

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
1.1	Opis zadania projektowego	2
1.2	Środowisko testowe i narzędzia	2
2	Implementacja	3
2.1	Parametryzacja skryptu	8
3	Przebieg badań	9
3.1	Funkcja Ackleys (wariant 2D)	9
3.2	Funkcja Ackleys (wariant 3D)	9
3.3	Funkcja Branin (wariant 3D)	10
3.4	Funkcja Schwefel (wariant 3D)	10
4	Podsumowanie	11

1 Wprowadzenie

Wstęp

1.1 Opis zadania projektowego

1.2 Środowisko testowe i narzędzia

2 Implementacja

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 graphs <- TRUE
11 isSingleTest <- FALSE
12 n <- 2 # default 7
13 GAPopulation <- 10 # default 500
14 GAlterations <- 2 # default 50
15 GAMutations <- 0.1 # % (def 0.1)
16 GACrossovers <- 0.8 # % (def 0.8)
17
18 # Functions ----
19
20 # d <- 0.5
21 # funcName <- "Branin" #2d
22
23 # d <- 0.5
24 # funcName <- "Gulf" #3d
25
26 # d <- 0.5
27 # funcName <- "CosMix4" #4d
28
29 # d <- 0.8
30 # funcName <- "EMichalewicz" #5d
31
32 # d <- 0.8
33 # funcName <- "Hartman6" #6d
34
35 # d <- 0.8
36 # funcName <- "PriceTransistor" #9d
37
38 # d <- 20
39 # funcName <- "Schwefel" #10d
40
41 # d <- 20
42 # funcName <- "Zeldasine20" #20d
43
44
45 # Processing ----
46
47 dim <- getProblemDimen(funcName)
48 B <- matrix(unlist(getDefaultBounds(funcName)), ncol=dim, byrow=TRUE)
49 f <- function(xx) goTest(par=c(xx, rep(0, dim-length(xx))), fnName=funcName,
50                           checkDim = TRUE)
51 globalOpt <- getGlobalOpt(funcName)
```

```

51
52 if (graphs) {
53   x <- seq(B[1,1], B[2,1], by = d)
54   y <- seq(B[1,2], B[2,2], by = d)
55   z <- outer(x, y, Vectorize(function(x,y) f(c(x,y))))
56   nbcol = 100
57   color = rev(rainbow(nbcol, start = 0/6, end = 4/6))
58   zcol = cut(z, nbcol)
59   persp3d(x, y, z, theta=50, phi=25, expand=0.75, col=color[zcol],
60           ticktype="detailed", axes=TRUE)
61   persp3D(x, y, z, theta = -45, phi = 20, color.palette = jet.colors)
62 }
63
64 if (isSingleTest) {
65   vector <- rep(NA,n)
66   for (i in 1:n) {
67     GAmin <- ga(type = "real-valued", fitness = function(xx) -f(xx),
68               min = c(B[1,]), max = c(B[2,]),
69               popSize = GAPopulation, maxiter = GAIterations,
70               pmutation = GAMutations, pcrossover = GACrossovers)
71     solution <- matrix(unlist(GAmin@solution), ncol=dim, byrow=TRUE)
72     vector[i] <- f(solution[1,])
73   }
74   result <- matrix(c(vector), nrow = n, ncol = 1)
75   write.table(result, file = "resultsSingle.csv", row.names=FALSE, na="",
76             col.names=FALSE, sep=";")
77
78 } else {
79
80   gMin <- .Machine$integer.max
81   gBest <- NA
82
83   temp <- c()
84   values <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
85   averages <- c()
86   for (mutation in values) {
87     sum <- 0
88     vector <- rep(NA,n)
89     for (i in 1:n) {
90       GAmin <- ga(type = "real-valued",
91                 fitness = function(xx) -f(xx),
92                 min = c(B[1,]), max = c(B[2,]),
93                 popSize = GAPopulation, maxiter = GAIterations,
94                 pmutation = mutation, pcrossover = GACrossovers)
95       solution <- matrix(unlist(GAmin@solution), ncol=dim, byrow=TRUE)
96       eval <- f(solution[1,])
97       if (eval < gMin) {
98         gMin <- eval
99         gBest <- GAmin
100       }
101       sum <- sum + eval
102       vector[i] <- eval
103     }
104     temp <- c(temp, vector)
105     averages <- c(averages, (sum / n))

```

```

106 }
107 result <- matrix(c(temp),nrow = n,ncol = length(values))
108 write.table(result, file = "resultsMutations.csv", row.names=FALSE, na="",
109             col.names=FALSE, sep=";")
110
111 if (graphs) {
112     plot(values, averages,
113          main="Goal function value for different mutation probabilities",
114          ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
115          type="l", col="red", xlab="params", ylab="value")
116     abline(globalOpt,0, col="green")
117 }
118
119 temp <- c()
120 values <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
121 averages <- c()
122 for (crossover in values) {
123     sum <- 0
124     vector <- rep(NA,n)
125     for (i in 1:n) {
126         GAmin <- ga(type = "real-valued",
127                    fitness = function(xx) -f(xx),
128                    min = c(B[1,]), max = c(B[2,]),
129                    popSize = GAPopulation, maxiter = GAIterations,
130                    pmutation = GAMutations, pcrossover = crossover)
131         solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
132         eval <- f(solution[1,])
133         if (eval < gMin) {
134             gMin <- eval
135             gBest <- GAmin
136         }
137         sum <- sum + eval
138         vector[i] <- eval
139     }
140     temp <- c(temp, vector)
141     averages <- c(averages, (sum / n))
142 }
143 result <- matrix(c(temp),nrow = n,ncol = length(values))
144 write.table(result, file = "resultsCrossover.csv", row.names=FALSE, na="",
145             col.names=FALSE, sep=";")
146
147 if (graphs) {
148     plot(values, averages,
149          main="Goal function value for different crossover probabilities",
150          ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
151          type="l", col="red", xlab="params", ylab="value")
152     abline(globalOpt,0, col="green")
153 }
154
155 temp <- c()
156 values <- c(0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5)
157 averages <- c()
158 for (elitism in values) {
159     sum <- 0
160     vector <- rep(NA,n)
161     for (i in 1:n) {

```

```

160     GAmin <- ga(type = "real-valued",
161               fitness = function(xx) -f(xx),
162               min = c(B[1,]), max = c(B[2,]),
163               popSize = GAPopulation, maxiter = GAIterations,
164               pmutation = GAMutations, pcrossover = GACrossovers, elitism =
                  elitism)
165     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
166     eval <- f(solution[1,])
167     if (eval < gMin) {
168       gMin <- eval
169       gBest <- GAmin
170     }
171     sum <- sum + eval
172     vector[i] <- eval
173   }
174   temp <- c(temp, vector)
175   averages <- c(averages, (sum / n))
176 }
177 result <- matrix(c(temp),nrow = n,ncol = length(values))
178 write.table(result, file = "resultsElitism.csv", row.names=FALSE, na="",
179             col.names=FALSE, sep=";")
180
181 if (graphs) {
182   plot(values, averages,
183         main="Goal function value for different elitism",
184         ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
185         type="l", col="red", xlab="params", ylab="value")
186   abline(globalOpt,0, col="green")
187 }
188
189 temp <- c()
190 values <- c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000)
191 averages <- c()
192 for (population in values) {
193   sum <- 0
194   vector <- rep(NA,n)
195   for (i in 1:n) {
196     GAmin <- ga(type = "real-valued",
197               fitness = function(xx) -f(xx),
198               min = c(B[1,]), max = c(B[2,]),
199               popSize = population, maxiter = GAIterations,
200               pmutation = GAMutations, pcrossover = GACrossovers)
201     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
202     eval <- f(solution[1,])
203     if (eval < gMin) {
204       gMin <- eval
205       gBest <- GAmin
206     }
207     sum <- sum + eval
208     vector[i] <- eval
209   }
210   temp <- c(temp, vector)
211   averages <- c(averages, (sum / n))
212 }
213 result <- matrix(c(temp),nrow = n,ncol = length(values))

```

```

213 write.table(result, file = "resultsPopulation.csv", row.names=FALSE, na="",
214             col.names=FALSE, sep=";")
215
216 if (graphs) {
217     plot(values, averages,
218          main="Goal function value for different population sizes",
219          ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
220          type="l", col="red", xlab="params", ylab="value")
221     abline(globalOpt,0, col="green")
222 }
223
224 temp <- c()
225 values <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
226 averages <- c()
227 for (iterations in values) {
228     sum <- 0
229     vector <- rep(NA,n)
230     for (i in 1:n) {
231         GAmin <- ga(type = "real-valued",
232                    fitness = function(xx) -f(xx),
233                    min = c(B[1,]), max = c(B[2,]),
234                    popSize = GAPopulation, maxiter = iterations,
235                    pmutation = GAMutations, pcrossover = GACrossovers)
236         solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
237         eval <- f(solution[1,])
238         if (eval < gMin) {
239             gMin <- eval
240             gBest <- GAmin
241         }
242         sum <- sum + eval
243         vector[i] <- eval
244     }
245     temp <- c(temp, vector)
246     averages <- c(averages, (sum / n))
247 }
248 result <- matrix(c(temp),nrow = n,ncol = 10)
249 write.table(result, file = "resultsIterations.csv", row.names=FALSE, na="",
250             col.names=FALSE, sep=";")
251
252 if (graphs) {
253     plot(values, averages,
254          main="Goal function value for different iteration quantities",
255          ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
256          type="l", col="red", xlab="params", ylab="value")
257     abline(globalOpt,0, col="green")
258 }
259
260 if (graphs) {
261     summary(GAmin)
262     filled.contour(x, y, z, color.palette = jet.colors, nlevels = 24,
263                   plot.axes = {
264                       axis(1);
265                       axis(2);

```



```
266     points(solution[1,1], solution[1,2], pch = 3, cex = 5, col = "black", lwd  
      = 2)  
267   }  
268 )  
269 plot(GAmin)  
270 }
```

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”.

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)

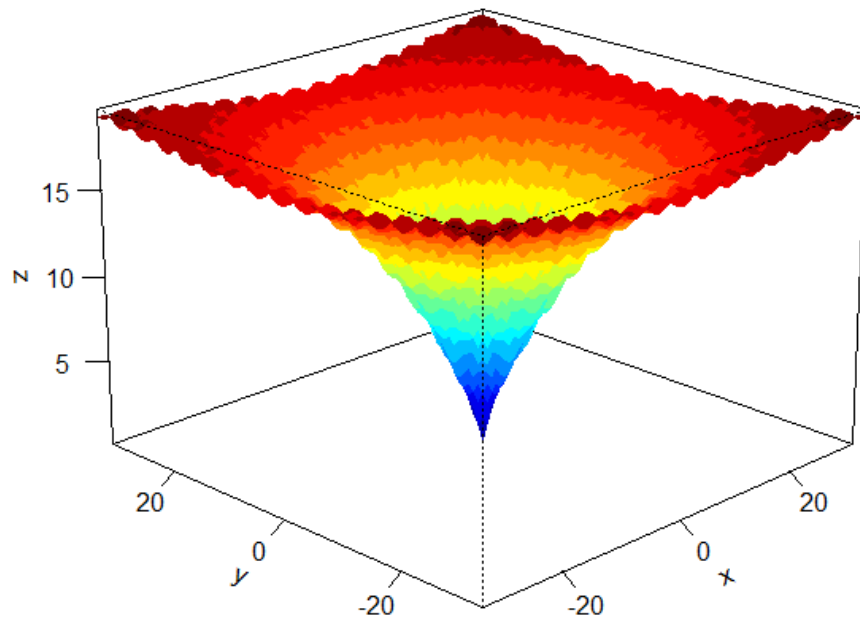
3.1 Funkcja Ackleys (wariant 2D)

Test

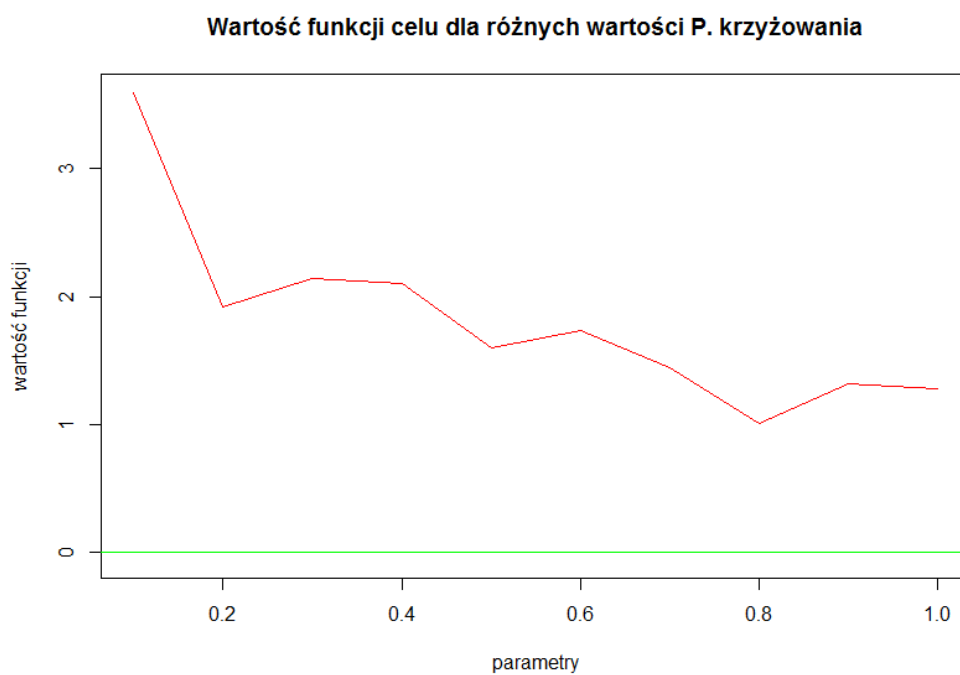
3.2 Funkcja Ackleys (wariant 3D)

Test

Na ilustracji (rys. 1) przedstawiono wykres omawianej funkcji.



Rysunek 1: Wykres funkcji Ackleys (d=3)



Rysunek 2: Wartość znalezionej minimum funkcji w zależności od P. krzyżowania

3.3 Funkcja Branin (wariant 3D)

Test

3.4 Funkcja Schwefel (wariant 3D)

Minimum -837,9658 w punkcie (420,9687; 420,9687).

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>