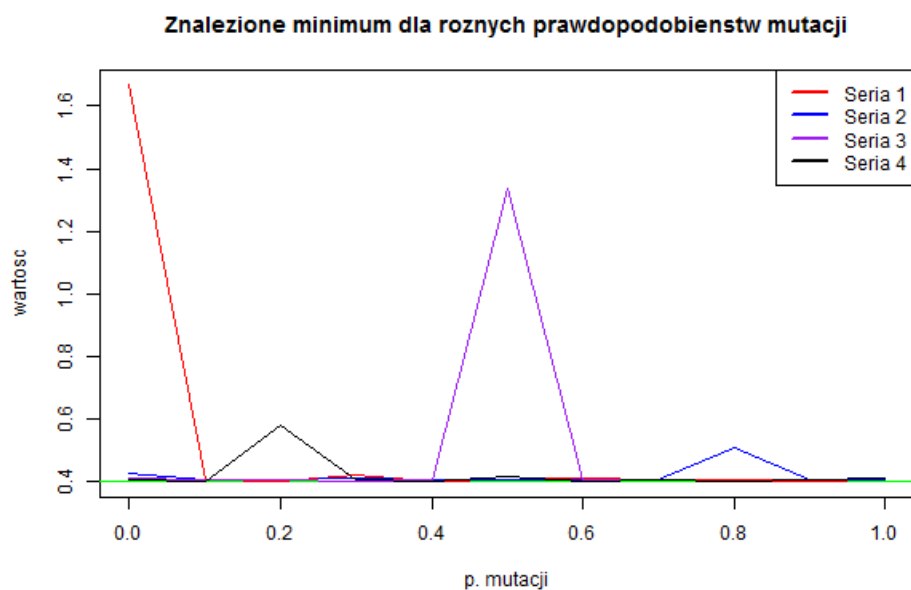
, gdzie $x_1 \in [-5, 10]$ oraz $x_2 \in [0, 15]$.



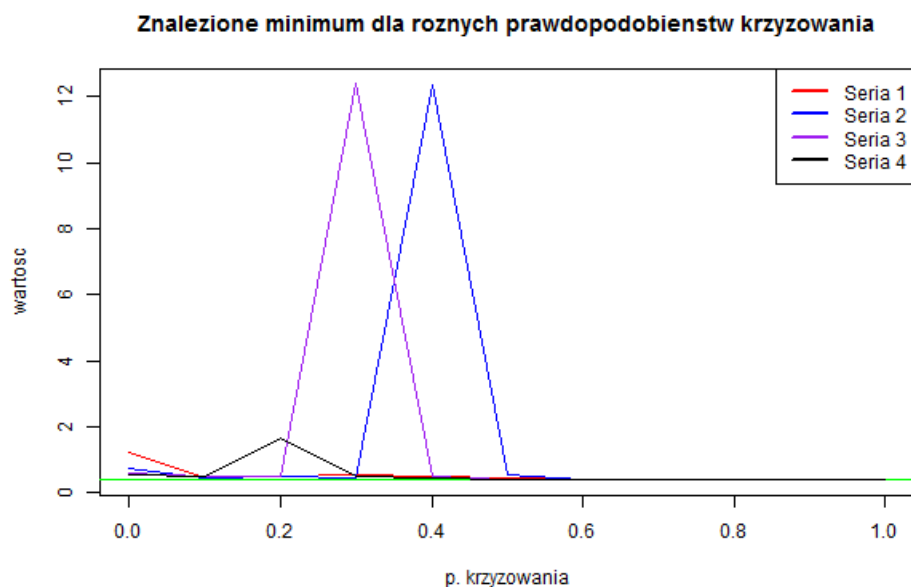
Rysunek 1: Wykres funkcji Branin

Powyższy wykres (rys. 1) pokazuje trójwymiarowy obraz funkcji Branin. Wynika z niego, że funkcja ta ma stosunkowo duży obszar w którym może znajdować się minimum oraz dwie strefy w których wartości są dużo większe.

Na kolejnych stronach zamieszczono wyniki pomiarów dla różnych wartości parametrów algorytmu genetycznego. Kolejno dokonano pomiarów dla różnych wartości: prawdopodobieństwa mutacji i krzyżowania, wielkości populacji, ilości iteracji oraz elityzmu. Wszystkie pomiary wykonano dla 4 różnych ustawień domyślnych parametrów.



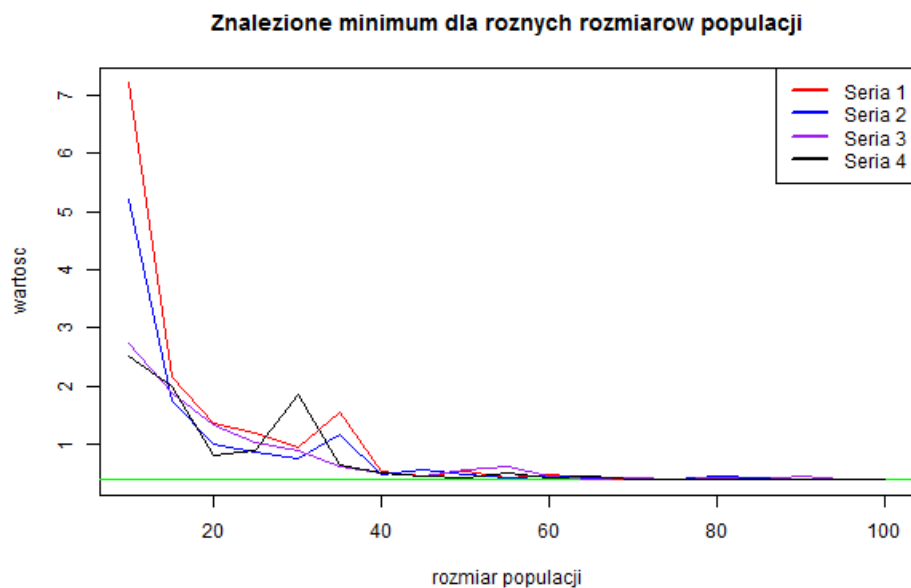
Rysunek 2: Wartość znalezione minimum funkcji Branin w zależności od prawdopodobieństwa mutacji



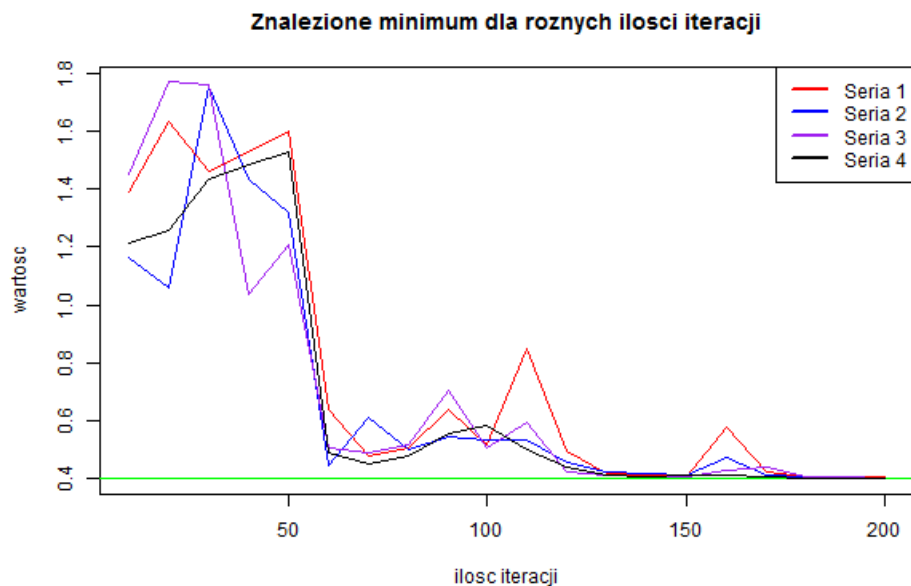
Rysunek 3: Wartość znalezione minimum funkcji Branin w zależności od prawdopodobieństwa krzyżowania

Na wykresie (rys. 2) można zauważyć niski wpływ ustawienia mutacji na znalezione rozwiązanie. Przy wszystkich parametrach domyślnych funkcja znajduje się w pobliżu optymalnej wartości. Miejscowe odchylenia są tu najprawdopodobniej związane z charakterem algorytmu i zbyt małą ilością prób poddanych uśrednieniu. Nie możemy tutaj określić czy przy wyłączonej zarówno mutacji jak i krzyżowaniu wyniki ulegają pogorszeniu, gdyż nie

ma w tym obszarze spójności. Podobne wnioski możemy wskazać dla wykresu krzyżowania (rys. 3).



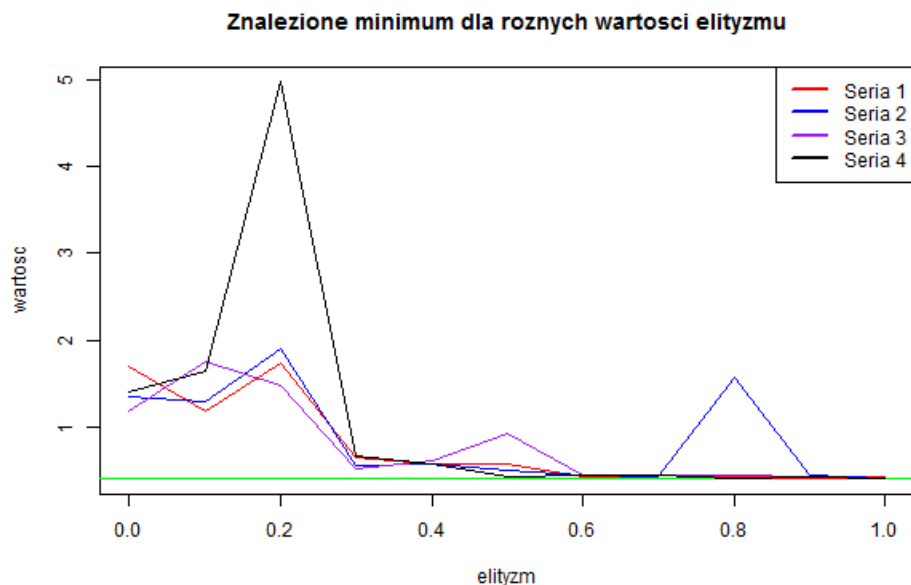
Rysunek 4: Wartość znalezione minimum funkcji Branin w zależności od rozmiaru populacji



Rysunek 5: Wartość znalezione minimum funkcji Branin w zależności od ilości iteracji

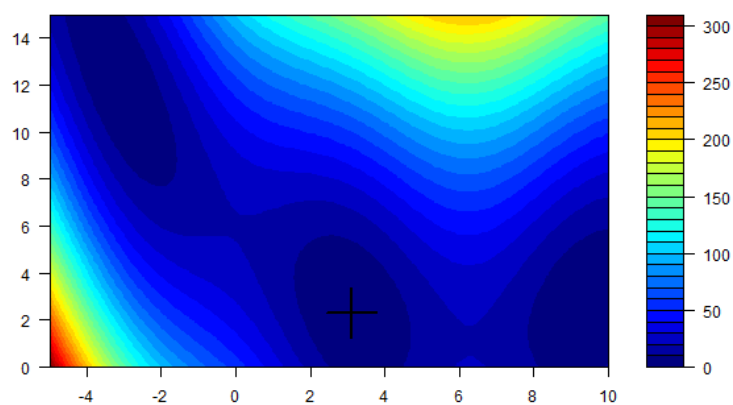
Z wykresu (rys. 4) można odczytać podatność funkcji na zmiany rozmiaru populacji. Wyniki zbliżone do oczekiwanych zostały uzyskane dla wartości wynoszącej 45 jednostek. Widać również, że przy małej populacji znaczenie mutacji i krzyżowania jest większe. Zauważalny jest wzrost jakości rozwiązania wraz ze wzrostem ilości jednostek populacji.

Wykres (rys. 5) wskazuje wyraźną zmianę jakości rozwiązań dla 60 i więcej iteracji. Poniżej tej wartości uzyskiwane wyniki są niestabilne, powyżej osiągają wartość zbliżoną do oczekiwanej szczególnie dla serii 4 (czyli z włączoną mutacją i krzyżowaniem).



Rysunek 6: Wartość znalezione minimum funkcji Branin w zależności od przyjętego elityzmu

Z wykonanych pomiarów (rys. 6) wynika, że dla uzyskania optymalnego rozwiązania należy zastosować wartość elityzmu na poziomie przynajmniej 0.4. Jego ustawienie poniżej tej wartości powoduje obniżenie się jakości rezultatów.



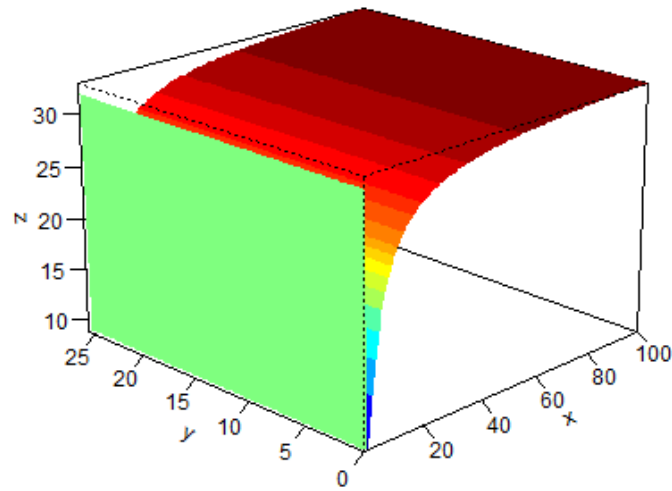
Rysunek 7: Poglądowa lokalizacja najlepszego znalezione minimum funkcji Branin dla pomiarów przy zmianach elityzmu

3.2 Gulf (3 parametry)

Gulf jest funkcją określoną dla ilości parametrów równej 3. Na ilustracji (rys. 8) przedstawiono jej wykres dla pierwszych dwóch.

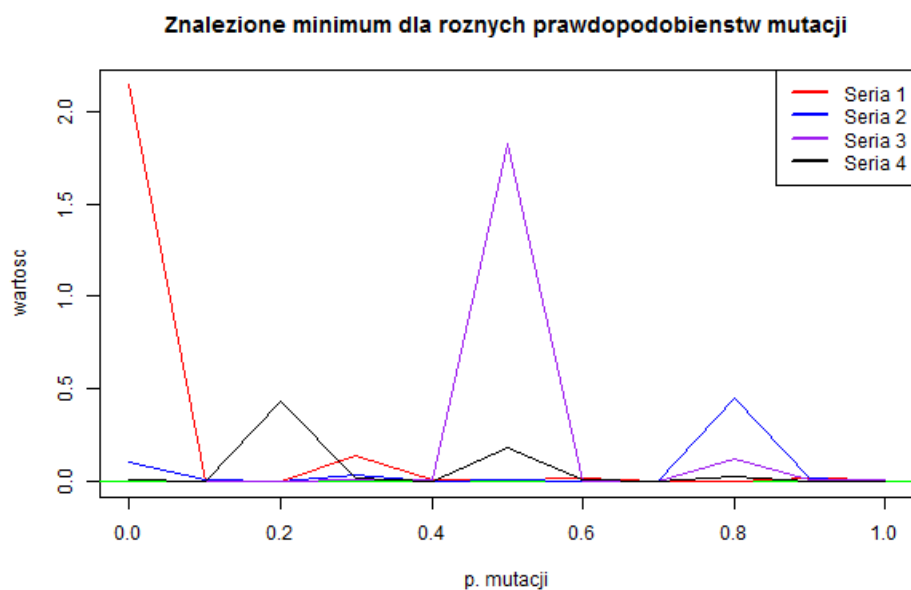
$$f(\mathbf{x}) = \sum_{i=1}^{99} [\exp(-\frac{(u_i - x_2)^{x_3}}{x_1}) - 0.01i]^2 \quad (2)$$

, gdzie $x_1 \in [0.1, 100]$, $x_2 \in [0, 25.6]$, $x_3 \in [0, 5]$ oraz $u_i = 25 + [-50 \ln(0.01i)]^{1/1.5}$.



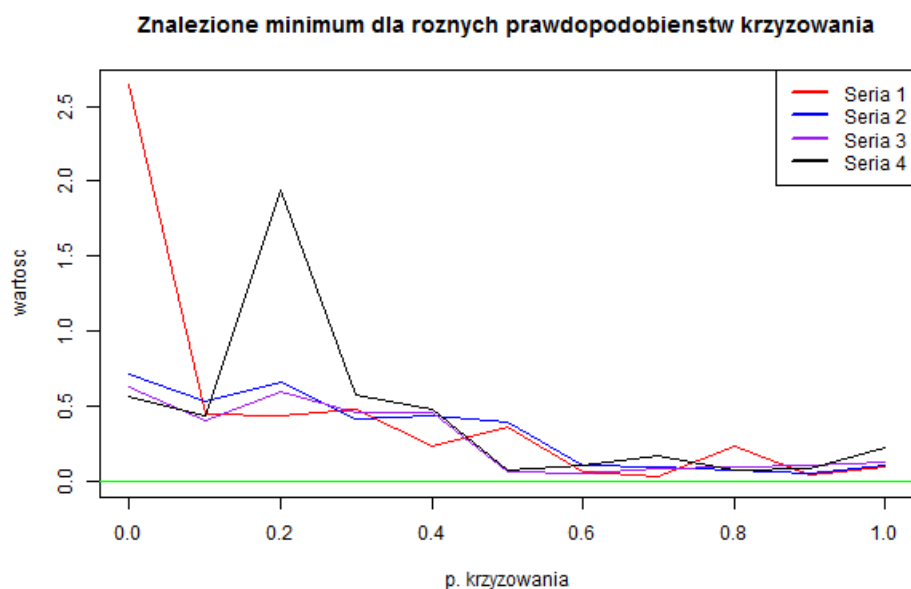
Rysunek 8: Wykres funkcji Gulf dla dwóch pierwszych parametrów

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



Rysunek 9: Wartość znalezionego minimum dla funkcji Gulf w zależności od prawdopodobieństwa mutacji

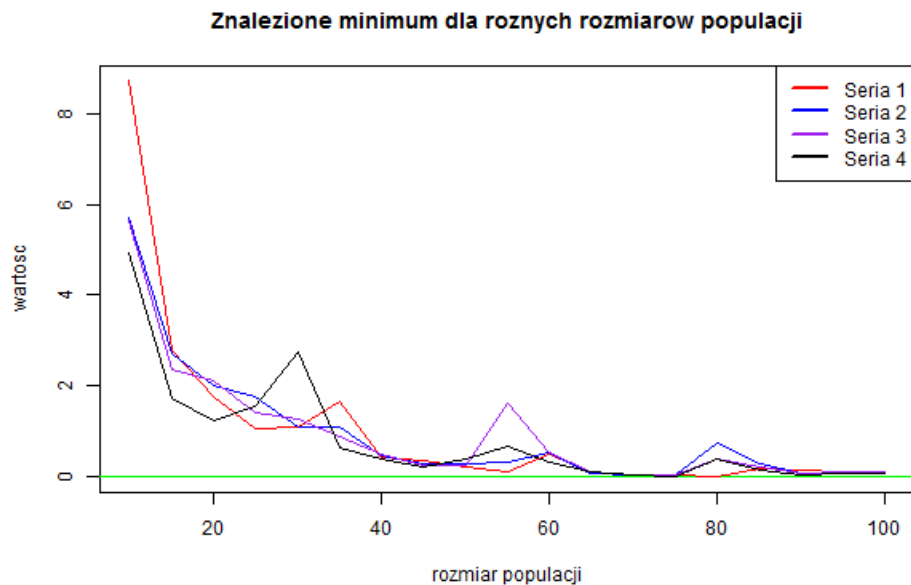
Wartości funkcji Gulf dla zadanego prawdopodobieństwa mutacji są zbliżone do wartości oczekiwanej w przedziale 0.1-0.6. Powyżej tego przedziału mutacja wywiera negatywny wpływ na otrzymywane wyniki.



Rysunek 10: Wartość znalezionego minimum dla funkcji Gulf w zależności od prawdopodobieństwa krzyżowania

Prawdopodobieństwo krzyżowania ma niski oraz niestabilny wpływ na otrzymane wy-

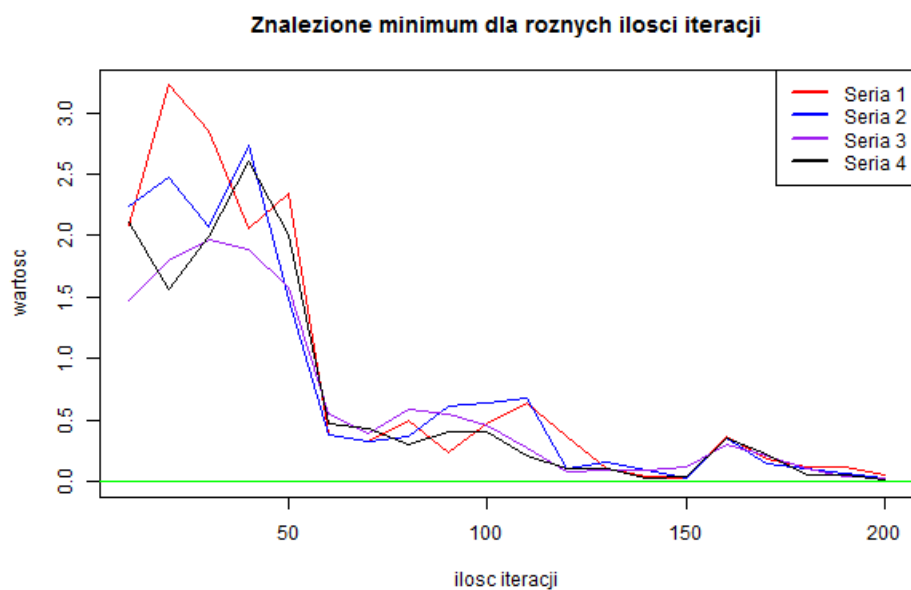
niki. Wspólnie (dla wszystkich ustawień domyślnych) najlepsze wyniki uzyskane zostały w przedziale 0.4-0.6. Przyjęcie wartości krzyżowania wykraczających poza wskazany przedział znacząco obniża jakość uzyskanych wyników.



Rysunek 11: Wartość znalezionego minimum dla funkcji Gulf w zależności od rozmiarów populacji

Wykres ten wyraźnie obrazuje pozytywny wpływ zwiększenia populacji na jakość wyników. Najlepsze wyniki uzyskano dla populacji wynoszącej przynajmniej 50 jednostek.

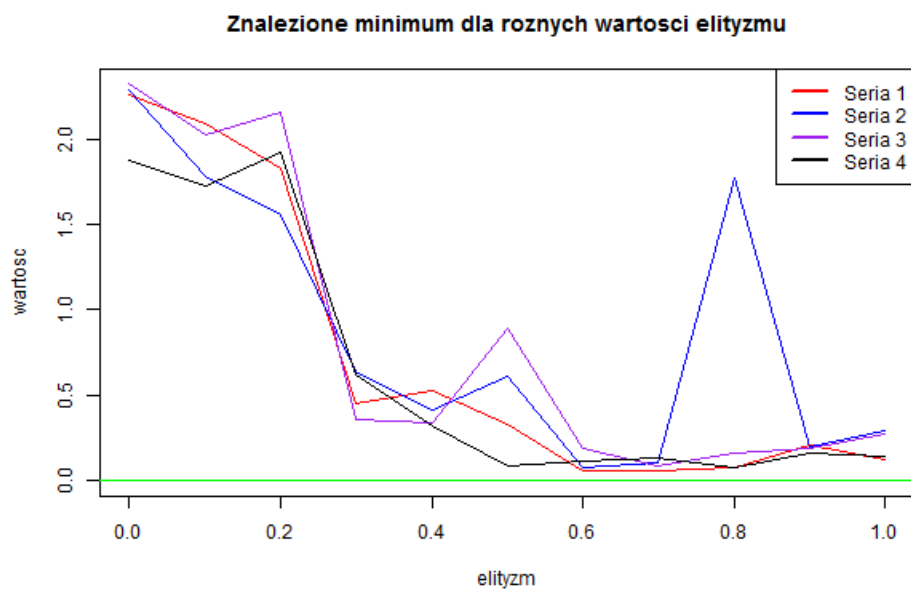
Zauważalne jest również pogorszenie wyników w przedziale 65-80[*todo: dlaczego*].



Rysunek 12: Wartość znalezionego minimum dla funkcji Gulf w zależności od ilości iteracji

Na wykresie można zauważyć znaczące poprawienie się rezultatów, gdy ilość iteracji wynosi przynajmniej 80. Poniżej tej wartości uzyskane wyniki są znacząco gorsze od optymalnego rozwiązania.

W przedziale 130-200 [todo: co się dzieje]



Rysunek 13: Wartość znalezionego minimum dla funkcji Gulf w zależności od przyjętego elityzmu

W przypadku funkcji Gulf elityzm ma znaczący wpływ na otrzymywane wyniki. W celu

ich optymalizacji wymagana jest wartość elityzmu na poziomie przynajmniej 0.4.

Dla wartości powyżej 0.6 wyniki zaczynają się pogarszać. [todo: dlaczego]

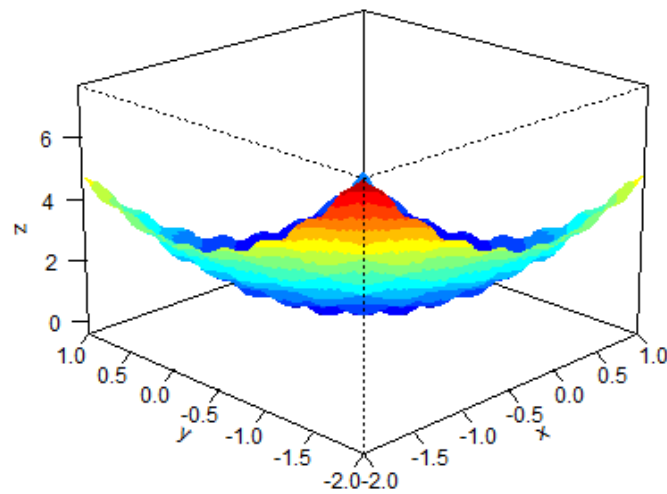
[todo: zmienić] Jak możemy zauważyć na ilustracji poniżej (rys. ??) przedstawiona lokalizacja optimum nie jest poprawna, gdyż optymalizacji poddano wersję z 3 parametrami. Ogólnie rzecz biorąc gdyby 3 wymiar przedstawić w postaci gradientu kolorystycznego wtedy byłaby to poprawna lokalizacja niemniej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

3.3 CosMix4 (4 parametry)

CosMix4 jest funkcją określoną dla ilości parametrów równej 4. Na ilustracji (rys. 14) przedstawiono jej wykres dla pierwszych dwóch.

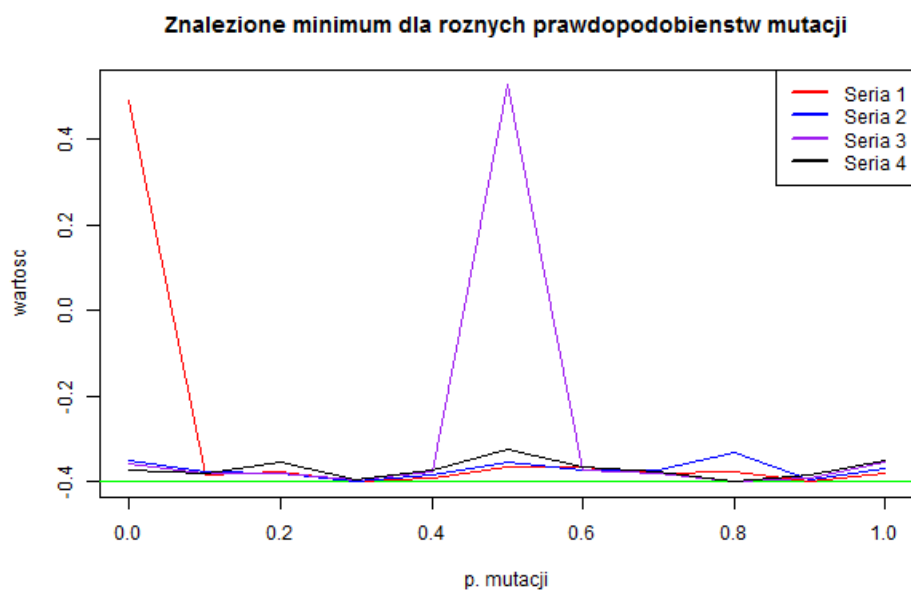
$$f(\mathbf{x}) = -0.1 \sum_{i=1}^4 \cos(5\pi x_i) - \sum_{i=1}^4 x_i^2 \quad (3)$$

, gdzie $x_i \in [-2, 1]$ oraz $i \in 1, 2, 3, 4$.



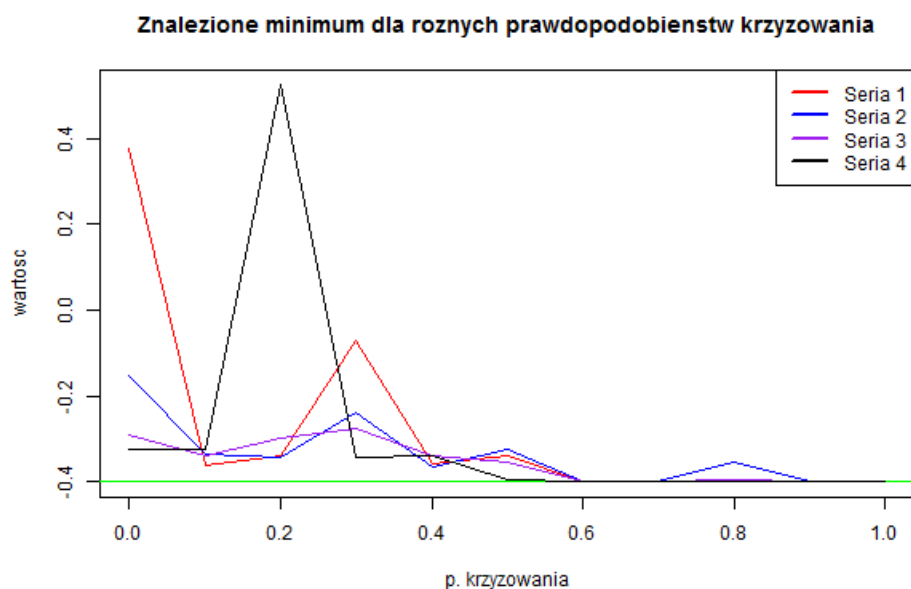
Rysunek 14: Wykres funkcji CosMix4 dla dwóch pierwszych parametrów

Jak widać funkcja ma dużo lokalnych optimów.



Rysunek 15: Wartość znalezionego minimum dla funkcji CosMix4 w zależności od prawdopodobieństwa mutacji

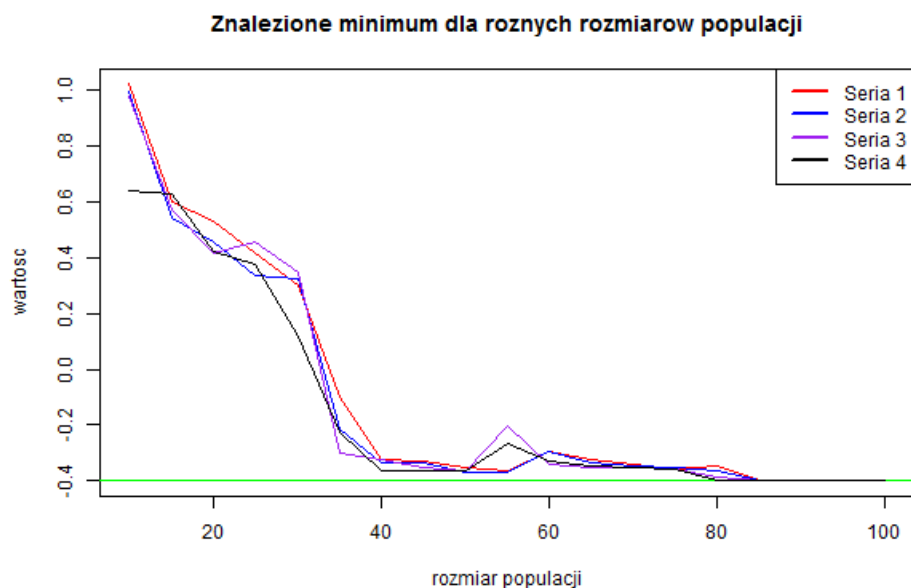
Ogólnie rzecz biorąc, niezależnie od pozostałych parametrów, jedyną niekorzystną sytuacją jest tu jednocześnie wyłączenie mutacji i krzyżowania co możemy zaobserwować na przykładzie serii 1-szej.



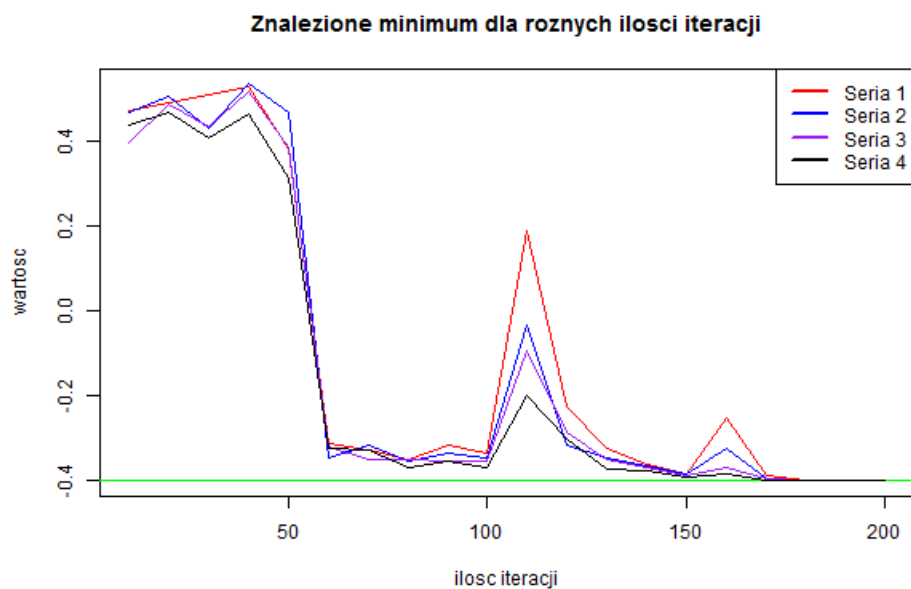
Rysunek 16: Wartość znalezionego minimum dla funkcji CosMix4 w zależności od prawdopodobieństwa krzyżowania

Podobnie jak na poprzednim wykresie (rys. 15) tak i na powyższym (rys. 16) ujawnia

się niekorzystny wpływ wyłączenia mutacji i krzyżowania.

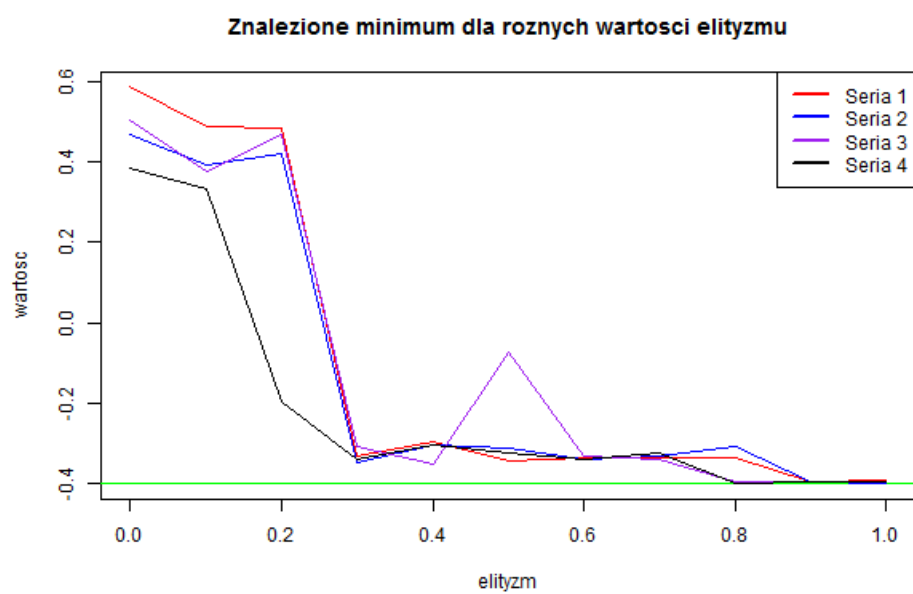


Rysunek 17: Wartość znalezione minimum dla funkcji CosMix4 w zależności od rozmiarów populacji



Rysunek 18: Wartość znalezione minimum dla funkcji CosMix4 w zależności od ilości iteracji

Na dwóch poprzedzających wykresach możemy zaobserwować, że domyślne wartości w postaci wielkości populacji w liczbie 50 i ilości iteracji równej 100 są wzajemnie optymalne.

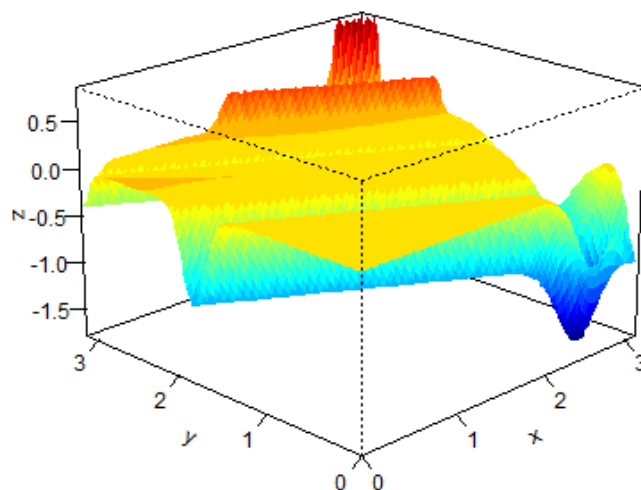


Rysunek 19: Wartość znalezione minimum dla funkcji CosMix4 w zależności od przyjętego elityzmu

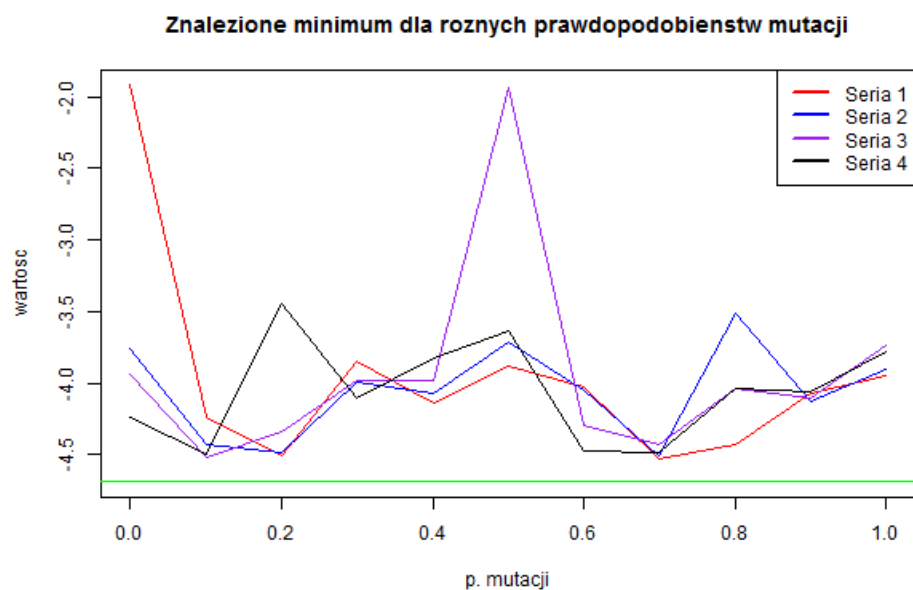
3.4 EMichalewicz (5 parametrów)

Poniżej zamieszczono wzór rozpatrywanej funkcji.

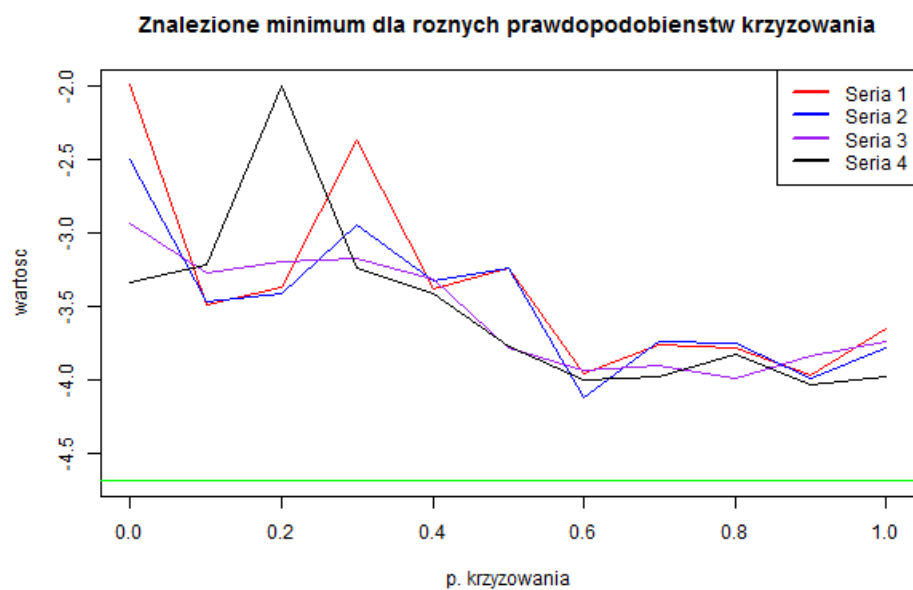
$$f(\mathbf{x}) = - \sum_{i=1}^d \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right) \quad (4)$$



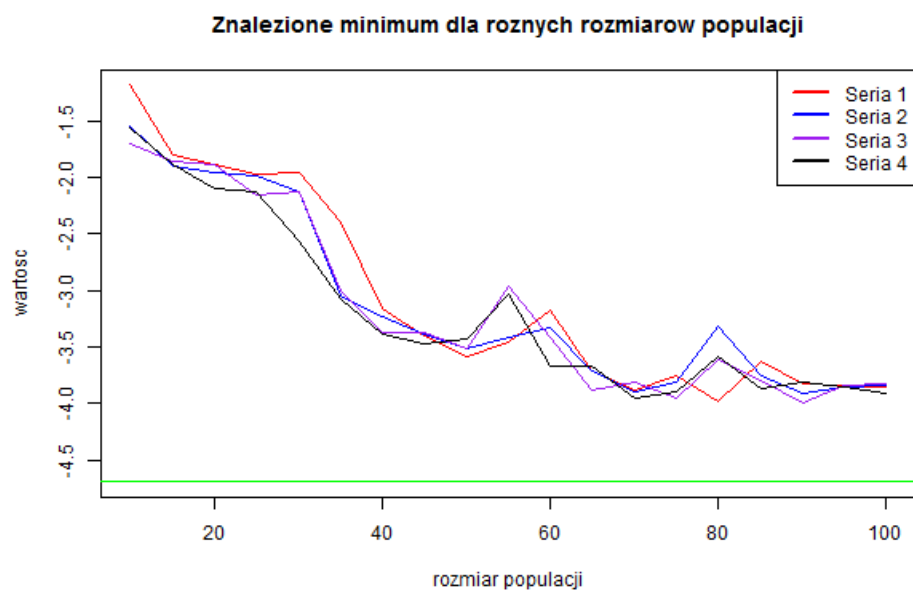
Rysunek 20: Wykres funkcji EMichalewicz (d=5)



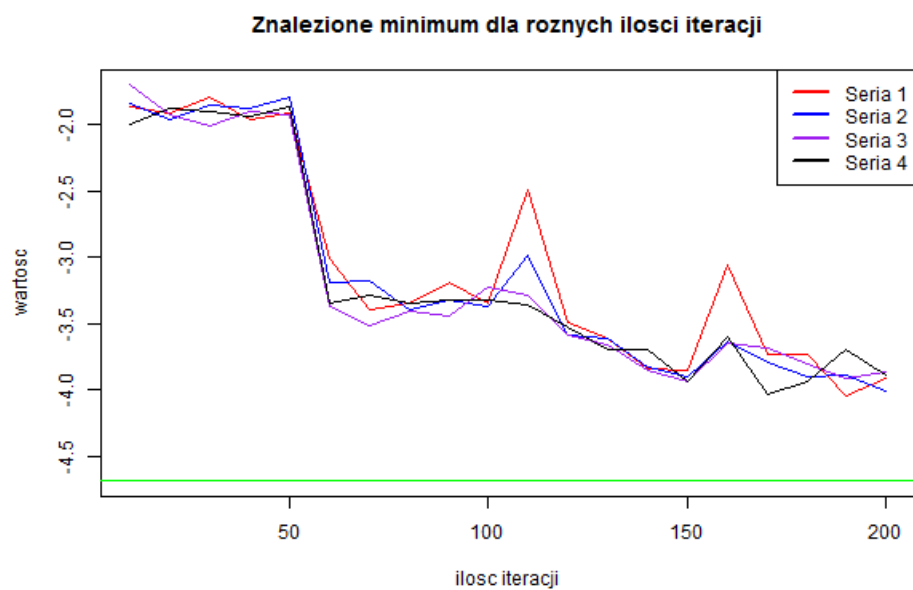
Rysunek 21: Wartość znalezione minimum dla funkcji EMichalewicz w zależności od prawdopodobieństwa mutacji



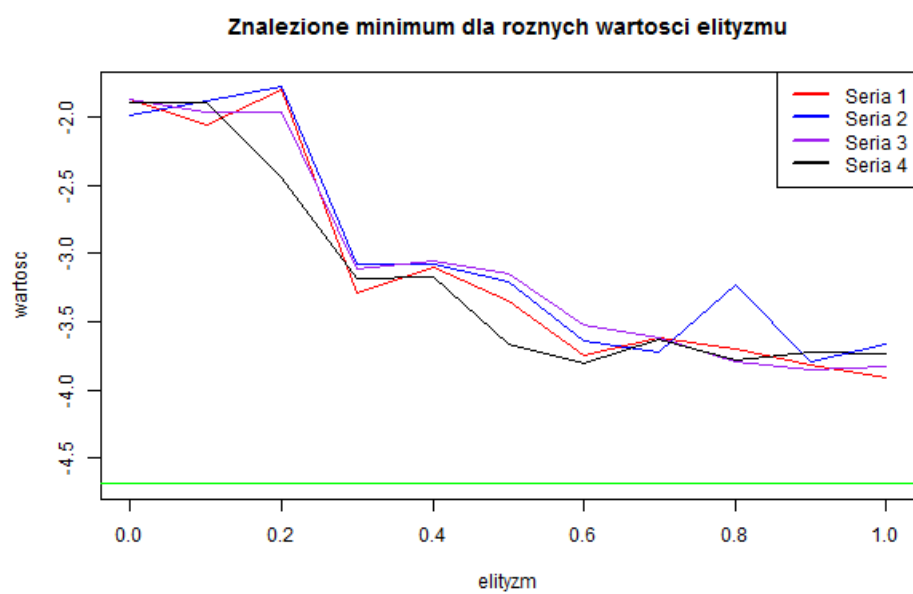
Rysunek 22: Wartość znalezione minimum dla funkcji EMichalewicz w zależności od prawdopodobieństwa krzyżowania



Rysunek 23: Wartość znalezione minimum dla funkcji EMichalewicz w zależności od rozmiarów populacji



Rysunek 24: Wartość znalezione minimum dla funkcji EMichalewicz w zależności od ilości iteracji



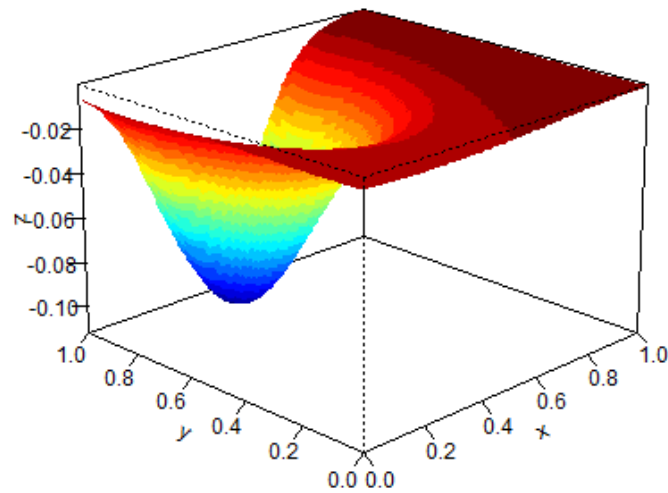
Rysunek 25: Wartość znalezione minimum dla funkcji EMichalewicz w zależności od przyjętego elityzmu

3.5 Hartman6 (6 parametrów)

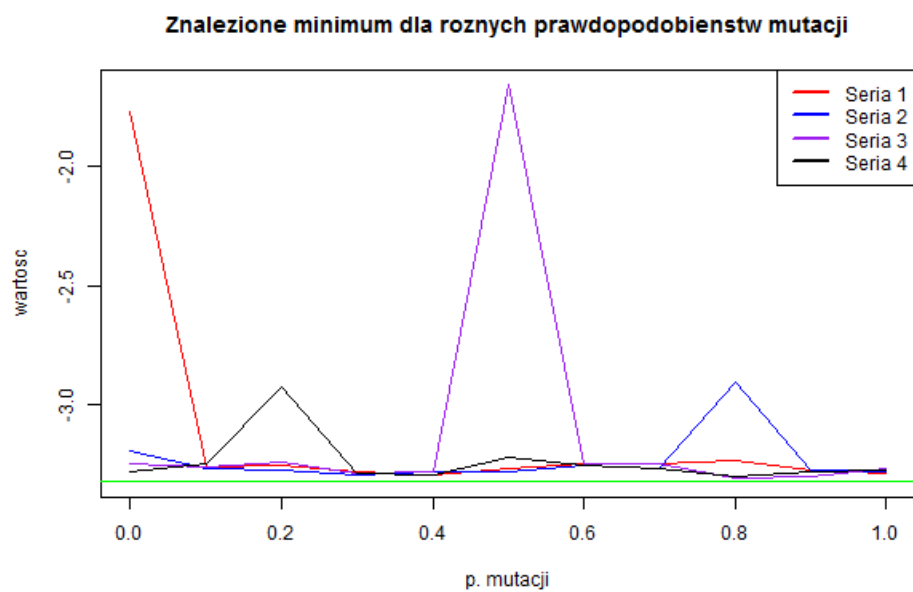
Hartman6 jest funkcją określoną dla ilości parametrów równej 6. Na ilustracji (rys. ??) przedstawiono jej wykres dla pierwszych dwóch.

$$f(\mathbf{x}) = -\sum_{i=1}^4 c_i \exp\left[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2\right] \quad (5)$$

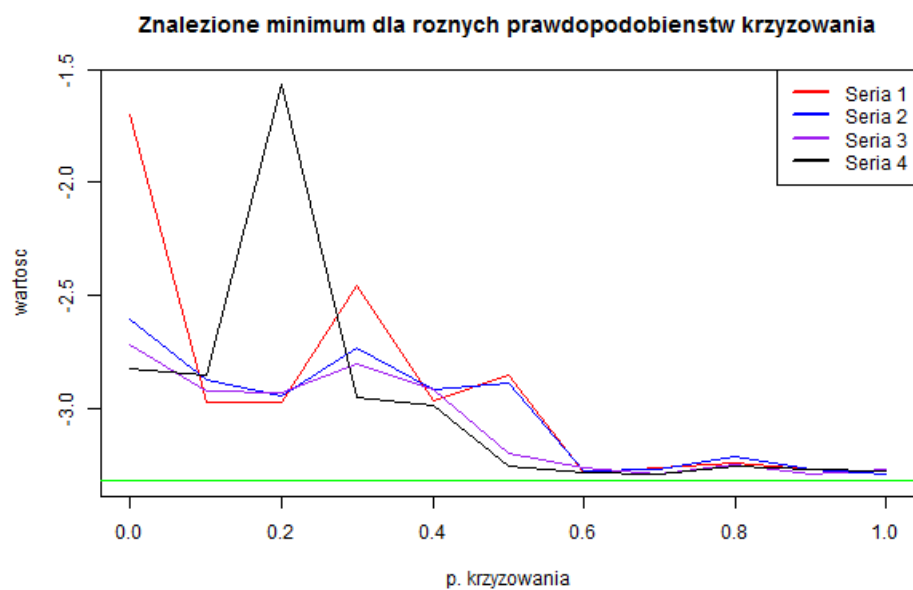
, gdzie $x_i \in [0, 1]$, $i \in \{1, \dots, 6\}$.



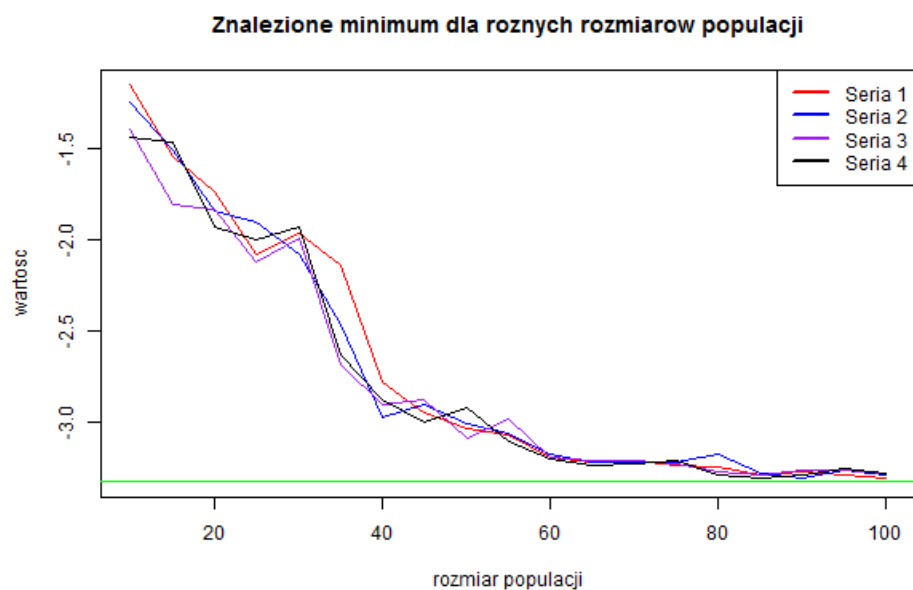
Rysunek 26: Wykres funkcji Hartman6 (d=6)



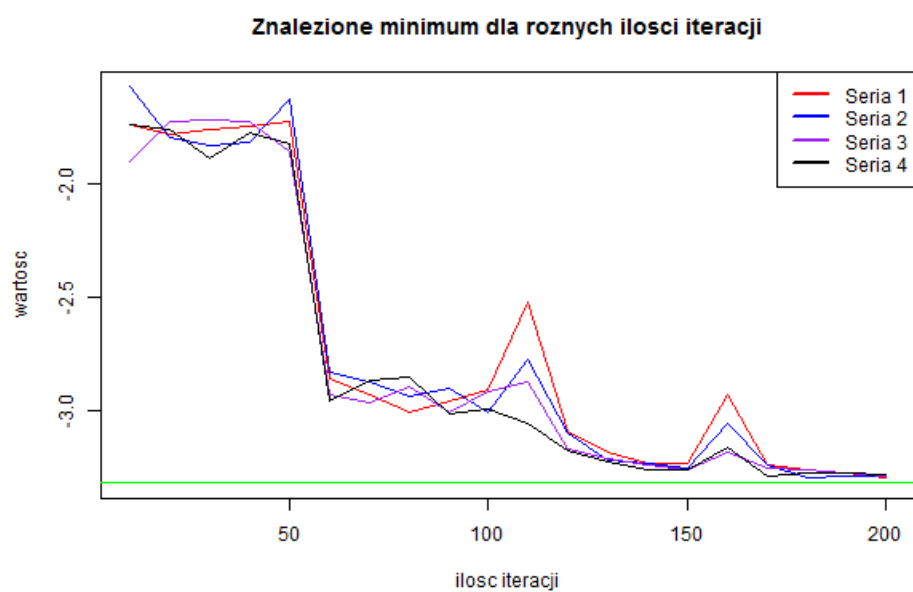
Rysunek 27: Wartość znalezione minimum dla funkcji Hartman6 w zależności od prawdopodobieństwa mutacji



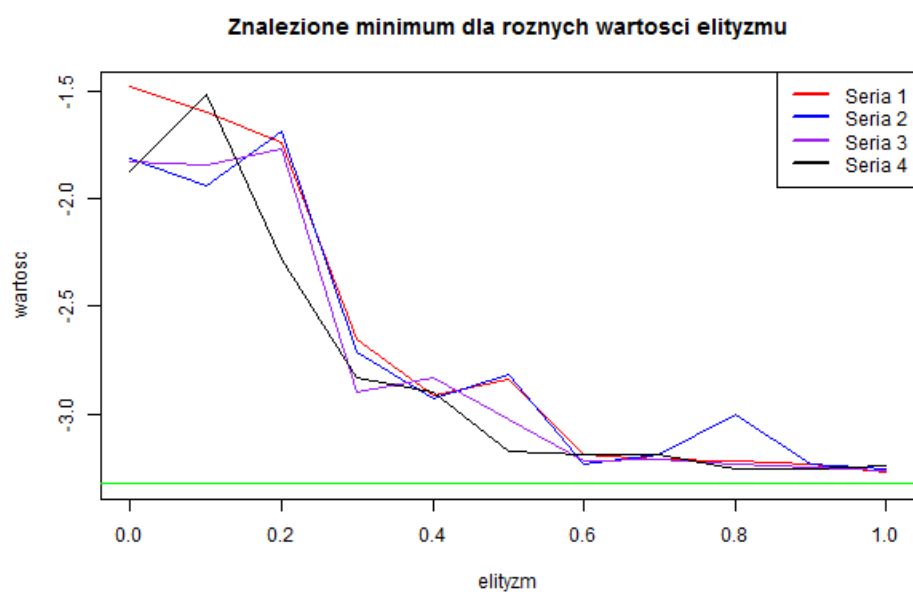
Rysunek 28: Wartość znalezione minimum dla funkcji Hartman6 w zależności od prawdopodobieństwa krzyżowania



Rysunek 29: Wartość znalezione minimum dla funkcji Hartman6 w zależności od rozmiarów populacji



Rysunek 30: Wartość znalezione minimum dla funkcji Hartman6 w zależności od ilości iteracji



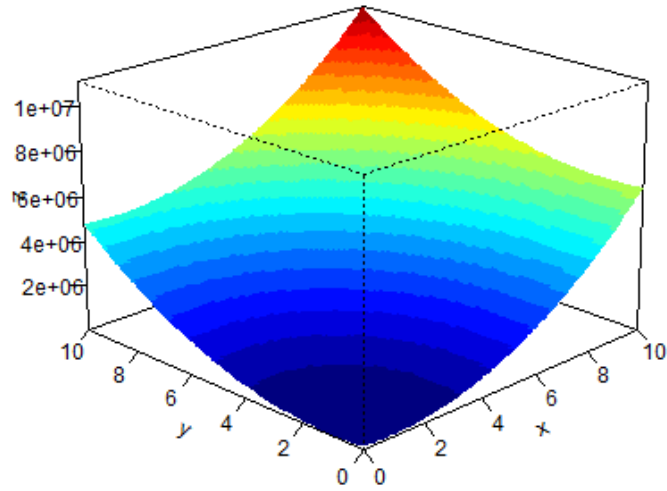
Rysunek 31: Wartość znalezione minimum dla funkcji Hartman6 w zależności od przyjętego elityzmu

3.6 PriceTransistor (9 parametrów)

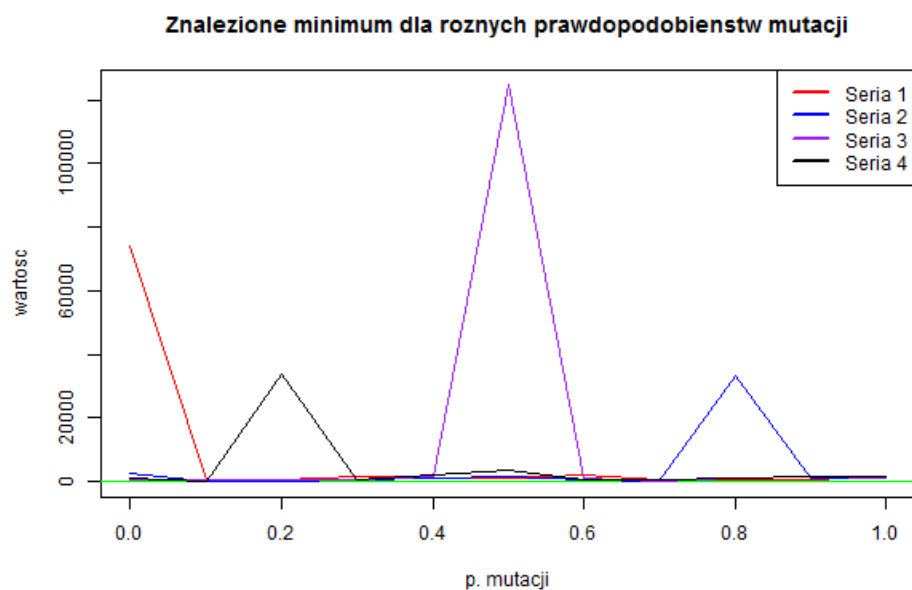
PriceTransistor jest funkcją określoną dla ilości parametrów równej 9. Na ilustracji (rys. 32) przedstawiono jej wykres dla pierwszych dwóch.

$$f(\mathbf{x}) = \gamma^2 + \sum_{k=1}^4 (\alpha_k^2 + \beta_k^2) \quad (6)$$

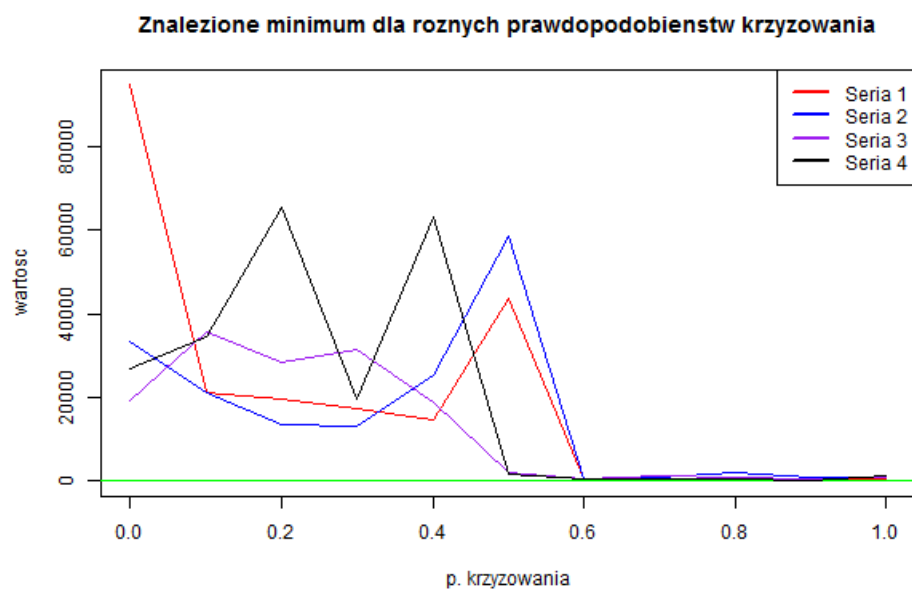
, gdzie $\alpha_k = (1 - x_1 x_2) x_3 \exp[x^5 (g_{1k} - g_{3k})]$.



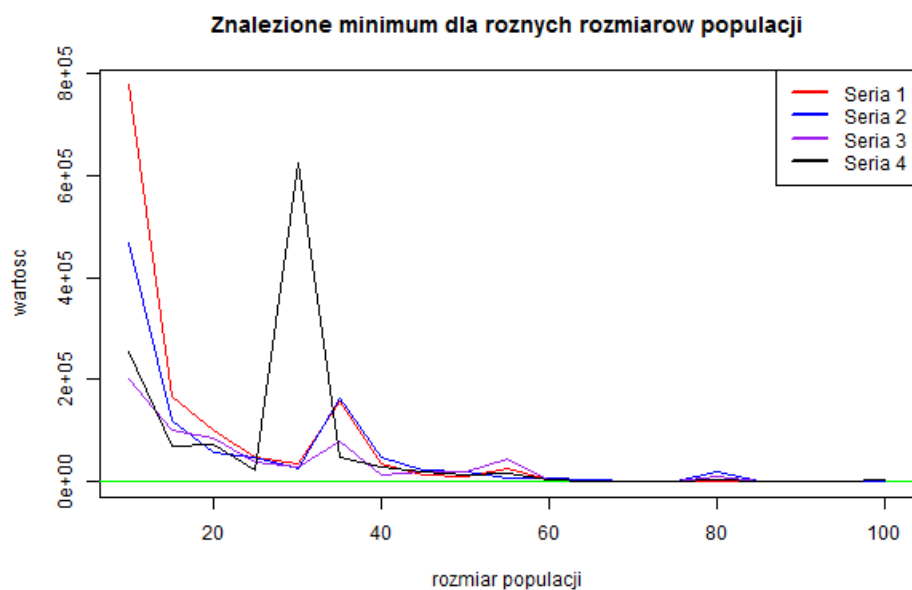
Rysunek 32: Wykres funkcji PriceTransistor (d=9)



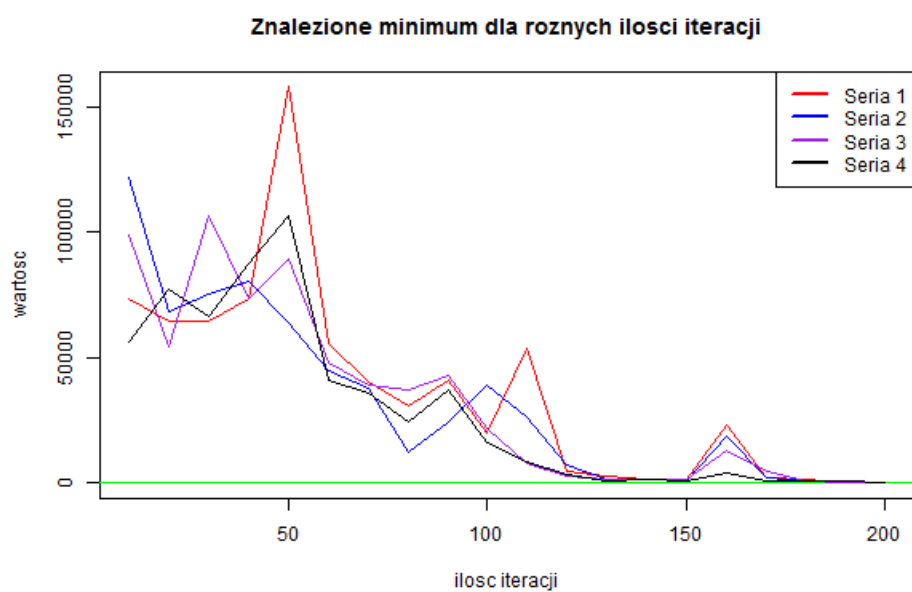
Rysunek 33: Wartość znalezione minimum dla funkcji PriceTransistor w zależności od prawdopodobieństwa mutacji



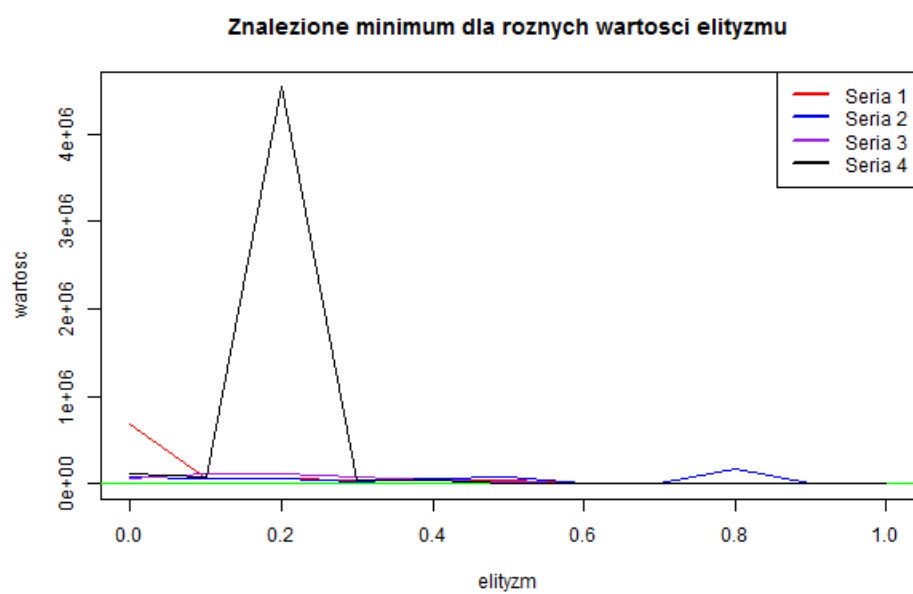
Rysunek 34: Wartość znalezione minimum dla funkcji PriceTransistor w zależności od prawdopodobieństwa krzyżowania



Rysunek 35: Wartość znalezione minimum dla funkcji PriceTransistor w zależności od rozmiarów populacji



Rysunek 36: Wartość znalezione minimum dla funkcji PriceTransistor w zależności od ilości iteracji

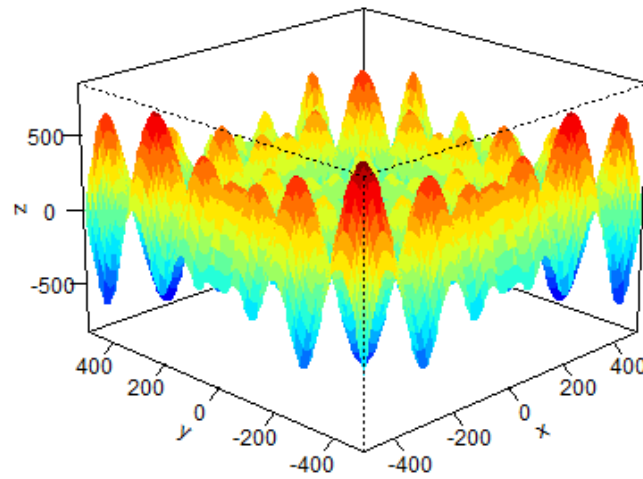


Rysunek 37: Wartość znalezione minimum dla funkcji PriceTransistor w zależności od przyjętego elityzmu

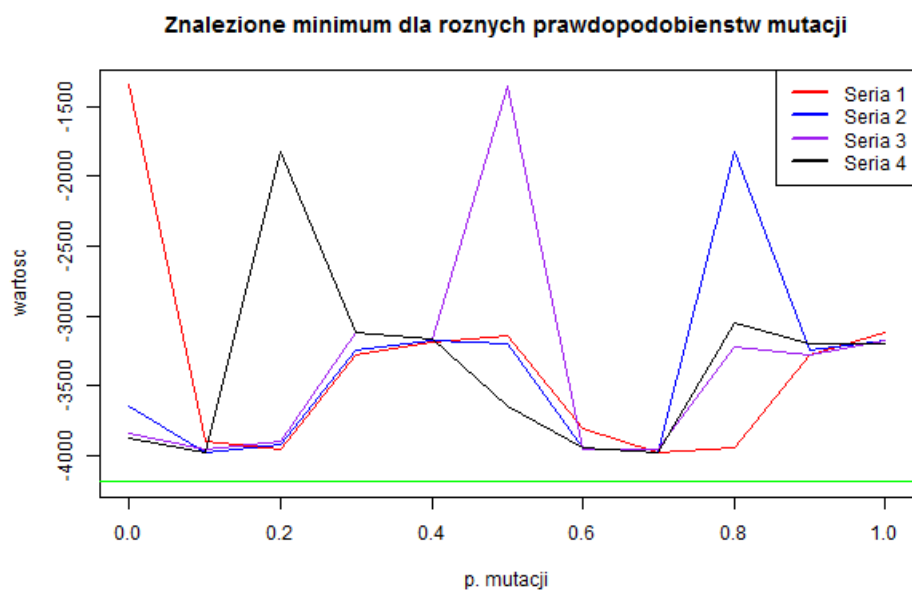
3.7 Schwefel (10 parametrów)

Schwefel jest funkcją określoną dla ilości parametrów równej 10. Na ilustracji (rys. 38) przedstawiono jej wykres dla pierwszych dwóch.

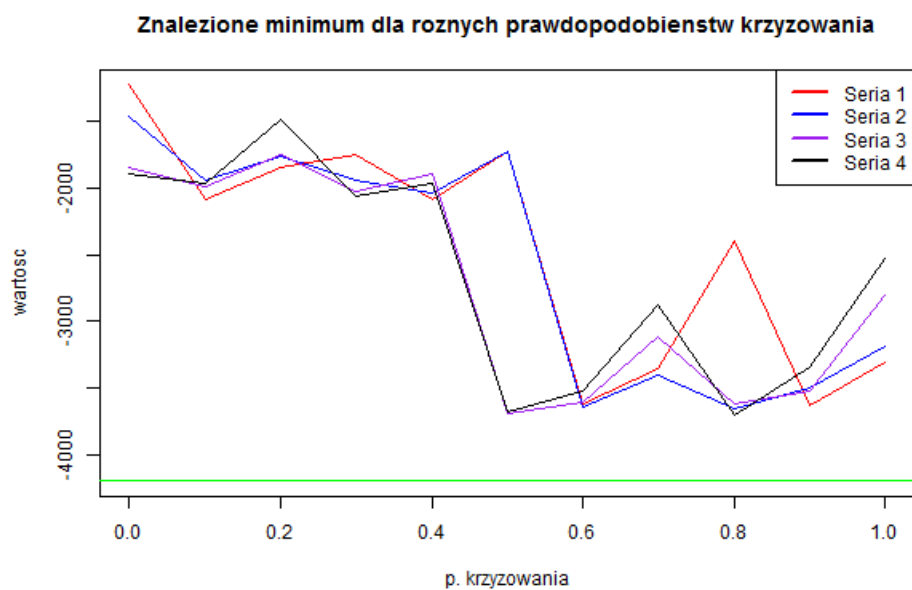
$$f(\mathbf{x}) = 418.9829d - \sum_{i=1}^d x_i \sin(\sqrt{|x_i|}) \quad (7)$$



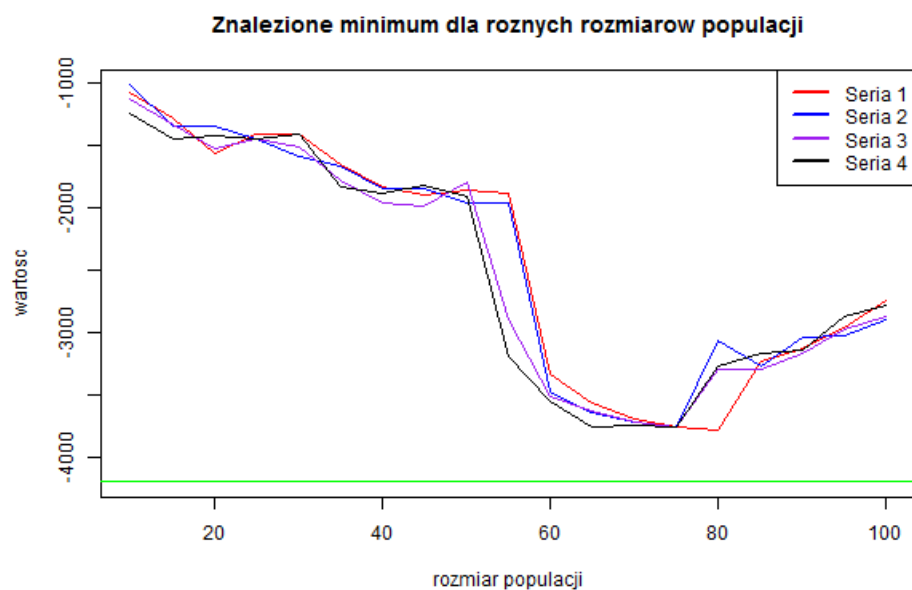
Rysunek 38: Wykres funkcji Schwefel (d=10) dla dwóch pierwszych wymiarów



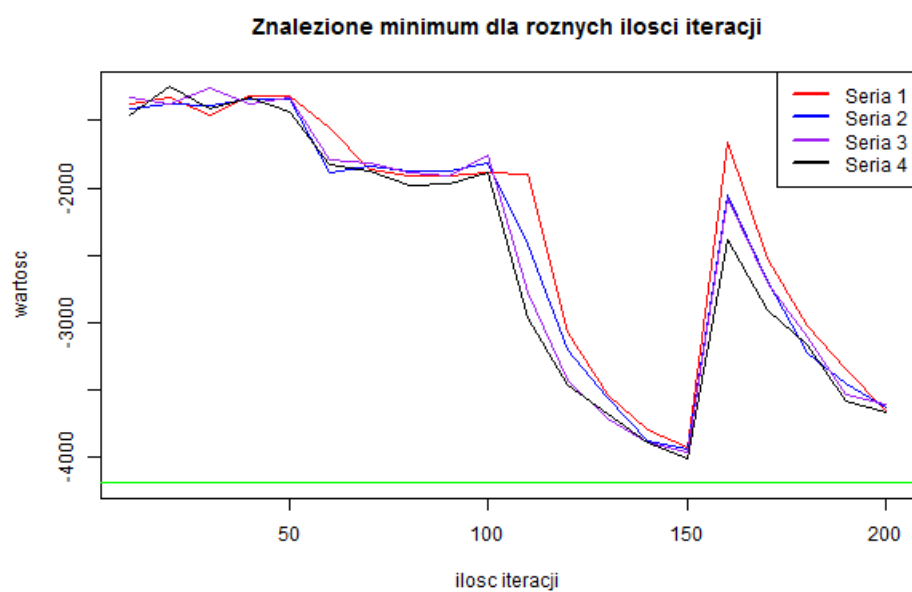
Rysunek 39: Wartość znalezionego minimum dla funkcji Schwefel w zależności od prawdopodobieństwa mutacji



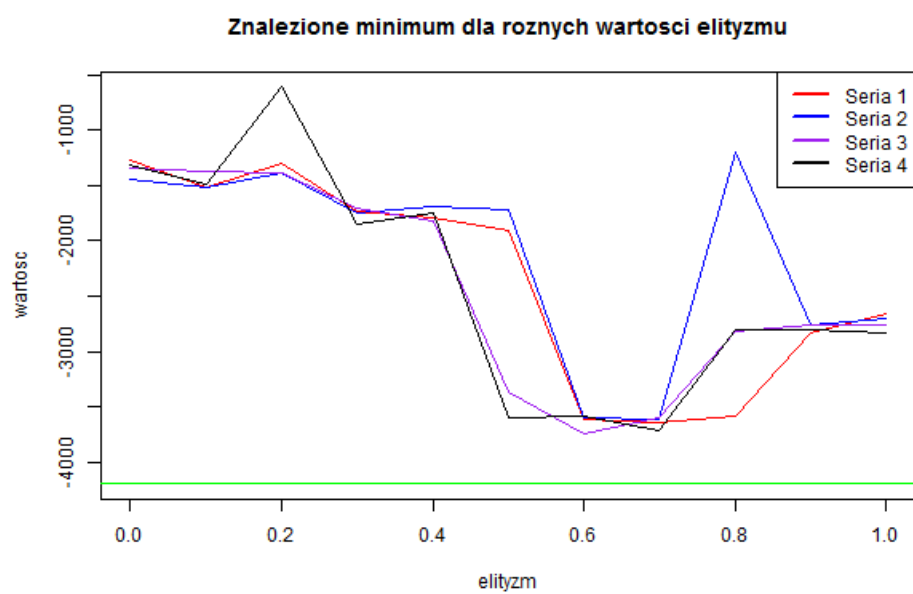
Rysunek 40: Wartość znalezionego minimum dla funkcji Schwefel w zależności od prawdopodobieństwa krzyżowania



Rysunek 41: Wartość znalezione minimum dla funkcji Schwefel w zależności od rozmiarów populacji



Rysunek 42: Wartość znalezione minimum dla funkcji Schwefel w zależności od ilości iteracji



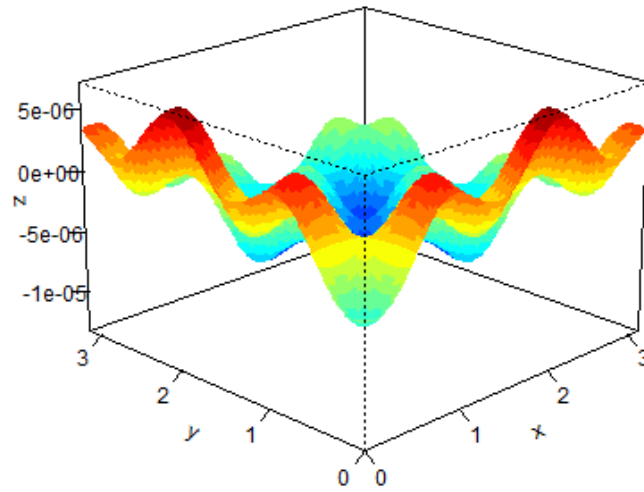
Rysunek 43: Wartość znalezione minimum dla funkcji Schwefel w zależności od przyjętego elityzmu

3.8 Zeldasine20 (20 parametrów)

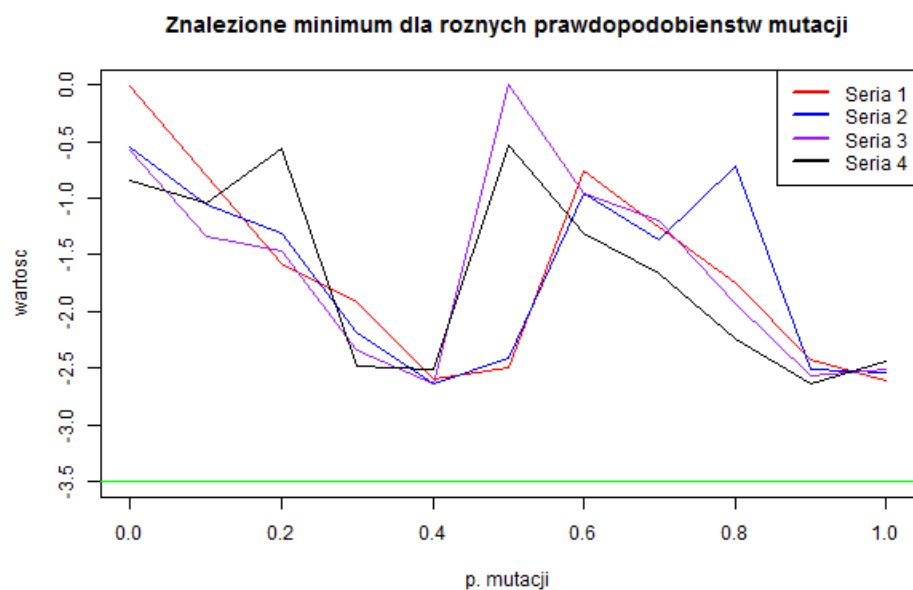
Zeldasine20 jest funkcją określoną dla ilości parametrów równej 20. Na ilustracji (rys. 44) przedstawiono jej wykres dla pierwszych dwóch.

$$f(\mathbf{x}) = -A \prod_{j=1}^D \sin(x_j - z) - \prod_{j=1}^D \sin(B \cdot (x_j - z)) \quad (8)$$

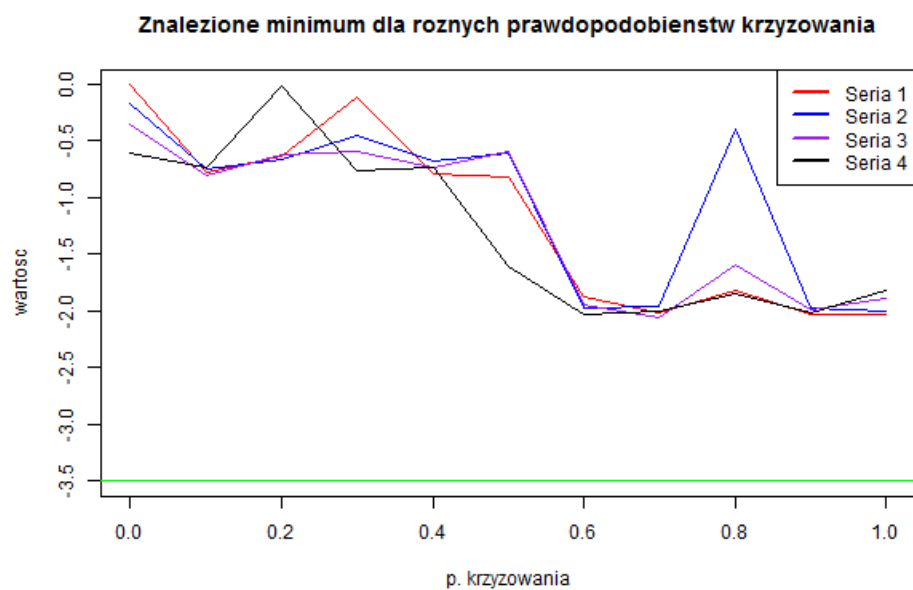
, gdzie $x_j \in [0, \pi]$ oraz $j = \{1, \dots, 10\}$.



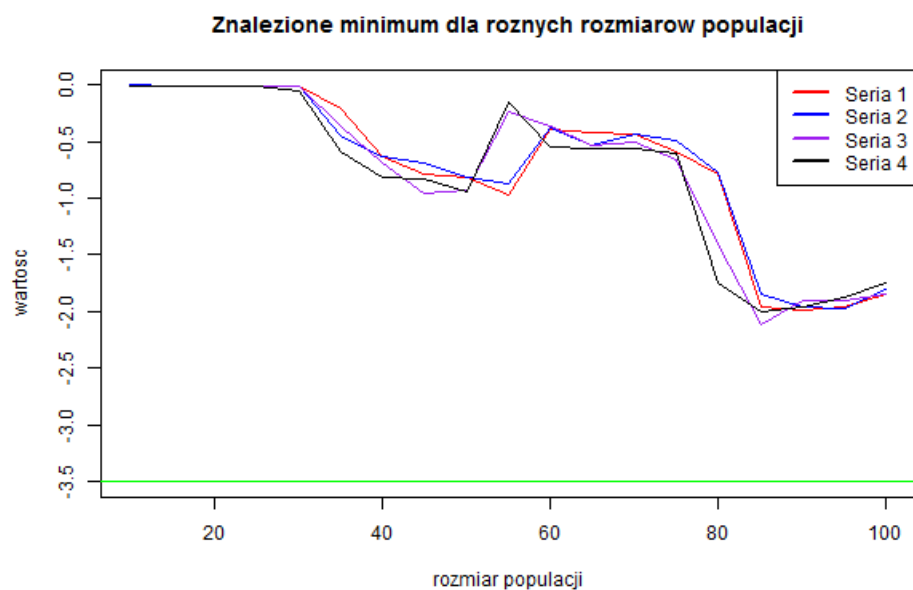
Rysunek 44: Wykres funkcji Zeldasine (d=20)



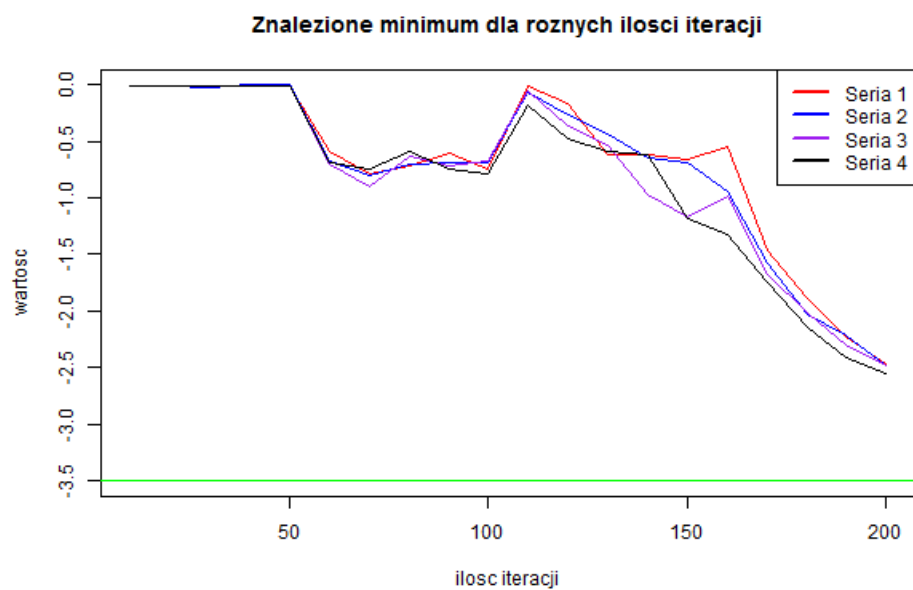
Rysunek 45: Wartość znalezionego minimum dla funkcji Zeldasine w zależności od prawdopodobieństwa mutacji



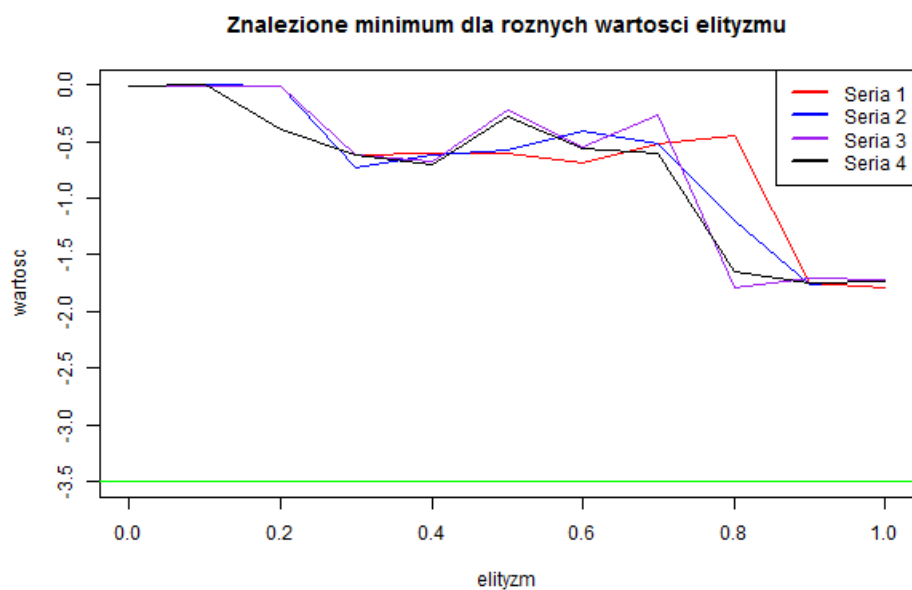
Rysunek 46: Wartość znalezionego minimum dla funkcji Zeldasine w zależności od prawdopodobieństwa krzyżowania



Rysunek 47: Wartość znalezione minimum dla funkcji Zeldasine w zależności od rozmiarów populacji



Rysunek 48: Wartość znalezione minimum dla funkcji Zeldasine w zależności od ilości iteracji



Rysunek 49: Wartość znalezione minimum dla funkcji Zeldasine w zależności od przyjętego elityzmu

Analizując otrzymane wyniki możemy stwierdzić, że nie udało się otrzymać wartości bliskiej szukanemu optimum. Jest to związane z dużą przestrzenią poszukiwań. Musimy pamiętać, że rozpatrujemy tu funkcję o 20 parametrach.

4 Podsumowanie

Wartość prawdopodobieństwa mutacji i krzyżowania zdaje się odgrywać drugorzędną rolę. Istotne jednak by chociaż jedna z nich była włączona z prawdopodobieństwem większym niż 0.

Najlepszym ustawieniem dla elityzmu jest prawdopodobieństwo rzędu 0,5.

Z pewnością należałoby zwiększyć ilość prób poddawanych uśrednianiu gdyż dla przyjętych 20 wyniki ciągle są niestabilne. Warto by również rozważyć pomijanie kilku najlepszych i najgorszych wyników przed uśrednianiem.

Literatura

- [1] Artur Suchwałko „Wprowadzenie do R dla programistów innych języków”
<https://cran.r-project.org/doc/contrib/R-dla-programistow-innych-jezykow.pdf>
- [2] Luca Scrucca „A quick tour of GA” <https://cran.r-project.org/web/packages/GA/vignettes/GA.html>
- [3] Surjanovic, S. & Bingham, D. (2013). Virtual Library of Simulation Experiments: Test Functions and Datasets. Retrieved April 3, 2017, from <http://www.sfu.ca/ssurjano>.