MAC 5701

Plano de Estudos

Aluno: Renato Lucindo

Orientadora: Profa. Dra. Yoshiko Wakabayashi

Área de Concentração: Otimização Combinatória e Grafos

— São Paulo, 16 de abril de 2005—

Partições Conexas Balanceadas em Grafos

1 Introdução

Grafos são estruturas muito usadas para representar a existência ou não de relações entre elementos de um dado conjunto. Assim, redes de comunicação, fluxos em redes de transporte, mapas geográficos e relações binárias em geral são muito bem representados por grafos, e neste caso, várias questões de interesse prático podem ser investigadas. Por exemplo, qual o seu grau de vulnerabilidade (se a queda de poucas ligações ou vértices causam a perda de conexidade), sua estabilidade (qual o número máximo de vértices independentes que contém), qual o seu diâmetro (maior distância entre quaisquer dois de seus vértices), se podem ser particionados em um certo número de subgrafos conexos, etc.

Essa última questão pode ser de interesse tanto pela sua aplicabilidade direta em diversas situações práticas, mas também no seguinte contexto. Algoritmos para problemas em grafos podem utilizar o seguinte procedimento recursivo: dado um grafo conexo G, particiona G em dois subgrafos conexos de tamanhos aproximadamente iguais, resolve os correspondentes subproblemas para esses subgrafos e de suas soluções obtém uma solução para o grafo original G. A exigência de que a partição seja tão balanceada quanto possível objetiva minimizar a profundidade da recursão e a discrepância dos tamanhos dos grafos usados em qualquer nível do processo de recursão.

Estamos interessados em problemas dessa natureza, onde $G = (V, E)$ é um grafo conexo com pesos associados aos seus vértices. Um dos problemas a ser investigado, chamado *Max q-Partição Conexa Balanceada* (Max-PCB$_q$), é o seguinte: dado um grafo conexo $G = (V, E)$ com uma função peso $w : V \to \mathbb{Z}$ definida sobre seus vértices, encontrar uma q-partição $\{V_1, V_2, \ldots, V_q\}$ de V tal que $G[V_i]$ (grafo induzido por V_i) é conexo para $1 \leq i \leq q$ e o peso do mais 'leve' deles seja o maior possível. Mais formalmente, queremos encontrar uma tal partição que maximiza a função $\min\{w(V_i) : i = 1, \ldots, q\}$, onde $w(S)$ denota o peso de um conjunto S, definido como a soma dos pesos de seus elementos.

2 Algoritmos

Não são muitos os algoritmos que foram desenvolvidos para o Max-PCB$_q$. Quando $q = 2$ e G é um grafo completo o problema é equivalente a uma variante do problema Max Mochila, denominada Max Soma de Subconjuntos, onde os lucros e os pesos são iguais. Como Max Mochila tem um esquema de aproximação completamente polinomial [13], segue que o Max-PCB$_2$ também tem um tal esquema para esta classe de grafos.

Ainda no caso $q = 2$, para grafos conexos (e 2-conexos), o melhor resultado conhecido é um algoritmo de aproximação obtido por Chlebíková [8] que resolve o problema dentro de uma razão de $\frac{4}{3}$. No caso $q = 3$. Salgado e Wakabayashi [19] obtiveram uma 2-aproximação para grafos 3-conexos.

No caso em que o grafo de entrada é uma árvore vários algoritmos polinomiais foram desenvolvidos. Destacamos, em particular, o algoritmo devido a Perl e Schach [18] cuja complexidade de tempo é $O(q^2\, r(T) + q\, n)$, onde $r(T)$ é o raio da árvore de entrada T, para qualquer q fixo.

No caso do Max-PCB$_q$ sem pesos, um resultado devido a Lovász [14] (veja também Györi [9]) garante que é possível obter uma q-partição conexa balanceada em tempo polinomial, quando o grafo é q-conexo.

Em suma, para grafos arbitrários e $q \geq 3$, o problema Max-PCB$_q$ ainda foi pouco investigado.

3 Aplicações

A grande motivação no estudo de algoritmos para a solução desse problema é a sua aplicabilidade em diversas situações práticas, dentre as quais destacamos as seguintes.

A primeira aplicação vem da área de recuperação de informação. Considere o grafo que modela a classificação automática de termos em classes representando termos mais gerais [20]. Cada termo é representado por um vértice do grafo e dois vértices estão conectados se os termos correspondentes são relacionados. O problema consiste em dividir o conjunto de vértices em conjuntos disjuntos de modo que cada conjunto de vértices irá representar um conjunto de termos para formar um termo mais geral. Tal conjunto de termos deve refletir a interrelação entre os diferentes termos no conjunto. Para a classificação em termos mais gerais ter sentido nenhum subconjunto deve ser muito pequeno. Para particionar o grafo em q termos mais gerais precisamos resolver o Problema Max q-Partição Conexa Balanceada.

Uma outra aplicação interessante é em sistemas operacionais [10, 11, 5]. Considere uma árvore enraizada onde os vértices representam procedimentos; sendo que existe uma aresta de um vértice A para um vértice B se o procedimento A chama o procedimento B. O peso de um vértice representa a quantidade de memória exigida para alocar o procedimento correspondente. Se a soma de memória exigida para todos os procedimentos é maior do que o espaço disponível na memória principal é necessário um sistema de paginação [22]. Um sistema de paginação é uma partição dos procedimentos em conjuntos disjuntos chamados páginas. Somente algumas páginas podem estar na memória simultaneamente, devido às limitações de espaço, as demais são armazenadas em memória secundária. Páginas são trocadas entre as memórias principal e secundária de acordo com a utilização de seus procedimentos na hora da execução. As operações de troca são lentas, devido às operações de entrada e saída que precisam ser feitas na memória secundária. Assim, para reduzir a quantidade destas operações, objetiva-se armazenar procedimentos que são chamados um pelo outro na mesma página. Ou seja, o objetivo é particionar a árvore em q subárvores, e definir para cada página uma tal subárvore.

O tamanho da memória de uma página deve ser suficientemente grande para armazenar todos os seus procedimentos. Visto que páginas diferentes são trocadas com outras (na memória principal), devemos alocar para cada página a quantidade de memória da página de maior tamanho, para permitir trocas entre quaisquer páginas. Neste caso, desejamos minimizar a quantidade de memória da página com o máximo tamanho. Ou seja, buscamos uma q-partição onde a componente com o menor tamanho seja o maior possível.

Existem muitas outras aplicações em diferentes áreas como: processamento de imagens [2, 15, 16], análise de *clusters* [17], simulações científicas [21], projeto de redes de telefonia móvel, projeto de circuitos VLSI [1], biologia computacional, etc.

4 Atividades a serem desenvolvidas

Investigaremos inicialmente vários algoritmos que foram desenvolvidos para o caso especial do Max-PCB$_q$ em o grafo de entrada é uma *árvore*.

Encontramos diversas referências bibliográficas a respeito de algoritmos polinomiais para esse problema [18, 12, 6, 4, 7, 3], e em particular à uma técnica (de *shifting*) desenvolvida por Becker e Perl [5] que permite resolver algumas variantes (outras funções objetivo) desse mesmo problema. Nossa pesquisa consistirá em estudar esses algoritmos, entender suas complexidades, e implementar um deles, possivelmente o algoritmos desenvolvido por Perl e Schach [18].

Referências

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–80, 1995.

[2] E. Aparo and B. Simeone. Un algoritmo di equipartizione e il suo impiego in un problema de contrasto ottico. *Ricerca Operativa*, 3:31 – 42, 1973.

[3] R. I. Becker and Y. Perl. Shifting algorithms for tree partitioning with general weighting functions. *J. Algorithms*, 4(2):101–120, 1983.

[4] R. I. Becker and Y. Perl. A shifting algorithm for constrained min-max partition on trees. *Discrete Applied Mathematics*, 45(1):1–28, 1993.

[5] R. I. Becker and Y. Perl. The shifting algorithm technique for the partitioning of trees. *Discrete Appl. Math.*, 62(1-3):15–34, 1995.

[6] R. I. Becker, S. R. Schach, and Y. Perl. A shifting algorithm for min-max tree partitioning. *J. ACM*, 29(1):58–67, 1982.

[7] R.I. Becker and Stephen R. Schach. A bottom-up algorithm for weight- and height-bounded minimal partitions of trees. In *CAAP*, pages 63–72, 1984.

[8] J. Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60:225 – 230, 1996.

[9] E. Györi. On division of graphs to connected subgraphs. In *Combinatorics (Proc. Fifth Hungarian Combinatorial Coll.,1976,Keszthely)*, pages 485 – 494. Bolyai – North-Holland, 1978.

[10] B. W. Kernighan. *Some Graph Partitioning Problems Related to Program Segmentation.* Ph.d. dissertation, Princeton Univ., 1969.

[11] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Sys. Tech. J.*, 49(2):291–308, 1970.

[12] S. Kundu and J. Misra. A linear tree partitioning algorithm. *SIAM J. Comput.*, 6(1):151–154, 1977.

[13] E. L. Lawler. Fast approximation algorithms for knapsack problems. *Mathematics of Operations Research*, 4(4):339 – 356, 1979.

[14] L. Lovász. A homology theory for spanning trees of a graph. In *Acta Math. Acad. Sci. Hunger*, volume 30, pages 241 – 251, 1977.

[15] M. Lucertini, Y. Perl, and B. Simeone. Image enhancement by path partitioning. In V. Cantoni and S. Levialdi, editors, *Recent Issues in Image Analysis*, Lectures Notes in Computer Science. Springer, 1989.

[16] M. Lucertini, Y. Perl, and B. Simeone. Most uniform path partitioning and its use in image processing. *Discrete Applied Mathematics*, 42:227 – 256, 1993.

[17] A. Maravalle, B. Simeone, and R. Naldini. Clustering on trees. *Computational Statistics and Data Analysis*, 24:217 – 234, 1997.

[18] Y. Perl and S. R. Schach. Max-min tree partitioning. *J. ACM*, 28(1):5–15, 1981.

[19] L. R. B. Salgado and Y. Wakabayashi. Approximations results on balanced connected partitions of graphs. *Eletronic Notes in Discrete Mathematics*, 18:207–212, 2004.

[20] G. Salton. *Automatic Information Organization and Retrieval.* McGraw-Hill, 1968.

[21] K. Schloegel, G. Karypis, and V. Kumar. Graph partitioning for high performance scientific simulations. In *The Sourcebook of Parallel Computing*, pages 491–538. Morgan Kaufmann, 1978.

[22] D. C. Tsichritzis and P. A. Bernstein. *Operating Systems.* Academic Press, 1974.