Estudo Comparativo entre Meta-heurísticas para Otimização

Leonardo Ferrari Soares*, Guilherme Palermo Coelho
Faculdade de Tecnologia (FT) - Universidade Estadual de Campinas (Unicamp)
* l201164@dac.unicamp.br

Resumo - Neste trabalho, a comparação entre dois algoritmos evolutivos de um framework voltado para a otimização é proposta. O problema escolhido a ser solucionado é a função de Rastrigin, um problema de teste muito empregado na literatura de otimização. De alguns frameworks levantados nesse estudo, o *pymoo* foi selecionado tendo em vista o critério da facilidade de utilização e aplicação frente aos outros. E, dentre as meta-heurísticas disponíveis para uso no *pymoo*, foram selecionados o Algoritmo Genético (GA) e o Algoritmo de Evolução Diferencial (DE). Os resultados obtidos mostraram que, na maioria das repetições, os dois algoritmos convergiram para o ótimo global do problema. No entanto, em algumas repetições o GA apresentou uma convergência prematura para ótimos locais.

Palavras-chave: *Otimização; Meta-heurísticas; Computação Evolutiva.*

Introdução

A otimização está fundamentalmente inserida em inúmeras esferas de trabalho, tais como na determinação de quais os melhores produtos a serem fabricados para maximizar o lucro de uma fábrica até a escolha das melhores rotas em redes, de forma a minimizar o atraso de pacotes.

Muitos dos problemas de otimização do mundo real são não-lineares e multimodais, além de poderem apresentar restrições que devem ser consideradas na busca por uma solução. Nesse contexto, diversas técnicas podem ser aplicadas a estes problemas, com diferentes resultados.

Neste trabalho buscamos estudar conceitos de otimização, identificar as principais bibliotecas e *frameworks* de otimização disponíveis na linguagem Python e realizar uma comparação de desempenho entre meta-heurísticas. Nesse estudo comparativo optou-se pela utilização de um problema de *benchmark* clássico de otimização da literatura, conhecido como *Rastrigin* (ÁVILA, 2002).

Fundamentação Teórica

Quando falamos de otimização, nos referimos ao ato de buscar a melhor solução possível para um problema, que atenda a determinadas condições pré-definidas. A qualidade de cada solução depende do problema sendo estudado e geralmente é descrita através de uma *função objetivo*. Tal função objetivo deverá ser minimizada ou maximizada durante a otimização, através do ajuste do conjunto de variáveis do problema. Além disso, eventualmente tais variáveis devem atender a um conjunto de restrições, que definem o domínio para o problema.

Muitos dos problemas reais de otimização pertencem à categoria de problemas NP-difíceis, enquanto que outros nem permitem uma modelagem matemática completa de sua função-objetivo e restrições. Por esses e outros fatores, muitas vezes acabamos recorrendo a algoritmos genéricos para "atacar" um problema de otimização, conhecidos como meta-heurísticas. Quando falamos de **meta-heurísticas** estamos tratando de técnicas muitas vezes consideradas estado-da-arte para problemas que não possuem algoritmos exatos definidos ou em que a aplicação dos algoritmos clássicos existentes não tiveram êxito (LUKE, 2015). Dentre as meta-heurísticas, neste trabalho foram utilizadas duas: o Algoritmo Genético (GA) e o Algoritmo de Evolução Diferencial (DE).

O **Algoritmo Genético (GA**, do inglês *Genetic Algorithm***)** faz parte de uma classe de métodos probabilísticos, de busca adaptativa, inspirados na Teoria da Evolução das Espécies de Darwin. O GA permite que uma população composta por indivíduos (soluções candidatas) evolua sob regras de seleção específicas, de forma a maximizar sua qualidade, ou seja, seu *fitness* (HAUPT R e HAUPT S, 2004, p. 22).

Já a **Evolução Diferencial** (**DE**, do inglês *Differential Evolution***)**é um algoritmo projetado para resolver problemas de otimização com variáveis em domínios contínuos (STORN e PRICE, 1995). É um otimizador também de base populacional, que utiliza operadores de mutação, cruzamento e seleção para gerar novos indivíduos em busca dos mais adaptados. Geralmente o algoritmo de evolução diferencial é utilizado para problemas de otimização não-lineares, sendo também um algoritmo estocástico, como o GA.

Frameworks para Otimização

Um bom *framework* deve ser simples o suficiente para ser entendido ao mesmo tempo que fornece recursos para que possa ser usado rapidamente, conectando os recursos que provavelmente mudarão (ROBERT e JOHNSON, 1997). Ao trabalhar com *frameworks*, buscamos reduzir o custo e o tempo de desenvolvimento de um projeto, facilitando a reutilização de código de algoritmos existentes e a compreensão do comportamento de técnicas existentes. Neste trabalho, o uso destes *frameworks* está voltado para um domínio específico, o de otimização, mais especificamente para meta-heurísticas.

Antes da definição de quais seriam os *frameworks* adotados no projeto, foi feito um levantamento e breve estudo de *frameworks* e/ou bibliotecas já existentes, voltados para otimização. Dentre os principais encontrados estão **jMetalPy**[1], **Inspired**[2], **PyGMO**[3], **DEAP**[4] e **pymoo**[5]. Dentre eles, destaca-se aqui o **pymoo** e o **DEAP** que, proporcionam uma facilidade maior de uso e flexibilidade frente ao uso dos algoritmos e parâmetros. Logo, neste trabalho a utilização dos algoritmos supracitados se deu através do **pymoo**, *framework* de otimização disponível para a linguagem Python.

O **pymoo** (BLANK, 2019) disponibiliza algoritmos de otimização mono-objetivo e multi-objetivo, além de muitos outros recursos tais como visualização e apoio à tomada de decisão.

[1] Mais informações: https://jmetalpy.readthedocs.io/
[2] Mais informações: https://pythonhosted.org/inspyred/overview.html
[3] Mais informações: https://esa.github.io/pygmo/index.html
[4] Mais informações: http://deap.readthedocs.io
[5] Mais informações: http://pymoo.org/

Metodologia Experimental

Para realização dos estudos comparativos entre GA e DE, a função de **Rastrigin** foi utilizada. Essa função é definida por:

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)]$$

$$-5.12 \leq x_i \leq 5.12 \quad i = 1, \ldots, n$$

onde *n* é a dimensão do espaço de busca em que a função é definida (correspondente ao número de variáveis do problema). O mínimo global, obtido analiticamente, é $x_i^* = 0$. No exemplo aqui estudado, a função Rastrigin foi formada por duas variáveis ($n = 2$) definidas no intervalo [-5,12; 5,12].

Para obtenção de resultados, uma série de etapas foram definidas: começamos com a implementação da função-objetivo (Rastrigin) e utilizamos o **pymoo** para a aplicação dos dois algoritmos (GA e DE).

Primeiramente, definimos, para os dois algoritmos, o critério de parada em 100 gerações e o tamanho da população inicial em 100 indivíduos (soluções-candidatas). Logo em seguida, escolhemos os seguintes operadores para GA e DE:

- A **população inicial** foi gerada pelo método chamado "*sampling*", que gera 100 cromossomos (vetores) com dois valores reais (número de variáveis do problema) aleatórios pertencentes ao domínio da função ([-5,12 ; 5,12]);
- Em seguida, o segundo operador utilizado foi o "*selection*", que consiste em um processo de **seleção** dos pais para reprodução. Este operador escolhe aleatoriamente soluções da população atual para serem usadas na recombinação (*crossover*).
- O terceiro operador utilizado foi "*mutation*", que realiza **mutação** polinomial (DEB e GOYAL, 1996), com uma probabilidade de 100%.
- O último operador utilizado no GA foi o "*crossover*", que implementa o SBX (DEB, 1995) como operador de **recombinação**.
- Já os parâmetros utilizados no DE mudaram um pouco. Aqui temos uma probabilidade de 50% de acontecer a troca de valores via "*crossover*". "*Sampling*" permanece o mesmo. O último parâmetro utilizado foi "*variant*", que indica como o operador cruzamento é aplicado. Pode ser de forma aleatória, exponencial (*exp*) ou binomial (*bin*). Neste algoritmo foi empregada a distribuição binomial aleatória, "rand/bin".

Resultados Experimentais

Após definidos os parâmetros e operadores de cada algoritmo, foram feitas 10 repetições dos experimentos para cada um deles. Para a comparação entre essas repetições foi gerado um *box plot* (Figura 1). Nele podemos ver que tanto o Algoritmo Genético quanto o algoritmo de

Evolução Diferencial apresentaram *outliers* nos experimentos, que são pontos extrapolados dos valores esperados. Com o gráfico é possível ver também que, na maioria das repetições, os dois algoritmos convergiram para o ótimo global, sem diferenças muito grandes. Porém o GA apresentou outliers mais distantes da mediana ao longo das execuções, indicando a convergência prematura para ótimos locais em alguns experimentos. Diante disso pode-se dizer que os resultados obtidos pelo DE são mais consistentes.

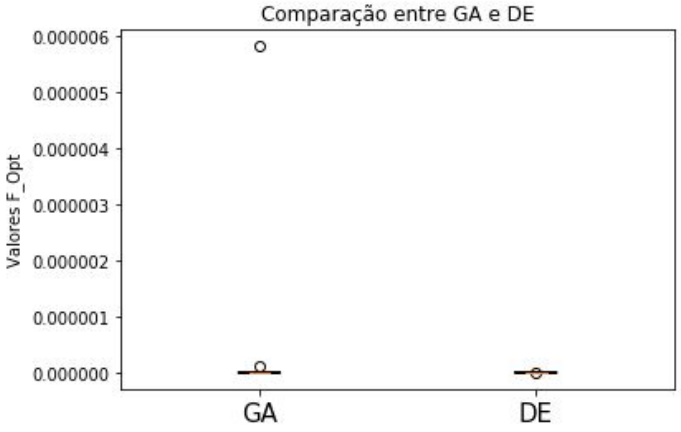

Figura 1 - *Box-plots* dos resultados obtidos pelos algoritmos GA e DE após 10 repetições dos experimentos utilizando a função de Rastrigin. Fonte: Autoria própria.

Além disso, para verificar se os algoritmos implementados estavam efetivamente convergindo com os parâmetros adotados, foram avaliadas as evoluções das populações ao longo das 100 gerações. Na Figura 2 são apresentadas as evoluções (para uma das repetições dos experimentos), tanto da melhor solução encontrada quanto da média das soluções candidatas, para os algoritmos DE e GA. Como é possível observar, ambos os algoritmos apresentaram convergência ao longo da busca.

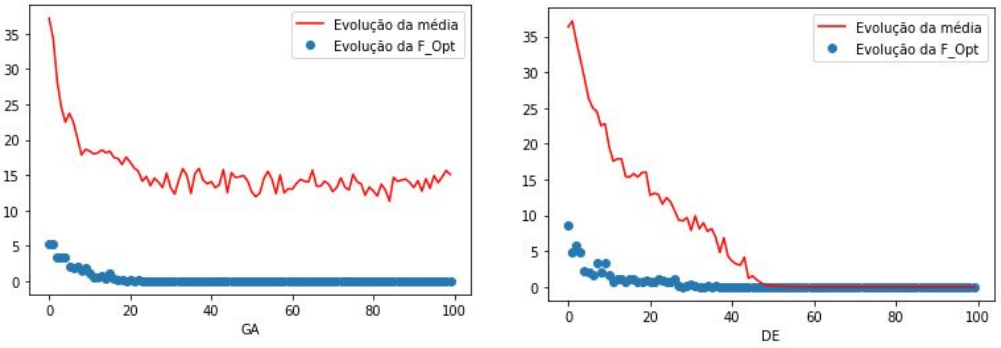

Figura 2 – Análise comparativa entre uma repetição dos dois algoritmos, GA e DE respectivamente. A análise deixa explícita a evolução da média e dos melhores resultados ao longo das gerações. Fonte: Autoria própria.

Conclusões

Neste trabalho comparou-se o desempenho de duas técnicas de otimização em um problema multimodal de *benchmarking*. Ambos os algoritmos geraram bons resultados,

principalmente em relação à convergência ao valor ótimo da função objetivo. Para os operadores descritos, observou-se que o DE acabou apresentando uma melhor evolução da busca em todas as repetições dos experimentos, diferentemente do GA. Como trabalhos futuros, recomenda-se a expansão do trabalho de forma a posicionar e comparar esses algoritmos com outras meta-heurísticas baseadas na computação evolutiva. Além disso, pode-se repetir o que foi feito no *pymoo* com outro *framework* disponível na literatura, com os mesmos algoritmos selecionados, de forma a comparar a facilidade na implementação, alteração e robustez dos *frameworks*.

Referências

ÁVILA, L, S. **Algoritmos Genéticos Aplicados na Otimização de Antenas Refletoras**. 98p. Dissertação de Mestrado - Universidade Federal de Santa Catarina, Santa Catarina, 2002.

BLANK, J. **pymoo - Multi-Objective Optimization Framework.** Versão 0.3.2., 2019. Página inicial. Disponível em: https://pymoo.org/. Acesso em: 13 de novembro de 2019.

DEB, K.; AGRAWAL, R. B. **Simulated Binary Crossover for Continuous Search Space**, Complex Systems, Vol. 9, No. 2, 1995, p. 115-148.

HAUPT, R., HAUPT, S. **Practical Genetic Algorithms**. 2nd. ed. Canada: John Wiley & Sons, Inc. 2004.

LUKE, S. **Essentials of Metaheuristics.** 2015**.** Disponível em: https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf. Acesso em: 27 de dezembro de 2019.

STORN, R.; PRICE, K. **Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces**. Em: Technical Report TR-95-012. Berkley: International Computer Science Institute; 1995.

ROBERTS, D.; JOHNSON, R. **Evolving frameworks: A pattern language for developing object-oriented frameworks**. Em: Pattern Languages of Program Design 3. Addison Wesley, 1997.