

Estudo de caso: Grupo D 3

Gilmar Pereira, Maressa Tavares e Victor Ruela

30 de Setembro, 2019

1 Summary

O presente trabalho realizou o delineamento e executou os testes estatísticos para avaliar o desempenho médio do algoritmo conhecido como Evolução 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 contínuas e variáveis numéricas discretas [2].

Para este 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 cálculo do número de blocos de acordo com [4] e o número de instâncias de acordo com [5]. Para teste das premissas de normalidade utilizou-se a ferramenta qqplot e o teste de shapiro-wilk. Para o teste da homogeneidade de variância utilizou-se o teste F. Para análise da hipótese nula utilizou-se o teste não paramétrico de Friedman.

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 existe alguma diferença entre duas configurações 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 hipóteses de teste;
- Cálculo dos tamanhos amostrais, determinando a quantidades de instâncias e número de iterações do algoritmo;
- Coleta e tabulação dos dados,
- Realização dos teste de hipóteses;
- Estimativa da magnitude das diferenças;
- Validação das premissas;
- Resultados e Conclusões.

2.3 Hipóteses

Para a análise comparativa entre as configurações do algoritmo DE, determinou-se as seguintes hipóteses 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 [5]. A coleta de dados foi dividida 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 dividi-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

	X	algorithm	replicate	instance	group	result
5	5	1	30	94	28	-1
6	6	1	29	140	42	-1
7	7	2	8	113	34	-1
8	8	2	16	7	3	-1
9	9	1	5	118	35	-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.

3 Análise Exploratória dos Dados

Nesta seção é apresentado uma análise exploratória dos dados, verificando normalidade, homocedasticidade, independência e realizando a validação das premissas.

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 criado 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)

```

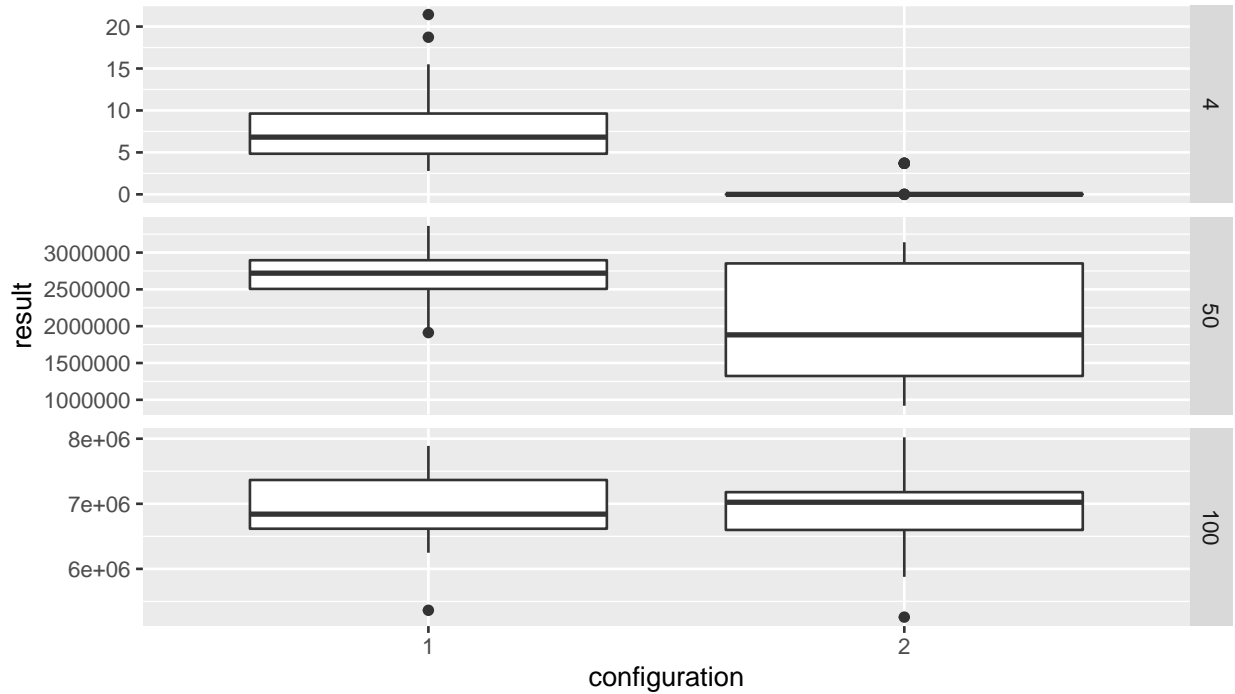


Figure 1: Boxplot dos dados

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. Entretanto, o mesmo não pode ser afirmado para dimensões maiores.

3.1 Cálculo do número de blocos

De acordo com [4], 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 3 exibe o gráfico de ridge para as instâncias selecionadas.

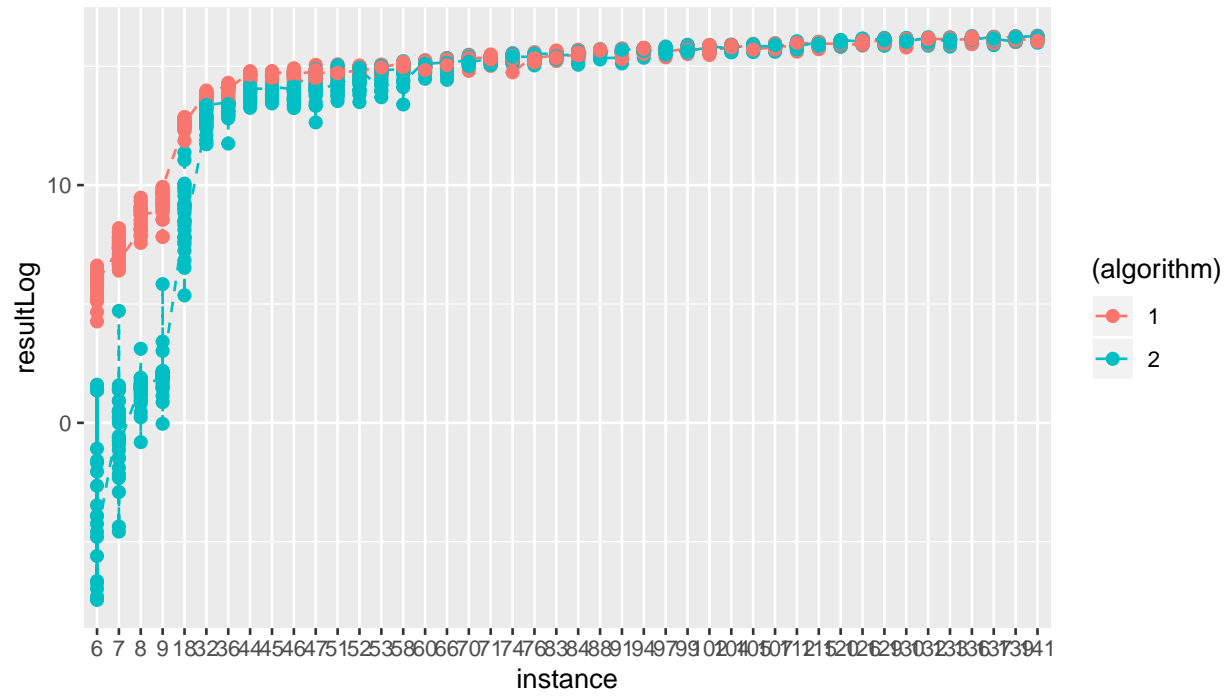


Figure 2: Ridge plot dos dados

```
## Picking joint bandwidth of 0.0939
## Picking joint bandwidth of 0.0939
```

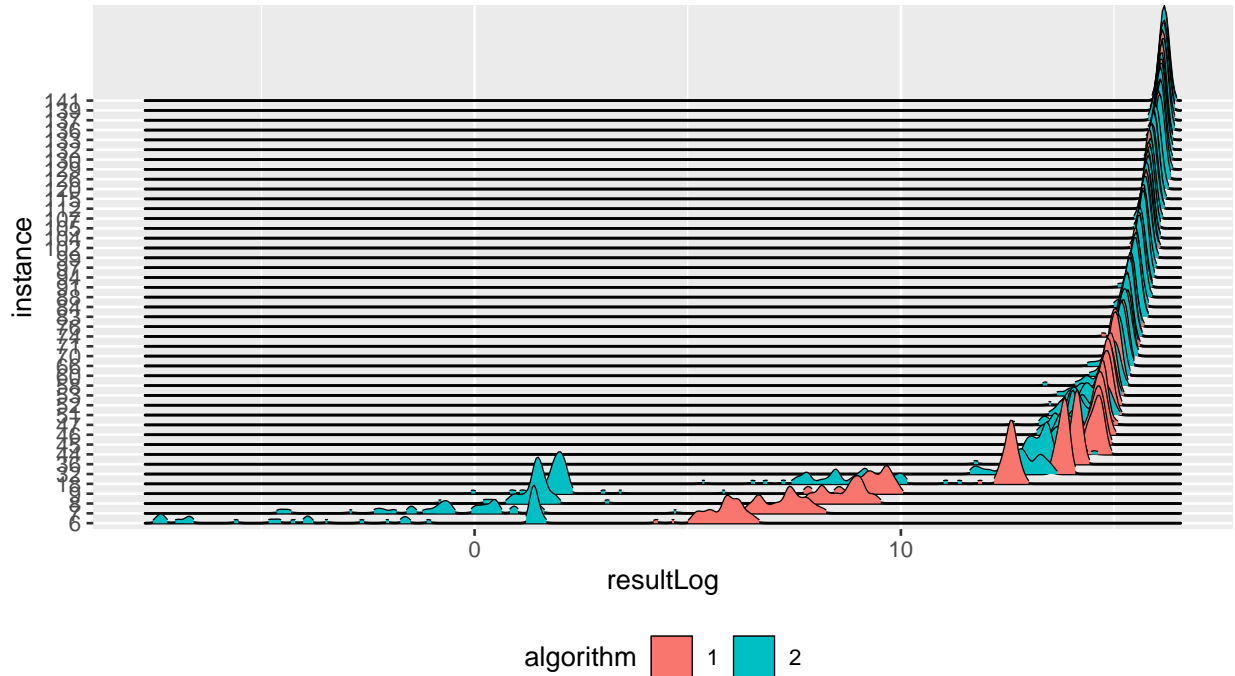


Figure 3: Ridge plot dos dados

3.2 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 teste.

A - Normalidade dos Resíduos

Para a validação desta premissa, aplica-se o teste ANOVA aos dados e depois avalia-se os resíduos obtidos pelo modelo. O gráfico quantil-quantil e teste de shapiro-wilk serão utilizados nessa validação.

```
##          Df Sum Sq Mean Sq F value Pr(>F)
## algorithm    1    539   539.1   369.3 <2e-16 ***
## instance    43 30891   718.4   492.2 <2e-16 ***
## Residuals 2595   3787     1.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

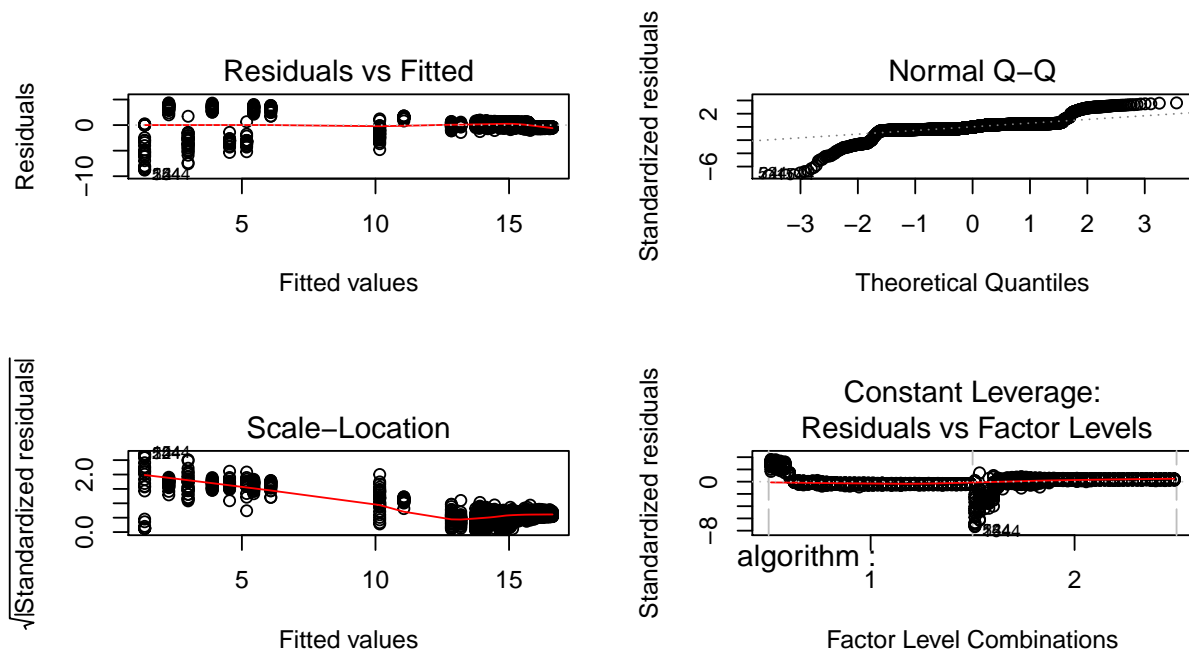


Figure 4: QQ-plot dos resíduos

```
##
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.73488, p-value < 2.2e-16

Conforme pode ser visto, a premissa de normalidade dos resíduos é violada.

##           Df Sum Sq Mean Sq F value Pr(>F)
## algorithm    1    539    539.1    369.3 <2e-16 ***
## instance    43   30891    718.4    492.2 <2e-16 ***
## Residuals   2595    3787     1.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

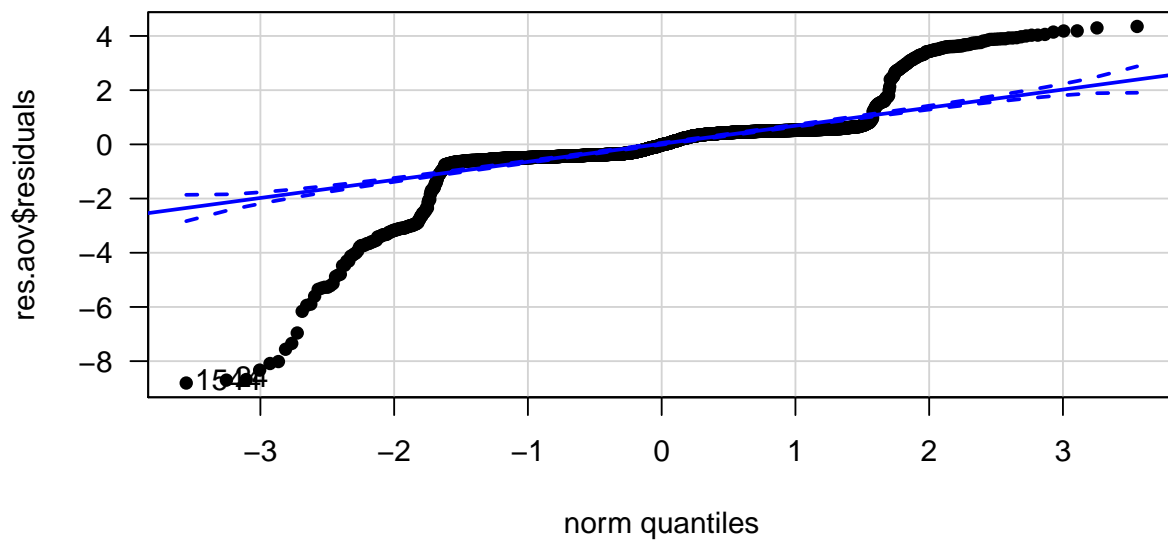


Figure 5: QQ-plot dos resíduos

```
##
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.73488, p-value < 2.2e-16
```

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

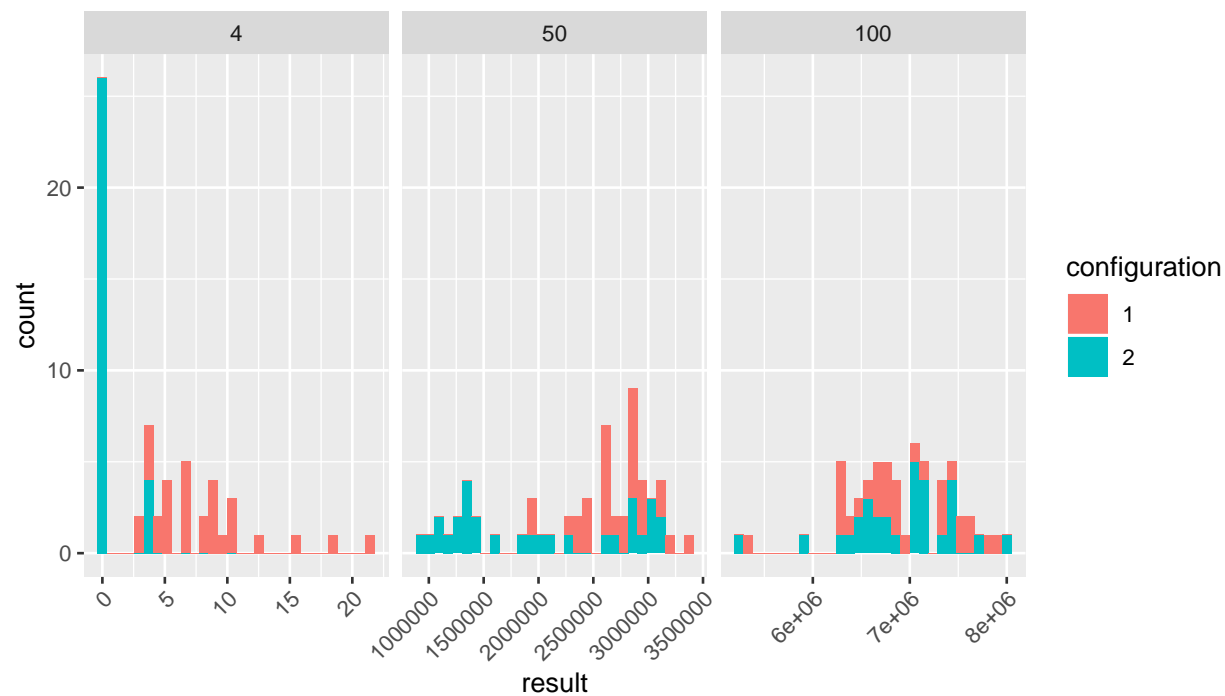


Figure 6: Histograma dos dados

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.

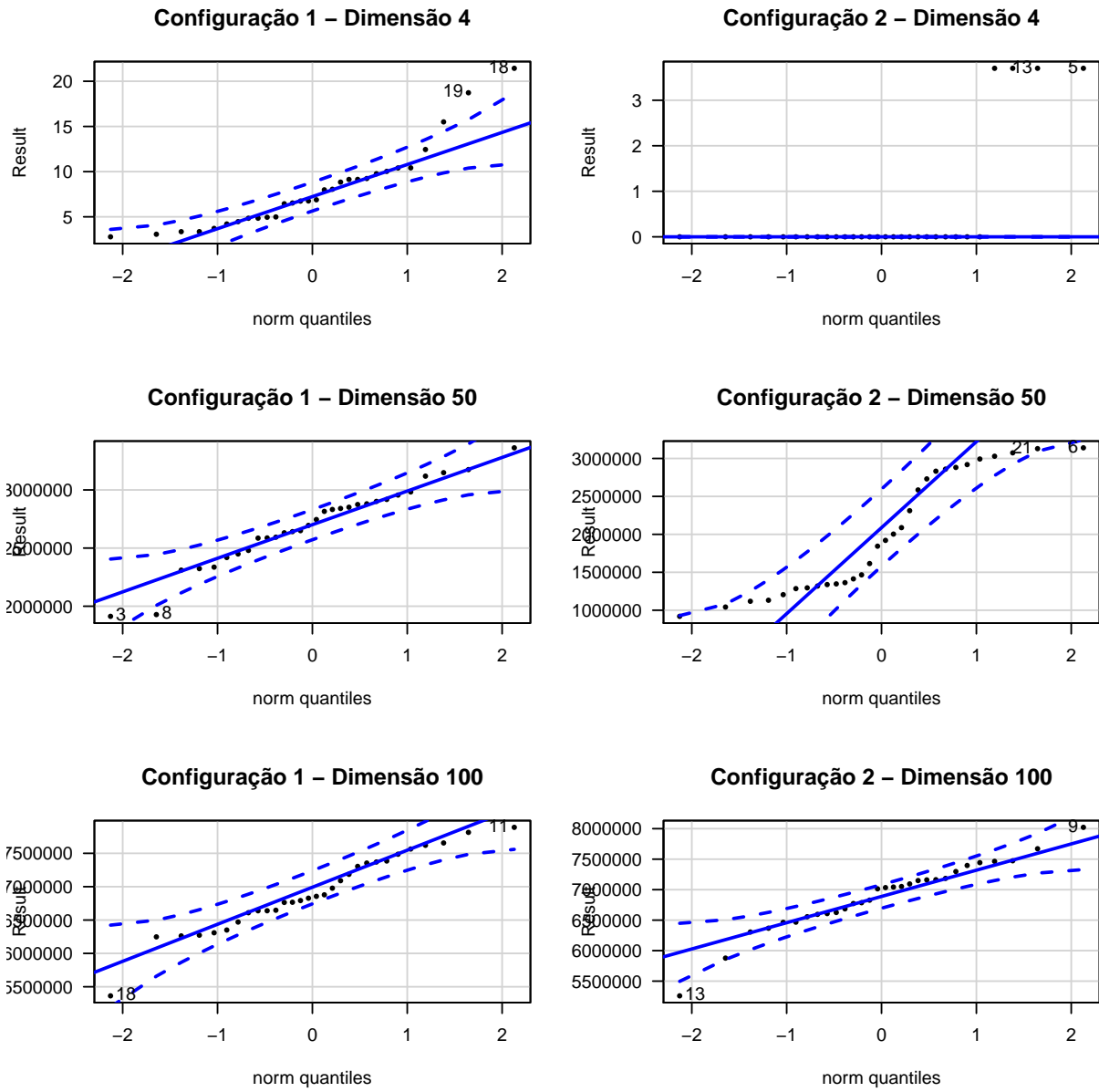


Figure 7: Gráfico quantil-quantil das instâncias avaliadas

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
6	0.0000009	0.9309204
7	0.0000000	0.0952691
8	0.0000000	0.2117700
9	0.0000000	0.7507335
18	0.0000001	0.2972384
32	0.3125588	0.0530113
36	0.5390465	0.0031382
44	0.0823744	0.2327750
45	0.0036558	0.3650279
46	0.0044446	0.9161466
47	0.0839617	0.0385525
51	0.0753235	0.1364449
52	0.0758525	0.0354189
53	0.0091924	0.3830805
58	0.0697282	0.5604240
60	0.0924252	0.2159456
66	0.0265610	0.2110943
70	0.9623741	0.0620700
71	0.0599036	0.7153322
74	0.3792093	0.0002792
76	0.1354236	0.3168043
83	0.7307143	0.2537269
84	0.0918314	0.4150641
88	0.1073720	0.3586552
91	0.0045456	0.4032497
94	0.0153487	0.0844764
97	0.7951191	0.1495584
99	0.7449118	0.3043869
102	0.2267074	0.0152054
104	0.0496420	0.4028618
105	0.0278883	0.3291645
107	0.0139989	0.4737125
112	0.9846658	0.0167655
115	0.1981531	0.0518414
120	0.3541607	0.3607403
126	0.0810109	0.4095614
129	0.0014765	0.6469189
130	0.7114775	0.3090016
132	0.0181213	0.0913901
133	0.0096410	0.0048931
136	0.0155369	0.2518921
137	0.0189678	0.1159964
139	0.3062163	0.3744098
141	0.2011834	0.4393923

Pelas tabelas 3 e 4, é possível notar que após a aplicação da transformação logarítmica, há resultados que deixam de ser normais. Portanto, não podemos afirmar que para todas as dimensões e configurações testadas os dados seguem uma distribuição normal.

B - Igualdade de Variância

Para validação dessa premissa utilizou-se o teste homogeneidade de variâncias no qual a hipótese nula é razão entre as variâncias igual 1. Primeiramente analisou-se as instâncias de dimensão 4, 50 e 100, verificando as variâncias das duas configurações. Os resultados do teste podem ser vistos na tabela 5.

Table 5: Resultados dos testes de variância

Instância	p-valor
4	0.0000000
50	0.0000212
100	0.8607916

Verifica-se que a instância de maior dimensão possui um p-valor alto, em relação ao nível de significância pré-determinado para a análise do experimento, desta maneira não se pode rejeitar a hipótese nula. Após analisou-se as duas configurações com as instâncias geradas forma aleatória. O resultado pode ser visto na tabela 6.

Table 6: p-valores do teste F

Instância	p-valor
6	0.0000000
7	0.0000000
8	0.0000000
9	0.0000000
18	0.0000011
32	0.0037336
36	0.4383798
44	0.6960527
45	0.1921058
46	0.0229743
47	0.0975879
51	0.0008317
52	0.0026071
53	0.0010530
58	0.0000169
60	0.0517480
66	0.1617131
70	0.3310122
71	0.7763353
74	0.2420485
76	0.6605738
83	0.2707051
84	0.1903662
88	0.3409592
91	0.6956770
94	0.4077931
97	0.9429186
99	0.5718182
102	0.0403736
104	0.3576964
105	0.3539264
107	0.1641992
112	0.7748390
115	0.9801664

Instância	p-valor
120	0.9651509
126	0.6185635
129	0.6617235
130	0.1419264
132	0.2281559
133	0.0343971
136	0.6794527
137	0.1266789
139	0.8289963
141	0.8248179

Verifica-se que para diferentes valores de instâncias, dimensões, o p-valor é maior que o nível de significancia pré-estabelecido. Desta forma não se pode rejeitar a hipótese nula de que as variâncias são iguais. Analisando de forma geral verifica-se que é possível que as variâncias entre as duas configurações são iguais.

4 Resultados

Nesta seção são apresentados os resultados realizando os testes de hipóteses e a determinação da melhor configuração juntamente com a estimação das magnitudes das diferenças.

4.1 Teste de Hipótese

Verificando que os dados gerados pelas duas configurações não são normais e que não há uma homogeneidade entre as variâncias, optou-se pela utilização de um teste não paramétrico. O teste utilizado para verificar a igualdade das médias entre as duas configurações foi o teste de Friedman. O teste de Friedman é um teste não paramétrico que generaliza o teste de sinais e possui um poder estatístico modesto para muitas distribuições não normais [7]. O teste Friedman está descrito abaixo.

```
## Levene's Test for Homogeneity of Variance (center = median)
##           Df F value  Pr(>F)
## group      1  8.1584 0.00432 **
##           2638
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Levene's Test for Homogeneity of Variance (center = median)
##           Df F value    Pr(>F)
## group     43 12.525 < 2.2e-16 ***
##           2596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

##
## Friedman rank sum test
##
## data:  result and algorithm and instance
## Friedman chi-squared = 11, df = 1, p-value = 0.0009111
```

Verifica-se pelo teste que o baixo p-valor garante a rejeição da hipótese nula de que as configurações possuem médias iguais.

4.2 Estimação das magnitudes das diferenças

```
TukeyHSD(res.aov, "algorithm")

## Tukey multiple comparisons of means
## 95% family-wise confidence level
##
## Fit: aov(formula = resultLog ~ algorithm + instance, data = data.block.instances)
##
## $algorithm
##          diff          lwr          upr p adj
## 2-1 -0.9037529 -0.9959638 -0.811542    0
```

5 Discussão e Conclusões

Neste trabalho foi feito um estudo estatístico ...

6 Divisão das Atividades

- Victor - coordenador
- Maressa - verificadora e monitora
- Gilmar - Relator

7 Apêndice

7.1 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, nrow),
               "replicate" = rep(seq(1,N), n.instances),
               "instance" = instance,
               "group" = groups,
               "result" = rep(-1, nrow))

return(X)
}

x.config.1 <- rcdb.configuration.generator(1, b, instances, N)
x.config.2 <- rcdb.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, 'rcdb.config.victor.csv', row.names=FALSE)
write.csv(x.config.all.shuffled.gilmar, 'rcdb.config.gilmar.csv', row.names=FALSE)
write.csv(x.config.all.shuffled.maressa, 'rcdb.config.maressa.csv', row.names=FALSE)

```

7.2 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
## 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.vector.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, “Lecture notes on design and analysis of experiments.” <https://github.com/fcampelo/Design-and-Analysis-of-Experiments>, 2018.
- [5] 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.
- [6] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers, (with cd)*. John Wiley & Sons, 2007.
- [7] D. W. Zimmerman and B. D. Zumbo, “Relative power of the wilcoxon test, the friedman test, and repeated-measures anova on ranks,” *The Journal of Experimental Education*, vol. 62, no. 1, pp. 75–86, 1993.