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

Wstęp

1.1 Opis zadania projektowego

1.2 Środowisko testowe i narzędzia

2 Implementacja

Listing 1: Skrypt w języku R wykorzystany do badań

```

rm(list=ls())

require("GA")
require("globalOptTests")
require("rgl")

# Params ----

graphs <- TRUE
isSingleTest <- FALSE
n <- 2              # default 7
GAPopulation <- 10 # default 500
GAIterations <- 2 # default 50
GAMutations <- 0.1 # % (def 0.1)
GACrossovers <- 0.8 # % (def 0.8)

# Functions ----

# d <- 0.5
# funcName <- "Branin" #2d

# d <- 0.5
# funcName <- "Gulf" #3d

# d <- 0.5
# funcName <- "CosMix4" #4d

# d <- 0.8
# funcName <- "EMichalewicz" #5d

# d <- 0.8
# funcName <- "Hartman6" #6d

# d <- 0.8
# funcName <- "PriceTransistor" #9d

# d <- 20
# funcName <- "Schwefel" #10d

# d <- 20
# funcName <- "Zeldasine20" #20d


# Processing ----

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)
```

```r

if (graphs) {
  x <- seq(B[1,1], B[2,1], by = d)
  y <- seq(B[1,2], B[2,2], by = d)
  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 <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
  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))
```

```r
}
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="Goal function value 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 <- c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0)
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="Goal function value 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 <- c(0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5)
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="Goal function value 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 <- c(100, 200, 300, 400, 500, 600, 700, 800, 900, 1000)
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="Goal function value 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 <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
  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 = 10)
  write.table(result, file = "resultsIterations.csv", row.names=FALSE, na="", col.names=FALSE, sep=";")

  if (graphs) {
    plot(values, averages,
         main="Goal function value for different iteration quantities",
         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)

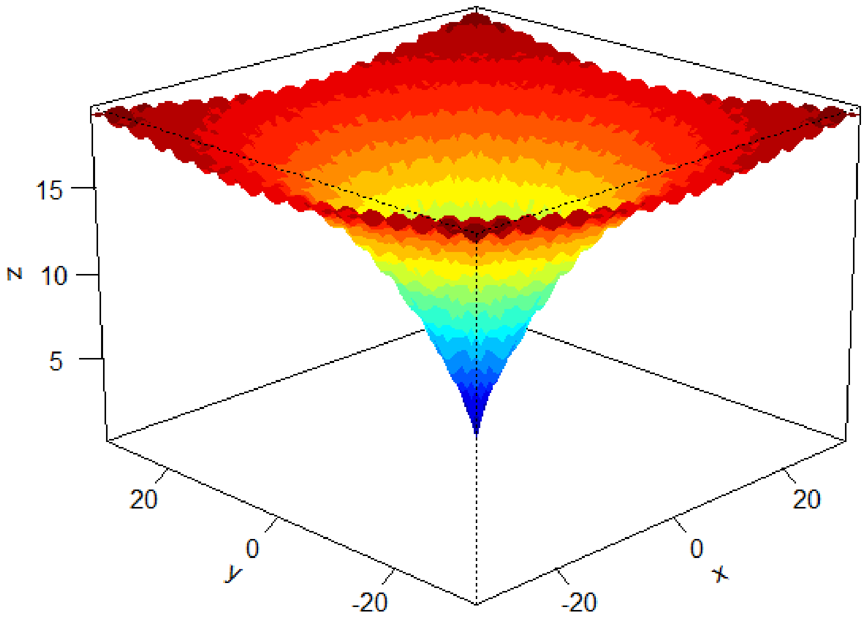
3.1 Funkcja Ackleys (wariant 2D)

Test

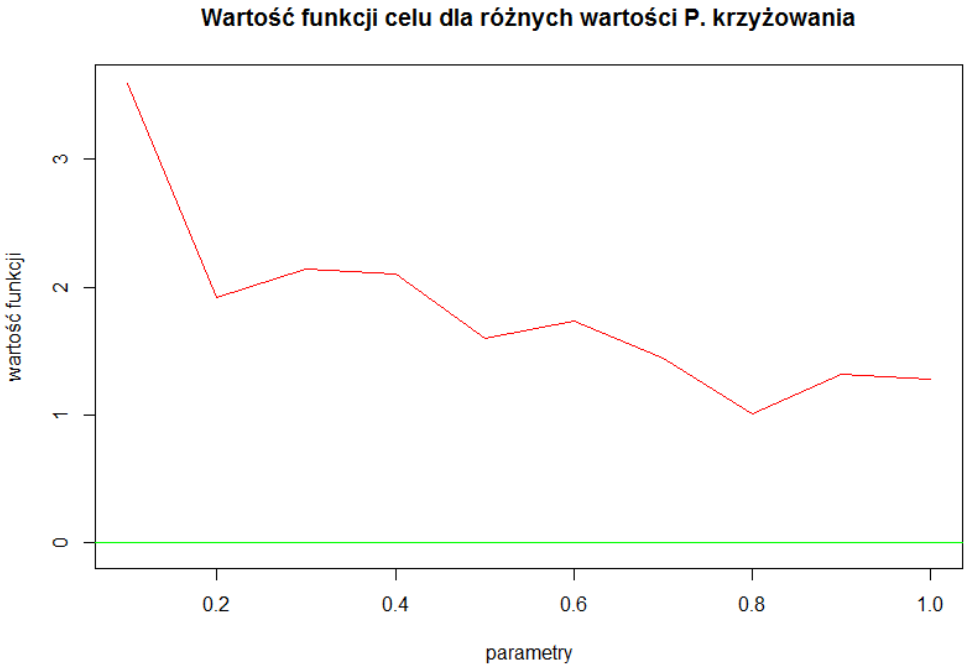
3.2 Funkcja Ackleys (wariant 3D)

Test

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



Rysunek 1: Wykres funkcji Ackleys (d=3)



Rysunek 2: Wartość znalezionego 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