Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
Disciplina de Grafos - DCC059 2017/2

Relatório trabalho de Grafos

Alunos:
Vinícius de Castro Sampaio - 201635002
Igor de Andrade Junqueira - 201635004
Almir Phelipe Arruda - 201635039

Professor orientador: STENIO SA ROSARIO FURTADO SOARES

Dezembro
2017

Universidade Federal de Juiz de Fora

Departamento de Ciência da Computação
Disciplina de Grafos - DCC059 2017/2

Relatório

Relatório final da disciplina de Grafos do segundo semestre de 2017 do curso de Ciência da Computação.

Alunos:
Vinícius de Castro Sampaio - 201635002
Igor de Andrade Junqueira - 201635004
Almir Phelipe Arruda - 201635039

Professor orientador: STENIO SA ROSARIO FURTADO SOARES

Dezembro
2017

Conteúdo

1 Resumo

O problema que o nosso grupo irá apresentar é o da Árvore de Steiner. Esse problema é NP completo, ou seja, não é garantido encontrar a melhor solução em tempo de processamento polinomial. Para isso, nosso grupo irá propor um algoritmo construtivo guloso e em seguida iremos randomizar ele e depois chamar a função que randomiza diversas vezes dando preferência para o valor de alfa que mais apresenta bons resultados.

O projeto foi desenvolvido na IDE CodeBlocks e desenvolvido em ambiente Linux, usando a linguagem C++.

A execução dos testes do algoritmo foi realizada em duas máquinas, tendo as seguintes configurações:

Máquina 1:

- Processador: Intel Core i5-4200U 1.7GHz (turbo boost 2.7GHz)
- Memória RAM: 6GB DDR3
- Sistema Operacional: Linux (Ubuntu)

Máquina 2:

- Processador: Intel Core i5-2300 2.8GHz
- Memória RAM: 8GB DDR3
- Sistema Operacional: Linux (Ubuntu)

2 Descrição do Problema

O problema da Árvore de Steiner se baseia em encontrar a árvore de menor peso de um determinado grafo. Nesse grafo, você tem o conjunto de vértices, arestas e o de vértices terminais. Nesse problema, você deseja encontrar a árvore que conecte esses vértices terminais com o menor peso póssível.

Por exemplo, no grafo abaixo uma solução para o problema seria a árvore de menor peso que conectaria os vértices 1,3,5 e 7.

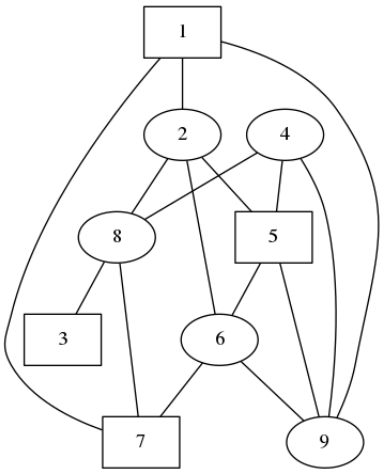
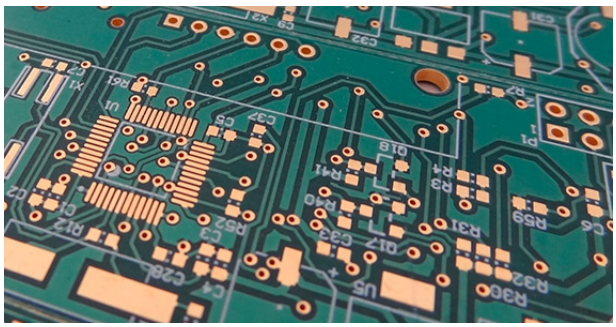


Imagem gerada no graphviz

Uma das aplicações do problema da Árvore de Steiner é em projetos de circuitos e layout de fios em placas de circuito impresso, onde você tem uma placa de circuitos e precisa ligar um conjunto de componentes minimizando os custos.



Fonte: http://www.griffuspcb.com.br/fabricacao-de-placas-de-circuito-impresso.html

3 Algoritmo Construtivo Guloso

Uma solução para o problema é um algoritmo construtivo guloso, onde você possui uma lista de candidatos e uma função critério na qual você irá escolher esses candidatos até você formar uma solução.

Para o nosso problema, escolhemos como sendo a lista de candidatos os vértices não-terminais, pois os vértices terminais nós já incluímos na solução chamando a função de subgrafo induzido para o conjunto de vértices terminais. A função critério é a busca pelos vértices que maximixa o valor do número de vértices terminais vizinhos sobre o número de vizinhos, assim, você busca pelos vértices que possuem maior porcentagem de vizinhos terminais.

Função critério:

$$max \frac{dT(Vi)}{d(Vi)} \tag{1}$$

Com a função critério, nós ordenamos a lista de candidatos em ordem decrescente, ou seja, deixando os melhores candidatos no inicio, armazenando todos em um vetor.

Após isso, ele vai gerando o subgrafo induzido por aquele conjunto de vértices até o grafo se tornar conexo, ou seja, todos os vértices terminais estarem conectados, pois no início do programa foi chamado a subgrafo induzido. Depois disso, ele chama a AGM (Kruskall) e por fim realiza as podas, removendo os vértices que possuem apenas um vizinho e esse mesmo vizinho sendo um vértice não terminal. Com isso, geramos uma solução para o problema da Árvore de Steiner em tempo de processamento polinomial, não garantindo solução ótimas mas para alguns casos soluções viáveis.

4 Algoritmo Construtivo Guloso Randomizado e Randomizado Reativo

No algoritmo randomizado, a função critério é a mesma do guloso, só que temos um parâmetro alfa(passado por valor) em que consideramos uma porcentagem da lista de candidatos.Temos o alfa e multiplicamos pelo tamanho da lista, considerando que o valor da lista diminui a cada interação. Adicionamos um vértice e suas aresta ao subgrafo induzido pela lista de terminais, esse processo continua até o grafo ser conexo. Após isso, geramos a árvore geradora de peso mínimo, e aplicamos uma busca por vértices folha e não terminal, e retiramos do grafo, porque ele não contribui na conectividade dos vértices terminais. O algoritmo gera 100 soluções e retorna a melhor.

No reativo, temos um vetor de alfas, que ele começa em 0.1 e vai a 1, diferenciando em 0.05 .Temos um vetor de probabilidade, em que cada posição corresponte a um alfa. A função de calcular a probabilidade recebe um vetor com as melhores soluções de cada alfa, e o vetor de probabilidade. Essa função começa encontrando o maior valor, depois temos um vetor em que são inseridos a diferença entre o maior valor e o elemento do vetor. Depois calculamos a porcentagem de cada alfa, e escrevemos o valor de alfa correspondente no vetor de probablidade. Como trabalhamos somente com valores de porcentagem inteiros, truncamos o valor antes de inserir n valores de alfa no vetor, sendo n a porcentagem. A função retorna o tamanho do vetor, sendo que ele é sempre menor ou igual a 100.

Temos uma função auxiliar em que ela executa uma interação do randomizado e retorna a solução. No inicio do algoritmo em sí, iniciamos o vetor de alfas e para cada valor chamamos a função auxiliar e calculamos a probabilidade.
Iniciamos um laço de mil interações em que ele vai escolher um alfa deforma aleatoria do vetor de probabilidade, depois ele vai chamar a função auxiliar, pasndo o valor de alfa escolhido, em que é retornado uma solução, ela é comparado com a melhor solução anterior e caso seja menor, o valor no vetor é atualizado. Verificamos também se essa solução, que foi alterado do vetor de melhores sóluções é a melhor de todas e guardamos essa solução. No final, se o valor foi alterado, recalculamos a probabilidade.

5 Instâncias escolhidas

Para testar o nosso algoritmo, recebemos do professor orientador instâncias da The Zuse Institute Berlin (ZIB), no qual escolhemos 3 instâncias consideradas fáceis, 4 médias e 3 difíceis.

Instâncias fáceis:

- c01.stp - 500 vértices, 625 arestas e 5 vértices terminais. Solução do autor: 85
- c10.stp - 500 vértices, 1000 arestas e 250 vértices terminais. Solução do autor: 1093
- c20.stp - 500 vértices, 12500 arestas e 250 vértices terminais. Solução do autor: 267

Instâncias Médias:

- lin11.stp - 816 vértices, 1460 arestas e 10 terminais. Solução do autor: 4280
- lin12.stp - 818 vértices, 1462 arestas e 12 vértices terminais. Solução do autor: 5250
- lin20.stp - 822 vértices, 1466 arestas e 16 vértices terminais. Solução do autor: 4609
- world66.stp - 666 vértices, 221445 arestas e 174 vértices terminais. Solução do autor: 122467

Instâncias Difíceis:

- d01.stp - 1000 vértices, 1250 arestas e 5 terminais. Solução do autor: 106
- d10.stp - 1000 vértices, 2000 arestas e 500 terminais. Solução do autor: 2110
- d20.stp - 1000 vértices, 25000 arestas e 500 terminais. Solução do autor: 537

6 Resultados obtidos

Tabela 1: Resultados Guloso

Instâncias	Resultado	Solução Autor	Diferença
c01.stp	997	85	912
c10.stp	1329	1093	236
c20.stp	270	267	3
lin11.stp	26243	4280	21963
lin12.stp	26420	5250	19170
lin13.stp	26594	4609	21985
world666.stp	123139	122467	672
d01.stp	1587	106	1481
d10.stp	2333	2110	223
d20.stp	546	537	9

Tabela 2: Resultados reativo

Instância	Média	DP	M. Solução / Alf	Diferença
c01.stp	842,7	13,83273	817 / 0,75	732
c10.stp	1192,2	7,62744	1180 / 0,8	87
c20.stp	270	0	270 / 0,1	3
lin11.stp	22608,1	539,9186	21867 / 0,35	17587
lin12.stp	23307	1317,52276	22431 / 0,25	17181
lin13.stp	22818,8	534,76803	22096 / 0,35	17487
world666.stp	123739	0	123739 / 0,1	1272
d01.stp	1477,6	9,1195817,25109	1449 / 0,55	1343
d10.stp	2322,5	9,11958	2300 / 0,9	190
d20.stp	546	0	546 / 0,1	9

Tabela 3: Resultados randomizado

Instância	Alf 0.1	DP 0.1	Alf 0.2	DP 0.2	Alf 0.3	DP 0.3	Melhor Sol.	Diferença
c01.stp	949,9	8,94936	907	14,65909	905	12,11021	882	797
c10.stp	1268,8	5,9963	1246,7	9,22617	1230,5	5,0448	1222	129
c20.stp	270	0	270	0	270	0	270	3
lin11.stp	25634,4	405,4	24653,2	954,19842	23816,4	1262,11029	24023	19743
lin12.stp	25671	199,62465	24490,2	878,5451182	23466,8	588,67739	23256	18006
lin13.stp	25915	75,46743	25262,8	834,04554	23686	700,56073	23396	18787
world666.stp	123739	0	123739	0	123739	0	123739	1272
d01.stp	1535,6	13,192242	1513,7	18,88003	1502,9	15,72295	1476	1370
d10.stp	2361,8	3,15524	2353,9	5,3427	1227,3	3,80025	2345	235
d20.stp	546	0	546	0	546	0	546	9

7 Conclusões

Ao final dos testes executados em todos os algoritmos para as instâncias escolhidas, verificamos que em instâncias onde o número de vértices terminais é inferior proporcionalmente aos vértices como um todo, o algoritmo não consegue encontrar boas soluções, mesmo executando o randomizado, ele consegue melhorar um pouco a solução, mas não consegue se aproximar batante do resultado em sí.Já em instâncias com um número de vértices terminais significativos, os resultados obtidos foram muito bons, até mesmo para instâncias de 1000 vértices, as quais foram as maiores que testamos.

Outro fato que pode ser verificado é o número de arestas, quanto mais arestas o grafo possuí, maior a chance de encontrar soluções próximas do autor, pois em grafos onde o número de arestas é restrito, o caso acaba se tornando particular e difícil de encontrar soluções boas.

Bibliografia

Instâncias: http://steinlib.zib.de/steinlib.php