Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada

Disciplina: DIM0406 — Algoritmos Avançados
Docente: Sílvia Maria Diniz Monteiro Maia
Discente: Felipe Cortez de Sá

Algoritmo exato para problema de Steiner com rotulação mínima

1 Introdução

Neste relatório é apresentado o problema da árvore de Steiner com rotulação mínima e um algoritmo exato para resolvê-lo baseado na técnica do *branch and bound*, sugerida por Consoli et al [1].

2 Problema

O problema de Steiner com rotulação mínima é a junção de dois problemas similares: *Minimum Labelling Spanning Tree*, que busca uma árvore geradora para um grafo $G = (V, E, L)$ utilizando a menor quantidade de rótulos possível, e o problema da árvore de Steiner, que busca num grafo ponderado $G = (V, E, w)$ uma árvore que conecta determinados vértices básicos $Q \subseteq V$ minimizando o custo das arestas. A combinação desses problemas consiste em achar uma árvore que contenha todos os vértices básicos Q utilizando o menor número de rótulos possível.

Formalmente, dado um grafo $G = (V, E, L)$, sendo V o conjunto de vértices, E o conjunto de arestas, L o conjunto de rótulos para as arestas e $Q \subseteq V$ um conjunto de vértices básicos, uma árvore de Steiner com rotulação mínima contém todos os vértices básicos conectados possivelmente utilizando os vértices não básicos $V - Q$ minimizando o número de rótulos utilizados.

O algoritmo tem aplicações na área de construção de circuitos *VLSI*, em que se deseja minimizar o cabeamento utilizado para conectar pinos, telecomunicações, engenharia civil, entre outros [2].

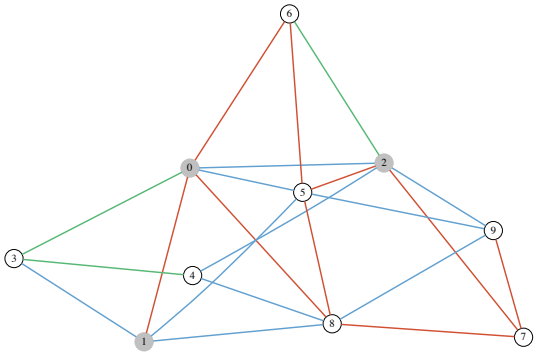

Figura 1: Um exemplo do problema com 10 vértices, 3 deles básicos e 3 cores

3 Algoritmo e técnica de solução

O algoritmo foi implementado utilizando C em um único arquivo **`steiner.c`** que não recebe entrada (já que seus casos de teste são gerados automaticamente) e que gera como saída dois arquivos **`.dot`**, que podem ser transformados em uma visualização do grafo inicial e final através do programa **`graphviz`** [3].

O funcionamento é baseado no algoritmo exato descrito por Consoli et al, que adiciona a técnica de *branch and bound* ao algoritmo guloso proposto por Cerulli et al. [4]

```
C = {}
H = (V, E(C)), E(C) = {e ∈ E : L(e) ∈ C}
C* = L
H* = (V, E(C*)), E(C*) = {e ∈ E : L(e) ∈ C}
Comp(C) // componentes conexos de  (Q, E(C))

Test(C) {
    if (|C| < |C*|) {
        if (Comp(C) = 1) {
            C* ← C
        } else if (|C| < |C*| − 1) {
            for each (c ∈ (L − C)) {
                Test(C ∪ {c})
            }
        }
    }
}
```

Listing 1: Pseudocódigo

O algoritmo adiciona uma nova cor a cada chamada do procedimento **`Test`**, permitindo a exploração do espaço de busca. C^* guarda a melhor solução até o momento numa variável global e $|C^*|$ informa quantos rótulos essa melhor solução possui. A poda (ou *bound*) é feita pela comparação $|C| < |C^*| - 1$, que ao ser avaliada como falsa significa que ao adicionar uma nova cor a $|C|$, sua cardinalidade será igual à da melhor solução, isto é, não se terá uma solução melhor e portanto não adianta explorar mais soluções a partir do conjunto C.

Comp(C) é calculado através de uma busca em profundidade.

Após executar **Test**, tem-se C^* atualizado com a melhor combinação de rótulos que garantem apenas um componente conexo contendo todos os vértices básicos Q.

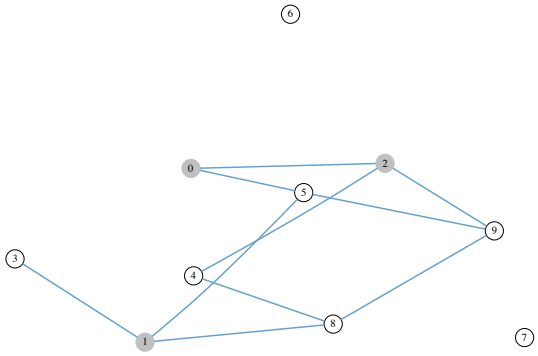

Figura 2: Grafo com arestas em C^*

Para encontrar uma solução, basta achar uma árvore geradora desse grafo e remover arestas não básicas $V - Q$ com grau 1.

4 Casos teste

Os casos de teste são gerados automaticamente de acordo com quatro parâmetros

- **SIZE**, a quantidade de vértices do grafo
- **COLORS**, a quantidade de rótulos (ou cores) em L
- **DENSITY**, valor de 0 a 100 que define a quantidade de arestas
- **BASIC**, a quantidade de vértices básicos

5 Resultados

Não foi possível testar extensamente e comparar a eficiência do algoritmo para casos grandes.

Referências

[1] S. Consoli, K. Darby-Dowman, N. Mladenovic, J.A. Moreno-Perez. *Variable neighbourhood search for the minimum labelling Steiner tree problem.* Annals of Operations Research, 2009.

https://www.researchgate.net/publication/225327721_Variable_neighbourhood_search_for_the_minimum_labelling_Steiner_tree_problem

[2] G. Robins, A. Zelikovsky. *Minimum Steiner Tree Construction.*
http://www.cs.virginia.edu/~robins/papers/Steiner_chapter.pdf

[3] *Graphviz — Graph Visualization Software.*
http://www.graphviz.org/

[4] R. Cerulli, A. Fink, M. Gentili e S. Voß. *Extensions of the minimum labelling spanning tree problem.* Journal of Telecommunications and Information Technology, 2006.
https://www.researchgate.net/publication/228668519_Extensions_of_the_minimum_labelling_spanning_tree_problem