Universidade Federal de Juiz de Fora
Departamento de Ciência da Computação
DCC059 - Teoria dos Grafos Semestre 2015-2

Problema de cobertura de vértices ponderados com minimização de vértices

Daniel André Carlos Cândido

Professor: Stênio Sã Rosário F. Soares

Relatório da terceira parte do trabalho de Teroria dos Grafos, parte integrante da avaliação da disciplina.

Juiz de Fora
Março de 2016

1 Introdução

Em Teoria dos Grafos, cobertura de vértices de um grafo trata-se de um conjunto de vértice tal que cada aresta do grafo é incidente, a pelo menos, um vértices do conjunto, ou seja, é um conjunto de vértices que contém pelo menos uma das extremidades de cada aresta. Em outras palavras, uma cobertura de vértices é um conjunto V de vértices que obedece a seguinte propriedade: toda aresta do grafo tem pelo menos uma das extremidades em um vértice de V.

Cobertura de vértices ponderado com minimização de vértices, concite em encontra um número mínimo de vértices que possua o menor peso possível e que atenda a propriedade citada acima.

Uma aplicação prática para o problema é definir a melhor localização "possível" para uma instalação de antena de celular ou uma minimização do número de instalações de câmera de segurança.

2 Metodologia utilizada

Dentre as linguaguens de programação propostas, C++ foi a linguagem escolhida para implementar deste trabalho. Pois, trata-se de uma abordagem flexível e multiparadigma permitindo ainda o uso de orientação a objetos, programação genérica e em alguns casos o uso de ambos em um mesmo código. Uma outra vantagem está nos problemas relacionados à memória, visto que nesses casos é possível adotar um estilo de mais baixo nível.

2.1 Estruturas de dados utilizadas

O algoritmo implementado possui as seguintes classes:

- Classe Grafo: Responsável pelas manipulações do grafo em si. O grafo é representado por listas de adjacências, e para cada vértice há uma lista encadeada contendo suas arestas com os demais vértices do grafo. As listas são compostas de item e são representações de grafos, logo grafos são listas, e essa relação pode portanto, ser definida como herança simples. Os vértices estão definidos como listas de arestas e itens do grafo, ou seja herança múltipla. As arestas, por sua vez, são apenas itens dos vértices o que nos faz apresentar outro exemplo de herança simples.
- Classe Vértice: Utilizada para manipulações referentes às arestas. Suas funções se referem basicamente a: verificar existência de arestas, encontrar arestas, e remover arestas (ambas apresentando o valor do id do vértice como parâmetro).

- Classe Item: Possui operações referentes à lista encadeada. Como por exemplo, pegar próximo, pegar anterior, pegar informação, entre outras.
- Classe Lista: Apresenta as operações referentes à lista de vértices. Como por exemplo, contar número total de itens, adicionar e deletar item, entre outras.
- Classe LeituraGravacao: Responsável pela leitura e gravação dos arquivos de texto. Sendo que os arquivos de leitura são as instâncias contendo os grafos.

2.2 Abordagens utilizadas

Foram adotadas três abordagem na elaboração do trabalho, Construtivo (Guloso), Construtivo Randômico e Construtivo Randômico Reativo ou Adaptativa.

- Algoritmo Construtivo: Concite em escolher sempre o melhor candidato de uma lista de canditato previamente ordenada.

```
1  ContrutivoCoberturaPonderada(G(V,E), alfa){
2    O ← ordenaVertice(G(V,E));
3    S ← Ø; // Armazena os vértice da solução
4    Enquanto O ≠ Ø faça{
5      i ← randRange(0, alfa*O.count);
6      v ← O[i]  // v recebe o primeiro vértice de O
7      para cada (u,v)  ε E de v faça{
8          decrementaGrau(u)
9      }
10     S ← S U {v};
11     atualizaLista(O,v);
12     ordenaLista (O);
13   }
14   returna S;
15 }
```

O algoritmo construtivo, recebe como parametro o grafo e um alfa que no caso cdo guloso esse valor é zero, para que o algoritomo comece do inicio do array. Na linha 2, na imagem acima, é criado um lista de candidatos ordenado pela divisão P ÷ g, onde P é o peso e g o grau do vértice, na linha 3 é criado um conjunto vazio. A condição de para do algoritmo é quando um array ciaro na linha 2 está vazio. Nas lihas 5 e 6, o indece i sempre recebe zero (nesse caso por conta do valor de alfa ser igual a zero) e o vetor v recebe o primeiro elemnte da lista O(lista de candidatos). Entre as linha 7 a 9, o grau dos vértice adjacência a v é decrementato em uma unidade. Nas linhas de 10 a 14, a lista de solução recebe o vértice v, a lista de cadidatos O é atualizada e a por fim é retornado a soução na linha 14.

- Algoritmo Contrutivo Randômico: Concite em escolher o melhor canditato entre os x melhores candidatos de uma lista L de canditato previamente ordenada. Onde o tamnha de x é dado pelo tamanha da L * alfa, onde 0 > alfa < 1.

```
1  CoberturaRamdomizao(G(V,E), alfa, numIteracao){
2      best ← S;
3      fara j=1 até numIteração faça {
4          S ← Ø; // Armazena os vértice da solução
5          S ← ContrutivoCoberturaPonderada(G(V,E), alfa);
6          se S.custo < best.custo então {
7                best = S;
8          }
9          J++;
10     }
11     retorna best;
12 }
```

No algoritmo Randômico é informado o grafo, um alfa entre 0.1 a 1.0 e um numero maximo de iterações. Nesse algoritmo exite um best que quarda a melhor solução gerado pelo algoritmo. Nas linha 5, o algoritmo Construtivo é chamado, retornando uma solu para o alfa informado. Nas linhas de 6 a 8 é verificafo se a solução corrente é a melhor, caso seja, best recebe a solução.

- Algoritmo Construtivo Randômico Reativo: Concite em escolher o melhor canditato entre os x melhores candidatos de uma lista L de canditato previamente ordenada. Onde o tamnha de x é dado pelo tamanha da L * alfa, onde 0 > alfa < 1. A cada iteração o tamanho de x é alterado escolhendo um novo alfa dentro de um array de alfas. O criterio de escolha do alfa, é dado pela probabilidade do alfa, que é calculada a cada iteração. O calculo da probabilidade de cada alfa é ralizado da seguinte forma.

$$P[i] = \frac{Q[i]}{somaQ} \tag{1}$$

Onde Q[i] é o valor quatitativo do alfa[i].

$$somaQ = \sum_{n=i}^{n} Q[i] \tag{2}$$

Soma de todos os valores quantitativo dos alfas.

$$Q[i] = \frac{best}{M[i]} \tag{3}$$

Onde best é o custo da melhor solução encontrada com o alfa[i].

$$M[i] = \frac{C[i]}{c[i]} \tag{4}$$

Onde M[i] é a media, C[i] é a soma de todos os custo das soluções do alfa[i] e c[i] a quatidade de vezer que o alfa[i] foi utilizado.

```
1  CoberturaRamdomizaoReativo(G(V,E), array alfa, numIteracao, numIteracaoBloco){
2      best ← S;
3      S → solução corrente.
4      while (i < numeroIteração){
5          while (j < numeroBloco){
6              k ← escolheAfla(P); escolhe um alfa de acordo com  probabilidade
7              S ← Contrutivo( alfa[k] );
8              AtualizaVetores(Conta , somaValor, k);
9              best ← atualizaMelhorSolução(best, S);
10         }
11         AtualizaProbabilidade(Q, P, best, M);
12     }
13     retorna best;
14 }
```

O algoritmo Construtivo Randômico Reativo, recebe como parametro o grafo, uma array de alfas, um numero maximo de iterações e um numero maximo de iterações por bloco. Na linha 2 e 3, são criados a variavel "best"que armazema e melhor solução e a variavel "S", que armazema solução corrente. Na linhas 6 uma posição i é escolhida, na linha 7 o algoritmo Construtivo e armazema o resultado na variavel S, na linha 8 os vetores c e C são atualizados na posição k, na linha 9 o best é atualizado com a melhor solução, na linha 11 os vetores Q, P e M são atualizados.

Funções auxiliares:

- ordenaVertice: função que cria e retorna um array de vértice ordenado de forma crescente. Para ordenar é usado a função "sort"da propria linguagem.
- decrementaGrau: decrementa o grau do vértice passado por parametro.
- atualizaLista: remove o vértice passado por parametro e todos os vértice com o grau igual a 0.
- melhorSolu: retorna a melhor solução entre a solução corrente e a solução atual e por referencia, retorna o custo da solução corrente e atualiza o melhor custo.
- escolheAlfa: escolhe o alfa de acordo com a sua probabilidade retornado o indece do mesmo.

3 Experimentos computacionais

As funções implementadas foram desenvolvidas e testadas em um computadore com sistema operacional Linux com as seguintes especificações.

- Processador: AMD FX 6300.
- Placa de Video: GeFoce GTX960.
- Menoria RAM: 12GB DDR3.

Para os teste foram utilizadas 15 instâcias de tamanhas diferentes, todas no formato .txt. AS instacia foram baixadas do link do site da DIMACS11. Todos os resuldados possuem os seguintes campos; nome da instacia, NUM-NOIS-IN -> quantidade de nos, MS -> melhor solução, TEMP -> tempo de execulção do algoritmo em segundos, MS -> melhor solução do algoritmo (MSG -> construtivo, MSR ->randomico, MSRE -> reativo), ALFA -> alfa utilizado para gerar a melhor solução, NUN-IT -> numero de iterações, NUM-NOS-S -> quantidade de nos da solução.

- Resuldado Algoritmo Construtivo

RESULTADOS							
INSTACIAS		CONSTRUTIVO					
Nome	NUM-NOS-IN	MS	TEMP	MSG	ALFA	NUM-IT	NUM-NOS-S
cc3-4n.txt	64	5887	0	6421	0	1	61
cc9-2n.txt	512	42691	20	44597	0	1	481
cc7-3n.txt	2187	199207	466	203005	0	1	2097
cc3-1n.txt	1728	169907	486	172021	0	1	1710
cc5-3n.txt	243	21429	5	22517	0	1	228
g10000.txt	10000	132742	515	132742	0	1	1513
cc6-3n.txt	729	64569	48	67019	0	1	697
cc1-2n.txt	2048	174975	359	175641	0	1	1889
cc4-2n.txt	1024	86243	81	87051	0	1	934
cc8-2n.txt	4096	350717	1589	352031	0	1	3768
g_1000.txt	1000	77608	2539	78098	0	1	846
cc3-5n.txt	125	10432	1	10891	0	1	121
cc3-10n.txt	1000	97523	141	99347	0	1	988
cc3-11n.txt	1331	129842	261	130959	0	1	1313
cc6-2n.txt	64	6287	0	7333	0	1	62

- Resuldado Algoritmo Construtivo Randômico

RESULTADOS							
INSTACIAS		CONSTRUTIVO RANDOMICO					
Nome	NUM-NOS-IN	MS	TEMP	MSR	ALFA	NUM-IT	NUM-NOS-S
cc3-4n.txt	64	5887	77	5887	0.5	100	57
cc9-2n.txt	512	42691	1978	42691	0.5	100	465
cc7-3n.txt	2187	199207	46047	199207	0.5	100	2059
cc3-1n.txt	1728	169907	48503	169907	0.5	100	1696
cc5-3n.txt	243	21429	507	21429	0.5	100	219
g10000.txt	10000	132742	54541	160826	0.5	100	1719
cc6-3n.txt	729	64569	4655	64773	0.5	100	680
cc1-2n.txt	2048	174975	35676	174975	0.5	100	1871
cc4-2n.txt	1024	86243	8144	86441	0.5	100	921
cc8-2n.txt	4096	350717	158747	352603	0.5	100	3745
g_1000.txt	1000	77608	220226	82687	0.5	100	878
cc3-5n.txt	125	10432	187	10487	0.5	100	116
cc3-10n.txt	1000	97523	13858	97523	0.5	100	976
cc3-11n.txt	1331	129842	26301	129842	0.5	100	1305
cc6-2n.txt	64	6287	38	6287	0.5	100	54

- Resuldado Algoritmo Construtivo Randômico Reativo

RESULTADOS							
INSTACIAS		CONSTRUTIVO RANDOMICO REATIVO					
Nome	NUM-NOS-IN	MS	TEMP	MSRE	ALFA	NUM-IT	NUM-NOS-S
cc3-4n.txt	64	5887	59	5997	0.3	100	58
cc9-2n.txt	512	42691	1954	43053	0.5	100	466
cc7-3n.txt	2187	199207	46885	199961	0.5	100	2067
cc3-1n.txt	1728	169907	48396	170573	0.4	100	1701
cc5-3n.txt	243	21429	493	21683	0.5	100	221
g10000.txt	10000	132742	54080	133702	0.1	100	1522
cc6-3n.txt	729	64569	4629	64569	0.5	100	679
cc1-2n.txt	2048	174975	35982	175017	0.2	100	1884
cc4-2n.txt	1024	86243	8107	86243	0.1	100	929
cc8-2n.txt	4096	350717	158658	350717	0.1	100	3759
g_1000.txt	1000	77608	235757	77608	0.1	100	843
cc3-5n.txt	125	10432	170	10432	0.5	100	115
cc3-10n.txt	1000	97523	13915	98134	0.5	100	980
cc3-11n.txt	1331	129842	26362	129871	0.5	100	1305
cc6-2n.txt	64	6287	37	6323	0.5	100	54

4 Conclusões

A partir do dados analisados nas tabela acima podemos concluir para as instâncias testadas e em outras com características semelhantes que o uso mais indicado para gera uma solução razoável mas a melhor solução entre as gerados para a instância que foi utilizada nos três algoritmos foi o Construtivo Randômico que teve 9 melhores resultados com um alfa igual a 0,5. Com tudo se o valos do alfas for menor que 0,5, a eficiência do algoritmo diminui consideravelmente. Enquanto o Construtivo Randômico Reativo, teve 5 melhores resultados e o Construtivo apenas 1, mas em alguns resultados, se aproximo do melhor resultado.