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 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 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 homogeniedade de variancia utilizou-se o teste F. Para análise da hipotese 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 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 [5]. 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

	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, independencia 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)
```

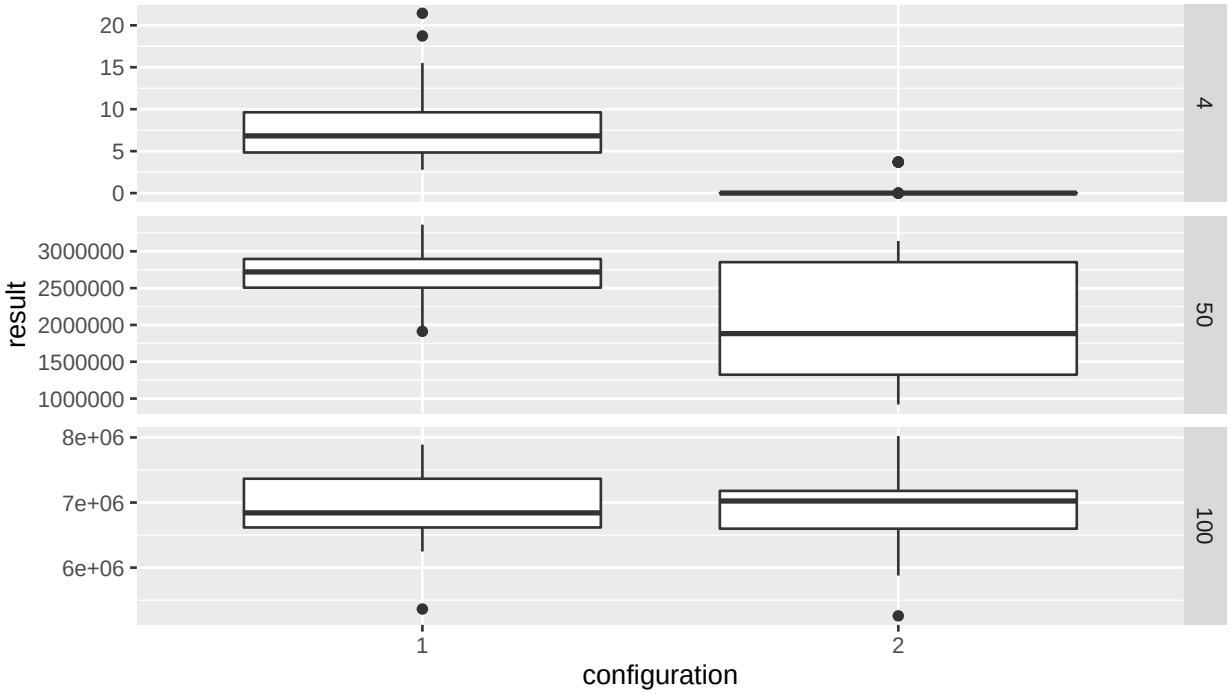


Figure 1: Boxplot dos dados

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)
}

b <- min((tb %>% filter(ratio <= 1 & ratio > 0))$b)
```

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.

```
data.all.instances.log <- sample.all %>% group_by(instance) %>%
  mutate(Max = max(result), Min = min(result), Std = sd(result), Avg = mean(result)) %>%
  mutate(resultLog = log(result))

set.seed(1234)
random.blocks <- sample(5:150, b)
data.block.instances <- data.all.instances.log %>%  filter(instance %in% random.blocks)

data.block.instances$instance = as.factor(data.block.instances$instance)
data.block.instances$algorithm = as.factor(data.block.instances$algorithm)
```

A figura 2 exibe os valores obtidos para as instâncias selecionadas. Na figura 3 é possível ver o respectivo gráfico de Ridge.

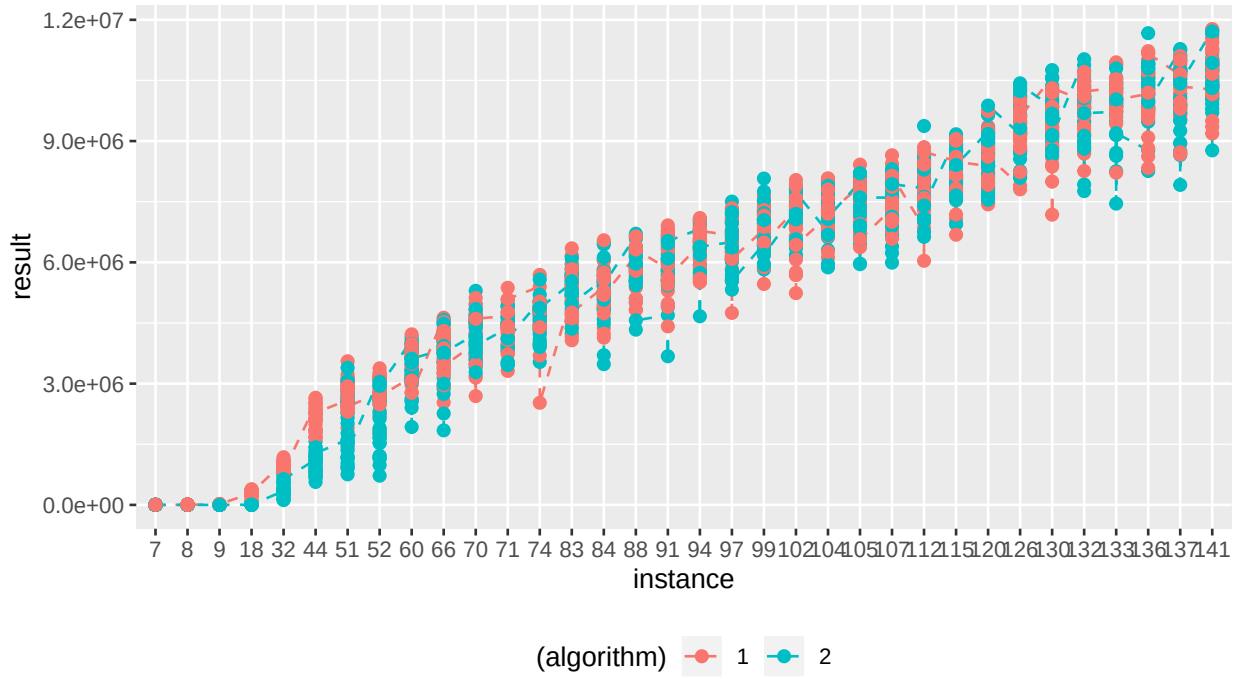


Figure 2: Instâncias selecionadas

```
## Picking joint bandwidth of 211000
## Picking joint bandwidth of 211000
```

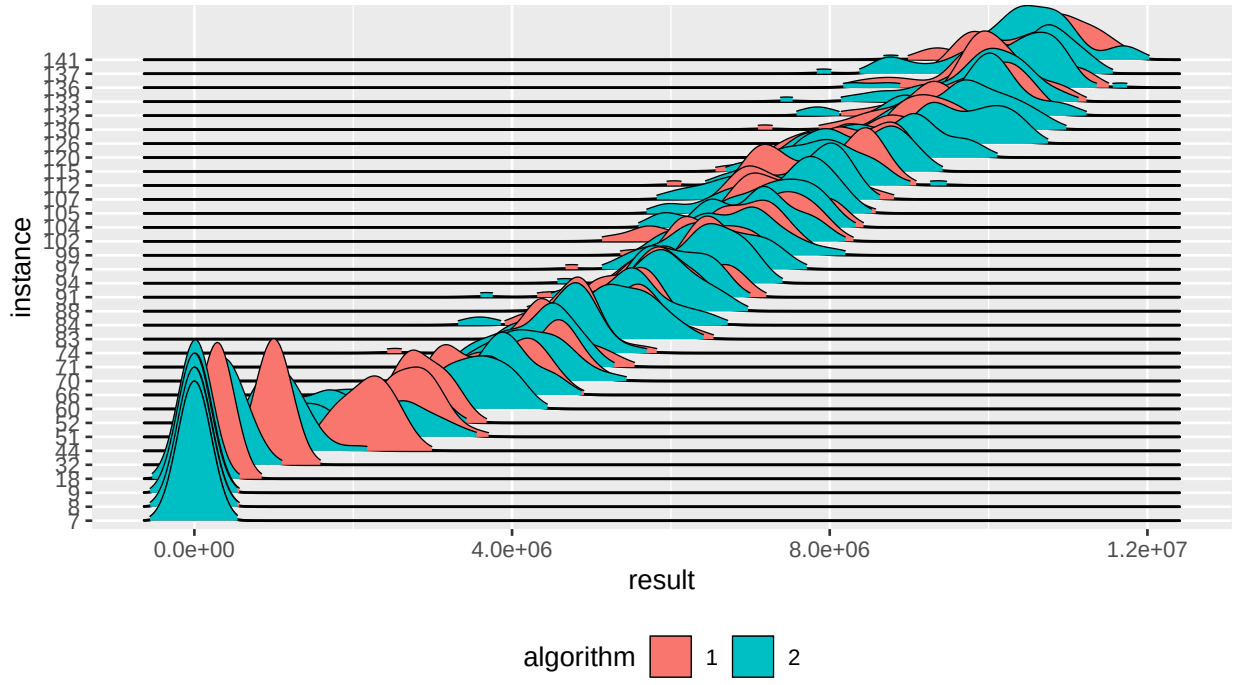


Figure 3: Ridge plot dos dados

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 tratam-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

Para a validação desta premissa, aplicou-se o teste ANOVA aos dados e depois foram avaliados os resíduos obtidos pelo moddelo. O gráfico quantil-quantil e teste de shapiro-wilk também foram utilizados nessa validação, como apresentados a seguir.

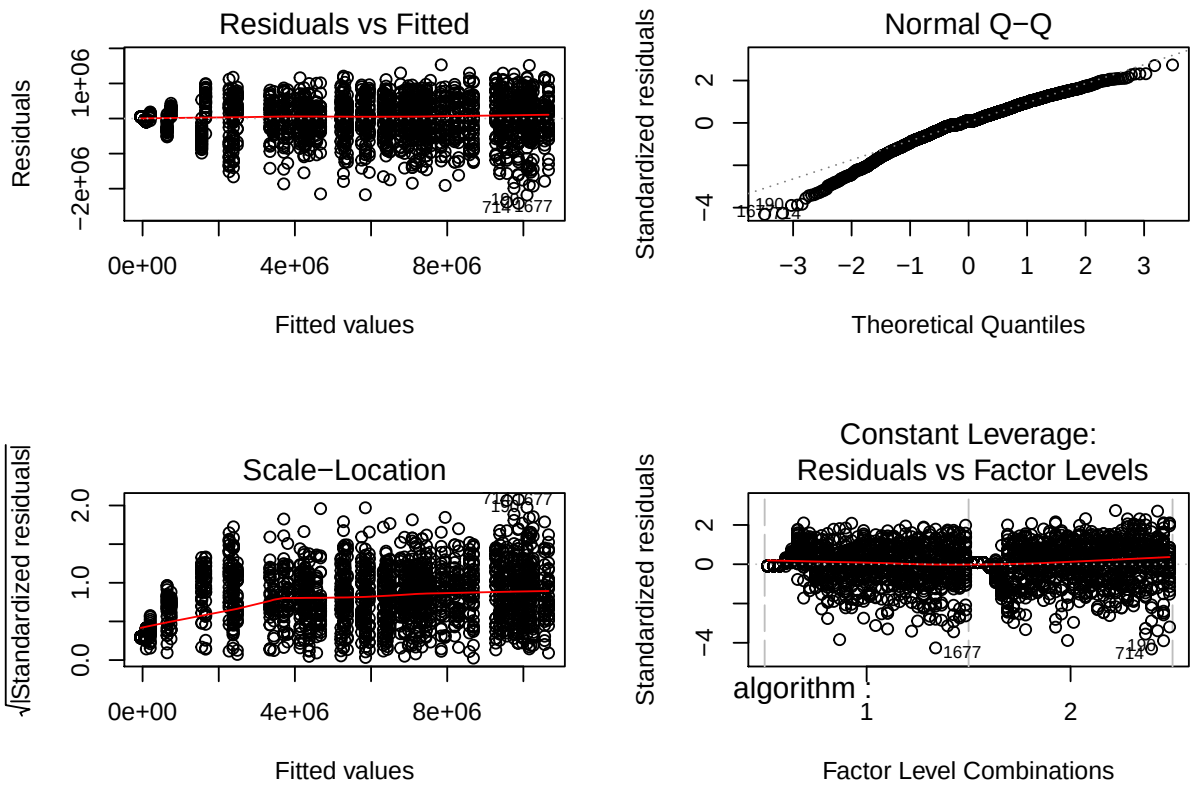


Figure 4: Resultados do ANOVA para transformação logarítmica

```
##
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.97513, p-value < 2.2e-16
```

Pelo gráfico quantil-quantil e teste de Shapiro-Wilk, é possível ver que a premissa de normalidade é violada. Nota-se que o gráfico quantil-quantil apresenta um desvio da normalidade maior à esquerda, o que pode ser explicado pela grande variabilidade das amostras em dimensões menores, conforme visto na figura 3. Logo, uma transformação logarítmica é aplicada aos dados como forma de tentar levá-los à normalidade e também reduzir os efeitos da diferença de magnitude dos resultados de cada instância.

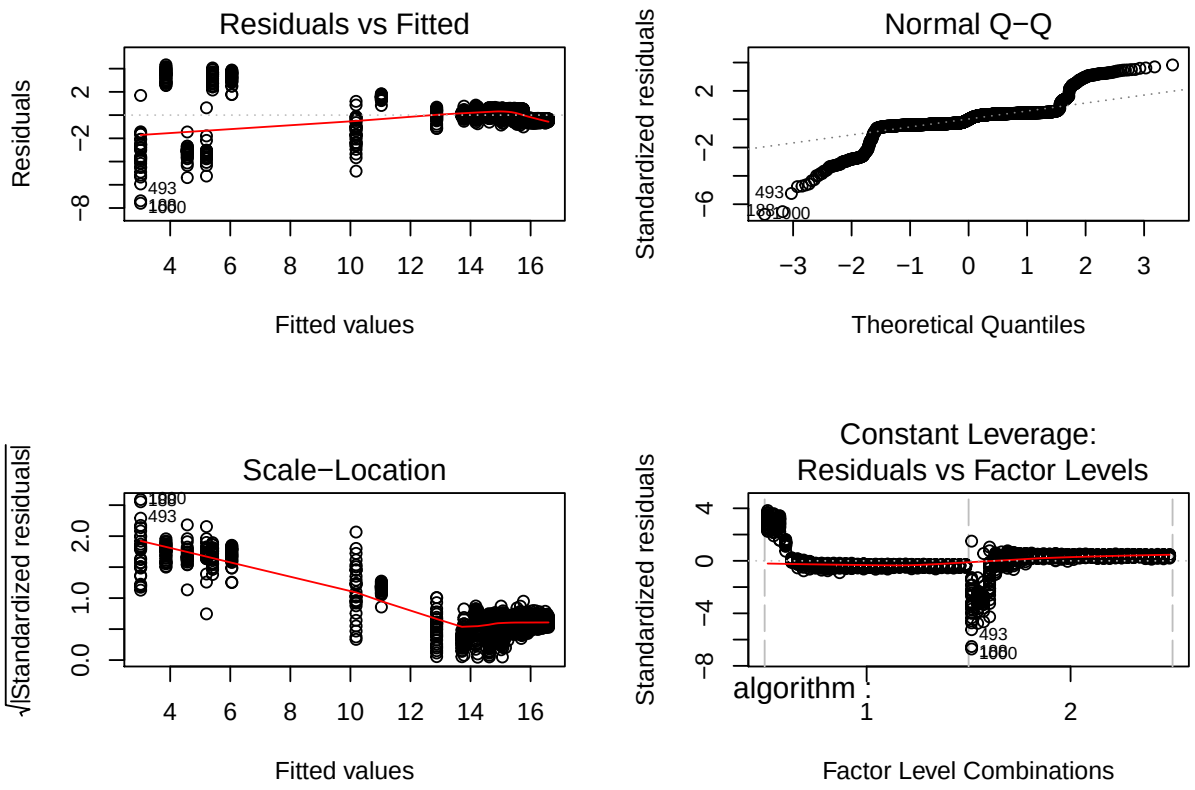


Figure 5: Resultados do ANOVA para transformação logarítmica

```
##
##  Shapiro-Wilk normality test
##
## data:  res.aov$residuals
## W = 0.76544, p-value < 2.2e-16
```

Aplicando essa transformação, é possível notar que o desvio da normalidade é ainda mais evidenciado, conforme pode ser visto no gráfico quantil-quantil. Portanto, a premissa é novamente violada e testes de hipótese não-paramétricos deverão ser utilizados para se obter resultados mais confiáveis.

B - Igualdade de Variâncias

Para validação dessa premissa utilizou-se o teste homogeniedade de variâncias no qual a hipotese nula é razão entre as variancias igual 1. Para isso, analisou-se as duas configurações com as instâncias selecionadas anteriormente. O resultado por ser visto na tabela 6.

Table 3: p-valores do teste F

Instância	p-valor
7	0.0000000
8	0.0000000
9	0.0000000

Instância	p-valor
18	0.0000011
32	0.0037336
44	0.6960527
51	0.0008317
52	0.0026071
60	0.0517480
66	0.1617131
70	0.3310122
71	0.7763353
74	0.2420485
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
120	0.9651509
126	0.6185635
130	0.1419264
132	0.2281559
133	0.0343971
136	0.6794527
137	0.1266789
141	0.8248179

Verifica-se que para diferentes valores de instâncias, dimensões, o p-valor é maior que o nível de significancia pré-estabelecido. Observa-se que a igualdade de variâncias é em sua maioria respeitada para dimensões maiores. Desta forma não se pode rejeitar a hipotese nula de que as variancias são iguais. Analisando de forma geral verifica-se que é possível que as variancias entre as duas configurações são iguais, conforme pode ser visto no gráfico `Residuals vs Fitted` na figura 4.

C - Independência

Como descrito anteriormente, a coleta de dados foi meticulosamente planejada de forma a garantir a independência entre as amostras. As características de cada amostras foram geradas e coletadas de modo aleatório, a fim de evnitar qualquer iterção entre a amostra e a instância avaliada. Desse modo, é possível garantir a independância entre as amostras.

4 Resultados

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

Como as premissas do teste paramétrico (ANOVA) foram parcialmente validadas, optou-se por realizar os testes paramétricos e não-paramétricos a fim de comparar os resultados. Os resultados das duas aborgagens

são apresentados e discutidos nas próximas seções.

4.1 Teste de Hipótese - Paramétrico

O teste paramétrico usado para avaliar a blocagem é o ANOVA

```
model <- aov(result~algorithm+instance,
             data = data.block.instances)
summary(model)
```

```
##               Df    Sum Sq   Mean Sq F value   Pr(>F)
## algorithm      1 5.429e+12 5.429e+12   17.03 3.83e-05 ***
## instance      33 2.195e+16 6.651e+14 2086.29  < 2e-16 ***
## Residuals   2005 6.392e+14 3.188e+11
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
## [1] 0.9717086
```

De acordo com o resultado do teste (p-valor < 0.05) existem evidências de que a média dos dois algoritmos não são iguais. Além disso, o modelo obteve um ajuste satisfatório com o $r^2 = 0.974$, isto é, o modelo explica mais de 97% dos resultados dos dois algoritmos.

Assim como na validação, foi realizado o teste de hipótese considerando uma transformação logarítmica dos resultados.

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## algorithm      1    362   362.0   278.4 <2e-16 ***
## instance      33  20996   636.2   489.3 <2e-16 ***
## Residuals   2005   2607     1.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
## [1] 0.8912031
```

Pela análise dos resultados, pode-se concluir que o modelo também ajusta bem aos dados, porém o modelo com os dados originais obteve melhor ajuste.

4.2 Teste de Hipótese - Não Paramétrico

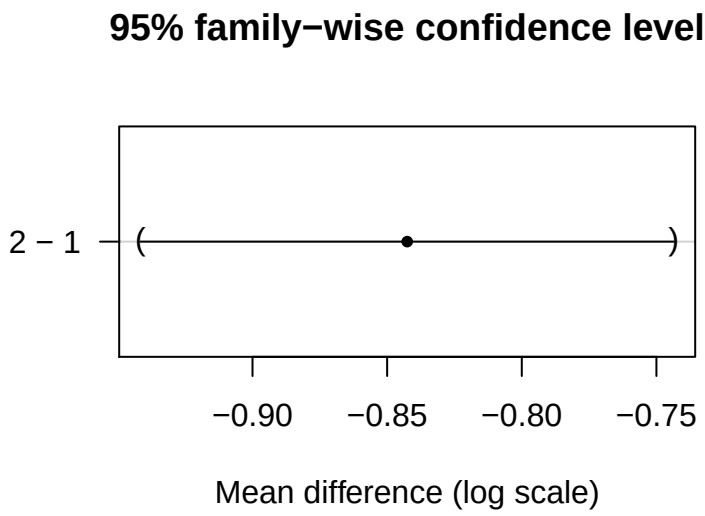
Verificando que os dados gerados pelas duas configurações não são normais e que não há a garantia de homegeniedade entre as variâncias, utilizou-se também um teste não paramétrico. O teste utilizado para verificar a igualdade das médias entre as duas configurações foi o teste de Kruskal-Wallis.

```
##
##  Kruskal-Wallis rank sum test
##
## data:  result by algorithm
## Kruskal-Wallis chi-squared = 0.55331, df = 1, p-value = 0.457
```

É importante ressaltar que o teste de Kruskal-Wallis não inclui os efeitos da blocagem. Pelos resultados obtidos (p-valor > 0.05), é possível concluir que a hipótese nula não pode ser rejeitada. Logo, diferente da conclusão obtida com a utilização do ANOVA, neste caso podemos afirmar que a mediana da diferença dos algoritmos é similar.

4.3 Estimação das magnitudes das diferenças

Para a estimativa do intervalo de confiança, utilizou-se o teste de Dunnet sobre os resultados do anova. Note que o intervalo obtido não contém o valor 0, ou seja, a hipótese nula é rejeitada.



O teste de Wilcoxon, por ser não-paramétrico, é aplicado para a estimativa de intervalo de confiança.

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  result by algorithm
## W = 530095, p-value = 0.457
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -167924.6  298979.6
## sample estimates:
## difference in location
##              15874.11
```

Note que, conforme esperaod, o intervalo de confiança contém o valor 0, ou seja, não podemos rejeitar a hipótese nula.

4.4 Apêndice

4.4.1 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)
```

4.4.2 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.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, "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.