ALGORITMO GENÉTICO PARA SOLUCÃO DO PROBLEMA DE COLORAÇÃO DE VÉRTICES

DA SILVA, Matheus Strutz Soares; GUEDES, Maycon

(1) Trabalho instruído sobre algoritmo genético da matéria de Tópicos Especiais 2
(2) Cursando na área de Tecnologia de analise e desenvolvimento de sistemas no IFES Campus Santa Teresa

RESUMO

O problema da coloração de vértices é um problema NP-Completo de dificuldade que escara conforme novos vértices e novas arestas são formados em um Grafo. Neste papel, será apresentado um simples algoritmo genético para solução deste problema. O algoritmo descrito aqui, ira usar um seleção por torneio, um cruzamento por mascara, e uma mutação customizada onde só sofrera a mutação um vértice que tenha um adjacente com a mesma cor que ele. O Algoritmo foi testado usando os dados do conhecido grupo DIMACS, onde 12 arquivos foram testados, com as definidas quantidades de cores para cada um deles. O Algoritmo teve alguns resultados satisfatórios porém sem apresentar nenhuma melhora significativa de performance comparado a outros algoritmos já apresentado na DIMACS.

PALAVRAS-CHAVE: Algoritmo genético, problema da coloração de vértices.

INTRODUÇÃO

Estudos realizados na última década mostraram que há implementação de algoritmos genéticos se prova superior na solução de problemas gráficos, especialmente em problemas não determinísticos de tempo polinomial (NP). No entanto, o principal desafio desse modelo é que o tamanho dos dados usados no processamento das informações cresce exponencialmente com o aumento do tamanho do problema. Esse fenômeno é chamado de explosão exponencial e é um dos maiores gargalos que impede a melhora de processamento em diversas áreas da computação.

Nesta seção, apresentamos um simples algoritmo genético para solução de um problema há muito discutido por vários estudiosos da área matemática, que é sobre o problema de coloração de vértice.

METODOLOGIA

Para resolução do problema em questão, foram lidos alguns dos arquivos que contem matrizes de coordenadas disponibilizadas pelo centro de discussão matemático e central de tecnologia, DIMACS, que foi proposto em uma competição em 2014 sobre Graph Partitioning and Graph Clustering realizada pelos mesmos,

Para executar os testes do algoritmo genético, foi usado um computado de mesa com as seguintes configurações: Processador Intel Core i5-8400 Coffee Lake, Cache 9MB, 2.8GHz, Placa-mãe Gigabyte X570 Gaming X, AMD AM4, ATX, DDR4, Memória HyperX Predator RGB, 16GB (2x8GB), 2933MHz, DDR4.

A linguagem de programação utilizada para o desenvolvimento do código foi Java SE Development Kit 8(JDK 8), assim como todo o mesmo está disponível no github.

O algoritmo genético conta com um cromossomo, onde o numero de genes corresponde sempre ao numero de vértices presente na matriz de coordenada que for carrega, e um fitness que e calculado por colisão, ou seja, quantos vértices adjacentes que tenham as mesmas cores, para isso foi aplicado um loop que percorre toda a matriz de coordenada, verificando se a primeira cor do gene na matriz é igual a sua respectiva ligação.

A população é gerada de forma aleatória, pela quantidade de cores que forem pré-definidas, porém, o tamanho da população também é pré-definido, onde após vários testes, uma população entre 50 a 150 indivíduos se mostrou ser o mais efetivo.

A seleção de pais é feita a partir de um torneio, onde uma quantidade de indivíduos é escolhida de forma aleatória da população, e o melhor é o pai. Isto é aplicado duas vezes, para escolher dois pais, porém, mesmo aplicado duas vezes de forma independente, um pai nunca poderá ser igual ao outro.

O cruzamento é feito a partir de mascara, ou seja, basicamente é sorteado de forma aleatória como o emaranhamento de genes vai ser feito, e qual gene de qual pai vai ser colocado no respectivo gene do filho, após o cruzamento ser finalizado, o novo filho tem seu fitness calculado usando o calculo de colisão.

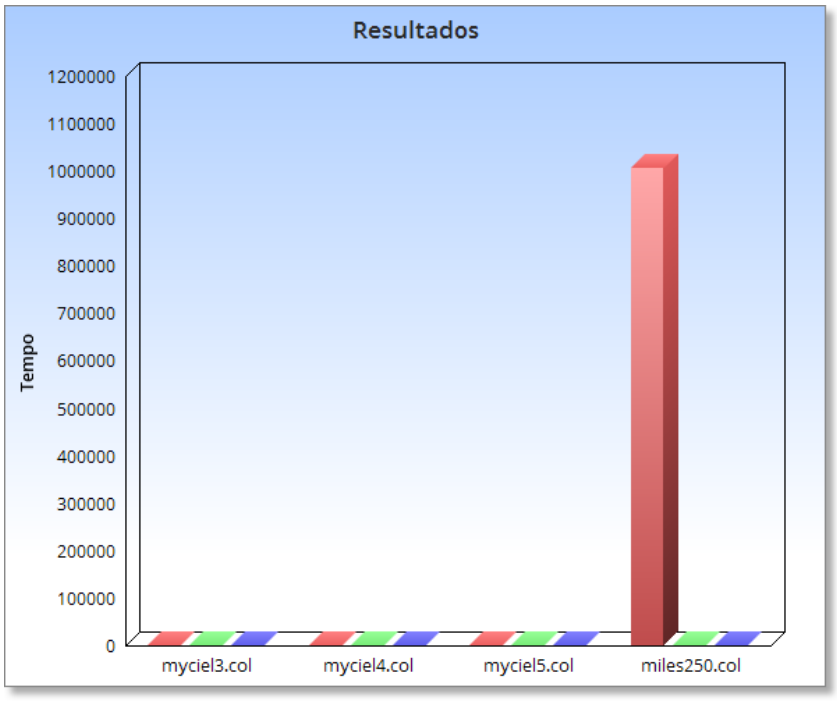
A mutação aplicada neste algoritmo é uma mutação exclusiva, não seguindo nenhum padrão comum de mutação, porém, com a ideia não é nova, e foi retirada de outros artigos. A mutação foi feita com 1% de chance de ocorrer, onde, será escolhido um gene aleatório, porém, um que ocorra uma colisão. Após ter o gene escolhido, e sorteado novamente uma cor, que não possa repetir com a que já esteja nesse gene, assim gerando sempre uma nova cor, e após todo esse processo é calculado novamente o fitness do novo cromossomo que sofreu a mutação. Usando este processo, um cromossomo, após a mutação, nunca poderá ter um fitness pior que o seu antigo.

A inserção é feita usando o pior pai. É verificado se o melhor filho que foi gerado, após todos os passos escritos acima, é melhor, que os melhores pais dos dois gerados pela seleção de torneio, assim, é retirado o pior dentre esses dois da população e inserido o melhor filho, mantendo sempre a população com o tamanho inicial.

RESULTADOS

O Algoritmo se provou bastante eficiente, encontrando os resultados para todos os arquivos que foi proposto. A performance também se mostrou bastante satisfatória, porém sem melhoria alguma em comparação a alguns dos artigos que foram apresentados na competição proposta pela DIMACS.

Assim dizendo, foi selecionado 4 arquivos para comparações principais, com o algoritmo de Musa M. Hindi and Roman V. Yampolskiy(Cor verde).e o algoritmo de Sidi Mohamed Douiri. (Cor azul):



	Myciel3.col	Myciel4.col	Myciel5.col	Miles250,col
Matheus Vemelho	0.030s	0.057s	0.304s	1007.103s
Musa and Roman Verde	0,003s	0.006s	0,014s	0,076s
Sidi Azul	0,001s	0,089s	5.140s	4.390s

FUNDAMENTAÇÃO TEORICA OU DISCUSSÕES

A seleção dos pais usando dois torneios distintos foi uma escolha visando melhor performance a longo prazo, pois dessa forma, mesmo que pequena, a chance de os dois pais terem um fitness um pouco melhor do que dois escolhidos por um mesmo torneio aumenta, então conforme o tempo passo, essa escolha vai se provando mais efetiva.

A inserção por pior pai foi aplicada de uma forma em que só ira ser removido o pior pai da população, se o melhor filho for melhor que o melhor pai, porém, pensando pelo lado do desempenho, retirar somente se isso acontecer, haveriam menos inserções, e como a seleção é feita de forma aleatório e os filhos não entram na população se não forem melhor que o melhor pai, a população ira demorar mais para “melhorar”, assim a seleção terá chances maiores de pegar melhores cromossomos no aleatório, porém, olhando pelo lado da teoria da evolução, fazendo com que o filho entre na população se ele for melhor que o pior pai, teríamos uma população não tanto, mas bem mais linear, já que conforme o tempo passaria, não iriamos ter mais tantos “piores pais”, já que eles vão sendo removidos pelos melhores filhos, que no caso, são os melhores pais.

CONSIDARAÇOES FINAIS

O Algoritmo foi implementado visando além de alcançar o melhor fitness, priorizar o melhor desempenho possível, porém, a muito oque ser melhorado, a implementação de um algoritmo de auto melhoramento da população usando resultados passados, assim como a maioria dos contribuintes do desafio da DIMACS usou, contribuiu muito para não alcançar o desempenho ideal, dito isso, o algoritmo compre com oque foi proposto para o projeto e a para qual foi implementado.

REFERÊNCIAS

HINDI, Musa M.; YAMPOLSKIY, Roman V. **Genetic Algorithm Applied to the Graph Coloring Problem,** Computer Engineering and Computer Science J.B. Speed School of Engineering Louisville, Kentucky.

DOUIRI, Sidi Mahamed. **Solving the graph coloring problem via hybrid genetic algorithms.** Journal of King Saud University - Engineering Sciences, 2015

The Center for Discrete Mathematics and Theoretical Computer Science DIMACS, <http://dimacs.rutgers.edu/>. Acesso em: 08 de Dez. de 2019

.