Pivotamento no Algoritmo Bron-Kerbosch para a Detecção de Cliques com Peso Máximo

Samuel Souza Brito[1], **Haroldo Gambini Santos**[1]

[1]Departamento de Computação – Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto (UFOP)
CEP: 35.400-000 – Minas Gerais – MG – Brasil

samuelsouza@iceb.ufop.br, haroldo@iceb.ufop.br

Abstract. *Finding the maximum weight clique is an NP-Hard problem, where one wants to find a complete subgraph with the highest weight in a graph. This paper considers a variant of this problem, where the objective is to find all cliques having height above a certain threshold. We propose a Bron-Kerbosch algorithm adaptation for computing cliques with weight in vertexes. As this algorithm becomes inefficient for graphs with many non-maximal cliques, some vertex pivoting methods were developed and analyzed to speedup the return from branches containing a reduced number of feasible solutions. Computational results for a set of test problems from the MIPLIB e DIMACS repository will be presented and discussed for each implemented method.*
KEYWORDS. Maximum Weight Clique Problem. Bron-Kerbosch Algorithm. Backtracking.
MAIN AREA. Combinatorial Optimization. Graph Theory and Algorithms. Mathematical Programming.

Resumo. *Encontrar a clique de peso máximo é um problema NP-Difícil, onde o objetivo é encontrar o subgrafo completo com maior peso num grafo. Este trabalho considera uma variante do problema, onde o objetivo é encontrar todos os cliques com peso acima de um limar. Propomos uma adaptação do algoritmo Bron-Kerbosch para computar cliques com peso nos vértices. Como este algoritmo se torna ineficiente para grafos com muitas cliques não maximais, são desenvolvidos e analisados alguns métodos de pivotamento dos vértices para acelerar o retorno de ramos com um número reduzido de soluções factíveis. Resultados computacionais para um conjunto de problemas teste dos repositórios MIPLIB e DIMACS serão apresentados e discutidos para cada método implementado.*
PALAVRAS CHAVE. Problema da clique com peso máximo. Algoritmo Bron-Kerbosch. Backtracking.
ÁREA DE CLASSIFICAÇÃO. Otimização Combinatória. Teoria e Algoritmos em Grafos. Programação Matemática.

1 Introdução

Dado um grafo não direcionado $G = (V, A)$, com um conjunto de vértices V e um conjunto de arestas A, então $G_1 = (V_1, A_1)$ denomina-se subgrafo de G se $V_1 \subseteq V$ e $A_1 \subseteq A$, onde cada aresta de A_1 incide nos vértices de V_1. Um subgrafo é completo se existir uma aresta para todos os pares de vértices. O subgrafo completo é denominado

clique. Por sua vez, uma clique é maximal se não está contida em outra clique. O problema de encontrar cliques de peso máximo é tão difícil quanto o problema da clique máxima, que é NP-Difícil [Garey (1979)].

No problema considerado neste artigo, deseja-se detectar as cliques maximais que satisfaçam o peso mínimo exigido. O peso de uma clique é calculado pela soma dos pesos dos vértices que a compõe. O método utilizado para tal problema foi o uso de uma extensão do algoritmo Bron-Kerbosch. Esse algoritmo encontra todas as cliques maximais de um grafo e ao mesmo tempo utiliza a técnica *branch-&-bound* para cortar os ramos que não levam a uma solução factível.[Bron (1973)]

2 Problema da Clique com Peso Máximo

O objetivo do PCPM consiste em determinar num dado grafo a clique de maior peso. Detectar todas as cliques do grafo para encontrar a que possui peso máximo implica em uma grande ineficiência. Por isso, muitas heurísticas para resolver esse problema têm sido desenvolvidas, como por exemplo, o uso do *GRASP* [Feo (1994)], da Busca Tabu [Cavique (2002)], de Algoritmos Genéticos [Marchiori (1998)], de *RLS* [Battiti (2001)] e de métodos exatos [Sørensen (2004)] e [Östergård (2001)].

Neste trabalho consideramos uma variante do PCPM: dado um grafo G e um inteiro $k > 0$, o objetivo é encontrar as cliques maximais que tenham peso maior ou igual à k. No contexto de Programação Inteira encontrar todas as cliques com peso acima de um limiar corresponde ao problema de encontrar todas as desigualdades violadas de clique. Esses cortes desempenham um papel fundamental na descoberta de desigualdades fortes [Chvátal (1973)]. Apesar do problema de separação de cortes ser formulado em termos de se encontrar a desigualdade mais violada, experimentalmente se verificou que mais importante do que a descoberta dessa desigualdade é a descoberta de um grande conjunto de cortes violados. Essa descoberta permitiu ganhos de desempenho bastante grandes e renovou o interesse nas desigualdades de Gomory [Cornuéjols (2007)]. Nesse sentido, métodos eficientes para a descoberta da clique mais violada como o método de Ostergard (2001), por exemplo, podem ter um desempenho pobre se usados como rotinas de separação de cortes, visto que a poda de soluções sub-ótimas pode limitar o número de desigualdades violadas encontradas.

3 Algoritmo Bron-Kerbosch Básico

Proposto por Coenraad Bron e Joep Kerbosch em 1973, a base desse algoritmo é a busca exaustiva, mas utiliza-se um conhecimento maior sobre a natureza do problema da clique máxima. A grande vantagem desse algoritmo é que ele gera apenas cliques maximais, evitando assim que cada conjunto gerado tenha que ser comparado com os previamente testados.

O algoritmo de Bron-Kerbosch é caracterizado pelo *backtraking*, que procura por todas as cliques maximais em um dado grafo G. Existem três estruturas que são a base do algoritmo:

C : conjunto de vértices já definidos como parte da clique;
P : vértices que têm ligação com todos os vértices de C; esse é o conjunto de candidatos a entrar na clique;

S : vértices que já foram analisados e não levam a uma extensão do conjunto de candidatos P, cujo objetivo é evitar comparação excessiva.

A execução do algoritmo é feita inicialmente com C e S vazios e P contendo todos os vértices do grafo. Dentro de cada chamada recursiva, se o conjunto P está vazio, o algoritmo encontra uma clique maximal (se S também estiver vazio) ou realiza *backtracking*. Para cada vértice v escolhido de P, faz-se uma chamada recursiva, adicionando v em C. Na recursão, os conjuntos P e S são restritos aos vizinhos de v, possibilitando encontrar todas as extensões da clique C que contêm v . Continuando a execução, move-se v de P para S, uma vez que v já foi analisado. Várias recursões são chamadas, até que se encontrem todas as cliques maximais do grafo.

Em Algoritmo 1 a versão básica do algoritmo de Bron-Kerbosch é apresentada. As linhas 3-5 indicam a verificação adicional necessária.

Algoritmo 1: Bron-Kerbosch Básico Adaptado

```
1  BK(C, P, S)
2  se P e S estão vazios então
3      se ω(C) ≥ PesoMin então
4          Adiciona a clique C no conjunto solução;
5      fim
6  fim
7  para cada Vértice v em P faça
8      BK(C ∪ {v}, P ∩ N(v), S ∩ N(v));
9      P := P\{v};
10     S := S\{v};
11 fim
```

Vale ressaltar que no pior caso, o algoritmo de Bron-Kerbosch apresenta complexidade exponencial O($2^{|V|}$).

4 Algoritmo Bron-Kerbosch com Pivotamento

A forma básica do algoritmo é ineficiente no caso de grafos com muitas cliques não maximais: faz uma chamada recursiva para cada clique, maximal ou não. Para poupar tempo e permitir que o algoritmo recue mais rapidamente dos ramos da pesquisa que não contêm cliques maximais, Bron e Kerbosch introduziram uma variante do algoritmo que envolve um "vértice pivô" u, escolhidos de P (ou de $P \cup S$) [Bron (1973)].

Qualquer clique maximal deve incluir tanto u ou um dos vértices não adjacentes a ele, pois caso contrário a clique poderia ser aumentada. Significa que se uma clique contém um vértice adjacente a u, necessariamente ele contém u. Portanto, só u e os vértices não adjacentes a ele precisam de ser testados como as escolhas para o vértice v, que é adicionado à C, em cada chamada recursiva do algoritmo.

O grande desafio do algoritmo com pivotamento é encontrar boas estratégias de seleção para o vértice pivô. Para isso, foram implementadas e avaliadas computacionalmente três métodos de seleção de pivô, que serão explorados a seguir. Contudo, como desejamos somente cliques que satisfaçam a condição do peso mínimo, podemos cortar alguns ramos em determinado ponto da execução, uma vez que estas não têm chance alguma de gerar soluções aceitáveis. Para implementar a poda, utilizamos uma estimativa de peso máximo que reduz a geração de cliques maximais com peso abaixo do limiar desejado.

Algoritmo 2: Bron-Kerbosch com Pivotamento e Adaptação

```
1  procedimento BK2(C, P, S)
2  se P e S estão vazios então
3      se ω(C) ≥ PesoMin então
4          Adiciona a clique C no conjunto solução;
5      fim
6  fim
7  Escolha um vértice pivô u de P ∪ S ;
8  para cada Vértice v em P\N(u) faça
9      BK2(C ∪ {v}, P ∩ N(v), S ∩ N(v));
10     P := P\{v};
11     S := S\{v};
12 fim
```

4.1 Pivotamento Aleatório

O primeiro método de pivotamento desenvolvido foi a seleção aleatória do vértice pivô. Esse método consiste em, a cada iteração, selecionar um vértice do conjunto P para ser o pivô. Apesar de aleatório, o método gerou boas soluções se comparadas ao Bron-Kerbosch determinístico, tendo bom desempenho até mesmo se comparado aos demais métodos desenvolvidos.

4.2 Pivotamento pelo Vértice de Grau Máximo

Muitos dos algoritmos que utilizam GRASP [Resende (2003)] para resolver o Problema da Clique Máxima realizam, em sua fase de construção, manipulações que visam a seleção de vértices que possuem os maiores graus do grafo. Isso se deve ao fato de que quanto maior o grau de um vértice, maior é a sua probabilidade de formar uma clique grande. Baseado nisso, foi criado um método de pivotamento no algoritmo Bron-Kerbosch que seleciona vértice de maior grau. De acordo com a descrição do pivotamento do algoritmo, pode-se perceber que quanto maior for o grau do vértice pivô, menor será a lista candidatos analisada, diminuindo o número de chamadas recursivas e consequentemente melhorando seu desempenho.

A seleção do vértice pivô, a cada iteração, é dada pela seguinte fórmula:

$$i_p \in P : d(i_p) \geq d(i) \quad \forall i \in P$$

ou seja, o vértice pivô será o vértice de maior grau contido no conjunto P.

4.3 Pivotamento pela Estratégia do Máximo Grau Modificado

A escolha do vértice de maior grau como pivô pode falhar significativamente, pois um grafo pode ter os vértices de maiores graus ligados a vértices com poucos vizinhos, o que levaria a formação de uma clique menor. Por esse motivo, mais importante do que analisar o grau de um único vértice é analisar os graus de todos os seus vizinhos, para que se garanta a maior probabilidade possível de gerar uma clique grande. Nesse contexto surgiu o novo método: o grau modificado ($wdeg_i$), que valorizará, além do grau de um vértice, os graus dos vértices adjacentes. O grau modificado de um vértice é o somatório dos graus de todos os seus vizinhos juntamente com seu respectivo grau, ou seja:

$$wdeg_i = deg(i) + \sum_{j \in N(i)} d(j)$$

onde $N(i)$ representa o conjunto dos vértices adjacentes a i.

Esse cálculo é feito para todo vértice do grafo, antes da execução do algoritmo. Em seguida, dentro do algoritmo Bron-Kerbosch, o vértice selecionado será aquele que possuir o maior grau modificado.

4.4 Técnica de Estimativa do Peso

Além do pivotamento no algoritmo Bron-Kerbosch, outra forma de melhorar seu desempenho é através da eliminação de determinados ramos antes do processamento de cliques maximais que não satisfaçam o peso mínimo. Isso é possível através da determinação do peso máximo que uma clique parcialmente construída poderá atingir.

Sendo assim, dado o peso da Clique parcialmente formada, $\omega(C)$, estimamos o peso máximo que ela poderá atingir, $h(C)$. A estimativa implementada para testes se deu através do somatório dos pesos de todos os vértices contidos em P, caracterizando-a como um limite superior para o peso da clique. Esse método foi aplicado juntamente com a seleção de pivô pelo maior grau modificado, devido a seu bom desempenho nos testes realizados.

Algoritmo 3: Pivotamento pelo Maior Grau Modificado + Estimativa de Peso

```
1  procedimento BK3(C, P, S)
2  se P e S estão vazios então
3      se ω(C) ≥ PesoMin então
4          Adiciona a clique C no conjunto solução;
5      fim
6  fim
7  se ω(C) + h(C) ≥ PesoMin então
8      Escolha o vértice de maior grau modificado de P ∪ S como pivô u ;
9      para cada Vértice v em P\N(u) faça
10         BK3(C ∪ {v}, P ∩ N(v), S ∩ N(v));
11         P := P\{v};
12         S := S\{v};
13     fim
14 fim
```

5 Experimentos Computacionais

Para analisar o desempenho dos métodos descritos foram utilizados 30 grafos com características apresentadas na Tabela 1. Esses grafos foram coletados de várias instâncias da biblioteca de problemas de Programação Inteira MIPLIB [Achterberg (2006)]. Especificamente, esses grafos correspondem a separação de cortes de clique para a relaxação linear do nó raiz.

Os algoritmos foram implementados em C++, utilizando o compilador GCC versão 4.4 e executados em uma máquina com Sistema Operacional Ubuntu 10.10, processeador Intel Core i5 3.2Ghz e 16GB RAM.

O critério de parada de cada algoritmo, quando necessário, foi dado pelo tempo de execução, apresentado em segundos. Estipulou-se que esse valor fosse de 300 segundos, denotado na tabela de experimento como "T.L.E."(Tempo Limite Excedido). Tempos muito pequenos são indicados pela expressão "$<$ *0,001*".

A grande ineficiência do algoritmo sem pivotamento pode ser observada através dos resultados de instâncias como *ns1686196*, *ns1688347* e *ns1745726*, que chegam a

Dados das Instâncias				
Instância	$\|V\|$	$\|E\|$	$\Delta(G)$	**Peso Total Encontrado**
10teams	160	925	18	47885
air04	291	1343	21	33888
air05	220	1223	24	27248
bab1	112	136	6	15238
brock400-1	400	20077	127	2723484
brock400-2	400	20014	125	3013382
C500-9	499	12363	67	245715
dsjc500-5	500	62126	279	92109405
eil33.2	30	288	29	21223
eilA101.2	69	859	37	56874
momentum1	210	440	25	3655
mzzv11	903	2931	33	56047
neos-1208135	326	1012	11	21575
neos-1605061	753	3784	21	12734
neos-693347	364	2021	22	19750
neos-807456	805	2688	12	15259
neos-849702	459	3387	23	103218
neos-933638	1335	2756	16	28834
neos788725	167	587	12	152114
ns1686196	48	482	33	8573
ns1688347	100	804	44	10345
ns1745726	49	584	37	9983
ns1769397	92	1404	42	26423
p6b	462	5852	48	7760500
pw-myciel4	578	5627	46	317359
san400-5-1	276	18867	160	72194959
sanr400-5	399	39601	238	157853594
sp100_500_50_1	100	2000	60	795502
t1722	320	3869	52	69792
uct-subprob	217	670	19	66844

Tabela 1. Dados das Instâncias

exceder o tempo limite, mas quando aplicadas aos outros algoritmos com pivotamento, são executadas em menos de um milésimo de segundo.

Os métodos de pivotamento aleatório e pelo maior grau tiveram um desempenho razoável, tanto em relação ao número de chamadas recursivas quanto ao tempo de execução. As exceções são nas instâncias *dsjc500-5* e *sanr400-5*, as quais nenhum algoritmo descrito neste artigo consegue executar em menos de 300 segundos. Esses são grafos com densidade maior que 50%.

Como pode ser observado na tabela, a estratégia de estimar o peso máximo que a clique pode atingir em conjunto com o pivotamento pelo maior grau modificado possui o menor número de chamadas recursivas executadas pelos algoritmos. Essa eficiência pode ser comprovada pela melhora obtida pivotando o algoritmo com o maior grau modificado juntamente com a estratégia de eliminar ramos que não levariam a cliques com o peso exigido. Desse modo, o algoritmo passa a realizar *backtracking* mais rapidamente, diminuindo a altura final da árvore de soluções geradas.

6 Conclusões

A estratégia de estimar o peso da clique mesclada com pivotamento pelo maior grau modificado teve o melhor desempenho em relação às outras tentativas de melhoria,

	Sem Pivotamento		Aleatório		Maior Grau		Grau Modificado		Estimativa de Peso + Grau Modificado	
Instância	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo
10teams	3071	< 0,001	1763	0,010	1896	< 0,001	1875	< 0,001	1240	0,010
air04	5780	0,010	2642	< 0,001	2404	0,010	2699	0,010	1579	< 0,001
air05	5514	< 0,001	2591	< 0,001	2319	0,010	2450	< 0,001	1379	0,010
bab1	320	< 0,001	259	< 0,001	248	< 0,001	247	< 0,001	180	< 0,001
brock400-1	543168	1,030	422933	0,970	425390	1,160	415408	1,090	278011	0,800
brock400-2	536458	1,020	416275	0,990	420616	1,150	412909	1,090	273167	0,780
C500-9	35252	0,060	32508	0,080	32677	0,080	32335	0,080	26206	0,080
dsjc500-5	137005882	T.L.E.	86174772	T.L.E.	66486485	T.L.E.	70747746	T.L.E.	65699086	T.L.E.
eil33.2	160416	0,300	995	0,010	489	< 0,001	181	< 0,001	179	< 0,001
eilA101.2	2147937	4,760	2756	< 0,001	3442	0,020	3044	0,010	2175	< 0,001
momentum1	1266	< 0,001	760	< 0,001	651	< 0,001	712	< 0,001	386	< 0,001
mzzv11	102254	0,180	6789	0,020	5605	0,030	6429	0,020	3292	0,030
neos-1208135	2077	< 0,001	1555	< 0,001	1566	< 0,001	1613	< 0,001	1157	0,010
neos-1605061	18495	0,020	6829	0,020	6362	0,030	7216	0,020	3629	0,020
neos-693347	7762	< 0,001	3878	0,010	3735	0,010	3988	0,010	2200	0,010
neos-807456	5491	0,010	4255	0,020	4151	0,020	4216	0,020	2981	0,010
neos-849702	20093	0,030	7783	0,010	7555	0,020	8223	0,020	4838	0,020
neos-933638	5813	0,030	4427	0,030	4255	0,040	4528	0,040	3287	0,030
neos788725	2238	< 0,001	850	< 0,001	840	< 0,001	1041	< 0,001	771	< 0,001
ns1686196	94713669	T.L.E.	127	< 0,001	533	< 0,001	96	< 0,001	94	< 0,001
ns1688347	79338099	T.L.E.	1474	< 0,001	902	0,010	218	< 0,001	145	< 0,001
ns1745726	87716597	T.L.E.	115	< 0,001	636	< 0,001	94	< 0,001	85	< 0,001
ns1769397	77419959	T.L.E.	1677	< 0,001	1579	0,020	2129	< 0,001	479	0,010
p6b	105997	0,150	25362	0,070	26709	0,070	34308	0,080	34293	0,080
pw-myciel4	15908410	40,820	11110	0,040	13847	0,060	21051	0,070	10007	0,040
san400-5-1	105250121	T.L.E.	38606247	165,720	56344228	295,240	61119351	T.L.E.	8779268	48,960
sanr400-5	141598787	T.L.E.	102840070	T.L.E.	79946261	T.L.E.	82781058	T.L.E.	74626852	T.L.E.
sp100_500_50_1	112448	0,180	37693	0,070	41576	0,100	32839	0,070	16182	0,040
t1722	57551	0,080	14800	0,030	13416	0,040	15086	0,030	6077	0,020
uct-subprob	1871	< 0,001	1271	< 0,001	1094	< 0,001	1307	< 0,001	867	< 0,001

Tabela 2. Experimentos Realizados

contudo, uma possível desvantagem desse método é o fato de calcular a cada chamada recursiva a função que estima o peso. Em algumas instâncias o tempo gasto no cálculo desse limite teve um impacto significativo e não compensou o ganho na poda de ramos.

Os experimentos mostram que apesar de ser um algoritmo exponencial no pior caso, o algoritmo de Bron-Kerbosch é uma alternativa bastante rápida em geral para o uso em rotinas de separação de cortes considerando Problemas de Programação Inteira. A determinação de bons critérios para escolha de sub-problemas na árvore de busca permite ainda que mesmo para execuções limitadas por tempo ou por chamadas recursivas o algoritmo encontre desigualdades violadas nos primeiros estágios da busca. Experimentos nesse sentido serão realizados em trabalhos futuros.

Referências

Achterberg, T., Koch, T., and Martin, A. (2006). Miplib 2003. *Operations Research Letters*, 34(4):361–372.

Battiti, R. and Protasi, M. (2001). Reactive local search for the maximum clique problem. *Algorithmica*, 29:610–637.

Bron, C. and Kerbosch, J. (1973). Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16:575–577.

Cavique, L., Rego, C., and Themido, I. (2002). Estruturas de vizinhança e procura local no problema da clique máxima. *Investigação Operacional*, 22:1–18.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337.

Cornuéjols, G. (2007). Revival of the gomory cuts in the 1990´s. *Annals of Operations Research*, 149(1):63–66.

Feo, T., Resende, M., and Smith, S. (1994). A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Marchiori, E. (1998). A simple heuristic based genetic algorithm for the maximum clique problem. In *Proc. ACM Symposium on Applied Computing*, pages 366–373. ACM Press.

Östergård, P. R. J. (2001). A new algorithm for the maximum-weight clique problem. *Nordic J. of Computing*, 8:424–436.

Resende, M. G. C. and Ribeiro, C. C. (2003). Greedy randomized adaptive search procedures. In Glover, F. and Kochenberger, G., editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 219–249. Springer New York.

Sørensen, M. M. (2004). New facets and a branch-and-cut algorithm for the weighted clique problem. *European Journal of Operational Research*, 154:57–70.