TP2 META - ALGORITMO GENÉTICO

META-HEURÍSTICAS - CCF480

Bruno Marra
Matrícula 3029
Aluno de Ciência da Computação
Universidade Federal de Viçosa - Florestal, MG
bruno.marra@ufv.br

Victor Hugo
Matrícula 3510
Aluno de Ciência da Computação
Universidade Federal de Viçosa - Florestal, MG
victor.h.santos@ufv.br

25 de setembro de 2021

RESUMO

Este relatório tem como objetivo explorar o uso de uma meta-heurística clássica, o Algoritmo genético. Dessa forma, será explorado tanto a implementação do algoritmo, bem como o uso para resolução de dois problemas distintos, fornecidos pela especificação. Será ainda discutido a forma como cada um deles foi implementado, parâmetros, e os resultados obtidos para o mesmo.

1 INTRODUÇÃO

Após, em ambiente virtual de aula, conhecer a meta-heurística **Algoritmo Genético**, o professor Marcus Henrique Soares Mendes propôs a implementação do algoritmo em **representação real** para minimizar as seguintes funções:

$$G6(x) = (x_1 - 10)^3 + (x_2 - 20)^3 \text{ e}$$

$$\tilde{F}(P_i) = a_i {P_i}^2 + b_i P_i + c_i + |e_i \text{sen}(f_i({P_i}^{min} - v_i))|$$

Vale lembrar que cada função objetivo acompanha algumas restrições, que serão melhores descritas no desenvolvimento deste relatório.

Os algoritmos genéticos foram inspirados no mecanismo da evolução das espécies, tendo como base os trabalhos de Darwin e Mendel. Tais algoritmos vêm sendo utilizados com sucesso para a resolução dos mais variados e complexos tipos de problemas. Isso é possível devido a sua estrutura genérica, que faz com que os Algoritmos Genéticos, ao contrário de outras meta-heurísticas, sejam aplicáveis aos mais variados problemas de otimização. Algoritmos Genéticos vêm sendo usados com sucesso para encontrar boas soluções desde sua introdução por Holland na década de 70. (LOPES, 2013).

2 DESENVOLVIMENTO

Para elaborar o desenvolvimento do algoritmo, a linguagem escolhida foi o **Python**. A implementação do algoritmo foi baseada na implementação feita por Jason Brownlee[1], de forma a resolver ambos os problemas. Apesar do algoritmo ser baseado nessa implementação, algumas variações importantes foram feitas para viabilizar a entrega deste trabalho prático, principalmente o fato da implementação original ser em codificação binária. Nas subseções a seguir serão descritas as decisões de projeto e também as alterações feitas em cada etapa do algoritmo genético: **População Inicial, Avaliação da Fitness, Seleção, Cruzamento, Mutação, Critério de Parada e tratamento de restrições**.

2.1 População Inicial

A etapa de geração da população inicial foi implementada do zero, visto que a implementação original foi feita em codificação binária.

[1] Acesso em: https://machinelearningmastery.com/simple-genetic-algorithm-from-scratch-in-python/

Inicialmente gera-se um indivíduo de forma aleatória e a respeitar os limites mínimos e máximos de cada variável de decisão, vale lembrar que, nesta parte, a solução gerada não é necessariamente factível, pois não considera as restrições. Esta etapa é feita *n_pop* vezes, sendo *n_pop* = número de indivíduos, ou seja, a população. Vale lembrar que a quantidade de indivíduos é um parâmetro extremamente importante para fazer ou não a solução convergir corretamente para o ótimo global. Uma população com poucos indivíduos tem grandes chances de convergir prematuramente para um ótimo local, enquanto uma população com muitos indivíduos pode tornar o tempo de execução inviável.

2.2 Avaliação da Fitness e Tratamento de Restrições

A avaliação da Fitness ocorre após cada geração de indivíduos, ou seja, após a geração inicial e após a etapa de cruzamento e mutação. É nela que cada um dos indivíduos são avaliados.

Inicialmente verifica-se se o indivíduo corrente é factível, ou seja, se além de respeitar os limites para as variáveis de decisão, ele também respeita as restrições de cada problema. As restrições do problema 1 são:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

$$13 \leq x_1 \leq 100$$

$$1 \leq x_2 \leq 100$$

Para o problema 1 e conforme os testes feitos durante execução, as restrições não precisaram ser tratadas conforme aprendido em aula. Provavelmente devido ao fato de se ter poucas variáveis de decisão e também pelas restrições serem simples. Caso o indivíduo seja infactível, é retornado um valor extremamente alto para este ser descartado na etapa de seleção.

As restrições do problema 2 são:

$$\sum_{i=1}^{n} P_i - P_L - P_D = 0, \text{ sendo } P_L = 1800 \text{ e } P_D = 0$$

$$P_i^{min} \leq P_i \leq P_i^{max}$$

Diferente do problema 1, as restrições do problema 2 precisaram e foram tratadas por uma penalidade estática de 0.5. Tal tratamento foi necessário devido ao fato de se ter várias variáveis de decisão. Encontrar valores exatos para satisfazer os limites de cada variável e ainda, a primeira restrição descrita anteriormente, está longe de ser viável.

2.3 Seleção

O procedimento de seleção implementado segue a ideia de seleção por torneio, considerando um torneio de *k = 3* indivíduos. Como aprendido em aula, o parâmetro *k = 3* é um valor que apresenta bons valores na prática. Nesta etapa, a implementação original não foi alterada.

2.4 Cruzamento

Assim como na etapa de seleção, a implementação original não foi modificada no cruzamento. Por mais que a implementação original seja em codificação binária, o método proposto também foi útil em codificação real. Nesta etapa, o método de cruzamento utilizado foi o cruzamento simples, que é a versão real do cruzamento de um ponto de corte binário. A probabilidade do cruzamento ocorrer foi definida em 90%.

2.5 Mutação

A mutação, como se sabe, apresenta formas distintas para cada uma das possíveis representações, ou seja, esta etapa também precisou ser implementada do zero. A mutação escolhida foi a mutação uniforme, que acrescenta, à cada variável de decisão, uma determinada perturbação aleatória. A probabilidade da mutação ocorrer foi definida em 1%.

2.6 Critério de Parada

O critério de parada utilizado foi o número de iterações (*n_iter*) realizado a cada execução. Este é um parâmetro extremamente importante para se considerar, pois determina com significância o quanto as soluções obtidas irão convergir para o ótimo global.

2.7 Resultados

Para testar o algoritmo implementado, foram definidas duas configurações A e B, sendo:

A:

- *n_pop* = 100
- *n_iter* = 200

B:

- *n_pop* = 500
- *n_iter* = 1000

Sabe-se que existem outros parâmetros a serem alterados e considerados em cada execução, como a probabilidade de mutação e de cruzamento, porém os valores escolhidos são valores que, como aprendido em aula, apresentam bons resultados na prática, em termos de viabilidade e qualidade. Além disso, os parâmetros *n_pop* e *n_iter* são os que mais influenciam na solução final obtida e por isso, foram os escolhidos para definir ambas as configurações.

A Tabela 1 mostra com detalhes quais os valores encontrados para as 30 execuções do algoritmo, seguindo os parâmetros listados anteriormente.

Algoritmo	**Mínimo**	**Máximo**	**Média**	**Desvio-padrão**
AG Configuração A	-7948.48052154997	-7883.737360374542	-7927.010269034697	16.41273376917945
AG Configuração B	-7950.915334639488	-7943.366864614684	-7948.861825911345	1.7798591125667167

Tabela 1: Problema com função objetivo 1

As Figuras 1 e 2 mostra como ficaram o boxplots das 30 execuções para cada configuração.

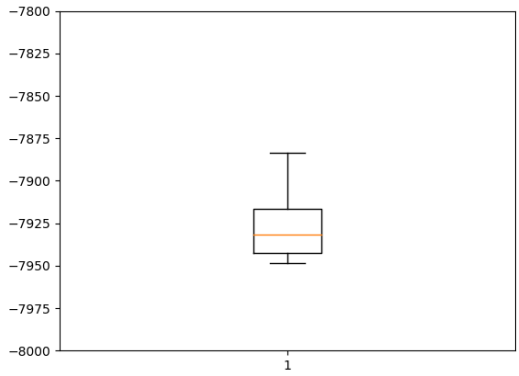

Figura 1: Boxplot - Função 1 e Configuração A

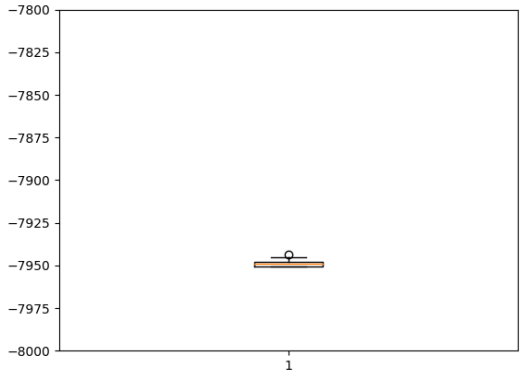

Figura 2: Boxplot - Função 1 e Configuração B

Os valores aproximados de x_1 e x_2 para as execuções que apresentaram menor valor e para cada configuração, respectivamente, são:

- x_1: 13.70707 e 13.66080
- x_2: 0.00048 e 2.0435x10^{-5}

A Tabela 2 mostra com detalhes quais os valores encontrados para as 30 execuções do algoritmo, seguindo os parâmetros da listados anteriormente.

Algoritmo	Mínimo	Máximo	Média	Desvio-padrão
AG Configuração A	18646.08820161271	19447.221248578295	19038.685253612482	168.60202649997134
AG Configuração B	18408.760842238204	18983.640533193633	18684.567863340817	137.87866251767713

Tabela 2: Problema com função objetivo 2

As Figuras 3 e 4 mostra como ficaram o boxplots das 30 execuções para cada configuração.

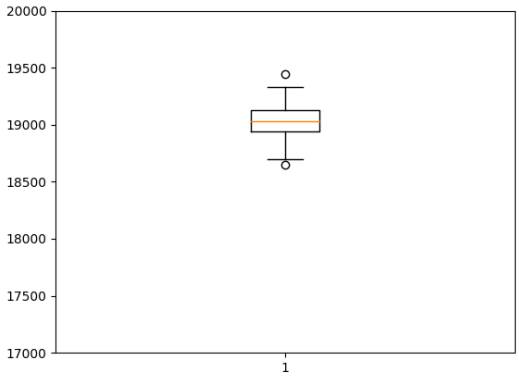

Figura 3: Boxplot - Função 2 e Configuração A

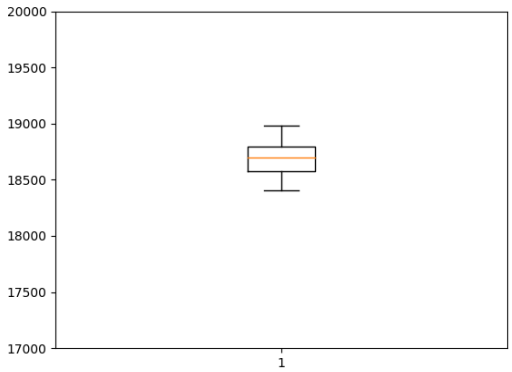

Figura 4: Boxplot - Função 2 e Configuração B

Os valores aproximados das variáveis de decisão para as execuções que apresentaram menor valor e para cada configuração, respectivamente, são:

- P_1: 354.694799 e 362.28775
- P_2: 301.41192 e 299.15794
- P_3: 157.75130 e 225.07221
- P_4: 115.18650 e 158.96395
- P_5: 160.07887 e 71.108885
- P_6: 104.32661 e 111.71746
- P_7: 150.06263 e 96.90443
- P_8: 72.78629 e 104.21
- P_9: 101.3402 e 159.180367
- P_{10}: 44.0738 e 60.13562
- P_{11}: 42.70351803104001 e 40.62851
- P_{12}: 93.00771 e 55.24601
- P_{13}: 102.08752 e 55.10603

3 CONCLUSÃO

Dessa forma, foi possível notar que o algoritmo converge para um valor bem específico quando são aumentados os parâmetros *n_pop* e *n_iter*. Tais valores são aproximadamente -7950 para a função objetivo 1 e configuração B e 18600 para a função objetivo 2 e configuração B (valores exatos nas tabelas 1 e 2).

A convergência para tais valores nos diz que o valor ótimo para cada função gira em torno destes e que, à medida que os parâmetros são aumentados, a convergência será maior e mais precisa, visto que há uma clara diminuição nos desvios

padrões. Para este trabalho prático, os parâmetros não foram tão aumentados devido ao tempo de execução aumentar consideravelmente. Uma possível configuração foi: $n\_pop = n\_iter = 1000$, na qual diminui consideravelmente o desvio padrão da execução da questões 1 e 2, porém, tal configuração não foi proposta pelo tempo de processamento ter sido muito grande e também pelo valor ótimo não estar tão distante dos obtidos com as configurações A e B, mas vale lembrar que é uma possibilidade caso queira uma solução mais precisa. Foi possível verificar com o aumento, principalmente da população, que os indivíduos irão convergir para o ótimo global e assim apresentar uma solução favorável.

Além de todas as conclusões observadas, foi um exercício importante para entender na prática como funciona os algoritmos genéticos, bem como sua implementação direta, podendo ser modificada para melhorar os resultados em caso de problemas específicos.

Referências

LOPES, H. *Meta-Heurísticas em Pesquisa Operacional*. Curitiba, PR: omnipax, 2013. 1