Estudo de Caso 3

Andre Avelar, Brayan Jaimes, Marcelo Carneiro, William de Paula

07 de novembro de 2017

Estudo de Caso 3

Introdução

Algoritmos computacionais são amplamente utilizados em problemas de otimização em engenharia. A otimização busca obter um ponto ótimo para uma dada função-objetivo, podendo este ponto ser um mínimo ou um máximo conforme o problema a ser otimizado. Entre os algoritmos disponíveis, temos aqueles baseados em populações, que consistem de um ciclo iterativo no qual um conjunto de soluções-candidatas são repetidamente sujeitas a operadores de variação e seleção, promovendo uma exploração do espaço de variáveis em busca do ponto ótimo.

Um pesquisador implementou um algoritmo baseado em populações utilizando o método *differential evolution* [1] e diversos operadores de forma padronizada no pacote ExpDE [2] do *RStudio*.

O presente estudo de caso tem o objetivo de comparar experimentalmente quatro diferentes configurações deste algoritmo para uma única função-objetivo. A comparação será feita através do desempenho médio do algoritmo, sendo que quanto menor o valor retornado, melhor o algoritmo. Adicionalmente, devemos responder às seguintes perguntas:

- Há alguma diferença no desempenho médio do algoritmo para as diferentes configurações?
- Caso haja diferença, qual a melhor configuração em termos de desempenho médio, e qual a magnitude das diferenças encontradas?
- Há alguma configuração que deva ser recomendada em relação às demais?

Para este estudo de caso, os papéis desempenhados por cada membro da equipe *Los Paisas* foram:

- *Coordenador*: Brayan
- *Relator*: André
- *Verificador*: William
- *Monitor*: Marcelo

O programa *RStudio* foi utilizado para a realização dos cálculos e dos gráficos deste estudo de caso, e o *R Markdown* foi utilizado para a consolidação e elaboração deste relatório.

Atividades

1. Formulação das Hipóteses de Teste

O objetivo é determinar se há alguma diferença no desempenho médio do algoritmo para as diferentes configurações. Portanto, para esta análise foram definidas as seguintes hipóteses de teste:

$$\begin{cases} H_0 : \tau_i = 0, \forall\ i \\ H_1 : \exists\ \tau_i \neq 0 \end{cases}$$

Se for possível rejeitar a hipótese nula (H_0), podemos afirmar com um certo nível de confiança que existe alguma diferença no desempenho médio do algoritmo para as diferentes configurações.

Utilizaremos o teste da distribuição tipo **F**.

$$F_0 = \frac{MS_{níveis}}{MS_E} \quad \Rightarrow \quad F_{(a-1),a(n-1)} \ \ graus\ de\ liberdade$$

Onde:
$MS_{níveis} = \frac{SS_{níveis}}{a-1}$ representa os quadrados médios dos níveis, dado pela soma dos quadrados dos níveis dividido pelo respectivo graus de liberdade.
$MS_E = \frac{SS_E}{a(n-1)}$ representa os quadrados médios dos resíduos, dado pela soma dos quadrados dos resíduos dividido pelo respectivo graus de liberdade.

Rejeitaremos H_0 com o nível de significância α se $f_0 > F^{1-\alpha}_{(a-1),a(n-1)}$.

Este teste pode ser executado no *RStudio* utilizando os comandos 'aov' e posteriormente 'summary.aov' (denominado tabela ANOVA), que retorna o valor da estatística de teste F_0 e o valor da probabilidade de ser maior que F_0 (ou p-valor). Neste caso, rejeitaremos H_0 com o nível de significância α se p-valor $< \alpha$.

2. Cálculo do Tamanho Amostral

Os seguintes parâmetros experimentais foram fornecidos:

- Mínima diferença de importância prática padronizada em termos do coeficiente d de Cohen: $d^* = \delta^*/\sigma = 0.25$, o que significa que a razão delta pelo desvio padrão deve ser igual a 0.25
- Nível de significância: $\alpha = 0.05$
- Potência mínima: $\pi = 0.85$

```
a<-4
k<-(a*(a-1))/2
alpha<-0.05
alpha_adj<-alpha/k
delta<-10
potencia_desejada<-0.85
sd<-40
```

Estamos testando quatro configurações diferentes do mesmo algoritmo, o que nos leva a um fator experimental único (tipo de configuração) com $a = 4$ níveis diferentes (Config 1, Config 2, Config 3 e Config 4) e número de réplicas n em cada nível sendo calculado na sequência.

Um dos objetivos do estudo de caso é determinar qual configuração é melhor do que as outras, sem interesse especial em uma configuração específica. Desta forma, após realizar o teste ANOVA, devemos realizar comparações de *todos* vs. *todos*. Neste caso, o número de comparações K é calculado por:

$$K = \frac{a(a-1)}{2}$$

Assim, K = 6. Com o intuito de manter a taxa geral de erro controlada no valor desejado $\alpha = 0.05$, devemos ajustar o valor α usado em cada teste de comparação de par de configurações e também no cálculo do tamanho amostral. Utilizando o método de correção de Bonferroni, o valor α_{ajust} é calculado como:

$$\alpha_{ajust} = \frac{\alpha_{família}}{K}$$

Assim, $\alpha_{ajust} = 0.0083333$.

Com estas informações, utilizamos o comando 'power.t.test' para duas amostras do *RStudio* para calcular o número de réplicas n em cada nível, ou seja, o tamanho amostral.

```
test_pow_N <- power.t.test(delta = delta, power = potencia_desejada, sd = sd, sig.level = alpha_adj, al
print(test_pow_N)
```

```
##
##      Two-sample t test power calculation
##
##               n = 433.8504
##           delta = 10
##              sd = 40
##       sig.level = 0.008333333
##           power = 0.85
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Com este resultado, o tamanho amostral em cada nível é $n = 434$.

3. Coleta e Tabulação dos Dados

Após o cálculo do número de réplicas em cada nível, ou seja, o tamanho amostral, fizemos a coleta dos dados utilizando os parâmetros fixos e os parâmatros de cada equipe através dos comandos descritos na sequência.

if(!require(ExpDE)){
install.packages("ExpDE")
library(ExpDE)
}

selpars <- list(name = "selection_standard")
stopcrit <- list(names = "stop_maxeval", maxevals = 50000, maxiter = 1000)
probpars <- list(name = "sphere", xmin = -seq(1,20), xmax = 20 + 5 * seq(5, 24))

%# Equipe Marcelo, Brayan, William, André

%## Config 1
recpars1 <- list(name = "recombination_blxAlphaBeta", alpha = 0, beta = 0)
mutpars1 <- list(name = "mutation_rand", f = 4)
popsize1 <- 200

%## Config 2
recpars2 <- list(name = "recombination_exp", cr = 0.6)
mutpars2 <- list(name = "mutation_best", f = 2)
popsize2 <- 130

%## Config 3
recpars3 <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
mutpars3 <- list(name = "mutation_rand", f = 4)
popsize3 <- 230

%## Config 4
recpars4 <- list(name = "recombination_wright")

```
mutpars4 <- list(name = "mutation_best", f = 4.8)
popsize4 <- 113

%# Run algorithm on problem:
Numero_it<-434
result_out1<-list()
result_out2<-list()

for (i in 1:Numero_it) {
print(i)
out1 <- ExpDE(popsize = popsize1,
mutpars = mutpars1,
recpars = recpars1,
selpars = selpars,
stopcrit = stopcrit,
probpars = probpars,
showpars = list(show.iters = "dots",
showevery = 20))

out2 <- ExpDE(popsize = popsize2,
mutpars = mutpars2,
recpars = recpars2,
selpars = selpars,
stopcrit = stopcrit,
probpars = probpars,
showpars = list(show.iters = "dots",
showevery = 20))

out3 <- ExpDE(popsize = popsize3,
mutpars = mutpars3,
recpars = recpars3,
selpars = selpars,
stopcrit = stopcrit,
probpars = probpars,
showpars = list(show.iters = "dots",
showevery = 20))

out4 <- ExpDE(popsize = popsize4,
mutpars = mutpars4,
recpars = recpars4,
selpars = selpars,
stopcrit = stopcrit,
probpars = probpars,
showpars = list(show.iters = "dots",
showevery = 20))

result_out1[[i]]<-c(out1Fbest, out2Fbest,out3Fbest, out4Fbest)
result_out2[[i]]<-c("out1","out2","out3","out4")
}

dados_total<-data.frame(label_out=(unlist(result_out2)),out=(unlist(result_out1)))
write.csv(dados_total, file = "dados_total.csv")
```

A coleta de dados inicial resultou no arquivo "dados_total.csv". Uma vez que a cada execução o algoritmo retorna um valor diferente, não é recomendado executar a rotina acima novamente. Se a rotina for executada novamente, teremos novos valores amostrados. Por esta razão, a rotina está em formato de texto e não em formato de código R executável neste relatório.

A importação dos dados foi executada utilizando o comando 'read_delim', e a análise exploratória inicial dos dados com os comandos 'summary', 'head' e 'boxplot'.

```
## Loading required package: car
```

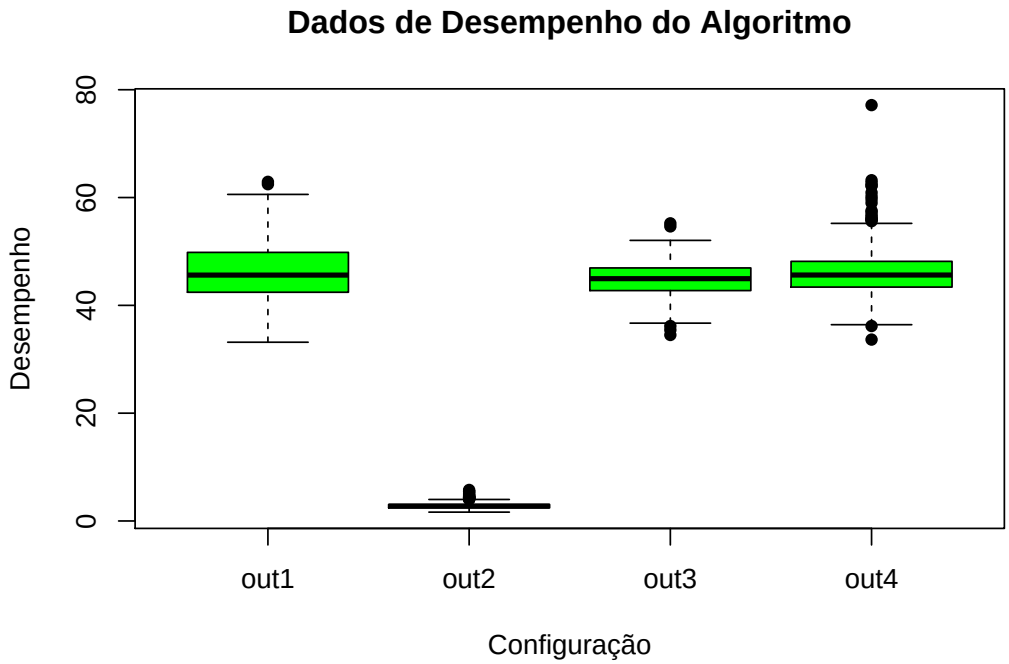
```
algoritmo <- read.table(file = "dados_total.csv", header = TRUE, sep = ",")
summary(algoritmo)
```

```
##        X          label_out       out        
##  Min.   :   1.0   out1:434   Min.   : 1.64  
##  1st Qu.: 434.8   out2:434   1st Qu.:26.31  
##  Median : 868.5   out3:434   Median :43.78  
##  Mean   : 868.5   out4:434   Mean   :34.98  
##  3rd Qu.:1302.2              3rd Qu.:47.03  
##  Max.   :1736.0              Max.   :77.15
```

```
head(algoritmo, 8)
```

```
##   X label_out       out
## 1 1      out1 44.265657
## 2 2      out2  3.471544
## 3 3      out3 44.210241
## 4 4      out4 48.889531
## 5 5      out1 47.084555
## 6 6      out2  2.858926
## 7 7      out3 42.828197
## 8 8      out4 47.596735
```

```
boxplot(out~label_out, data = algoritmo, xlab = "Configuração", ylab = "Desempenho", main = "Dados de D
```



O boxplot sugere que: (i) a configuração 2 possui um desempenho melhor do que as demais configurações; (ii) as configurações 1, 3 e 4 possuem desempenho similar; (iii) todas as configurações possuem simetria dos dados e (iv) possíveis *outliers* devem ser levados em consideração, principalmente nas configurações 3 e 4.

4. Teste das Hipóteses

O teste das hipóteses foi realizado no *RStudio* utilizando os comandos 'aov' e 'summary.aov', conforme descrito na seção 1.

```
model <- aov(out~label_out, data = algoritmo)
summary.aov(model)
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## label_out      3 599221  199740   12985 <2e-16 ***
## Residuals   1732  26642      15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O resultado do teste das hipóteses retornou um p-valor < 2 x 10^{-16}. Como o p-valor é menor do que o valor de $\alpha = 0.05$, temos evidência suficiente para rejeitar H_0 com o nível de significância de 0.05. Desta forma, concluímos que existe alguma diferença no desempenho médio do algoritmo para as diferentes configurações.

5. Verificação das Premissas dos Testes

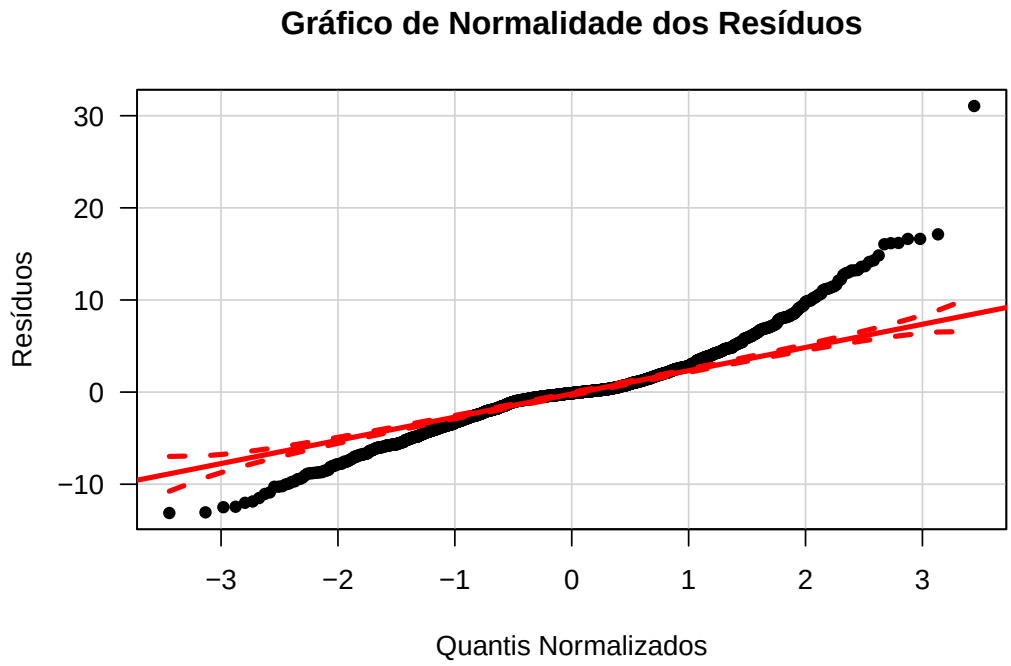
O teste de hipóteses realizado na seção 4 assume as premissas de normalidade, homoscedasticidade e independência dos resíduos. Vamos agora verificar se estas premissas foram atendidas.

A premissa de normalidade pode ser verificada através do teste de Shapiro-Wilk em conjunto com o gráfico tipo qqPlot.

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

```
##
##  Shapiro-Wilk normality test
##
## data:  model$residuals
## W = 0.9392, p-value < 2.2e-16
```

```
sht <- shapiro.test(model$residuals)
qqPlot(model$residuals, pch = 16, lwd = 3, cex = 1, las = 1, xlab = "Quantis Normalizados", ylab = "Resi
```




```
## p-valor:  4.340659e-26
```

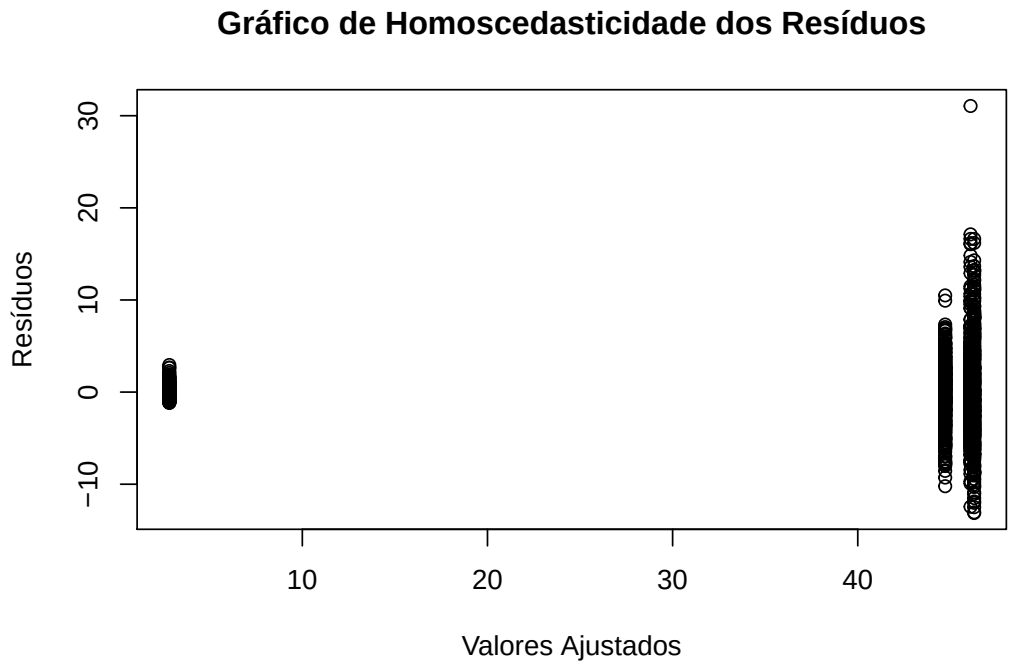
O resultado do teste de Shapiro-Wilk retornou um p-valor = $4.3406593 \times 10^{-26}$. Este p-valor muito baixo nos leva a rejeitar o teste de normalidade e concluir que os resíduos não possuem uma distribuição normal. No gráfico qqPlot, observamos que os dados estão fora da linha de normalidade, com valores abaixo da linha para quantis baixos e acima da linha para quantis altos. Este tipo de gráfico representa distibuições não normais com caudas leves.

A premissa de homoscedasticidade pode ser verificada através do teste de Fligner-Killeen em conjunto com o gráfico por valores ajustados.

```
fligner.test(out~label_out, data = algoritmo)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  out by label_out
## Fligner-Killeen:med chi-squared = 557.16, df = 3, p-value <
## 2.2e-16
```

```
flt <- fligner.test(out~label_out, data = algoritmo)
plot(x = model$fitted.values, y = model$residuals, xlab = "Valores Ajustados", ylab = "Resíduos", main =
```




```
## p-valor:  1.950843e-120
```

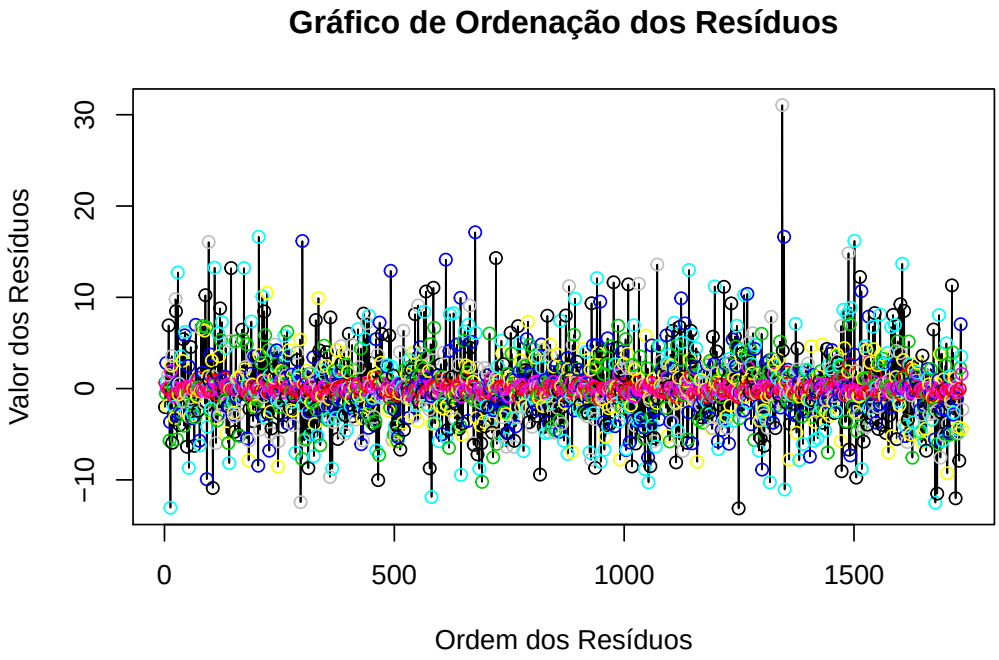
O resultado do teste de Fligner-Killeen retornou um p-valor = $1.9508428 \times 10^{-120}$. Este p-valor muito baixo nos leva a rejeitar o teste de homoscedasticidade e concluir que os resíduos possuem heteroscedasticidade. No gráfico por valores ajustados, observamos uma forma visual do tipo "megafone", onde para maiores valores ajustados no eixo x, temos maiores valores de resíduos no eixo y. Esta forma de gráfico caracteriza desigualdade de variâncias entre grupos, ou heteroscedasticidade.

A premissa de independência foi observada na fase de planejamento, onde a avaliação do algoritmo foi executada através de amostragem sequencial da configuração 1 até a configuração 4 (amostra 1 da config 1, amostra 1 da config 2, amostra 1 da config 3, amostra 1 da config 4; amostra 2 da config 1, amostra 2 da config 2, amostra 2 da config 3, amostra 2 da config 4; e assim por diante), ao invés de amostrar todos os valores da configuração 1 até a configuração 4 (434 amostras da config 1, 434 amostras da config 2, 434 amostras da config 3 e 434 amostras da config 4). Esta premissa também pode ser verificada através do teste de Durbin-Watson para correlações seriais em conjunto com o gráfico de ordenação dos resíduos.

```
durbinWatsonTest(model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.02403142      2.047712   0.316
##  Alternative hypothesis: rho != 0
```

```
dwt <- durbinWatsonTest(model)
plot(x = seq_along(model$residuals), y = model$residuals, type = "l", xlab = "Ordem dos Resíduos", ylab
points(x = seq_along(model$residuals), y = model$residuals, type = "p", col = as.numeric(algoritmo[,1])
```




```
## p-valor:  0.294
```

O teste de Durbin-Watson retornou um p-valor = 0.294. Este p-valor indica independência dos resíduos. O gráfico de ordenação dos resíduos não possui tendência de alta ou de baixa, indicando a independência dos resíduos.

Os resultados desta seção demonstram que as premissas de normalidade e de homoscedasticidade dos resíduos não foram atendidas. Por isto, vamos realizar a adequação destas premissas na próxima seção.

6. Adequação das Premissas dos Testes

Como as premissas de homoscedasticidade e de normalidade não foram atendidas e a análise de variância (ANOVA) depdende dessas premissas, mostrou-se necessário o uso de técnicas não paramétricas, pois essas assumem poucas (ou nenhuma) hipóteses sobre a distribuição de probabilidade da população [3]. Como alternativa ao ANOVA, o teste não-paramétrico de Kruskal-Wallis pode ser utilizado, o qual realiza a comparação entre médias de 3 ou mais amostras independentes [4]. O teste de kruskal-wallis é análogo ao ANOVA 1 fator, porém sem nenhuma restrição sobre as premissas de normalidade e homoscedasticidade.

No R studio, o teste de kruskal-wallis pode ser executado através da função kruskal.test:

```
kruskal.test(out~label_out, data = algoritmo)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  out by label_out
## Kruskal-Wallis chi-squared = 986.88, df = 3, p-value < 2.2e-16
```

O resultado do teste das hipóteses retornou um p-valor $< 2.2 \times 10^{-16}$. Como o p-valor é menor do que o valor de $\alpha = 0.05$, temos evidência suficiente para rejeitar H_0 com o nível de significância de 0.05. Desta forma, concluímos que existe alguma diferença no desempenho médio do algoritmo para as diferentes configurações.

7. Estimação dos Tamanhos de Efeito e dos Intervalos de Confiança

8. Derivação de Conclusões

9. Discussão sobre Possíveis Limitações do Estudo e Sugestões de Melhoria

Referências

[1] Storn R, Price K (1997). "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *J. of Global Optimization*, **11**(4), 341-359.
[2] Campelo F, Botelho M (2016). "Experimental Investigation of Recombination Operators for Differential Evolution." In *Proc. Genetic and Evolutionary Computation Conference - GECCO'2016*.
[3] Portal Action. "Técnicas Não Paramétricas". Disponível em: http://www.portalaction.com.br/tecnicas-nao-parametricas. Acesso em 31/10/2017 [4] Portal Action. "Teste de Kruskal Wallis". Disponível em: http://www.portalaction.com.br/976-4-teste-de-kruskal-wallis. Acesso em 31/10/2017