

Estudo de caso: Grupo D 3

Gilmar Pereira, Maressa Tavares e Victor Ruela

11 de Novembro, 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 o desenvolvimento do trabalho o algoritmo DE (Differential Evolution) foi ajustado com duas configurações alterando a forma de recombinação e mutação dos algoritmos. As classes de funções para este experimento foi composta pela função Rosenbrock [3] de dimensões entre 2 e 150. Para analisar os dados utilizou-se a técnica de blocagem determinando a quantidade de blocos e seus tamanhos, assim como o número de amostras por instância.

Realizou-se o cálculo do número de blocos de acordo com [4] e o número de instâncias de acordo com [5]. Foram realizados os testes das premissas, porém não foi possível de validar completamente todas elas. Assim, foi realizado teste paramétrico e não paramétrico e comparados os resultados. Os dois testes levaram a conclusões diferentes, que serão apresentados nas seções deste relatório.

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 configuração do algoritmo DE dentre as classes de funções, determinando a configuração de melhor desempenho e ressaltando as magnitudes das diferenças encontradas.

2.2 Delineamento

Para execução do experimento foram realizadas as seguintes etapas, as quais estão detalhadas nas próximas seções.

- Formulação das hipóteses de teste;
- Cálculo dos tamanhos amostrais, estabelecimento das quantidades de instâncias e número de iterações do algoritmo;
- Coleta e tabulação dos dados,
- Realização dos testes 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 amostrais das configurações 1 e 2 dos algoritmos, 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, recomendado como suficiente por Campelo e Takahashi [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 A.

2.4.1 Geração do 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 e níveis a . Um último passo consiste em randomizar o arquivo de configuração e dividi-lo em 3 arquivos separados, executados por cada membro do grupo. Isso garante que as amostras geradas sejam 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 | result |
|-----------|-----------|----------|--------|
| 1 | 30 | 94 | -1 |
| 1 | 29 | 140 | -1 |
| 2 | 8 | 113 | -1 |
| 2 | 16 | 7 | -1 |
| 1 | 5 | 118 | -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 de configuração (.csv) 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 que seja possível continuar a execução do arquivo sem perder as amostras realizadas anteriormente, caso ocorra algum problema.

2.5 Análise Exploratória dos Dados

Nesta seção é apresentado uma análise exploratória dos dados, a fim de verificar as premissas de normalidade, homocedasticidade, independência, que devem ser validadas antes da realização dos testes estatísticos.

Como o estudo consiste na comparação entre os resultados da execução de duas configurações de um algoritmo de otimização, a dimensão da função objetivo é um fator importante. Logo, a análise exploratória foi feita considerando amostras de instâncias de baixa, média e alta dimensão. Inicialmente, os dados do experimento foram carregados, sendo as instâncias 4, 50 e 100 escolhidas para avaliação, e um gráfico boxplot foi criado para a análise preliminar (Figura 1).

```
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)

ggplot(sample.all.eda, aes(x=configuration, y=result)) +
  geom_boxplot() + facet_grid(rows = vars(instance), scales = "free_y")
```

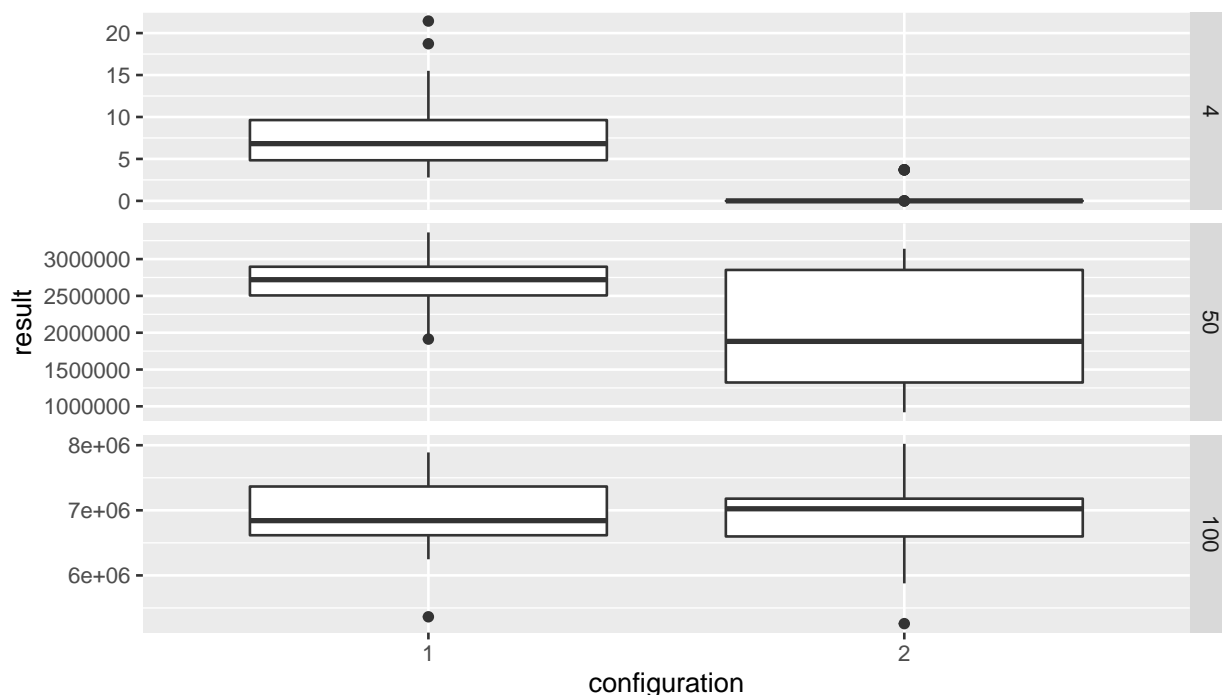


Figure 1: Boxplot dos dados

Através da figura 1, 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 a 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 ao mínimo da função. Entretanto, o mesmo não pode ser afirmado para dimensões maiores.

2.6 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 2.

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.

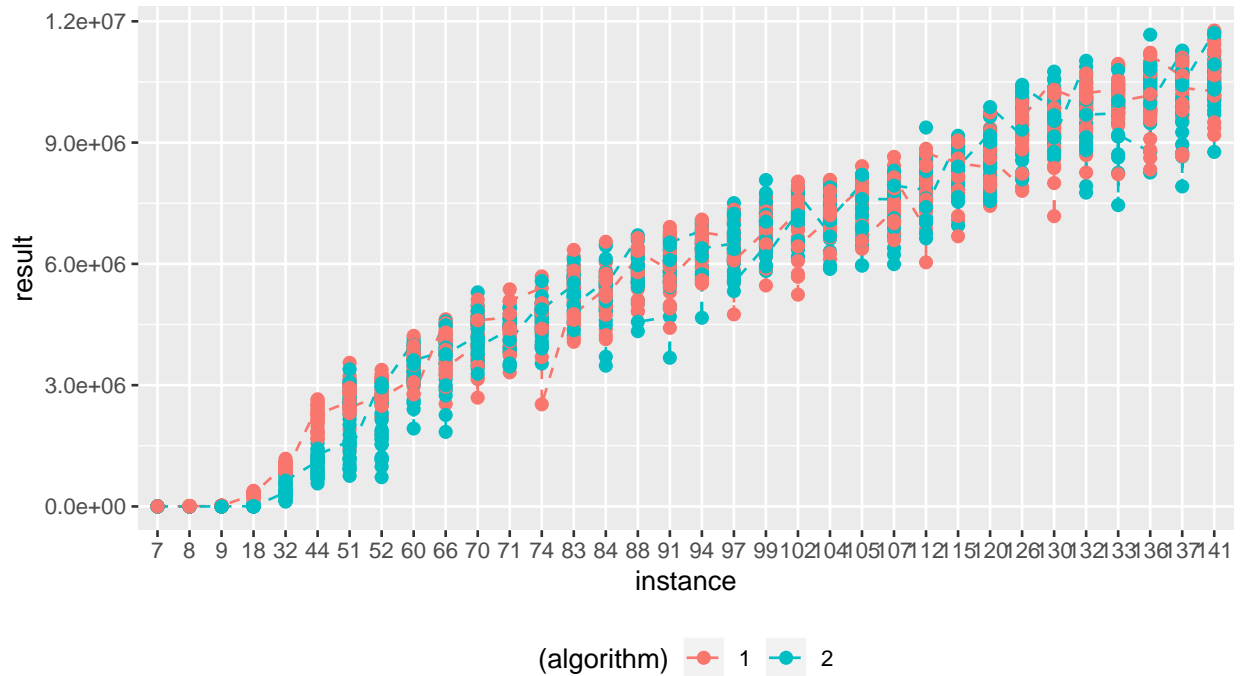


Figure 2: Instâncias selecionadas

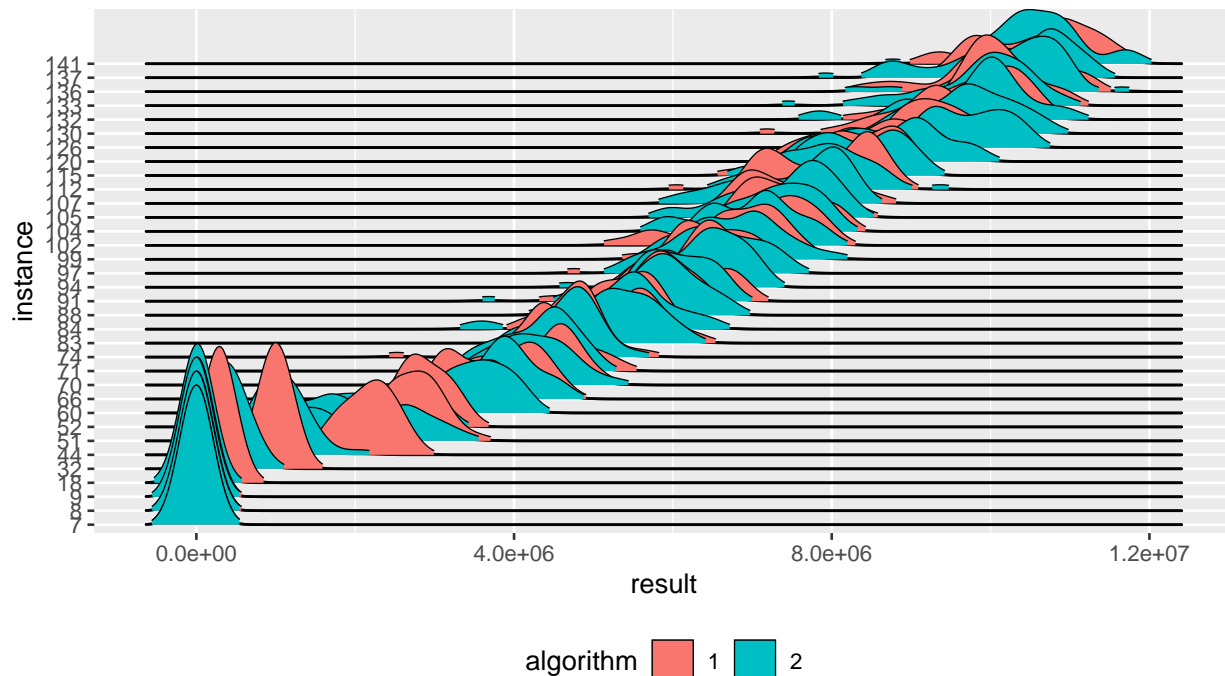


Figure 3: Ridge plot dos dados

Analisando as figuras percebe-se claramente a influência do número de instância no desempenho do algoritmo, porém, o mesmo não se pode afirmar para as diferentes configurações.

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 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 pela blocagem (ANOVA).

A - Normalidade

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

```
res.aov <- aov(result ~ algorithm + instance, data = data.block.instances)
par(mfrow=c(2,2))
plot(res.aov)
```

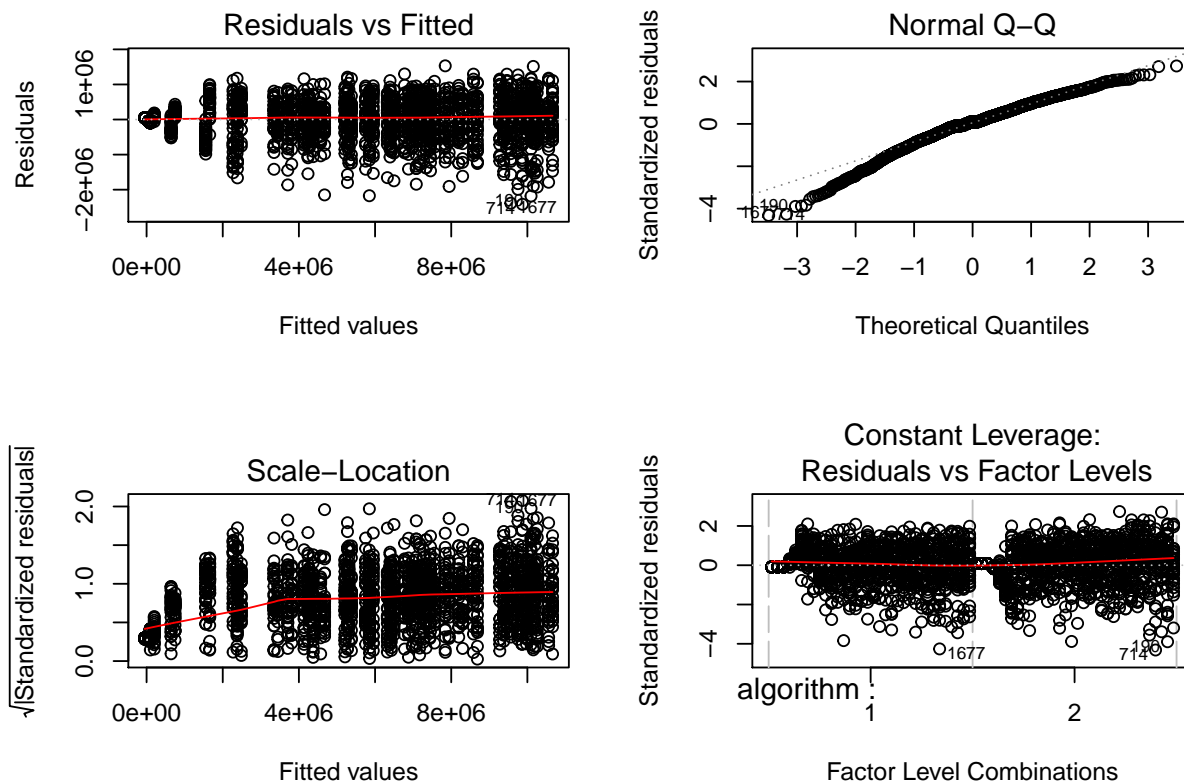


Figure 4: Resultados do ANOVA

```
shapiro.test(res.aov$residuals)
```

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

Pelo gráfico quantil-quantil e o teste de Shapiro-Wilk, é possível ver que a premissa de normalidade é violada com o p-valor muito abaixo da significância estabelecida. 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 foi 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.

```
res.aov <- aov(resultLog ~ algorithm + instance, data = data.block.instances)
par(mfrow=c(2,2))
plot(res.aov)
```

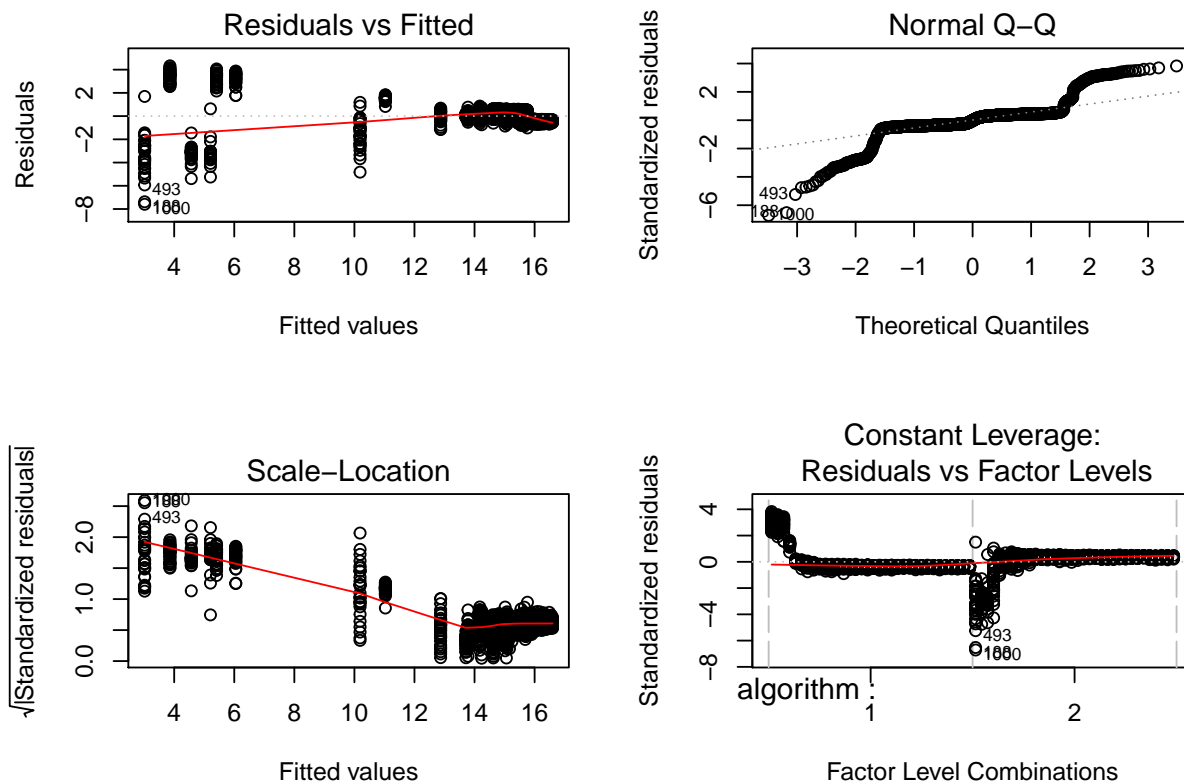


Figure 5: Resultados do ANOVA para transformação logarítmica

```
shapiro.test(res.aov$residuals)
```

```
##
##  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 (p-valor baixo) 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 de homogeneidade de variâncias no qual a hipótese nula considera que razão entre as variâncias é igual 1. Para isso, analisou-se as duas configurações com as instâncias selecionadas anteriormente. O resultado por ser visto na tabela 3.

```
sw.test.result <- data.frame("p.value" = rep(0,b), instance = rep(0,b))
for (i in seq(1,b)){
  bi <- sort(random.blocks)[i]
  data.block.algo1 <- data.block.instances %>% filter(instance == bi & algorithm == 1)
  data.block.algo2 <- data.block.instances %>% filter(instance == bi & algorithm == 2)
```



```

sw.test.result[i,] <- c(var.test(data.block.algo1$result, data.block.algo2$result)$p.value,bi)
}

kable(sw.test.result[,c(2,1)], col.names = c("Instância","p-valor"),
      caption = 'p-valores do teste F \\label{tab:varteste1}', align = 'l')

```

Table 3: p-valores do teste F

| Instância | p-valor |
|-----------|-----------|
| 7 | 0.0000000 |
| 8 | 0.0000000 |
| 9 | 0.0000000 |
| 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 significância pré-estabelecido. Observa-se que a igualdade de variâncias é em sua maioria respeitada para dimensões maiores. 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 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 amostra foram geradas e coletadas de modo aleatório, a fim de evitar qualquer interação entre a amostra e a instância avaliada. Desse modo, pode-se garantir a independência entre as amostras.

3 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.

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 abordagens são apresentados e discutidos nas próximas seções.

3.1 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 $r^2 = 0.971$, 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.

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

```
##              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
```

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

```
## [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. Apesar dos bons resultados de correlação foi realizado também o teste não paramétrico, devido às falhas na validação das premissas.

3.2 Teste de Hipótese - Não Paramétrico

Verificando que os dados gerados pelas duas configurações não são normais e que não há a garantia de homogeneidade 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 [7].

```
(krus.test <- kruskal.test(result~algorithm,data=data.block.instances))
```

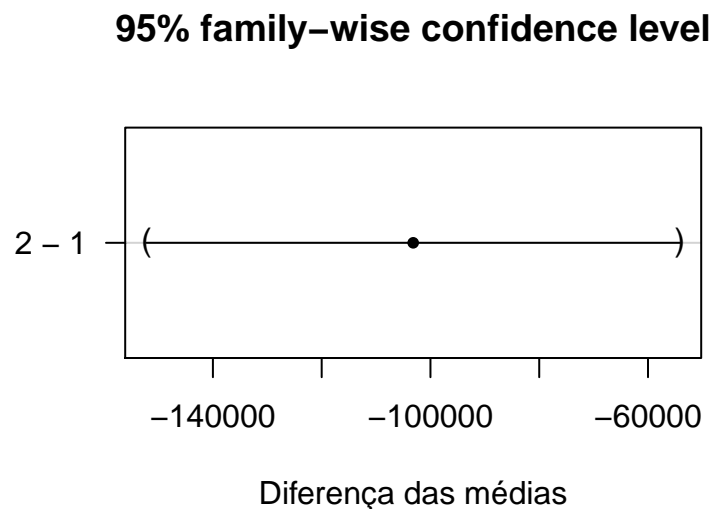
```
##  
## Kruskal-Wallis rank sum test  
##  
## data: result by algorithm  
## Kruskal-Wallis chi-squared = 0.55331, df = 1, p-value = 0.457
```

É importante ressaltar que o teste de Kruskal-Wallis não inclui os efeitos da blocagem. Pelos resultados obtidos ($p\text{-valor} > 0.05$), 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.

3.3 Estimação das magnitudes das diferenças

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

```
duntest <- glht(model, linfct = mcp(algorithm = "Dunnett"))  
duntestCI <- confint(duntest)  
par(mar = c(5, 8, 4, 2), las = 1)  
plot(duntestCI, xlab = "Diferença das médias")
```



O teste de Wilcoxon [8], por ser não-paramétrico e válido para o caso da comparação entre duas populações, é aplicado para a estimativa de intervalo de confiança.

```
wilcox.test(result ~ algorithm ,data=data.block.instances, conf.int = TRUE,  
            alternative='two.sided', conf.level=0.95, paired=FALSE)
```

```
##  
## 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 esperado, o intervalo de confiança contém o valor 0, ou seja, não podemos rejeitar a hipótese nula de que as médias são iguais.

4 Conclusão

O presente trabalho realizou o delineamento do experimento que testou o desempenho de duas configurações diferentes de um algoritmo DE. Como não foi possível validar todas as premissas com convicção, optou-se por realizar os testes paramétricos e comparar os resultados com os de testes não paramétricos, que não precisam das premissas serem respeitadas.

Após a realização dos testes verificou-se que, provavelmente, devido à não validação das premissas e à influência das instâncias nos resultados, não foi possível chegar à mesma conclusão pelos testes paramétrico e não paramétrico.

Enquanto o teste paramétrico apresenta evidências de rejeição da igualdade das médias ($p\text{-valor} < 0.05$), o teste não paramétrico leva a concluir que não há diferença entre os resultados dos dois algoritmos ($p\text{-valor} = 0.4569679$) para a significância especificada.

4.1 Apêndice A

4.1.1 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)

```

4.1.2 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] “Kruskal.test function | r documentation.” <https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/kruskal.test>.
- [8] “Wilcox.test function | r documentation.” <https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/wilcox.test>.