Trabalho final da disciplina

Universidade Federal de Minas Gerais

Instituto de Ciências Exatas

Departamento de Ciência da computação

Projeto e analise de algoritmos

Aluno: João Marcos Martins da Costa Cota

Modelagem, algoritmo exato e plano de experimentos.

Encontrar cliques em redes de colaboração, estudo de caso dos 30 anos do SBBD

1 Descrição do problema

O Simpósio Brasileiro de Bancos de Dados (SBBD) é o evento oficial de bancos de dados da Sociedade Brasileira de Computação (SBC). Segundo informações do site da SBC é o maior evento da América Latina para apresentação e discussão de resultados de pesquisas e aplicações na área de bancos de dados reunindo mais de 400 pessoas nos últimos anos dentre pesquisadores, alunos e profissionais da área.

Dada a relevância do evento, em 2016, ele completou 30 anos. Devido ao tempo de existência e participação desses pesquisadores, alunos e profissionais da área foi possível obter uma base de dados notável das participações e dos trabalhos ali apresentados. Então é possível analisar a colaboração entre esses autores criando uma representação na forma de um grafo.

1.1 Modelagem do problema em Grafos

Seja um grafo G = (V, E) não direcionado, V representa o conjunto de vértices e E o conjunto de arestas. Os vértices são os autores de vários trabalhos e as arestas conectam vários autores de um mesmo trabalho, formando assim a rede de colaboração. Autores que participaram de apenas um único trabalho em todo evento do SBBD, portanto não estarão relacionados a outros autores. A tabela abaixo mostra um exemplo:

Tabela 1 - Relacionamento entre autores e trabalhos

Trabalho	Autores participantes
1	2
1	3
2	4
2	5
2	6
3	1
4	5
4	11
5	11
7	8

Isso é representado pelo seguinte grafo:

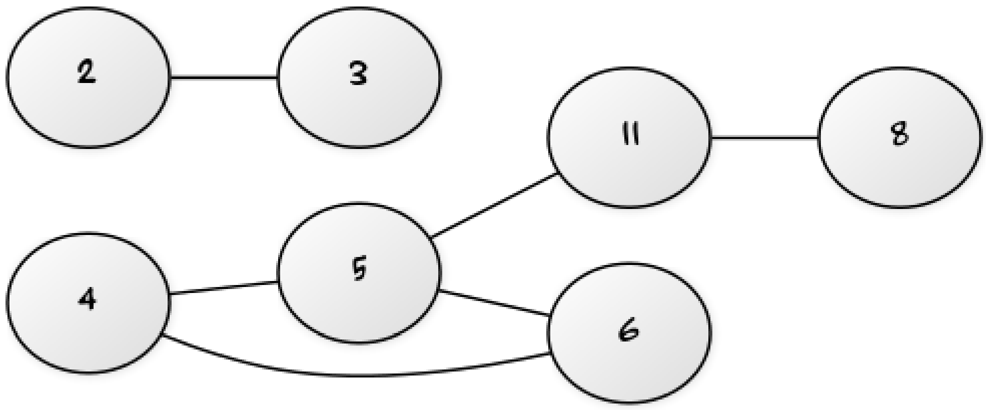

Figura 1 - Conjunto modelado em forma de Grafo

Dessa forma, pode-se observar pelo grafo, que o conjunto de vértices e arestas possuem um clique máximo, que é representado pelos vértices S = {4, 5, 6}. Esse clique máximo representa que esses autores trabalharam juntos várias vezes. Enquanto o conjunto dado por S' = {2, 3}; indica que os autores apenas participaram juntos e possivelmente uma única vez no evento.

2 Implementação do algoritmo exato (*baseline*)

Foram selecionados dois algoritmos exatos (PROSSER, 2012) para o problema do clique máximo. O primeiro algoritmo, MC (FAHLE, 2002), possui uma forma simplista, que computa todos cliques no grafo e ao final define qual o maior deles. O segundo algoritmo, MCQ (TOMITA; SEKI, 2003), apresenta uma forma melhorada do algoritmo anterior ao utilizar um princípio chamado de colaboração de vértices para então descobrir o limite superior do clique máximo (SUYUDI et al., 2014).

O pseudocódigo do Algoritmo MC é apresentado logo a seguir. A solução do algoritmo é armazenada em *cliqueVertice* e o tamanho do clique máximo em *cliqueTam.* O conjunto *C* contém um clique em crescimento e o conjunto *P* contém os candidatos para a solução. Quando um vértice *v* é selecionado em P, sendo adicionado em C, cria então um novo conjunto de vértices candidatos, *novoP*, para qual os vértices serão adjacentes ao vertice *v*. Ao ser notado que o tamanho do conjunto *C* ficou maior do que a solução atual, *cliqueVertice* e *cliqueTam* são atualizados e ao final da execução tem o tamanho e os vértices do clique máximo.

O pseudocódigo do Algoritmo MCQ é apresentado logo a seguir. Como já mencionado, ele é baseado no primeiro algoritmo onde trata a busca pelo clique máximo de forma semelhante, com o diferencial de buscar por um limite superior. A função *EXPAND* é semelhante ao algoritmo MC e ele possui uma função extra, *NUMBERSORT* que determina a coloração dos vértices e armazena esses vértices num conjunto *colour*. A função *NUMBERSORT* é chamado no início da função *EXPAND* que recebe o conjunto *P* atualizado e entrega a saída o mesmo conjunto *P*, já com os vértices ordenados em ordem crescente de acordo com sua colocação.

Os algoritmos *baselines* foram testados utilizando de grafos da coleção DIMACS, muito comuns para análise de *benchmark* para clique máximo. Na Tabela 2 fica evidente a relação dos algoritmos MC e MCQ com os grafos utilizados, é importante salientar que os algoritmos encontraram o clique máximo do mesmo tamanho do que informado pela base coleção DIMACS, isso prova que os *baselines* são capazes de detectar o clique máximo.

- Algoritmo MC

```
int cliqueTam
List cliqueVertice

função expand(Graph g, List C, List P)
        para i ← |P|-1 até 0 faça
                se |C| + |P| ≤ cliqueTam então return
                fim se
                int v ← P(i)
                C ← v
                List novoP
                para u ← 0 até |P| faça
                        se u ∈ adj(v) então
                                novoP ← u
                        fim se
                fim para
                se novoP = ∅ e |C| > cliqueTam então
                        cliqueVertice ← C
                        cliqueTam ← |C|
                fim se
                se novoP != ∅ então
                        EXPAND(g, C, novoP)
                fim se
                C ← remove(v)
                P ← remove(v)
        fim para
fim função
```

Tabela 2 - Testes Preliminares

Grafo	Vértices	Arestas	Clique Máximo	Tempo MC	Tempo MCQ
Hamming8-4	256	20864	16	15601	28658
Mann-A9	45	918	16	1469	2719
Johnson8-4-4	70	1825	14	313	469

- Algoritmo MCQ

```
int cliqueSize
List cliqueVertex

função expand(Graph g, List C, List P)
    NUMBERSORT(g, P)
    para i ← |P|-1 até 0 faça
        se |C| + |P| ≤ cliqueSize então return
        fim se
        int v ← P(i)
        C ← v
        List newP
        para j ← 0 até i faça
            u ← P(j)
            se u ∈ adj(v) então
                newP ← u
            fim se
        fim para
        se newP = ∅ e |C| > cliqueSize então
            cliqueVertex ← C
            cliqueSize ← |C|
        fim se
        se newP != ∅ então
            EXPAND(g, C, newP)
        fim se
        C ← remove(|C|)
        P ← remove(i)
    fim para
fim função

função numberSort(Graph g, List P)
    int colours
    para i ← 0 até |P| faça
        v ← P(i)
        k ← 0
        enquanto adj(v ∈ colour(k)) faça
            k++
        fim enquanto
        colour(k) ← v
        colours ← max(colours, k+1)
    fim para
    P.clear()
    para k ← 0 até colours faça
        para j ← 0 até |colour(k)| faça
            v ← colour(k)(j)
            P ← v
        fim para
    fim para
fim função
```

3 Heurísticas

Pelo emprego de técnicas de *Branch and Bound*, pretende-se determinar os limites superiores e inferiores para tamanho de um clique máximo (JOHNSON, 1973). Desse modo a determinação do limite superior será dado pela coloração dos vértices e o limite inferior pela busca gulosa do clique máximo.

Determinado os valores inferiores e superiores não será necessário buscar por cliques maiores e verificar vértices com grau menor, visto que se o clique máximo será maior ou igual ao limite inferior e não haverá clique maior que o limite superior.

A coloração é fazer com que vértices adjacentes sejam coloridos de formas diferentes, tal que o número cromático é o menor número de cores necessário para colorir todos vértices de um grafo (CORMEN et al., 2001). Assim o número cromático pode ser usado como limite superior.

A busca gulosa parte pelo princípio que o clique máximo pode ser encontrado pelo vértice que possui maior grau. Então a busca se inicia por esse vértice, inserindo-o no conjunto solução e fazendo isso para os próximos vértices que possuem maior grau e não estão no conjunto solução. A cada passo os vértices não adjacentes ao conjunto solução são removidos da busca (SUYUDI et al., 2014).

4 Plano de Experimentos

Após desenvolver os algoritmos propostos, será realizado testes para verificar a eficácia do algoritmo, isso é, se ele é capaz de encontrar o clique máximo. Para tal, será comparado com os algoritmos *baselines* apresentados, com o uso da coleção DIMACS.

Por fim, será aplicado os algoritmos, *baseline* e desenvolvidos, em cima da base de dados do SBBD. Dessa forma calculando o clique máximo da base de dados, problema tratado nesse trabalho.

Referências

CORMEN, T. H. et al. *Introduction to algorithms*. [S.l.]: MIT press Cambridge, 2001. v. 2. page.66

DIMACS, DIMACS Challenge, Disponível em: < http://dimacs.rutgers.edu/Challenges/ >

FAHLE, T. Simple and fast: Improving a branch-and-bound algorithm for maximum clique. In: *In 10th Annual European Symposium on Algorithms (ESA 2002), volume 2461 of LNCS*. [S.l.]: Springer Verlag, 2002. p. 485–498. page.33

JOHNSON, D. S. Approximation algorithms for combinatorial problems. In: ACM. *Proceedings of the fifth annual ACM symposium on Theory of computing*. [S.l.], 1973. p. 38–49. page.33

PROSSER, P. Exact algorithms for maximum clique a computational study tr-2012-33. page.33

SUYUDI, M. et al. Solution of maximum clique problem by using branch and bound method. *Applied Mathematical Sciences*, v. 8, n. 2, p. 81–90, 2014. page.33, page.66

TOMITA, E.; SEKI, T. An efficient branch-and-bound algorithm for finding a maximum clique. In: *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science*. Berlin, Heidelberg: Springer-Verlag, 2003. (DMTCS’03), p. 278–289. ISBN 3-540-40505-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1783712.1783736>. page.33