Resolução de Coloração de Vértices com GA

TOMÁS SILVA QUEIROGA *

*Ciência da Computação - Graduação
E-mail: tomasqueiroga@gmail.com

***Resumo* – Este trabalho teve como objetivo mostrar a capacidade de resolver problemas usando algoritmos genéticos, em particular coloração de vértices, observando diferentes parâmetros do algoritmo. Com o solucionador de problemas calibrado, isto é, otimizando o valor de tamanho da população, taxa de mutação e de reprodução, essa abordagem se mostrou eficaz para resolver o problema, e os demais parâmetros não se demostram cruciais para encontrar ou não a solução.**

***Palavras-chave* – Teoria de grafos, coloração de vértices, problema NP-Completo, inteligência artificial, algoritmo genético, Julia**

I. INTRODUÇÃO

Em Teoria dos grafos, coloração de vértices de um dado grafo é o problema de atribuir uma cor para cada vértice de forma que vértices vizinhos sempre tenham cores diferentes. O problema aqui tratado é a coloração de vértices mínima, ou seja, descobrir o número mínimo de cores que satisfaça a restrição citada, é um problema é NP-Completo[1].

Uma das naturezas do problema que o leva parecer NP-Completo é o número exponencial de combinações possíveis de cores (mesmo que erradas) que os vértices podem assumir, e poucos indícios de se a coloração testada será correta até que seja colorido um número "grande" de vértices se o grafo for "grande" sendo inviável testar todas. Dado este cenário, utilizar algoritmo genético (GA) irá reduzir o número de colorações testadas, e tornar tratável o problema.

Existem diversas formas de implementar soluções de GA, algumas mais verbosas do que outras, e a linguagem de programação Julia é próxima de modelos matemáticos, fácil de se aprender e programar, além de rápida como linguagens mais tradicionais como Java e C.

II. TRABALHO PROPOSTO

O presente trabalho se propõe a implementar em Julia um solucionador de problemas genéricos utilizando algoritmos genéticos e resolver o problema de coloração de vértices.

III. MÉTODOS

Para ser alcançado os resultados desejados foram executados os seguintes passos:

1) Estudou-se a linguagem de programação Julia
2) Criou-se um repositório online para guardar o código utilizado, bem como este relatório
3) Implementou-se um solocionador básico de problemas genéricos com algoritmo genético
4) Resolveu-se o problema N-Rainhas com ele, com ajuda dos slides apresentados em sala de aula, adaptando o solucionador
5) Resolveu-se o problema do caxeiro viajante, adaptando o solucionador
6) Resolveu-se o problema de encontrar um caminho que resolva um labirinto, realizando as adaptações finais ao solucionador
7) Resolveu-se o problema da mochila, um problema NP simples
8) Resolveu-se o problema de coloração de vértices.

Foram feitos diversos passos antes de se resolver o problema principal porque tratar problemas NP-Difíceis não é uma tarefa fácil. E quis-se ter certeza de que, caso falha-se solucionar o problema com GA (o que não ocorreu) era interessante saber se o problema estava na formulação do problema como GA, ou no solucionador.

Quando diz-se "resolveu-se", quer-se dizer que foram testadas algumas instâncias do problema, e que todas chegaram a resultados esperados.

Para alcançar o último passo com sucesso, motivo deste trabalho, foi necessário obter uma boa formulação do problema em termos de algoritmos genéticos, ou seja definir uma função fitness e o que é um indivíduo da população.

O indivíduo é um vetor de tamanho v, onde v é o número de vértices do grafo, e cada posição i do vetor contém a cor atribuída ao vértice i do grafo. Sendo que cada cor é um número inteiro, de 1 a G, onde G é igual ao maior grau que algum vértice possúi no grafo. Assim, todo indivíduo é uma atribuição de cores aos vértices do grafo, não necessariamente correta, e o quão correta é solução que o indivíduo representa será dado pela função fitness.

O problema foi tratado como de minimização, então, quanto menor o valor da função fitness melhor, sendo que o valor fitness para cada indivíduo é dado por: $2|c - G| - m$, onde c é o número de cores utilizadas, G o número de cores ideal, m número de vértices adjacentes com mesma cor. Dessa forma, é mais valorizado individuos com menos cores, porém também considerarando os erros que ele fez para atingir essa quantidade mais baixa de cores. Note que ao usar o menor número de cores possíveis G, é possível que a função fitness retorne zero.

Utilizou-se de eletismo - o melhor indivíduo da geração g_k é colocado na geração g_{k+1} - normalização do ranking dos indivíduos, e seleção de pais com o método da roleta, conforme visto em sala de aula.

Critério de parada utilizado foram dois, excludentes, ou encontrou-se alguma solução cuja função fitness seja 0 ou o mesmo melhor valor de função fitness é encontrado a mais de 5.000 gerações.

A coloração de vértices foi realizada em grafos $G = (V, E)$, onde V é o conjunto de vértices, e E o de arestas, gerados aleatoriamente, com $|V| = 50$ e $|E| = 4|V|$, sendo que não há auto-arestas. Não sendo testado em grafos completos portanto, nem em direcionados.

O solucionador aceita variação no tamanho da população, taxa de mutação, reprodução, pontos de crossover, 1 ou 2, tipo de mutação, novo gene ou swap. Tais parâmetros foram alterados para avaliar a solução do problema.

IV. Resultados e Discussão

Os gráficos contidos nas figuras 1, 2, 3, 4, 5, 6, 7 mostram valores de função fitness do melhor e pior indivíduo e a média da população por geração, em um grafo com grau máximo de vértice 13, para diferentes valores dos parâmetros:

- Tamanho da população p
- Taxa de mutação $m_r \in [0\%, 100\%]$
- Utilizar swap na mutação $m_s \in \{t, f\}$
- Taxa de reprodução $r \in [0\%, 100\%]$
- Pontos de crossover $c \in \{1, 2\}$

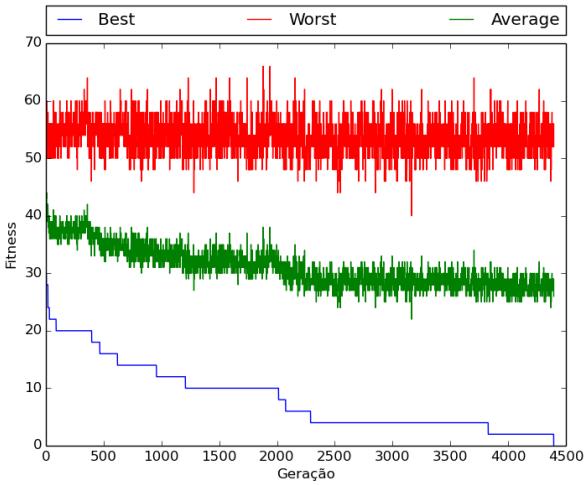

Figura 1. $p = 50$, $m_r = 5\%$, $m_s = t$, $r = 95\%$, $c_p = 1$

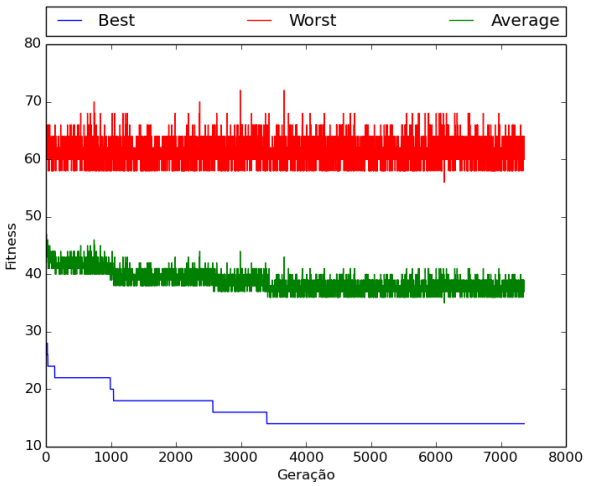

Figura 2. $p = 1000$, $m_r = 5\%$, $m_s = t$, $r = 95\%$, $c_p = 1$

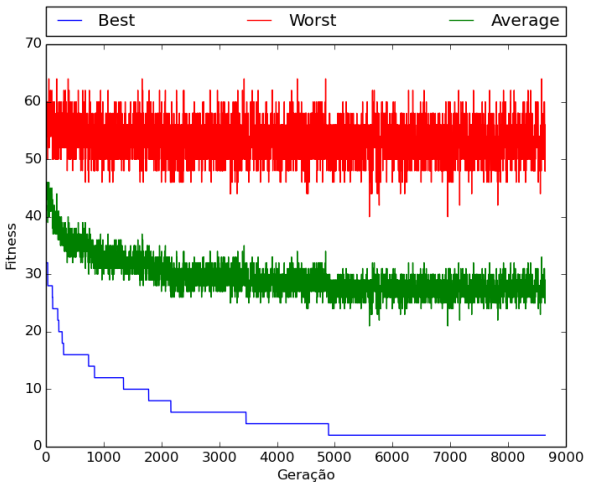

Figura 3. $p = 50$, $m_r = 0.5\%$, $m_s = t$, $r = 98\%$, $c_p = 1$

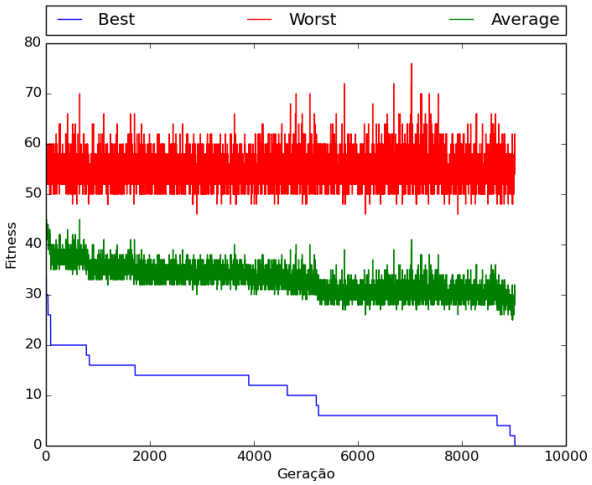

Figura 4. $p = 50$, $m_r = 50\%$, $m_s = t$, $r = 95\%$, $c_p = 1$

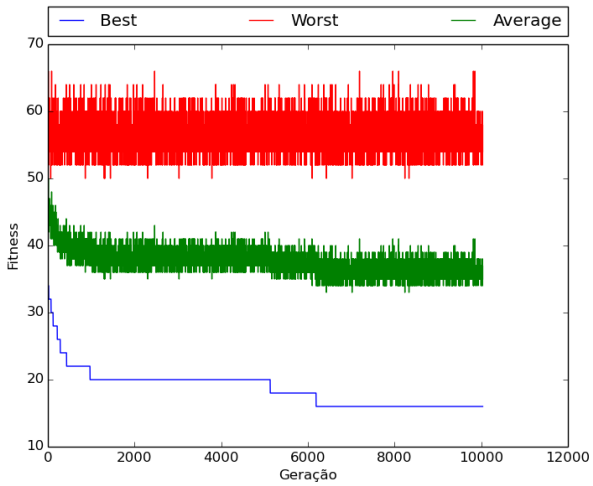

Figura 5. $p = 50$, $m_r = 50\%$, $m_s = t$, $r = 50\%$, $c_p = 1$

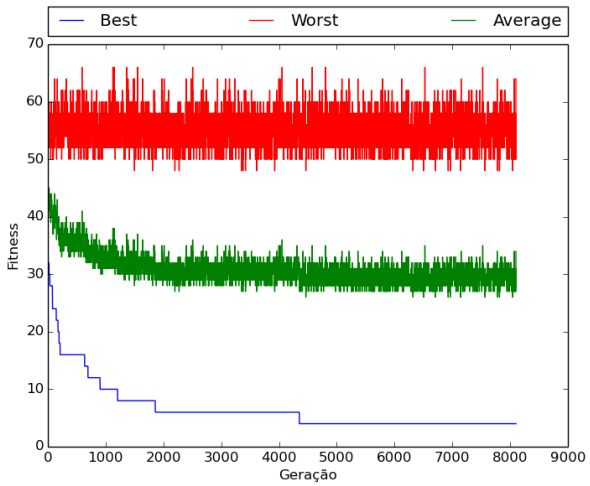

Figura 6. $p = 50$, $m_r = 5\%$, $m_s = t$, $r = 90\%$, $c_p = 2$

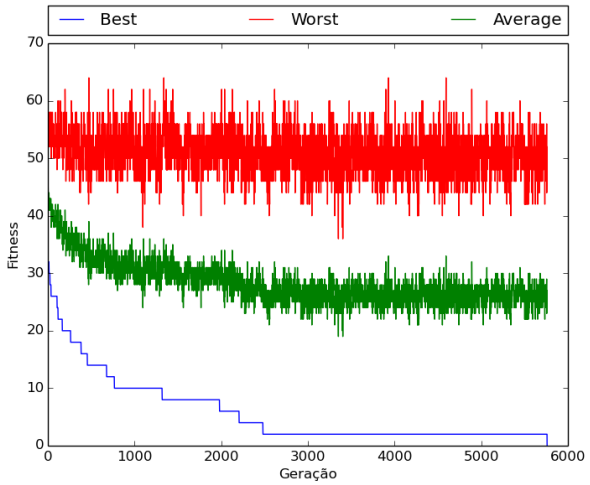

Figura 7. $p = 50$, $m_r = 5\%$, $m_s = t$, $r = 90\%$, $c_p = 2$

Em todos os casos, o valor médio e máximo oscila a cada geração, porém, de alguma forma e velocidade, decai, dando indícios que, talvez, depois de muitas gerações mesmo, tenha-se apenas indivíduos ótimos.

Observa-se que em 3 casos (Figuras 1, 4 e 7) encontrou-se a melhor solução ($fitness = 0$), 13 cores, sem nenhum vértice adjacente de mesma cor. Os demais casos não obtiveram exito. Apesar de cada um encontrar tal solução em gerações diferentes, não há como dizer qual foi mais eficaz, visto que há aleatoriedades no processo, porém a figura 1 apresentou a solução ótima em quase metade das gerações se comparada a figura 4, e antes também da figura 7. O encontro tardio da solução na figura 4 se deu pela alta taxa de mutação, que provavelmente alterava individuos melhores para nem tanto.

Vê-se também que ter uma população muito maior que a entrada (Figura 2) não é uma boa estratégia, não houve grande variação na população, visto que a curva da média se assemelha a grosso modo ao melhor indivíduo, e não se aproximou ao valor ótimo. Pela Figura 5, onde há grande taxa de mutação e baixa de reprodução (note que na Figura 4 possui mesma taxa de mutação, porém de reprodução mais elevada), os resultados foram semelhantes a ter uma superpopulação.

Uma leve mudança para cima na taxa de reprodução (Figura 3) pareceu dar as curvas um decaimento inicial, gerando vários superindivíduos precocemente, diminuido a variedade genética, fazendo com que estagna-se em uma solução não ótima. Neste caso é necessário aumentar também a taxa de mutação, ou adotar outra forma de senão a normalização de ranking, para fugir deste "mínimo local".

Alterar de crossover simples (um ponto) para duplo (dois pontos) abaixando a taxa de reprodução (Figura 6) também não se mostrou eficaz.

Fazer a mutação por swap ou por criar um novo gene se mostrou como algo indiferente, pois a solução fora encontrada em ambos os meios.

Outro resultado importante pode ser conferido neste link, que contém todas as implementações feitas.

V. Conclusões

Utilizar algoritmos genéticos se mostrou ser uma forma eficiente para encontrar soluções de problemas NP-Difíceis e rápida implementação, dado que o solucionador já esteja pronto (como já está, para todos que encontrarem o meu repositório). A linguagem julia se mostrou eficaz também para este tipo de problema, e bastante flexível. E, finalmente, determinar os parâmetros do solucionador ainda parece ser algo que precisa de um pouco de conhecimento sobre o problema, e testes em casos menores antes de se utilizar em instâncias maiores do problema.

Referências

[1] G. SCHMIDT and E. W. MAYR, *GRAPH-THEORETIC CONCEPTS IN COMPUTER SCIENCE*, 1995. 1

[2] E. Colombini, *Notas de aula MC906/MO416*. Unicamp, 2016. 3

[3] O. P. E. N. Source. Julia programming language. [Online]. Available: http://julialang.org/ 3