

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	Branin (variant 2D)	9
3.2	Gulf (variant 3D)	9
3.3	CosMix4 (variant 4D)	9
3.4	EMichalewicz (variant 5D)	9
3.5	Hartman6 (variant 6D)	9
3.6	PriceTransistor (variant 9D)	9
3.7	Schwefel (variant 10D)	9
3.8	Zeldasine20 (variant 20D)	9
3.9	Test	10
4	Podsumowanie	12

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 n <- 5 # default 7
11 GAPopulation <- 10 # default 500
12 GAlterations <- 10 # default 50
13 GAMutations <- 0.1 # % (def 0.1)
14 GACrossovers <- 0.8 # % (def 0.8)
15
16 isSingleTest <- FALSE
17 graphs <- TRUE
18 quality <- 100 #graph probes
19
20 mutationTests <- seq(0, 1, 0.05)
21 crossoverTests <- seq(0, 1, 0.05)
22 elitismTests <- seq(0, 1, 0.05)
23 populationTests <- seq(10, 200, 10)
24 iterationTests <- seq(1, 20, 1)
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
39 dim <- getProblemDimen(funcName)
40 B <- matrix(unlist(getDefaultBounds(funcName)),ncol=dim,byrow=TRUE)
41 f <- function(xx) goTest(par=c(xx, rep(0, dim-length(xx))), fnName=funcName,
42   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
```

```

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

```

```

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

```

```

159     solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
160     eval <- f(solution[1,])
161     if (eval < gMin) {
162         gMin <- eval
163         gBest <- GAmin
164     }
165     sum <- sum + eval
166     vector[i] <- eval
167 }
168 temp <- c(temp, vector)
169 averages <- c(averages, (sum / n))
170 }
171 result <- matrix(c(temp),nrow = n,ncol = length(values))
172 write.table(result, file = "resultsElitism.csv", row.names=FALSE, na="",
173             col.names=FALSE, sep=";")
174
175 if (graphs) {
176     plot(values, averages,
177          main="Goal function value 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         }
201         sum <- sum + eval
202         vector[i] <- eval
203     }
204     temp <- c(temp, vector)
205     averages <- c(averages, (sum / n))
206 }
207 result <- matrix(c(temp),nrow = n,ncol = length(values))
208 write.table(result, file = "resultsPopulation.csv", row.names=FALSE, na="",
209             col.names=FALSE, sep=";")
210
211 if (graphs) {
212     plot(values, averages,
213          main="Goal function value for different population sizes",
214          ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),

```

```

213     type="l", col="red", xlab="params", ylab="value")
214     abline(globalOpt,0, col="green")
215 }
216
217 temp <- c()
218 values <- iterationTests
219 averages <- c()
220 for (iterations in values) {
221     sum <- 0
222     vector <- rep(NA,n)
223     for (i in 1:n) {
224         GAmin <- ga(type = "real-valued",
225             fitness = function(xx) -f(xx),
226             min = c(B[1,]), max = c(B[2,]),
227             popSize = GAPopulation, maxiter = iterations,
228             pmutation = GAMutations, pcrossover = GACrossovers)
229         solution <- matrix(unlist(GAmin@solution),ncol=dim,byrow=TRUE)
230         eval <- f(solution[1,])
231         if (eval < gMin) {
232             gMin <- eval
233             gBest <- GAmin
234         }
235         sum <- sum + eval
236         vector[i] <- eval
237     }
238     temp <- c(temp, vector)
239     averages <- c(averages, (sum / n))
240 }
241 result <- matrix(c(temp),nrow = n,ncol = 10)
242 write.table(result, file = "resultsIterations.csv", row.names=FALSE, na="",
243     col.names=FALSE, sep=";")
244
245 if (graphs) {
246     plot(values, averages,
247         main="Goal function value for different iteration quantities",
248         ylim=c(min(c(averages,globalOpt)),max(c(averages,globalOpt))),
249         type="l", col="red", xlab="params", ylab="value")
250     abline(globalOpt,0, col="green")
251 }
252 }
253
254 if (graphs) {
255     summary(GAmin)
256     filled.contour(x, y, z, color.palette = jet.colors, nlevels = 24,
257         plot.axes = {
258             axis(1);
259             axis(2);
260             points(solution[1,1], solution[1,2], pch = 3, cex = 5, col = "black", lwd
261                 = 2)
262         }
263     )
264     plot(GAmin)
265 }

```


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 Branin (variant 2D)

Test

3.2 Gulf (variant 3D)

Test

3.3 CosMix4 (variant 4D)

Test

3.4 EMichalewicz (variant 5D)

Test

3.5 Hartman6 (variant 6D)

Test

3.6 PriceTransistor (variant 9D)

Test

3.7 Schwefel (variant 10D)

Test

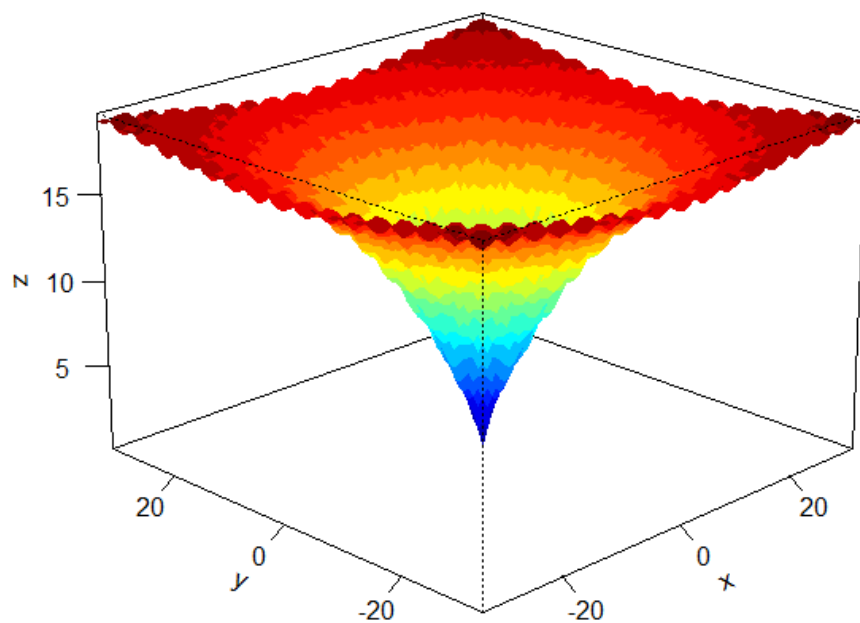
3.8 Zeldasine20 (variant 20D)

Test

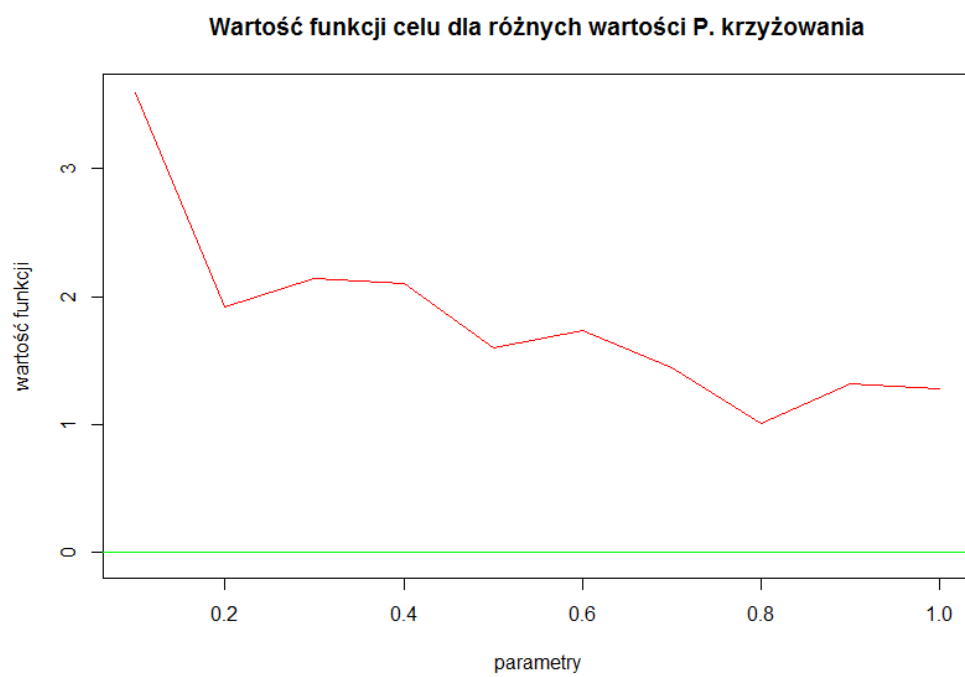
3.9 Test

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

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>