Problema da Árvore Geradora Mínima

Patrícia Castro[1], Thiago Leite[2], Vanessa Soares[3],
Orientador: Leandro G. M. Alvim[4]

[1]Departamento de Ciência da Computação –
Universidade Federal Rural do Rio de Janeiro (UFRRJ)
R. Governador Roberto Silveira S/N – Nova Iguaçu –
Rio de Janeiro – RJ – Brasil

alvim.lgm@gmail.com, patriciadecastrowang@yahoo.com.br

thiagoflosino@gmail.com, vanessa.ssoares@hotmail.com

Abstract. *Due to the need to carry services such as roads, sewer and power lines to every corner of the city during the Industrial Revolution, there were methods that guaranteed to get the lowest possible cost for such services. Generated from a non-directional graph, and related costs or weights at the edges, an acyclic tree is formed by connecting all the vertices of the graph. Taking into account the existence of more than one possibility to solve this problem, we will greater emphasis on Kruskal's algorithm, which happens to be one of the most widely used along with Prim algorithm. The purpose of the report is to explain and elaborate on the problem of minimum spanning tree, with the Algorithm Kruskal as a possible solution, prove it and show its complexity.*

Resumo. *Devido à necessidade de se levar serviços como rodovias, esgoto e linhas de energia à todos os cantos da cidade, durante a Revolução Industrial, surgiram métodos que garantiam conseguir os menores custos possíveis para tais serviços. Gerada a partir de um grafo não direcional, conexo e com pesos ou custos nas arestas, uma árvore acíclica é formada, ligando todos os vértices do grafo. Levando em conta a existência de mais de uma possibilidade para a resolução deste problema, daremos uma ênfase maior ao Algoritmo de Kruskal, que vem a ser um dos mais utilizados atualmente, juntamente com o Algoritmo de Prim. O objetivo do trabalho se resume a explicar e dissertar sobre o problema de árvores geradoras mínimas, apresentando o Algoritmo de Kruskal como possível solução, prová-lo e mostrar sua complexidade.*

1. Introdução

Desenvolvido em 1926 pelo cientista checo Otakar Borůvka[1], o algoritmo de Borukva foi o primeiro a encontrar uma árvore de extensão mínima e foi criado com o objetivo de fornecer uma cobertura elétrica eficiente na área rural da cidade de Morávia do Sul. Baseado no algoritmo de Borůvka, também foi criado o mais rápido algoritmo de árvore mínima de extensão que foi desenvolvido por Bernard Chazelle[2], cujo tempo chega muito perto de O(m), sendo m o número de arestas. Entretanto, atualmente existem dois algoritmos que são normalmente utilizados: algoritmo de Prim e algoritmo de Kruskal. Ambos são gulosos, ou seja, escolhe a opção que parece ser a melhor no momento, e rodam em tempo polinomial. Logo, o problema de encontrar as árvores geradoras de custo mínimo é de complexidade $O(ElogV)$.

Assim, dado um grafo não orientado com custos nas arestas e considerando que esse é um grafo conexo, uma árvore de extensão desse grafo é definida como um subgrafo que conecta todos os vértices, levando em conta os pesos nas arestas que ligam dois vértices. Um exemplo de uso de uma árvore de extensão mínima seria a instalação de fibras óticas num condomínio com muitos prédios. Cada trecho de fibra ótica entre os prédios possui um custo associado (isto é, o custo da fibra, somado ao custo da instalação da fibra, mão de obra, etc). Sabendo todos os custos, pode-se construir uma árvore de extensão que seria uma solução para conectar todos os predios mostrando o menor custo para fazer essa ligação.

2. Problema e Motivação

Além de áreas ligadas à computação, como projeto de redes de telecomunicação, o problema da árvore geradora de custo mínimo também é aplicável a outros exemplos em outras áreas, dentre eles estão: projetos de rodovias e ferrovias, projeto de redes de transmissão de energia. Assim, sendo de grande importância devido ao grande número de problemas que podem ser resolvidos usando a mesma solução, daremos ênfase à uma das formas de resolver tal problema, utilizando o algoritmo de Kruskal.

3. Objetivo

O objetivo do relatório é dissertar sobre o problema da árvore geradora mínima (MST) e considerando sua solução o algoritmo de Kruskal, mostrando seu algoritmo, suas provas de corretude, complexidade e análise empírica.

4. Proposta

4.1. O algoritmo de Kruskal

Algoritmo usado para encontrar uma árvore geradora mínima num dado grafo conexo não orientado e com pesos nas arestas, o algoritmo de Kruskal é considerado um algoritmo guloso e ganancioso, já que funciona de acordo com a ordenação das arestas do grafo em ordem crescente de peso.

```
MST-KRUSKAL(G,w)
1 A := {}
2 for cada vértice v em V[G]
3     do MAKE-SET(v)
4 Ordene as arestas de E em ordem crescente de peso (w)
5 for cada aresta (u,v), tomadas em ordem crescente de peso (w)
6     do if FIND-SET(u) != FIND-SET(v)
7           then A := A união {(u,v)}
8                UNION(u,v)
9 return A
```

Figura 1. Pseudocódigo Algoritmo de Kruskal[3]

O algoritmo utiliza uma estrutura de dados de conjuntos disjuntos para manter vários conjuntos disjuntos de elementos. Cada conjunto contém os vértices em uma árvore da floresta que está sendo formada. A operação $FIND - SET(u)$ retorna um elemento que representa o conjunto que contem u. Assim, conseguimos determinar se dois vértices

u e v pertencem a mesma árvore testando se $FIND - SET(u)$ é igual a $FIND - SET(v)$. Já a combinação das árvores é realizada pela operação $UNION()$.

O algoritmo funciona da seguinte forma: inicializa-se o conjunto A como sendo um conjunto vazio e criam $|V|$ árvores, cada uma contendo um vértice. Já as arestas são ordenadas por peso, em ordem crescente. A seguir, um loop verifica se para cada aresta(u,v), se os pontos externos u e v pertencem à mesma árvore e somente serão adicionados se não pertencerem à mesma árvore. Já que, se estiverem na mesma árvore, significa que a adição de tal aresta formaria um ciclo.

4.2. Prova de Corretude

Seja $G = (V, E)$ um grafo conexo não direcionado e valorado. O problema consiste em achar um sub-conjunto T de E, tal que T forme uma árvore e a soma dos pesos é o menor possível. Aplicando o algoritmo guloso nesse caso, teremos as seguintes definições:

O conjunto S de candidatos é um conjunto de arestas distintas de G. O conjunto S é viável se ele não contém nenhum circuito. O conjunto S é uma solução se ele contém todos os vértices de G. Um conjunto S de arestas é promissor se ele pode ser completado para produzir uma solução ótima. E um conjunto vazio é considerado promissor. Uma aresta toca um conjunto S de vértices se exatamente um vértice ligado por essa aresta existe em S.

Teorema : O algoritmo de Kruskal é correto, isto é, ele retorna sempre uma árvore geradora mínima.

Prova: Vamos provar por indução. Seja $G = (V, E)$ o grafo conexo processado pelo algoritmo.
Base: No início o conjunto T é vazio. Considerando que G é conexo, sabemos que necessariamente ele contém uma árvore geradora. Então por definição T vazio é promissor.
Hipótese de indução: Suponhamos que T é promissor para 0 e k arestas.
Passo indutivo: Tendo T de k arestas, o algoritmo seleciona a aresta de menor peso no conjunto de arestas não visitadas. Seja ak+1 essa aresta. Há quatro casos:

T já contém n-1 arestas. Nesse caso, a solução T que por indução é uma árvore mínima, é retornada. A aresta ak+1 liga dois vértices de T que são no mesmo componente. Nesse caso, ak+1 é rejeitada e T não muda. Um dos dois vértices ligados por ak+1 não pertence a T. Seja B o conjunto que contém somente esse vértice. É fácil verificar que T não toca esse conjunto então, podemos concluir que T U ak+1 é promissor. A aresta ak+1 liga dos vértices de T. Nesse caso, são dois vértices de componentes diferentes. Seja C1 e C2 esses dois componentes. Seja B os vértices de um desses dois componentes. Podemos ver facilemente que T não toca B, pois toda aresta de T ou não contém nenhum vértice de B ou contém dois. Portanto, percebemos que T U ak+1 é promissor. Como a cada passo do algoritmo obtemos um conjunto promissor, necessariamente ele retorna uma solução.

4.3. Análise de sua Complexidade

O tempo de execução do algoritmo para um grafo G, depende da forma como foi implementada a estrutura de dados característica do algoritmo de Kruskal. Supondo que tais estruturas foram implementadas de acordo com as heurísticas de união por ordenação.

A inicialização do conjunto demora o tempo $O(1)$ e o tempo de ordenação das arestas é $O(ElogE)$. O loop que vem logo em seguida, executa em $O(E)$ operações $FIND - SET$ e $UNION$ sobre a floresta de conjuntos disjuntos. Junto com as $|V|$ operações $MAKE - SET$, essas operações totalizam o tempo de $O((V + E)\alpha(V))$, onde α é a função de crescimento muito lento. Assim, tomamos que o tempo de execução total do algoritmo $O(ElogV)$.

5. Experimentos

Para os experimentos foram utilizados 4 grafos diferentes com pesos diferentes nas arestas para que pudéssemos avaliar a variação de tempo em função do número de arestas e vértices.

Vétices	Arestas	Tempo(ms)
5	10	326
10	45	736
40	780	5711
60	1770	29621

Tabela 1. Exemplo de tabela de experimentos

Acima podemos ver a tabela com os dados encontrados nos experimentos. Para estes foi utilizado um computador com sistema operacional Ubuntu 14.04, que trabalha a 1.7 GHz, com 6GB de RAM.

6. Conclusões

Algoritmos de árvores geradoras de custo mínimo vêm sendo usados em uma série de áreas devido à sua importancia causada pela sua aplicabilidade em diversos problemas. Por isso, o trabalho propôs a implementação do algoritmo de Kruskal para mostrar uma possível solução para o problema, já que existem outros algoritmos que podem ser usados. Sendo um deles tão utilizados quanto o algoritmo de Kruskal, que é o algortimo de Prim.

Utilizando grafos não orientados, valorados e conexos, realizamos experimentos, mostrando a variação do tempo de execução em função do número de arestas e vértices pertencentes ao grafo criado. Foram utilizados também estruturas de dados de conjuntos disjuntos na implementação para facilitar o funcionamento do algoritmo na obtenção da árvore geradora de custo mínimo.

Assim, neste trabalho, abordamos de forma prática o uso do algortimo de Kruskal como solução para o problema da árvore geradora de custo mínimo, percebendo sua simples implementação e assim, de fácil uso em diversos casos.

7. Referências

[1]Algoritmo de Boruvka. Disponível em: `https://prezi.com/rjel80npvke6/algoritmo-de-boruvka/` Acesso em: 05 de julho de 2015.

[2] Natural Algorithms. Disponível em: `https://www.cs.princeton.edu/~chazelle/pubs/soda09.pdf` Acesso em: 10 de julho de 2015.

[3] Árvores Espalhadas Mínimas. Disponível em: http://www.ic.unicamp.br/~meidanis/courses/mo417/2003s1/aulas/2003-05-16.html Acesso em: 10 de julho de 2015.

Árvores Geradoras de Grafos. Disponível em: http://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/spanningtrees.html Acesso em: 13 de julho de 2015.

Árvores . Disponível em: http://www.professeurs.polymtl.ca/michel.gagnon/Disciplinas/Bac/Grafos/Arvores/arvores.html#Kruskal Acesso em: 13 de julho de 2015.

CORMEN, THOMAS, et al. Algoritmos: Teoria e Prática. Editora Campus, pag 470-475.

PASTI, DIEGO, et al. Instituto Federal do Espírito Santo - Campus Serra*Problema da Árvore Geradora Mínima* Jefferson Rios