Planejamento e Análise de Experimentos (EEE933)
Estudo de Caso 4

Pedro Vinícius, Samara Silva e Savio Vieira

5 de Outubro de 2020

Introdução

O algoritmo de Evolução Diferencial (DE, do inglês *Differential Evolution*) é uma estratégia baseada em população comumente utilizada para resolver problemas de otimização cujos métodos baseados em gradiente são inviáveis ou apresentam déficit de desempenho. Na linguagem R, sua implementação está disponível no pacote `ExpDE` [1] com o seguinte protótipo:

```
ExpDE(popsize, mutpars = list(name = "mutation_rand", f = 0.2),
      recpars = list(name = "recombination_bin", cr = 0.8, nvecs = 1),
      selpars = list(name = "standard"), stopcrit, probpars, seed = NULL,
      showpars = list(show.iters = "none"))
```

onde `popsize` é o tamanho da população, `mutpars`, `recpars` e `selpars` são as listas com as definições dos parâmetros de mutação, recombinação e seleção, respectivamente, `stopcrit` é a lista com as definições dos critérios de parada, `probpars` é a lista com os parâmetros relacionados a instância do problema, `seed` é a semente do gerador de números aleatórios e, por fim, `showpars` é a lista que controla o histórico que será impresso no terminal durante a execução do algoritmo.

No presente estudo serão investigadas duas configurações diferentes deste algoritmo:

Algoritmo 1:
Recombinação *Blend Alpha Beta* com $\alpha = 0,4$ e $\beta = 0,4$
`recpars` <- list(name = "recombination_blxAlphaBeta", alpha = 0.4, beta = 0.4)
Mutação nos indivíduos aleatórios com um fator de escala $f = 4$ para o vetor de diferenças
`mutpars` <- list(name = "mutation_rand", f = 4)

Algoritmo 2:
Recombinação binomial com probabilidade $c_r = 0,9$
`recpars` <- list(name = "recombination_eigen", othername = "recombination_bin", cr = 0.9)
Mutação nos melhores indivíduos com um fator de escala $f = 2,8$ para o vetor de diferenças
`mutpars` <- list(name = "mutation_best", f = 2.8)

A função de teste escolhida foi a Rosenbrock, cuja equação é dada por (1):

$$f(\mathbf{x}) = \sum_{i=1}^{D-1} (100(x_i^2 + x_{i+1})^2) + (x_i - 1)^2) \tag{1}$$

onde D é a dimensão do problema e x_i é a i-ésima variável do indivíduo $\mathbf{x}$, tal que $x_i \in$ [-5, 10] para $i \in [1, D]$. A Figura 1 apresenta a superfície bidimensional da função descrita e ressalta sua principal propriedade que é ter um vale estreito do ótimo local ao ótimo global.

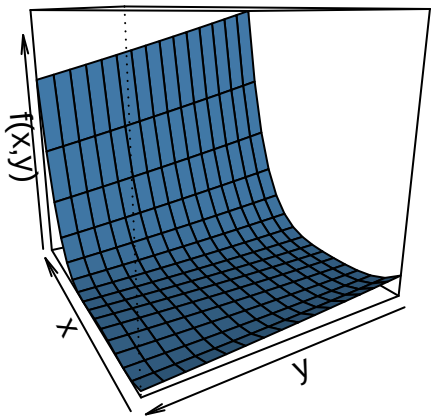

Figura 1: Função Rosenbrock ($D = 2$).

Os demais parâmetros foram definidos igualmente para ambas configurações durante toda a experimentação, sendo tamanho da população `popsize` $= 5 \times D$, número máximo de avaliações na função objetivo `maxevals` $= 5000 \times D$ e número máximo de iterações `maxiter` $= 100 \times D$.

Planejamento dos Experimentos

Dado que o propósito do estudo é analisar o desempenho de duas configurações de algoritmos de otimização, os valores da função objetivo (*fitness*) serão utilizados como métrica de qualidade das soluções encontradas. É interessante ressaltar que em problemas de minimização, como é o caso da função Rosenbrock, soluções boas apresentam valores pequenos de $f(\mathbf{x})$ e soluções ruins apresentam valores maiores de $f(\mathbf{x})$. Assim, ao final de cada execução de um algoritmo para uma determinada instância do problema, apenas o valor de fitness do melhor indivíduo da população será tomado como referência.

As hipóteses estatísticas foram definidas com o objetivo de verificar as seguintes proposições:

- Há alguma diferença no desempenho médio das duas configurações propostas do algoritmo de Evolução Diferencial para a classe de problemas de interesse?
- Caso haja, qual a magnitude dessa diferença encontrada?
- Se possível ainda, qual configuração é superior em termos da qualidade média das soluções encontradas no conjunto de instâncias definido?

Considerando as questões propostas, foram estabelecidas as seguintes hipóteses de teste sobre o *fitness* médio das duas configurações de algoritmo [2]:

$$\begin{cases} H_0 : \mu_{\text{algo}_1} \geq \mu_{\text{algo}_2} \\ H_1 : \mu_{\text{algo}_1} < \mu_{\text{algo}_2} \end{cases} \tag{2}$$

Os parâmetros experimentais considerados para realização dos testes foram nível de significância $\alpha = 0,05$, mínima diferença de importância prática (padronizada) $d^* = \delta^*/\sigma = 0,5$ e potência mínima desejada $\pi = 1 - \beta = 0,80$ para tamanho de efeito $d = d^*$.

Coleta de Dados

A classe de funções de interesse para o presente experimento é composta por instâncias de dimensão no domínio $2 \leq D \leq 150$. De modo a permitir que o Teorema Central do Limite (TCL) seja evocado, se necessário, cada algoritmo foi executado 33 vezes para cada instância do problema. Supondo todas as instâncias no intervalo desejado, ao todo foram cerca de $n_{\text{algoritmos}} \times n_{\text{instâncias}} \times n_{\text{execuções por instância}} = 2 \times 149 \times 33 = 9834$ execuções. Essa experimentação exaustiva foi realizada por uma máquina Intel(R) Xeon(R) Gold 6140 de 18 núcleos e 36 threads operando a 2.30 GHz, 256 GB de RAM, RAID 5 com SSDs de 1TB e Placa de Vídeo Nvidia Quadro P5000 com 2560 cores CUDA e 16GB GDDR5X.

Embora para este estudo de caso o orçamento computacional não tenha sido um gargalo insuperável, problemas de engenharia, cuja solução emprega heurísticas para otimizar modelos numéricos, podem requerer uma limitação no número de instâncias. Assim, diferenças no desempenho médio que excedem algum limite mínimo de relevância prática d^* ainda podem ser alcançados em um subconjunto das instâncias disponíveis, a um custo computacional muito menor do que o que seria necessário para a investigação completa [2].

O método `calc_instances` do pacote `CAISEr` em linguagem R permite estimar o número de instâncias mínimas necessárias para se comparar múltiplos algoritmos, de modo que os requisitos experimentais α, d^* e π sejam atingidos [3]. O parâmetro número de comparações `ncomparisons` é dado pela Equação (3):

$$\texttt{ncomparisons} = \frac{K \times (K-1)}{2} \tag{3}$$

onde K é o número de algoritmos que se deseja comparar. Para o estudo em questão, $K = 2$ e, portanto, $\frac{2\times(2-1)}{2} = 1$.

```
out <- calc_instances(ncomparisons = 1,
                      d = 0.5,
                      power = 0.80,
                      sig.level = 0.05,
                      alternative.side = "one.sided",
                      power.target = "mean")
cat('Número de instâncias necessárias:', out$ninstances)
```

```
## Número de instâncias necessárias: 27
```

O número de instâncias necessárias para se realizar o experimento descrito é de apenas 27 instâncias, ou seja, 1782 execuções no total, o que demanda cerca de $81,88\%$ a menos de processamento computacional que a experimentação integral. Desse modo, as instâncias seriam amostras igualmente espaçadas no domínio $D \in [2, 150]$.

```
options(width = 90)
round(linspace(2, 150, out$ninstances), digits = 0)
```

```
##   [1]   2   8  13  19  25  30  36  42  48  53  59  65  70  76  82  87  93  99 104 110 116
## [22] 122 127 133 139 144 150
```

Análise Exploratória de Dados

Diante do demasiado número de instâncias do problema, ainda que sejam consideradas apenas as instâncias mínimas definidas anteriormente, analisar estatísticas amostrais, como média, mediana e desvio, par a par, com o intuito de gerar algumas suposições sobre potenciais diferenças nos algoritmos se torna impraticável. Portanto, a fim de compreender melhor os dados em estudo e, posteriormente, inferir sobre as populações de onde as amostras provêm, serão analisadas algumas representações gráficas.

No que se refere ao gráfico de *fitness* médio por instância do problema, é possível evidenciar que, para um intervalo visualmente estimado de $10 < D < 60$, a média de desempenho do Algoritmo 1 é substancialmente pior que a média de desempenho do Algoritmo 2. À medida que a dimensão do problema cresce, as diferenças

entre os valores médios já não são mais perceptíveis. Essa verificação ressalta ainda mais a importância de se realizar uma amostragem uniformemente distribuída de instâncias no intervalo de interesse. Caso contrário, as análises podem ser completamente enviesadas. No presente estudo, o Algoritmo 2, ao que tudo indica, apresenta uma melhor performance para dimensões relativamente menores. Se porventura o subconjunto de instâncias abrangesse apenas problemas de baixa dimensão, por exemplo, as conclusões tiradas seriam polarizadas.

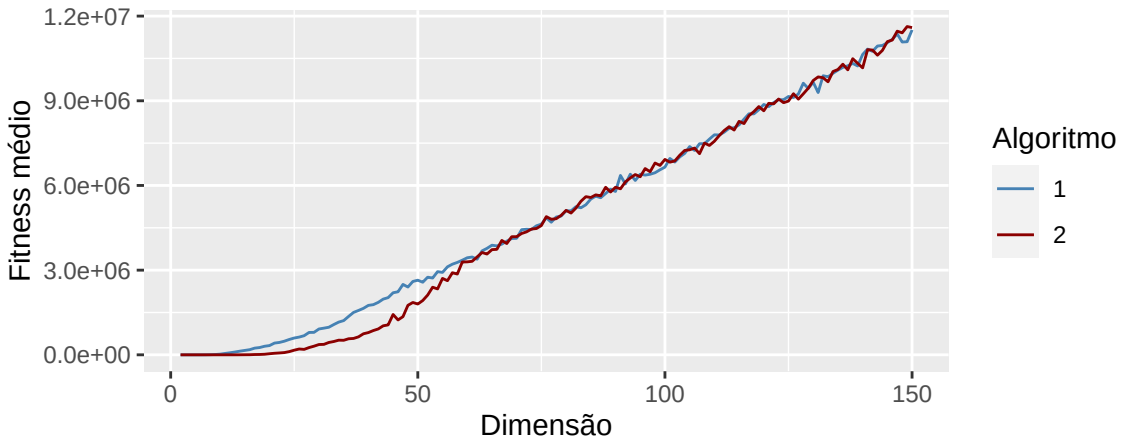

Figura 2: Média de desempenho dos algoritmos em todas as 149 instâncias.

Com a finalidade de verificar o comportamento dos algoritmos nas primeiras instâncias, foram gerados diagramas de caixa par a par no intervalo $2 \leq D \leq 11$. As diferenças evidentes no gráfico anterior têm origem na instância $D = 8$, onde já se pode perceber uma mediana maior do Algoritmo 1 em relação ao Algoritmo 2, bem como uma tendência de crescimento apenas do Algoritmo 1 nas instâncias seguintes. Além disso, a dispersão da primeira configuração apresenta um comportamento com princípio exponencial ($D \geq 8$), enquanto que para a segunda configuração não é possível visualizar o intervalo interquatilico entre o terceiro e o primeiro quartil para o mesmo domínio.

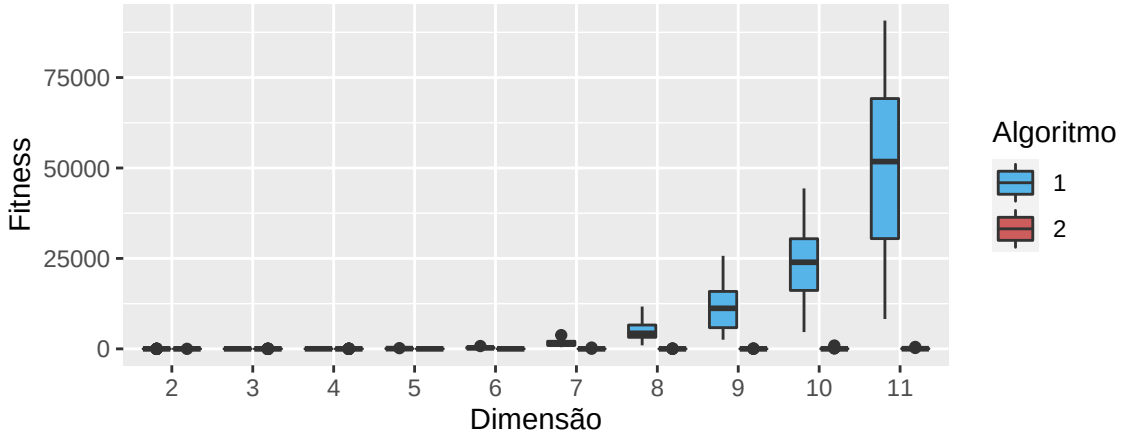

Figura 3: Boxplot das dez primeiras instâncias.

Analogamente, os diagramas de caixa par a par das últimas instâncias também foram produzidos. Conforme mencionado anteriormente para o gráfico de *fitness* médio por instância, as diferenças nos desempenhos dos algoritmos se tornam imperceptíveis em dimensões maiores. Nas instâncias $D = 141, 142, 148$ e 149, por exemplo, a mediana do Algoritmo 1 se encontra inferior a mediana do Algoritmo 2. Em contrapartida, nas

instâncias $D = 143$ e 144, se observa o inverso. Por fim, não se pode concluir nada a respeito dos segundos quartis nas instâncias $D = 145$ e 146. Essa incerteza acerca das comparações entre as duas configurações propostas do algoritmo de Evolução Diferencial, torna ainda mais clara a necessidade da análise estatística para o estudo de caso que se segue.

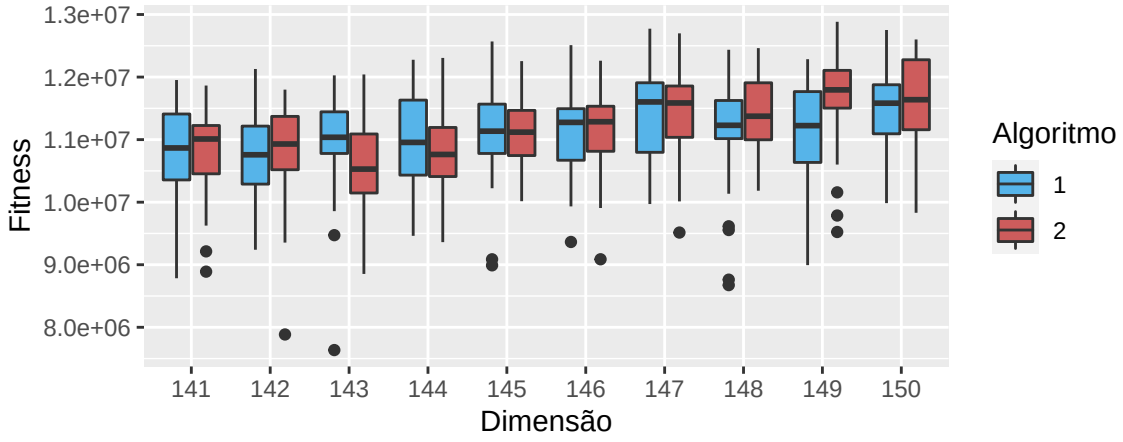

Figura 4: Boxplot das dez últimas instâncias.

Potência do Teste

Tendo em vista que a análise estatística do presente trabalho será realizada sobre a experimentação completa, isto é, as 149 instâncias do problema ao invés apenas da quantidade mínima calculada, a potência obtida no teste sofrerá uma mudança e, portanto, deve ser calculada para o novo tamanho amostral. Dado que o tamanho de efeito não é afetado pelo tamanho da amostra ($d^* = 0,5$), tem-se a seguinte relação entre potência do teste e número de instâncias para um nível de significância constante e igual a $\alpha = 0,05$.

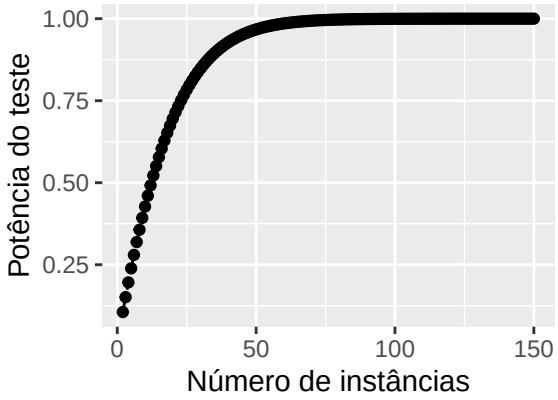

Figura 5: Relação entre o poder do teste e o número de instâncias.

Aumentar o número de instâncias torna o teste de hipótese mais sensível, isto é, mais provável de rejeitar a hipótese nula quando ela é, de fato, falsa. Em outras palavras, a probabilidade de cometer um erro do tipo II (β) diminui à medida que o tamanho amostral aumenta. Como a potência do teste é o complemento do erro do tipo II ($\pi = 1 - \beta$), então consequentemente ela aumenta. Desse modo, tem-se uma potência de $99,99\%$ ao se considerar 149 instâncias do problema com os demais parâmetros experimentais fixos.

```r
out <- calc_instances(ncomparisons = 1,
                      d = 0.5,
                      ninstances = 149,
                      sig.level = 0.05,
                      alternative.side = "one.sided",
                      power.target = "mean")
cat('Potência alcançada:', out$power)
```

```
## Potência alcançada: 0.9999953
```

Validação de Premissas

Análise Estatística

Conclusões

Discussão de Melhorias

Atividades Desempenhadas

Referências

[1] Felipe Campelo. Modular Differential Evolution for Experimenting with Operators. https://www.rdocumentation.org/packages/ExpDE/versions/0.1.2, 2016. Version 0.1.2.

[2] Felipe Campelo and Fernanda Takahashi. Sample Size Estimation for Power and Accuracy in the Experimental Comparison of Algorithms. *Journal of Heuristics*, 25(2):305–338, 2019.

[3] Felipe Campelo, Fernanda Takahashi, and Elizabeth Wanner. CAISEr: Comparing Algorithms with Iterative Sample-size Estimation in R. https://www.rdocumentation.org/packages/lmtest/versions/0.9-37/topics/dwtest. Documentation reproduced from package CAISEr, version 1.0.16.