**Árvore Geradora de Rótulos Minimos**

**Relatório**

**Ana Carolina Frozza, Renan Kodama Rodrigues**

Instituto de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)  
Caixa Postal 271 – 87301-899 – Campo Mourão – PR – Brasil

Departamento de Ciência da Computação – UTFPR, Campo Mourão PR-BR

Teoria dos Grafos

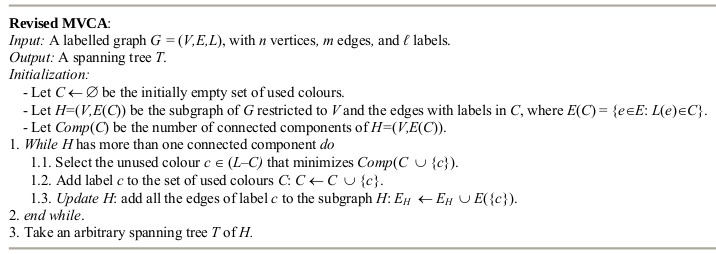
{annafrozza@gmail.com, renan\_kdm\_rdg@hotmail.com}

# 1. Introdução

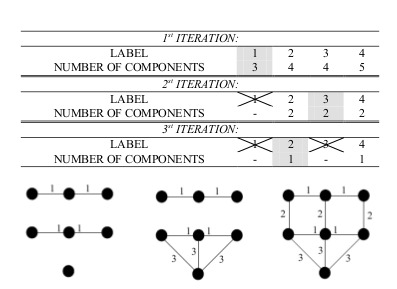
Introduzido por Chang and Leu em 1997 o conceito de Árvore Geradora de Rótulos Mínimos, para este problema determinaram que sua complexidade pertence à classe NP-Completo e propuseram um algoritmo em tempo polinomial, o MVCA (*maximum vertex covering algorithm*) para encontrar a solução do problema.

Esta heurística começa tendo o grafo inicializado e sucessivamente tendo os rótulos que não foram visitados sendo adicionado ao grafo. A heurística consiste em gerar em um grafo final porém mesmo o grafo de entrada ser conectado o algoritmo não garante encontrar a solução correta e irá falhar consequentemente.

Em 1998 Krumke e Wirth propuseram uma versão correta do MVCA. Este começa com um grafo inicializado e sucessivamente adiciona os rótulos diminuindo assim o número de componentes conexas do grafo até restar apenas uma única componente conexa.

*“Imagem1.jpg” Versão modificada do MVCA por Krumke e Wirth (1998).*

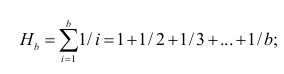
Para exemplificar o funcionamento do algoritmo temos a seguinte imagem :



*“Imagem2.jpg” Figura ilustrativa do funcionamento do MVCA modificado.*

No primeiro passo o algoritmo adiciona o rótulo “1”, pois o número de componentes é três e o algoritmo só para quando o número de componentes conectadas for igual à um. No segundo passo ainda resta os rótulos que não foram utilizados (2,3), então é escolhido de forma aleatória algum rótulo que não foi utilizado, no exemplo citado o rótulo escolhido foi o “3” e consequentemente tendo seu rótulo adicionado aos rótulos usados, na antiga versão de Chang e Leu (1997) o algoritmo iria parar aqui, como no segundo grafo da esquerda para a direita na “*imagem2.jpg*”, temos duas componentes onde todos os vértices que estão dentro são conectados, porém as duas componentes não estão conectadas entre -si resultando assim em um erro no resultado. Entretanto a versão do modificada do MVCA proposto por Krumke e Wirth adiciona o último rótulo restante “2”, tendo como resultado o último grafo do canto direito mostrado na “*imagem2.jpg*”, restando assim apenas uma única componente conexa.

Krumke e Wirth concluiram que a versão proposta do MVCA tinha um custo de O(1+2 logn), onde n é o número total de nós. Após outras versões do algoritmo MVCA surgirem como as de, Wan (2002) tinha uma melhor custo O(1+ log(n-1)) para n>1, Brüggemann (2003) usou uma abordagem diferente, aplicando técnicas aprimoradas de busca local, Xiong (2005) modificou a versão do algoritmo de Wan (2002) e mostrou que o pior caso é representado por uma série harmônica mostrado na seguinte imagem:



*“Imagem3.jpg”, série harmonica para o pior caso .*

Onde “b” significa a frequência de dado rótulo, este cálculo é válido somente se:

Hb < (1 + log(n − 1)) && b ≤ (n − 1) e o custo desta nova versão é de O(1+ log(n-1)) dado por Wan (2002).

A regra usado por Krumke e Wirth (1998) é de selecionar as a cor que minimiza o número de componentes conectados a cada passo, resultando em uma rápida solução. O problema desta abordagem ocorre quando mais de uma cor com o mesmo resultado de componentes conectados aparece em algum caso específico, o resultado dependerá da regra escolhida para selecionar o candidato deste conjunto.

Xiong em 2006 propôs uma outra modificação para o MVCA, a cada passo ele considera as três cores mais frequentes e atribui a probabilidade de seleção em relação a frequência de cada cor, então de forma aleatória seleciona um desses candidatos e adiciona à uma solução parcial.

# 2. Problema

Em um PAGRM, é dado um grafo com cada uma de suas arestas marcadas com uma label e procura-se um grafo que seja induzido pelo menor número de labels possı́veis. O problema pode ser aplicado no mundo real em problemas de comunicação de redes, redes elétricas, circuitos lógicos, entre outros. E está arvore pode reduzir tanto o custo quanto a complexidade quando aplicado à um problema desta espécie.

**Definição:** Dado um grafo de rotulos não-dirigido G = {V, E, L}, ou seja o conjunto de n vértices, E = {(u, v) | u, v ∈ V ∧u 6 = v}, o conjunto de m arestas e L = {l 1 , l 2 . . . l |L| }, o conjunto de l rótulos, encontrar uma árvore geradora T apartir de G tal que |LT | é mı́nimo. Define-se LT como o conjunto de diferentes rótulos das arestas em uma árvore geradora T . Uma solução factı́vel é definida como um conjunto de rótulos C contidos em L, tal que todas as arestas com rótulos em C representam um subgrafo conexo de G o qual contém todos os vértices de G. Se C é uma solução realizável, então qualquer árvore geradora possui no máximo |C| rótulos. Se a solução é ótima, então qualquer árvore geradora C é uma árvore geradora de rótulos mı́nimos. Assim, para resolver o problema, devemos encontrar uma solução possı́vel com o mı́nimo número de rótulos possı́veis.

# 3. Algoritmo

Para a solução do problema de Árvores Geradoras Mínimas propusemos um algoritmo baseado no MVCA. As operações de cruzamento entre os vértices de dado grafo pode ser representado da seguinte forma, se v1⊂L e v2⊂L então no subgrafo P será adicionado os rótulos P = P1 ∪ P2, até que obtenha-se uma solução viável.

O processo de mutação consiste em adicionar um novo rótulo aleatoriamente e depois tentar remover as suas adjacências, o algoritmo foi baseado nos princípios da evolução com operações de cruzamento e mutação, onde o cruzamento se dá realizando a combinação de cada elemento do grafo representado por uma matriz de adjacência e o processo de mutação ocorre quando um rótulo é escolhido para ser adicionado ao subgrafo.

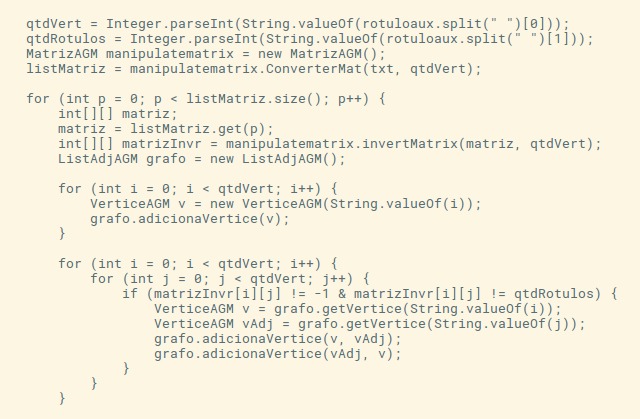
Basicamente o algoritmo cria um subgrafo contendo apenas vértices e enquante este subgrafo não totalmente conexo ele adicionas as arestas que contenham o rótulo que mais liga as demais componentes.

As principais classes do algoritmo implementado na linguagem java são, “PrincipalAGM”, “ConectadosAGM”, “MvcaAGM”, onde:

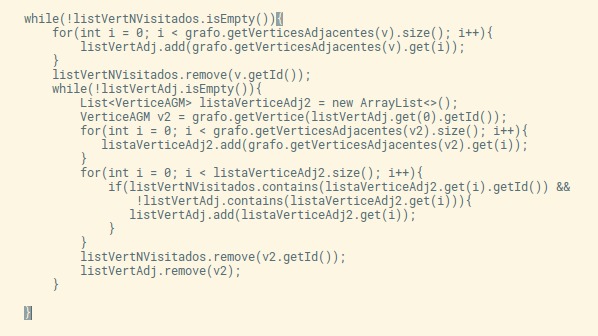
A classe “PrincipalAGM” tem a funcionalidade de inicializar o algoritmo chamando as demais funções para a solução do problema, ler o arquivo e construir a matriz de adjacência e o grafo na forma de um hashmap contendo um vértice como chave e um vetor de vértices como valor, representado pela figura “*imagem4.jpg*”.

A classe “ConectadosAGM” tem como função contar o número de componentes conectados no subgrafo passado por parâmetro, representado pela figura “*imagem5.jpg*”.

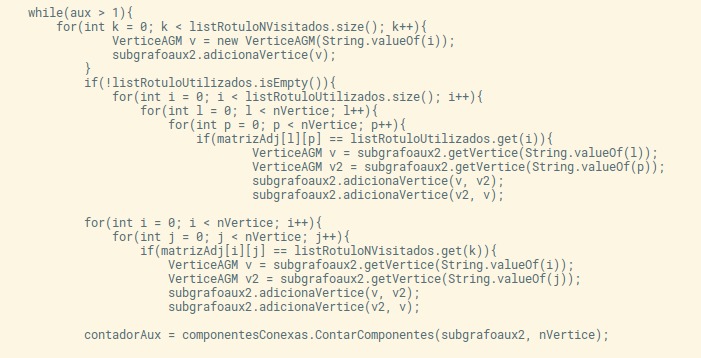
A classe “MvcaAGM” tem como função selecionar um rótulo e ligar todos os vértices que contenham este rótulo e torná-los uma componente, até que no subgrafo tenha apenas apenas uma única componente formado pela união dos rótulos, inicialmente o subgrafo é inicializado com todos os rótulos, e através da adição gradativa destes é verificado a cada passo se todos já são conectados, o algoritmo é representado pela figura “*imagem6.jpg*”.



“*Imagem4.jpg*”, trecho de código extraido da classe “PrincipalAGM”.



“*Imagem5.jpg*”, trecho de código extraido da classe “ConectadosAGM”.



“*Imagem6.jpg*”, trecho de código extraido da classe “McvaAGM”.

# 4. Resultados Obtidos

Usando como base de comparação entre os resultados do artigo criado por Yupei Xiong, Bruce Golden e Edward Wasil em “A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem (2005)” obtivemos os seguintes valores para determinados conjuntos:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grupo | Arquivo de Entrada | Execução | Artigo 6 | PILOT | MGA | EXACT | GRASP | VNS |
| I | HD20\_20 | 2.5 | 2.5 | 2.4 | 2.4 | 2.4 | 2.4 | 2.4 |
| HD30\_30 | 2.8 | 2.65 | 2.8 | 2.8 | 2.8 | 2.8 | 2.8 |
| HD40\_40 | 2.7 | 2.95 | 2.9 | 2.9 | 2.9 | 2.9 | 2.9 |
| HD50\_50 | 3.0 | 3.00 | 3.0 | 3.0 | 3.0 | 3.0 | 3.0 |
| LD20\_20 | ## | 7.00 | 6.7 | 6.7 | 6.7 | 6.7 | 6.7 |
| LD30\_30 | 7.9 | 7.70 | 7.4 | 7.4 | 7.4 | 7.4 | 7.4 |
| LD40\_40 | 8.7 | 7.85 | 7.6 | 7.4 | 7.4 | 7.4 | 7.4 |
| LD50\_50 | ## | 8.25 | 8.6 | 8.6 | 8.6 | 8.6 | 8.6 |
| MD20\_20 | 3.1 | 3.40 | 3.2 | 3.1 | 3.1 | 3.1 | 3.1 |
| MD30\_30 | 4.0 | 3.80 | 3.7 | 3.7 | 3.7 | 3.7 | 3.7 |
| MD40\_40 | 4.0 | 3.95 | 3.7 | 3.7 | 3.7 | 3.7 | 3.7 |
| MD50\_50 | 4.4 | 4.20 | 4.0 | 4.1 | 4.0 | 4.0 | 4.0 |
|  | | | | | | | | |
| II | HD200\_50 | 2.0 | 2.00 | 2.0 | 2.0 | 2.0 | 2.0 | 2.0 |
| HD200\_100 | 2.6 | 2.65 | 2.6 | 2.6 | 2.6 | 2.6 | 2.6 |
| HD200\_200 | 4.0 | 4.00 | 4.0 | 8.3 | 4.0 | 4.0 | 4.0 |
| HD200\_250 | 4.6 | 4.75 | 4.0 | 4.0 | 4.0 | 4.1 | 4.0 |
| LD200\_50 | 5.6 | 4.30 | 5.2 | 5.2 | 5.2 | 5.2 | 5.2 |
| LD200\_100 | 8.9 | 6.90 | 8.3 | 8.3 | NF | 8.1 | 7.9 |
| LD200\_200 | 13.4 | 10.90 | 12.4 | 12.4 | NF | 12.2 | 12.0 |
| LD200\_250 | 15.9 | 12.55 | 13.9 | 14.0 | NF | 13.9 | 13.9 |
| MD200\_50 | 2.4 | 2.0 | 2.2 | 2.2 | 2.2 | 2.2 | 2.2 |
| MD200\_100 | 3.8 | 3.75 | 3.4 | 3.4 | 3.4 | 3.4 | 3.4 |
| MD200\_200 | 6.2 | 5.80 | 5.5 | 5.4 | NF | 5.4 | 5.4 |
| MD200\_250 | 7.1 | 6.50 | 6.3 | 6.3 | NF | 6.3 | 6.3 |

# 5. Analise dos Resultados

1. Analisando os resultados obtidos acima, podemos concluir que o algoritmo implementado obteve um desempenho um tanto quanto abaixo do relatado no artigo, para alguns casos, o número de rótulos se manteve ao que estava descrito no artigo, porém para grande parte das outras instâncias o algoritmo implementado utilizou uma maior quantia de rótulos, uma diferença pequena, porém em uma aplicação real isto poderá causar sérios problemas.

# 6. Conclusão

Neste trabalho compreendemos o problema da Árvore Geradora Mínima e exemplos de aplicações reais, juntamente com alguns algoritmos criados e aprimorados ao longo da história para resolver este problema, mencionados anteriormente neste relatório. Estabelecido como objetivo a implementação de um algoritmo para resolver o problema da árvore geradora mínima, tendo apenas as principais funções mencionadas, afim de comparar os resultados obtidos por Yupei Xiong, Bruce Golden, and Edward Wasil em A One-Parameter Genetic Algorithm for the Minimum Labeling Spanning Tree Problem(2005).

# 7. References

[1] R. S. Chang and S. J. Leu, “The minimum labelling spanning trees,” Information Processing Letters,

Springer-Verlag, 2005.

[2] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). Introduction To Algo-

rithms. MIT Press.

[3] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual

comparison,” ACM Computing Surveys, vol. 35(3), pp. 268–308, 2003.

[4] R. Cerulli, A. Fink, M. Gentili and S. Voss, “Metaheuristics comparison for the minimum labelling

spanning tree problem,” in The Next Wave on Computing, Optimization, and Decision Technologies, B.

[5] Y. Wan, G. Chen, and Y. Xu, “A note on the minimum label spanning

tree,” Inf. Process. Lett., vol. 84, pp. 99–101, 2002.