Case Study 03

Frederico Augustos (Relator), Mariana Pimenta (Monitora), Patrícia Vaz (Coordenadora), Pedro Maia (Verificador)

11 de novembro de 2019

Introdução

A análise de desempenho de algoritmos é uma prática comum entre pesquisadores que buscam fortes evidências para a adoção de um novo modelo em comparação com o estado da arte ou diferentes configurações do novo método proposto. Dessa forma, este trabalho apresenta a análise estatística de comparação por blocagem do desempenho de duas diferentes configurações de um método de otimização baseado no algoritmo de Evolução diferencial Storn and Price (1997). Para isso, foram geradas duas configurações do algoritmo através do pacote ExpDE Campelo and Botelho (2016), e a elas foram aplicadas as funções de interesse composta por funções de Rosenbrock (1960), também conhecida como função Valley ou Banana, que é um problema de teste popular para algoritmos de otimização baseados em gradiente. A função é unimodal, e o mínimo global está em um vale estreito e parabólico e, embora este vale seja fácil de encontrar, a convergência ao mínimo é difícil. A dimensão da função de Rosenbrock define a instância do problema.

Os seguintes parâmetros experimentais são dados para este estudo:

- Mínima diferença de importância prática (padronizada): $(d^* = \delta^*/\sigma) = 0,5$;
- Nível Significância: $\alpha = 0,05$;
- Potência mínima (para o caso $d = d^*$): $\pi = 1 - \beta = 0,8$.

Instalação dos pacotes necessários

```
# clean workspace
rm(list=ls())

# install required packages
packages_needed <- c("ExpDE","smoof")
for (package_name in packages_needed) {
      if (!(package_name %in% rownames(installed.packages()))){
            install.packages(package_name)
      }
}
```

```
## Installing package into '/home/pedro/R/x86_64-pc-linux-gnu-library/3.4'
## (as 'lib' is unspecified)
```

```
## also installing the dependencies 'curl', 'httr', 'plotly'
```

```
## Warning in install.packages(package_name): installation of package 'curl'
## had non-zero exit status
```

```
## Warning in install.packages(package_name): installation of package 'httr'
## had non-zero exit status
```

```
## Warning in install.packages(package_name): installation of package 'plotly'
## had non-zero exit status
```

```
## Warning in install.packages(package_name): installation of package 'smoof'
## had non-zero exit status
```

Formulação das Hipóteses

A hipótese a ser testada é a existência de alguma diferença de desempenho médio entre as duas configurações fornecidas para o problema em questão. E por se tratar de um problema de blocagem, visto que as instâncias influenciam na análise do problema, o parâmetro de interesse a ser avaliado no teste de hipóteses é o efeito de cada algorítmo. Na hipótese nula, H_0, foi considerado que os efeitos dos algoritmos τ_i da **configuração 1** e da **configuração 2** são iguais a zero. Isto é, H_0 estabelece que as configurações não apresentam diferença significativa em relação ao estado da arte. Já a hipótese alternativa do teste H_1 é de que existe pelo menos um algoritmo que possui efeito diferente de zero. Assim:

$$\begin{cases} H_0 : \tau_i = 0 \\ H_1 : \tau_i \neq 0 \end{cases} \tag{1}$$

Cálculo do número de instâncias

Para aplicação da Blocagem, se faz necessário o cálculo do número de blocos em que os dados serão divididos para a análise posterior. No estudo de caso 03, cada bloco possui uma única instância e foi utilizado o algoritmo proposto por Campelo (2019) para definir o número de blocos. Desta forma, o número de blocos foi cálculado a partir de um teste com potência estatística predefinida, $\pi = 1 - \beta = 0.8$), de forma a detectar diferenças iguais ou superiores ao mínimo tamanho de efeito relevante $\delta^* = 0.5$, em um nível de significância $\alpha = 0.05$. Com isso, obteve-se o número mínimo necessário de blocos para a comparação dos dois algoritmos, obedecendo a seguinte relação:

$$N^* = \min N | t^{N-1}_{1-\alpha/2} \leq t^{N-1}_{\beta^* ncp}$$

```
alpha <- 0.05
delta <- 0.5
beta <- 0.2

n <- 2
while (qt(1 - alpha/2, n-1) > qt(beta, n - 1, delta*sqrt(n))) n <- n + 1
print(n)
```

```
## [1] 34
```

Coleta de dados e Execução do projeto piloto

A classe de funções de interesse para a aplicação do teste é composta por funções Rosenbrock (Rosenbrock, 1960) (Pohlheim, 2005) com dimensão de 2 a 150. Desta forma a blocagem foi feita de forma a agrupar certo número de dimensões das funções de Rosenbrock e Pohlheim. Inicialmente utilizou-se o número de blocos obtido no teste anterior n. E de forma aleatória optou-se por realizar 10 repetições em cada bloco.

```
arquivo="pilot.csv"

nRuns <- 10
#suppressPackageStartupMessages(library(smoof))
suppressPackageStartupMessages(library(ExpDE))

if (file.exists(arquivo)) {
    data <- read.csv(file=arquivo, header = TRUE, sep=",")
    dims <- data$dim
    meanFx1 <- data$mean.config1
    meanFx2 <- data$mean.config2
    sdFx1 <- data$sd.config1
```

```
    sdFx2 <- data$sd.config2
    nDims <- length(dims)
} else {

    nDims <- n
    dims <- sort(sample(2:150, nDims))

    meanFx1 <- vector(,nDims)
    meanFx2 <- vector(,nDims)
    sdFx1 <- vector(,nDims)
    sdFx2 <- vector(,nDims)
    for (d in 1:nDims)
    {
        dim <- dims[d]
        fn <- function(X)
        {
            if(!is.matrix(X)) X <- matrix(X, nrow = 1) # <- if a single vector is passed as X
            Y <- apply(X, MARGIN = 1,
            FUN = smoof::makeRosenbrockFunction(dimensions = dim))
            return(Y)
        }

        ## Config 1
        recpars1 <- list(name = "recombination_mmax", lambda = 0.25)
        mutpars1 <- list(name = "mutation_best", f = 4)
        ## Config 2
        recpars2 <- list(name = "recombination_npoint", N = round(dim / 2))
        mutpars2 <- list(name = "mutation_rand", f = 2.2)

        selpars <- list(name = "selection_standard")
        stopcrit <- list(names = "stop_maxeval", maxevals = 5000*dim, maxiter = 100*dim)
        probpars <- list(name = "fn", xmin = rep(-5, dim), xmax = rep(10, dim))
        popsize <- 5*dim

        fx1 <- vector(,nRuns)
        fx2 <- vector(,nRuns)
        for (i in 1:nRuns)
        {
            # Run algorithm 1 on problem:
            out <- ExpDE(mutpars = mutpars1,
            recpars = recpars1,
            popsize = popsize,
            selpars = selpars,
            stopcrit = stopcrit,
            probpars = probpars,
            showpars = list(show.iters = "dots", showevery = 20))
            # Extract observation:
            fx1[i] <- out$Fbest

            # Run algorithm 2 on problem:
            out <- ExpDE(mutpars = mutpars2,
            recpars = recpars2,
            popsize = popsize,
```

```
            selpars = selpars,
            stopcrit = stopcrit,
            probpars = probpars,
            showpars = list(show.iters = "dots", showevery = 20))
            # Extract observation:
            fx2[i] <- out$Fbest
        }
        meanFx1[d] <- mean(fx1)
        meanFx2[d] <- mean(fx2)
        sdFx1[d] <- sd(fx1)
        sdFx2[d] <- sd(fx2)
    }

    data <- data.frame(dim = dims, mean.config1 = meanFx1, mean.config2 = meanFx2,
                       sd.config1 = sdFx1, sd.config2 = sdFx2)
    write.csv(data, file = arquivo)
}

data$n1 <- rep(nRuns, nDims)
data$n2 <- rep(nRuns, nDims)
```

Cálculo do número de amostras para cada instância

Após avaliar o comportamento dos dados do projeto piloto, utilizou-se o Algoritmo 1 proposto por Campelo e Takahashi (2018) para calcular o número mínimo de amostras necessárias em cada instância para cada algoritmo. É importante notar que cada instância possui uma variância, dessa forma, o número de observações por instância e para cada algoritmo poderá variar. O número máximo de obervações foi estipulado como 40, somando ambos tamanhos amostrais dos dois algoritmos, número que foi limitado devido ao tempo de execução do experimento.

```
arquivo="data.csv"
if (file.exists(arquivo)) {
    data <- read.csv(file=arquivo, header = TRUE, sep=",")
} else {
  nmax <- 40
  for (i in 1:nDims)
  {
    dim <- data$dim[i]

    fn <- function(X)
          {
              if(!is.matrix(X)) X <- matrix(X, nrow = 1) # <- if a single vector is passed as X
              Y <- apply(X, MARGIN = 1,
              FUN = smoof::makeRosenbrockFunction(dimensions = dim))
              return(Y)
          }

    ## Config 1
    recpars1 <- list(name = "recombination_mmax", lambda = 0.25)
    mutpars1 <- list(name = "mutation_best", f = 4)
    ## Config 2
    recpars2 <- list(name = "recombination_npoint", N = round(dim / 2))
    mutpars2 <- list(name = "mutation_rand", f = 2.2)
```

```
selpars <- list(name = "selection_standard")
stopcrit <- list(names = "stop_maxeval", maxevals = 5000*dim, maxiter = 100*dim)
probpars <- list(name = "fn", xmin = rep(-5, dim), xmax = rep(10, dim))
popsize <- 5*dim

mu1 <- data$mean.config1[i]
s1 <- data$sd.config1[i]
n1 <- data$n1[i]

mu2 <- data$mean.config2[i]
s2 <- data$sd.config2[i]
n2 <- data$n2[i]

se <- sqrt(s1^2/n1 + s2^2/n2)
seStar <- 0.05

while( se > seStar && n1+n2 < nmax)
{
  ropt <- s1/s2
  if (n1/n2 < ropt)
  {
    # Run algorithm 1 on problem:
    out <- ExpDE(mutpars = mutpars1,
    recpars = recpars1,
    popsize = popsize,
    selpars = selpars,
    stopcrit = stopcrit,
    probpars = probpars,
    showpars = list(show.iters = "dots", showevery = 20))
    # Extract observation:
    xnew <- out$Fbest

    munew <- (mu1*n1 + xnew)/(n1+1)
    sdnew <- sqrt(((n1-1)*s1^2 + (xnew - munew)*(xnew - mu1))/n1)

    mu1 <- munew
    s1 <- sdnew
    n1 <- n1 + 1
  }
  else
  {
    # Run algorithm 2 on problem:
    out <- ExpDE(mutpars = mutpars2,
    recpars = recpars2,
    popsize = popsize,
    selpars = selpars,
    stopcrit = stopcrit,
    probpars = probpars,
    showpars = list(show.iters = "dots", showevery = 20))
    # Extract observation:
    xnew <- out$Fbest

    munew <- (mu2*n2 + xnew)/(n2+1)
```

```
        sdnew <- sqrt(((n2-1)*s2^2 + (xnew - munew)*(xnew - mu2))/n2)

        mu2 <- munew
        s2 <- sdnew
        n2 <- n2 + 1
      }

      se <- sqrt(s1^2/n1 + s2^2/n2)

    }

    data$mean.config1[i] <- mu1
    data$sd.config1[i] <- s1
    data$n1[i] <- n1

    data$mean.config2[i] <- mu2
    data$sd.config2[i] <- s2
    data$n2[i] <- n2
  }
}

write.csv(data, file = arquivo)
```

```
data.frame(dimensao=data$dim, Alg1=data$n1, Alg2=data$n2)
```

```
##    dimensao Alg1 Alg2
## 1        10   10   30
## 2        12   10   30
## 3        13   10   30
## 4        15   10   30
## 5        25   17   23
## 6        26   17   23
## 7        28   18   22
## 8        32   23   17
## 9        36   25   15
## 10       38   20   20
## 11       41   22   18
## 12       44   25   15
## 13       46   26   14
## 14       47   23   17
## 15       57   26   14
## 16       59   23   17
## 17       63   20   20
## 18       72   27   13
## 19       77   26   14
## 20       83   19   21
## 21       84   20   20
## 22       85   20   20
## 23       96   20   20
## 24       98   21   19
## 25      106   25   15
## 26      111   17   23
## 27      112   20   20
## 28      114   25   15
```

```
## 29      118   21   19
## 30      120   24   16
## 31      131   20   20
## 32      132   17   23
## 33      135   22   18
## 34      140   21   19
```

Teste de Hipótese

Anova

Após realizar a blocagem, tendo definido número de instâncias e número de observações por instância, aplicou-se o ANOVA para investigar as hipóteses definidas a priori. Observou-se que existe uma diferença nos efeitos dos algoritmos, visto que a hipótese nula foi rejeitada.

```
d <- data.frame(dim = rep(data$dim,2), config = c(rep(1,nDims), rep(2,nDims)), Y = c(data$mean.config1,

for (i in 1:2){
      d[, i] <- as.factor(d[, i])
}

model <- aov(Y~dim+config, data = d)
summary(model)
```

```
##             Df    Sum Sq   Mean Sq F value   Pr(>F)
## dim         33 3.923e+14 1.189e+13   4.346 2.90e-05 ***
## config       1 1.297e+14 1.297e+14  47.422 7.29e-08 ***
## Residuals   33 9.028e+13 2.736e+12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.lm(model)$r.squared
```

```
## [1] 0.852563
```

```
model2 <- aov(log(Y)~dim+config, data = d)
summary(model2)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## dim         33  422.1  12.790   327.6 <2e-16 ***
## config       1   25.9  25.891   663.2 <2e-16 ***
## Residuals   33    1.3   0.039
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary.lm(model2)$r.squared
```

```
## [1] 0.9971322
```

Estimação da magnitude da diferença entre os métodos e intervalo de confiança

Para saber qual das configurações possui melhor desempenho, aplicou-se o t.test. A hipótese nula diz que não existe diferença entre a diferença das médias das configurações 1 e 2. E a hipótese alternativa diz que a média da configuração 2 é maior que a média da configuração 1. Como a hipótese nula foi rejeitada, concluiu-se que o algortimo da configuração 1 é melhor que o algoritmo da configuração 2, visto que a média do último é maior.

```r
muD <- (data$mean.config2 - data$mean.config1)/data$mean.config2
t_test <- t.test(muD, alternative = "greater", mu = 0, conf.level = 1-alpha)
t_test
```

```
##
##  One Sample t-test
##
## data:  muD
## t = 57.19, df = 33, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  0.67827     Inf
## sample estimates:
## mean of x
## 0.6989536
```

O valor da estatística de teste da configuração 1 é em média 70% do valor da estatística de teste da configuração 2, portanto o algoritmo 1 é em média 30% melhor que o algoritmo 2.

Validação das premissas

Para realizar o experimento, além de garantir as premissas da blocagem, uma replicação por bloco e randomização dentro de cada bloco, avaliou-se separadamente a normalidade dos resíduos e a independência dos blocos.

Normalidade dos resíduos

Foi verficado graficamente que os resíduos são normais, possuem médias iguais a zero e a variância entre blocos é aproximadamente igual, condições essas que foram melhor verificadas em instâncias de dimensões maiores. Logo, conclui-se que o modelo explica bem os dados do experimento.

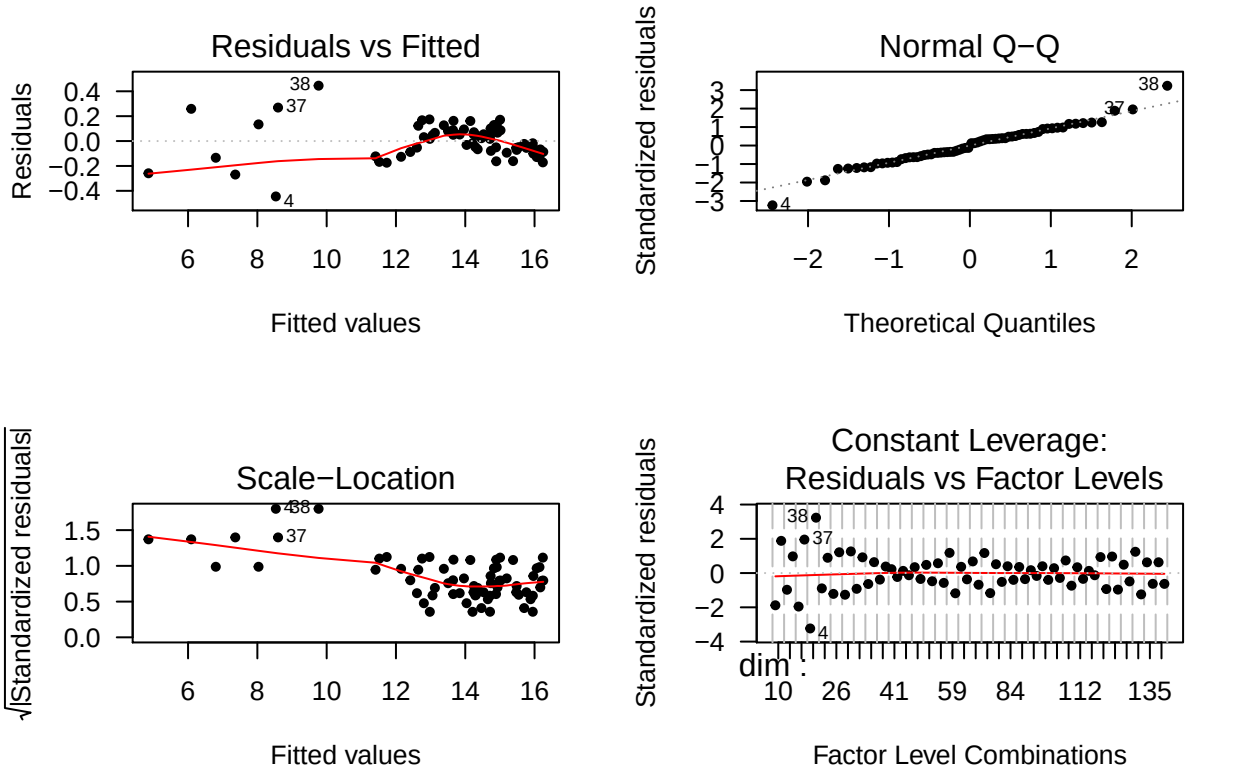
```r
shapiro.test(model$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.92423, p-value = 0.0004974
```

```r
shapiro.test(model2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  model2$residuals
## W = 0.9751, p-value = 0.1906
```

```r
par(mfrow = c(2, 2))
plot(model2, pch = 20, las = 1)
```




```
#par(mfrow = c(1, 1))
#qqPlot(model2$residuals, pch = 20, las = 1)
```

Indenpendência

No caso deste trabalho, a suposição de independência pode ser garantida pelo projeto do experimento. As amostras geradas para os dois algoritmos em qualquer instância são produzidas sem uma observação influenciar o valor de qualquer outra - foram utilizadas sequências aleatórias para diferentes execuções dos algoritmos. Garantir execuções algorítmicas independentes, como mencionado anteriormente, também ajuda a garantir essa suposição.

Conclusões

Por meio dos testes realizados podemos concluir que há diferença estatística significante entre os desempenhos das configurações 1 e 2. Além disso, demonstrou-se que a configuração 1 apresenta um desempenho médio superior a 30% da configuração 2. Portanto, se recomenda a configuração 1 em relação a configuração 2. Também concluímos que todas as premissas necessárias para realizar o teste de blocagem foram atendidas e logo o modelo proposto é condizente com o comportamento do fenômeno estudado.

Discussão sobre possíveis limitações do estudo e sugestões de melhoria

O presente estudo se limitou a analisar apenas o desempenho das configurações em relação ao valor da função objetiva. Outros parâmetros também poderiam ser analisados, como por exemplo o tempo de execução de cada configuração e o consumo de memória. Levando em conta o tempo e memoria consumidos pelas configurações diferentes poderíamos comparar o custo entre "precisão" e tempo de execução. Além disso, outros problemas de benchmark poderiam ser analisados e o desempenho das configurações comparadas para termos certeza que a configuração 1 é melhor que a configuração 2 não somente para o problema testado.