Estudo de caso: Grupo D 3

Gilmar Pereira, Maressa Tavares e Victor Ruela

29 de Outubro, 2019

1 Summary

O presente trabalho realizou o delineamento e executou os testes estatísticos para avaliar o desempenho médio do algoritmo conhecido como Evoluçao Diferencial [1].O algoritmo foi desenvolvido no ano de 1997 por Storn e Price e é um algoritmo simples de otimização multimodal, primeiramente desenvolvido para otimização de funções continuas e variaveis numéricas discretas [2].

Para o presente trabalho o algoritmo DE (Differential Evolution) é equipado com duas configurações alterando a forma de recombinação e mutação dos algoritmos. As classes de funções para este experimento foi composta por funções Rosenbrock [3] de dimensões entre 2 e 250. Para se analisar os dados utilizou-se da técnica de blocagem determinando a quantidade de blocos e seus tamanhos, assim como o número de amostras por instância.

Realizou-se a análise dos dados pela técnica ....

2 Planejamento do Experimento

Nesta seção é apresentado o planejamento do experimento, descrevendo os objetivos e o delineamento do experimento.

2.1 Objetivo do Experimento

O objetivo deste experimento é analisar se exite alguma deferença entre duas configuração do algoritmo DE dentre as classes de funções, determinando a configuração de melhor desempenho ressaltando as magnitudes das diferenças encontradas.

2.2 Delineamento

Para o seguinte experimento serão realizados as seguintes etapas:

- Formulação das hipoteses de teste;
- Cálculo dos tamanhos amostrais, determinando a quantidades de instâncias e número de iterações do algortimo;
- Coleta e tabulação dos dados,
- Realização dos teste de hipoteses;
- Estimação da magnitude das diferenças;
- Validação das premissas;
- Resultados e Conclusões.

2.3 Hipóteses

Para a análise comparativa entre as configurações do algortimo DE, determinou-se a seguintes hipoteses a serem testadas.

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_1 : \mu_1 \neq \mu_2 \end{cases}$$

Onde μ_1 e μ_2 são as médias amostras das configurações 1 e 2, respectivamente.

Além disso, foram definidos os seguintes parâmetros experimentais:

- Significância desejada: $\alpha = 0.05$.
- Mínima diferença de importância prática (padronizada): $d^* = \delta^*/\sigma = 0.5$
- Potência mínima desejada $\pi = 1 - \beta = 0.8$

2.4 Coleta dos Dados

Neste trabalho, cada amostra consiste em uma execução do algoritmo DE, para cada instância (dimensão da função objetivo) e configuração do algoritmo em questão (níveis). Foram escolhidas $N = 30$ repetições de cada par instância-configuração, conforme recomendado como suficiente em [4]. A coleta de dados foi divida em duas etapas, descritas nas seções a seguir. O código utilizado para a coleta de dados está disponível no apêndice deste trabalho.

2.4.1 Geração de arquivo de configuração do experimento

Esta etapa consiste em permitir que seja gerada um arquivo .csv contendo a configuração descrita a seguir. As rotinas foram implementadas de forma a permitir que seja criada a configuração para qualquer número de repetições N, instâncias I, grupos b e níveis a. Um último passo consiste em randomizar o arquivo de configuração e dividí-lo em 3 arquivos separados, a serem executados por cada membro do grupo. Isso garante que as amostras geradas são independentes e que o experimento seja completamente randomizado. Como o algoritmo demora um tempo considerável para sua execução, a divisão entre os participantes permitiu a sua execução em paralelo para otimizar o tempo necessário para gerar todos os dados. A tabela abaixo exibe um exemplo de arquivo de configuração gerado.

Table 1: Exemplo de arquivo de configuração

algorithm	replicate	instance	group	result
2	27	4	1	-1
2	6	89	21	-1
2	20	73	17	-1
2	29	2	1	-1
1	30	94	22	-1

2.4.2 Execução de arquivo de configuração

Com o arquivo de configuração disponível, o experimento está pronto para ser executado. A rotina desenvolvida carrega um arquivo informado e executa cada linha em sequência, para os seus respectivos parâmetros. À medida em que uma amostra é finalizada, o resultado é salvo no próprio arquivo de configuração, na coluna `result`. Isso garante com que seja possível continuar a execução do arquivo sem perder as amostras realizadas anteriormente, caso ocorra algum problema.

2.5 Análise Exploratória dos Dados

Como o estudo consiste na comparação entre resultados da execução de um algoritmo de otimização, a dimensão da função objetivo é um fator importante. Logo, a análise exploratória será feita considerando exemplos de instâncias de baixa, média e alta dimensão. Inicialmente, os dados do experimento são carregados, sendo as instâncias 4, 50 e 100 escolhidas para avaliação, e um gráfico boxplot é criad para uma análise preliminar.

```
sample.all <- read.csv('data.all.instances.csv', header = TRUE)
sample.all$configuration <- as.factor(sample.all$algorithm)
sample.all <- sample.all %>% mutate(logresult = log(result))

sample.all.eda <- sample.all %>% filter(instance == 100 | instance == 50 | instance == 4)
```

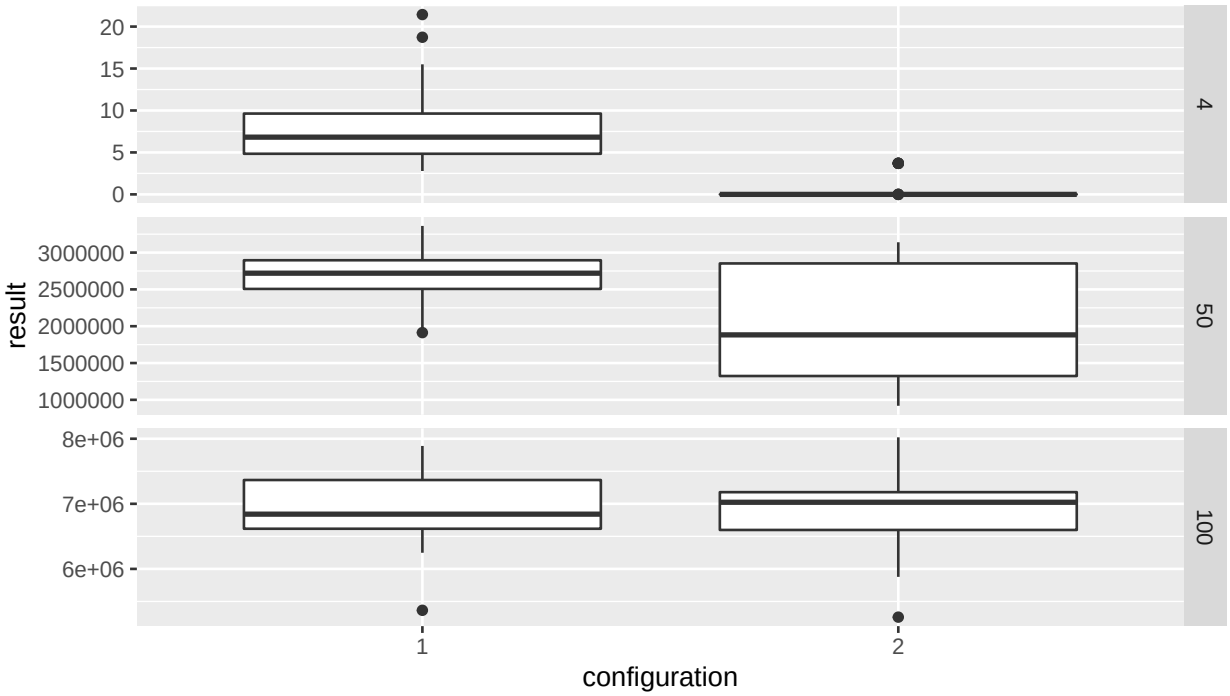

Figure 1: Boxplot dos dados

Através deste gráfico, as seguintes observações podem ser feitas:

- Os valores da função objetivo final possuem magnitudes muito diferentes dependendo da dimensão. Portanto, uma normalização dos dados para uma escala comum pode ser necessária para correta análise dos experimentos.
- Há algumas repetições do algoritmo que poderiam ser considerados outliers. Elas devem ser removidas de forma a não prejudicar os testes de hipótese e validação das premissas.
- A configuração 2 parece obter melhores resultados para dimensões baixas, quase sempre chegando o mínimo da função. Entretnato, o mesmo não pode ser afirmado para dimensões maiores.

2.6 Cálculo do número de blocos

De acordo com [5], o número de blocos ideal é calculado variando a quantidade de blocos enquanto a relação

$$F(1-\alpha) \leq F(\beta, \phi)$$

é respeitada. Onde ϕ é o parâmetro de não-centralidade, definido por:

$$\phi = \frac{b\sum_{i=1}^{a}\tau_i}{a\sigma^2}$$

De acordo com a definição do experimento, temos que $a = 2$, tamanho de efeito normalizado $d = 0.5$, potência desejada de $\pi = 0.8$ e significância $\alpha = 0.05$. Logo, é possível calcular o número de blocos b de acordo com a rotina abaixo.

```
a <- 2
d <- 0.5
alpha <- 0.05
beta <- 0.2
```

```
tau <- c(-d, d, rep(0, a - 2)) # define tau vector
b <- 5

tb <- data.frame(b = rep(-1, 50), ratio = rep(-1,50), phi = rep(-1,50))

for(i in seq(1,40,by=2)){

  b <- i + 5
  f1 <- qf(1 - alpha, a - 1, (a - 1)*(b - 1))
  f2 <- qf(beta, a - 1, (a - 1)*(b - 1), (b*sum(tau^2)/a))
  phi <- b*sum(tau^2)/a

  tb[i, ] = c(b, f1/f2, phi)
}
```

Portanto, o número mínimo de blocos necessários b é de 34. As iterações podem ser vistas na tabela abaixo.

Table 2: Iterações para cálculo do número de blocos

Blocos	Razão	Phi
6	22.3119377	1.5
10	8.3089036	2.5
14	4.3118995	3.5
18	2.7150544	4.5
22	1.9226732	5.5
26	1.4656488	6.5
30	1.1734386	7.5
34	0.9725686	8.5
38	0.8269777	9.5
42	0.7171273	10.5

Portanto, devemos escolher b blocos aleatoriamente das instâncias disponíveis. A figura 2 exibe o gráfico de ridge para as instâncias selecionadas.

```
## Picking joint bandwidth of 0.078
## Picking joint bandwidth of 0.078
```

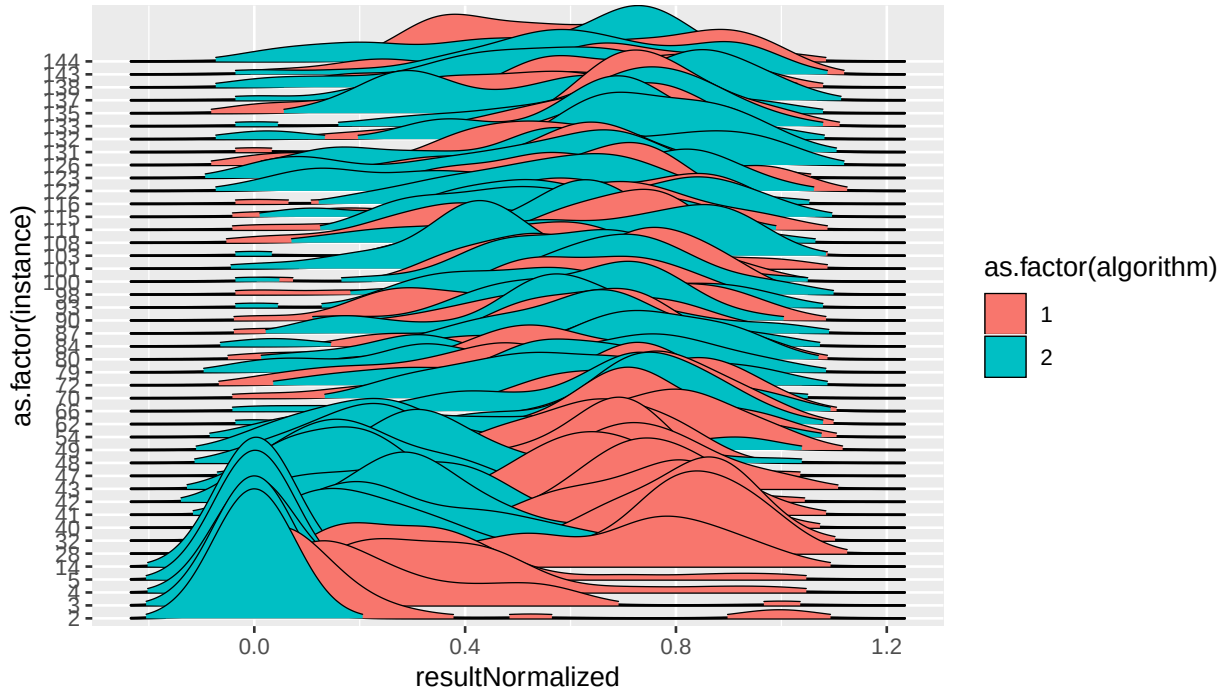

Figure 2: Ridge plot dos dados

2.7 Validação das premissas

Para realizar as inferências estatísticas sobre as duas configurações do algoritmo de otimização é necessário validar as premissas antes de executar o teste. Neste caso, como trata-se de duas configurações em um espectro amplo de dimensões, existe um fator conhecido e controlável que pode influenciar no resultado do teste. Então, para eliminar o efeito desse fator indesejável uma opção é realizar a blocagem [6]. A seguir são apresentados os testes realizados para validar as premissas exigidas pelo anova.

A - Normalidade dos Resíduos

Para se ter uma idéia inicial da normalidade dos dados, o histograma para as instâncias em avaliação é gerado a seguir.

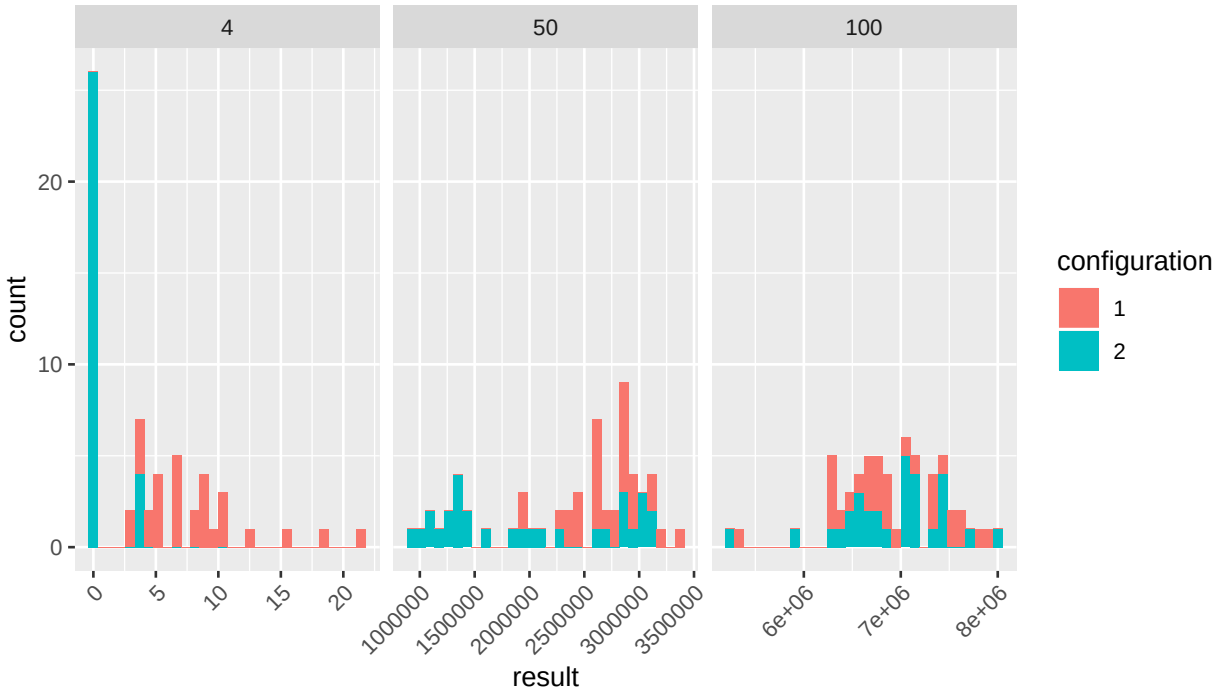

Figure 3: Histograma dos dados

Pelo histograma apresentado, é possível notar que os dados não apresentam uma distribuição visivelmente normal. Os gráficos quantil-quantil e os testes de Shapiro-Wilk são apresentados para se comprovar essa observação.

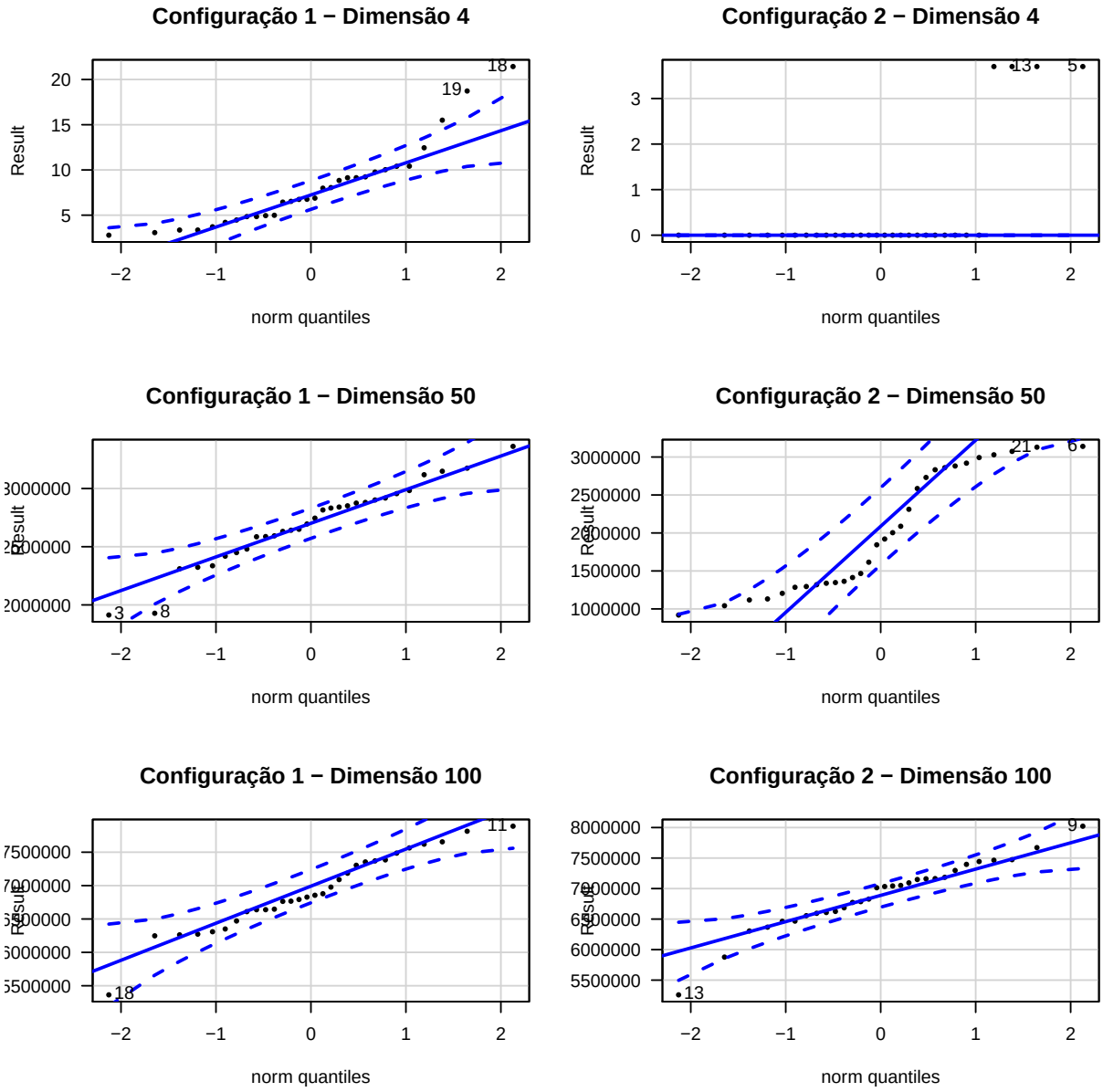

Figure 4: Gráfico quantil-quantil das instâncias avaliadas

Table 3: Resultados dos testes de shapiro-wilk

Instância	Configuração	p-valor
4	1	0.0016068
4	2	0.0000000
50	1	0.5191989
50	2	0.0019100
100	1	0.3923896
100	2	0.3056838

Table 4: p-valores dos testes de shapiro-wilk

Instância	Configuração 1	Configuração 2
2	0.0000000	0.0000000
3	0.0000000	0.0005915
4	0.0000000	0.0016068
5	0.0000000	0.0003446
14	0.0000063	0.1966538
28	0.0675749	0.1188241
32	0.3125588	0.0530113
40	0.2051146	0.2855580
41	0.0361824	0.8783769
42	0.1209133	0.8197129
43	0.2262048	0.5241492
47	0.0839617	0.0385525
48	0.0028819	0.2134882
49	0.0049658	0.5737687
54	0.1526723	0.0483240
62	0.0002044	0.5037147
66	0.0265610	0.2110943
70	0.9623741	0.0620700
72	0.6793350	0.9362853
79	0.1830647	0.5587759
80	0.0168388	0.1359909
84	0.0918314	0.4150641
87	0.1732496	0.8121814
90	0.3237680	0.1009137
93	0.8673279	0.8229739
98	0.7859124	0.0002289
100	0.3056838	0.3923896
101	0.0760142	0.8189769
103	0.0080130	0.6654325
108	0.4074819	0.7786390
111	0.1647580	0.3863659
115	0.1981531	0.0518414
116	0.9776951	0.0076466
122	0.0304172	0.3349377
123	0.0227588	0.4736620
126	0.0810109	0.4095614
131	0.5461694	0.0045566
132	0.0181213	0.0913901
133	0.0096410	0.0048931
135	0.1716773	0.8990398
137	0.0189678	0.1159964
138	0.5054000	0.3755507
143	0.2430332	0.0213012
144	0.0055289	0.0765569

Conforme visto anteriormente, não foi possível atestar a normalidade dos dados. Logo, uma transformação logarítmica é aplicada aos dados e os testes de Shapiro-Wilk são executados novamente.

Table 5: Resultados dos testes de shapiro-wilk com aplicação de transformaçao logarítimica

Instância	Configuração	p-valor
4	1	0.6439926
4	2	0.0012907
50	1	0.0870792
50	2	0.0065506
100	1	0.1687062
100	2	0.0582035

Pelas tabelas 2 e 3, é possível notar que após a aplicação da transformação logarítmica, há resultados que passa e deixam de ser normais. Portanto, não podemos afirmar que para todas as dimensões e configurações testadas os dados seguirão uma distribuição normal.

B - Igualdade de Variância

Table 6: Resultados dos testes de variância

Instância	p-valor
4	0.0000000
50	0.0000212
100	0.8607916

Table 7: p-valores dos testes de shapiro-wilk

Instância	p-valor
2	0.0000000
3	0.0000000
4	0.0000000
5	0.0000000
14	0.0000000
28	0.0263514
32	0.0037336
40	0.7391286
41	0.0038719
42	0.1434864
43	0.2625872
47	0.0975879
48	0.0003533
49	0.0032971
54	0.0046567
62	0.0310612
66	0.1617131
70	0.3310122
72	0.8082301
79	0.1540014
80	0.9174343
84	0.1903662
87	0.9970596
90	0.0107843

Instância	p-valor
93	0.0236607
98	0.3428729
100	0.8607916
101	0.4994527
103	0.0214749
108	0.4502423
111	0.3341492
115	0.9801664
116	0.8517553
122	0.4421971
123	0.4628551
126	0.6185635
131	0.0719711
132	0.2281559
133	0.0343971
135	0.6186219
137	0.1266789
138	0.7931851
143	0.5043999
144	0.1865931

Pela tabela 4, é possível notar que para os

1 - Replicação por bloco:

2 - Independência dos blocos:

3 - Randomização dos blocos:

2.8 Apêndice

2.8.1 Geração de configuração

```
# Load packages ------------------------------------------------------------
if (!require(ExpDE, quietly = TRUE)){
  install.packages("ExpDE")
}


if (!require(smoof, quietly = TRUE)){
  install.packages("smoof")
}


if (!require(CAISEr, quietly = TRUE)){
  install.packages("CAISEr")
}


# RCBD functions -----------------------------------------------------
set.seed(15632) # set a random seed
instances <- seq(2, 150) # number of instances
N <- 30 # number of replicates per instance
```

```
rcbd.configuration.generator <- function(level, b, instances, N){
  nrows <- length(instances) * N
  n.instances <- length(instances)
  instance <- sort(rep(instances, N))
  groups <- sapply(instance, function(i){ ceiling(i/(n.instances/b))  })

  X <- data.frame("algorithm" = rep(level, nrows),
                  "replicate" = rep(seq(1,N), n.instances),
                  "instance" = instance,
                  "group" =  groups,
                  "result" = rep(-1, nrows))
  return(X)
}


x.config.1 <- rcbd.configuration.generator(1, b, instances, N)
x.config.2 <- rcbd.configuration.generator(2, b, instances, N)
x.config.all <- rbind(x.config.1, x.config.2)
x.config.all.shuffled <- x.config.all[sample(nrow(x.config.all)), ]

split.size <- (nrow(x.config.all.shuffled)/3)
x.config.all.shuffled$member <- ceiling((1:nrow(x.config.all.shuffled))/split.size)

x.config.all.shuffled.victor <- x.config.all.shuffled[x.config.all.shuffled$member == 1,1:5]
x.config.all.shuffled.gilmar <- x.config.all.shuffled[x.config.all.shuffled$member == 2,1:5]
x.config.all.shuffled.maressa <- x.config.all.shuffled[x.config.all.shuffled$member == 3,1:5]

write.csv(x.config.all.shuffled.victor, 'rcbd.config.victor.csv', row.names=FALSE)
write.csv(x.config.all.shuffled.gilmar, 'rcbd.config.gilmar.csv', row.names=FALSE)
write.csv(x.config.all.shuffled.maressa, 'rcbd.config.maressa.csv', row.names=FALSE)
```

2.8.2 Execução de configuração

```
# Load packages ------------------------------------------------------------
if (!require(ExpDE, quietly = TRUE)){
  install.packages("ExpDE")
}


if (!require(smoof, quietly = TRUE)){
  install.packages("smoof")
}

# Execute a RCBD test configuration ----------------------------------------

# Define a class to store the levels configuration
level.config <- function(mp, rp, id) {
  value <- list(mutparsX = mp, recparsX = rp, id = id)
  class(value) <- append(class(value),"level.config")
  return(value)
}

## Equipe D
```

```r
## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars1 <- list(name = "mutation_rand", f = 4)

## Config 2
recpars2 <- list(name = "recombination_eigen", othername = "recombination_bin", cr = 0.9)
mutpars2 <- list(name = "mutation_best", f = 2.8)

config.1 <- level.config(mutpars1, recpars1, 1)
config.2 <- level.config(mutpars2, recpars2, 2)


fname = 'rcbd.config.victor.csv'
Z <- read.csv(fname)
set.seed(15632) # set a random seed

my.ExpDE <- function(mutp, recp, dim){

  fn.current <- function(X){
    if(!is.matrix(X)) X <- matrix(X, nrow = 1) # <- if a single vector is passed as Z

    Y <- apply(X, MARGIN = 1, FUN = smoof::makeRosenbrockFunction(dimensions = dim))
    return(Y)
  }


  assign("fn", fn.current, envir = .GlobalEnv)

  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

  out <- ExpDE(mutpars = mutp,
               recpars = recp,
               popsize = popsize,
               selpars = selpars,
               stopcrit = stopcrit,
               probpars = probpars,
               showpars = list(show.iters = "none"))

  return(list(value = out$Fbest))
}

for (row in 1:nrow(Z)){

  if(Z[row, "result"] == -1){ # start from the last execution
    dim <- Z[row, "instance"]
    algo <- Z[row, "algorithm"]
    replicate <- Z[row, "replicate"]

    if(algo == 1)
      algo.config <- config.1
```

```
    else
      algo.config <- config.2

    print(paste("Started Instance:", dim, "; Algo:", algo, "; Repetition:", replicate))

    out <- my.ExpDE(algo.config$mutparsX, algo.config$recparsX, dim)

    Z[row, "result"] <- out$value
    print(paste("Finished. Instance:", dim, "; Algo:", algo, ";
                Repetition:", replicate, "; Result=", out$value))
    print(paste("Progress = ", 100 * row / nrow(Z) , "%"))
    write.csv(Z, fname)

  }
}
```

Referências

[1] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[2] K. V. Price, "Differential evolution," in *Handbook of optimization*, Springer, 2013, pp. 187–214.

[3] H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[4] F. Campelo and F. C. Takahashi, "Sample size estimation for power and accuracy in the experimental comparison of algorithms," *CoRR*, vol. abs/1808.02997, 2018.

[5] F. Campelo, "Lecture notes on design and analysis of experiments." https://github.com/fcampelo/Design-and-Analysis-of-Experiments, 2018.

[6] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers, (with cd)*. John Wiley & Sons, 2007.