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

Samuel Souza Brito Haroldo Gambini Santos

Departamento de Computação
Universidade Federal de Ouro Preto

XIX Seminário de Iniciação Científica, 2011

Sumário

Definições

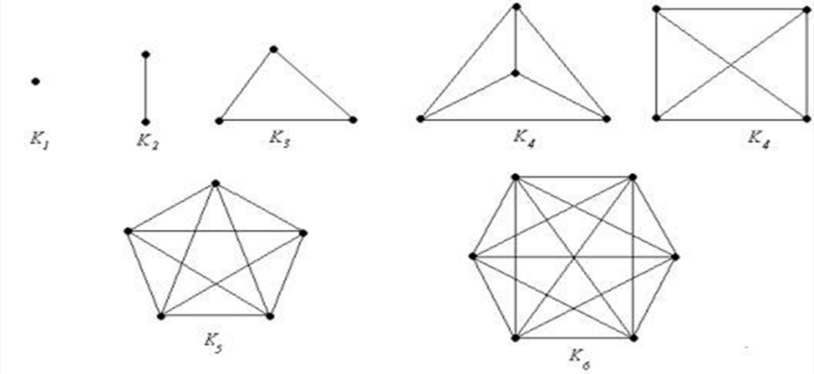
- Dado um grafo não direcionado $G = (V, A)$, com um conjunto de vértices V e um conjunto de arestas A, então $G_1 = (V_1, A_1)$ denomina-se subgrafo de G se $V_1 \subseteq V$ e $A_1 \subseteq A$.
- Um subgrafo é completo se existir uma aresta para todos os pares de vértices. O subgrafo completo é denominado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

Definições

- Uma clique é maximal se não está contida em outra clique.
- O peso de uma clique é a soma dos pesos dos vértices que a compõe.
- O problema de encontrar cliques com peso máximo é tão difícil quanto o problema da clique máxima, que é NP-Difícil.[Garey (1979)]

Definições

Exemplo



Aplicações

- **Programação Inteira**
- Biologia Computacional (Distância de Edição)[Mitchell (1990)]
- Recuperação da Informação
- Transmissão de Sinais [Horaud (1989)]
- ...

Problema da Clique com Peso Máximo

- O objetivo do PCPM consiste em determinar num dado grafo a clique de maior peso.
- 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.

UFOP

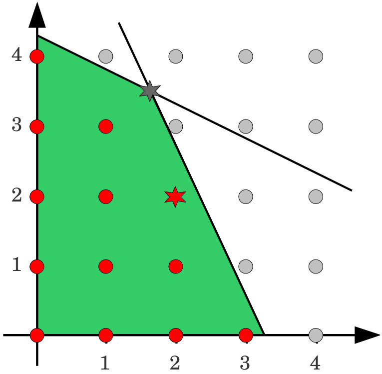
Problema da Clique com Peso Máximo

- No contexto de Programação Inteira, encontrar todas as cliques com um peso acima de um limiar corresponde ao problema de encontrar todas as desigualdades violadas.
- 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.

Cortes em Programação Inteira

Maximize: $6x_1 + 5x_2$
Sujeito a:

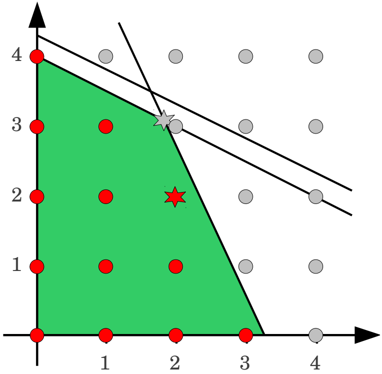
1. $15x_1 + 7x_2 \leq 49$
2. $2x_1 + 4x_2 \leq 17$
3. $x_1, x_2 \in \mathbb{Z}^+$



Cortes em Programação Inteira

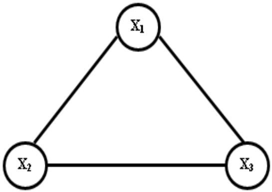
Maximize: $6x_1 + 5x_2$
Sujeito a:

1. $15x_1 + 7x_2 \leq 49$
2. $2x_1 + 4x_2 \leq 17$
3. $\mathbf{x_1 + 2x_2 \leq 8} \Rightarrow$ **Corte inserido**
4. $x_1, x_2 \in \mathbb{Z}^+$



Melhorando uma Formulação

Maximize $z = x_1 + x_2 + x_3$
Sujeito a: $x_1 + x_2 = 1$
$x_2 + x_3 = 1$
$x_1 + x_3 = 1$
$x_1, x_2, x_3 \in \{0, 1\}$



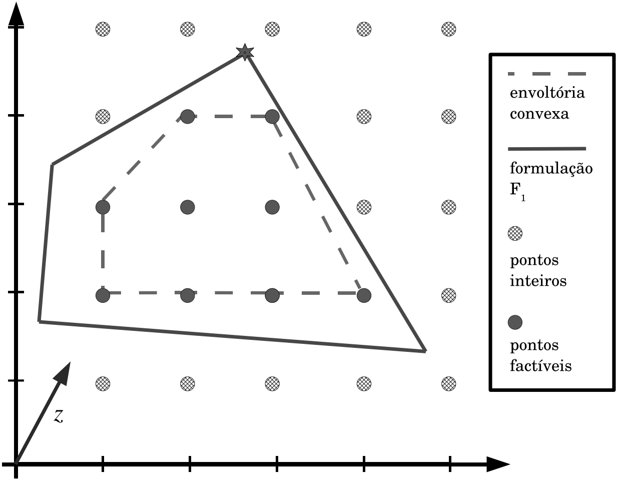
Ótimo da Relaxação:

$$x = \{0.5, 0.5, 0.5\} \; z = 1.5$$

Corte de Clique:

$$x_1 + x_2 + x_3 = 1 \text{ (violação de 0.5)}$$

Resumindo...


z
envoltória
convexa
formulação
F_1
pontos
inteiros
pontos
factíveis

Algoritmo Bron-Kerbosch

- 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.
- Gera apenas cliques maximais, evitando assim que cada conjunto gerado tenha que ser comparado com os previamente testados. [Bron (1973)]
- Caracterizado pelo *backtracking*.

Estrutura

- **Conjunto** C: vértices já definidos como parte da clique.
- **Conjunto** P: vértices que têm ligação com todos os vértices de C (candidatos).
- **Conjunto** S: vértices já analisados e que não levam a uma extensão do conjunto C. Usado para evitar comparação excessiva.

Execução

- Chamada inicial: C e S vazios e P contendo todos os vértices do grafo.
- Em cada chamada recursiva, se P está vazio, uma clique maximal é encontrada (se S também estiver vazio). Se S não estiver vazio, o algoritmo realiza *backtracking*.
- Para cada vértice v escolhido, faz-se uma chamada recursiva adicionando v em C.
- Os conjuntos P e S são restritos aos vizinhos de v, possibilitando encontrar todas as extensões de C que contém v.

Execução

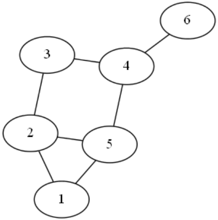
- Quando todas as extensões de C que contém v foram analisados, v é movido de P para S.
- Assim, todos os cliques maximais contidos no grafo são encontrados.
- No pior caso, o algoritmo apresenta complexidade exponencial $O(2^{|V|})$.

Pseudocódigo

Algoritmo 1: Bron-Kerbosch Básico Adaptado

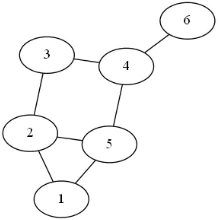
BK(C**,** P**,** S**)**
se *P e S estão vazios* **então**
    se $\omega(C) \geq PesoMin$ **então**
        Adiciona a clique C no conjunto solução;
    fim
fim
para cada *Vértice v em P* **faça**
    BK($C \cup \{v\}$, $P \cap N(v)$, $S \cap N(v)$);
    $P := P \setminus \{v\}$;
    $S := S \setminus \{v\}$;
fim

Exemplo

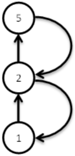


Execução

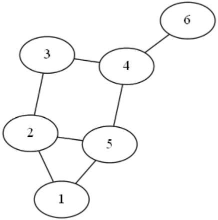
Exemplo



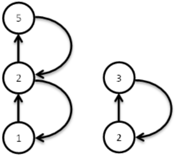
Execução



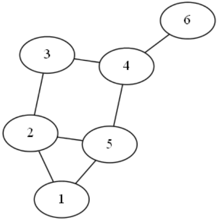
Exemplo



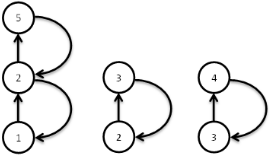
Execução



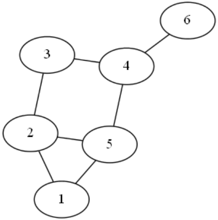
Exemplo



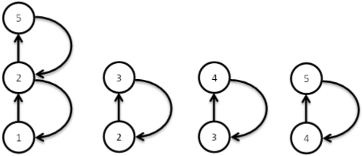
Execução



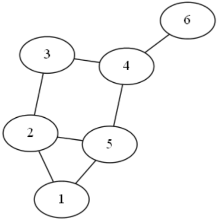
Exemplo



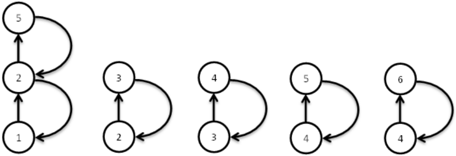
Execução



Exemplo



Execução



Pivotamento

- O algoritmo básico é ineficiente para grafos com muitas cliques não maximais: uma chamada para cada clique, maximal ou não.
- Para economizar tempo e permitir o recuo mais rápido dos ramos que não levam a cliques maximais, os criadores do algoritmo introduziram o pivotamento de vértices.
- O vértice pivô u é escolhido de P (ou de $P \cup S$).[Bron (1973)]
- Somente u e os vértices não adjacentes a ele precisam ser testados como escolhas para o vértice v.

Pivotamento

- O grande desafio do algoritmo com pivotamento é encontrar boas estratégias de seleção para o vértice pivô.
- Buscando a melhoria do algoritmo, foram implementados e avaliados computacionalmente três métodos de seleção de pivô.
- Criada uma nova estratégia para podar ramos com soluções infactíveis.

Pivotamento Aleatório

- Consiste na seleção aleatória de um vértice pivô.
- A cada iteração um novo vértice pivô é selecionado aleatoriamente, de P.

Pivotamento pelo Vértice de Grau Máximo

- A cada iteração, é selecionado o vértice de maior grau do conjunto P como pivô.
- Quanto maior o grau de um vértice, maior é a sua probabilidade de formar uma clique grande.
- Quanto maior for o grau do vértice pivô, menor será a lista de candidatos analisada, diminuindo o número de chamadas recursivas.

Pivotamento pelo Vértice de Máximo Grau Modificado

- Utilizar o vértice de maior grau pode não gerar melhorias no algoritmo.
- Mais importante do que analisar o grau de um vértice é analisar os graus dos vértices adjacentes à ele.
- A cada iteração o pivô selecionado é o que possui o maior grau modificado do conjunto P.
- Cálculo do Grau Modificado:

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

Estimativa do Peso

- Outra forma de melhorar o desempenho do algoritmo é através da eliminação de alguns ramos antes do processamento de cliques maximais que não satisfaçam o peso mínimo.
- Estimando o peso máximo que a clique poderá atingir, essa melhoria pode ser aplicada ao algoritmo.
- O peso estimado da clique é calculado através de uma função de limite superior, $h(C)$, onde:

$$h(c) = \sum_{i \in P} w(i)$$

- Essa técnica foi aplicada juntamente com o pivotamento pelo Máximo Grau Modificado.

UFOP

Pseudocódigo

Algoritmo 2: Pivotamento pelo Maior Grau Modificado + Estimativa de Peso

BK(C, P, S)
se P *e* S *estão vazios* **então**
    se $\omega(C) \geq PesoMin$ **então**
        Adiciona a clique C no conjunto solução;
    fim
fim
se $\omega(C) + h(C) \geq PesoMin$ **então**
    Escolha o vértice de máximo grau modificado em P como pivô u;
    para cada *Vértice v em* $P \backslash N(u)$ **faça**
        BK($C \cup \{v\}$, $P \cap N(v)$, $S \cap N(v)$);
        $P := P \backslash \{v\}$;
        $S := S \backslash \{v\}$;
    fim
fim

Resultados

- **Utilizados 30 grafos coletados de várias instâncias da MIPLIB.[Achterberg (2006)]**

Resultados

- Utilizados 30 grafos coletados de várias instâncias da MIPLIB.[Achterberg (2006)]
- **Especificamente, esses grafos correspondem a separação de cortes de clique para a relaxação linear do nó raiz.**

Resultados

- Utilizados 30 grafos coletados de várias instâncias da MIPLIB.[Achterberg (2006)]
- Especificamente, esses grafos correspondem a separação de cortes de clique para a relaxação linear do nó raiz.
- **Algoritmos implementados em C++.**

Resultados

- Utilizados 30 grafos coletados de várias instâncias da MIPLIB.[Achterberg (2006)]
- Especificamente, esses grafos correspondem a separação de cortes de clique para a relaxação linear do nó raiz.
- Algoritmos implementados em C++.
- **Tempo de execução máximo fixado em 300 segundos.**

Resultados

- Utilizados 30 grafos coletados de várias instâncias da MIPLIB.[Achterberg (2006)]
- Especificamente, esses grafos correspondem a separação de cortes de clique para a relaxação linear do nó raiz.
- Algoritmos implementados em C++.
- Tempo de execução máximo fixado em 300 segundos.
- **O tempo de execução final de cada instância, para o pivotamento aleatório, é dado pela média das 10 execuções realizadas.**

Resultados

- **Em 5 instâncias, o algoritmo sem pivotamento excedeu o tempo de execução. Dentre essas, 3 foram executadas em menos de um milésimo de segundo pelo pivotamento por Máximo Grau Modificado.**

Resultados

- Em 5 instâncias, o algoritmo sem pivotamento excedeu o tempo de execução. Dentre essas, 3 foram executadas em menos de um milésimo de segundo pelo pivotamento por Máximo Grau Modificado.
- **A estratégia do máximo grau modificado foi superior às demais em 26 dos problemas.**

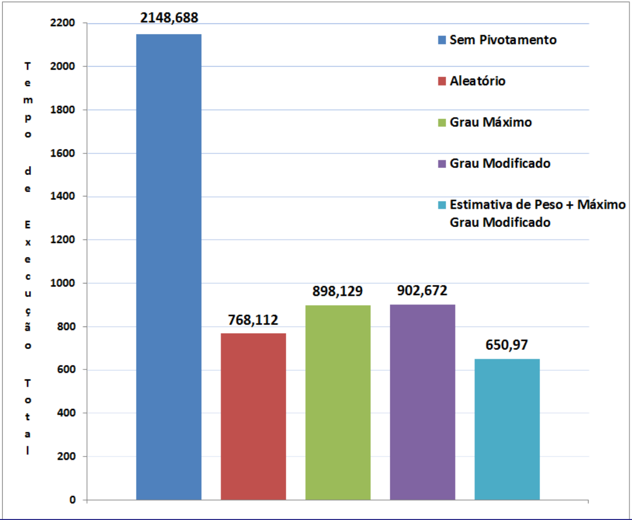
Resultados

- Em 5 instâncias, o algoritmo sem pivotamento excedeu o tempo de execução. Dentre essas, 3 foram executadas em menos de um milésimo de segundo pelo pivotamento por Máximo Grau Modificado.
- A estratégia do máximo grau modificado foi superior às demais em 26 dos problemas.
- **2 instâncias excederam limite de tempo para todos os algoritmos implementados. Essas instâncias possuiam densidade maior que 50%.**

Resultados

- Em 5 instâncias, o algoritmo sem pivotamento excedeu o tempo de execução. Dentre essas, 3 foram executadas em menos de um milésimo de segundo pelo pivotamento por Máximo Grau Modificado.
- A estratégia do máximo grau modificado foi superior às demais em 26 dos problemas.
- 2 instâncias excederam limite de tempo para todos os algoritmos implementados. Essas instâncias possuiam densidade maior que 50%.
- **Em algumas instâncias o tempo gasto no cálculo da estimativa de peso teve um impacto significativo e não compensou o ganho na poda de ramos.**

Resultados - Tempo de Execução Total



Resultados

Instância	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo	Chamadas	Tempo
brock400-1	543168	1,030	422933	0,970	425390	1,160	415408	1,090	278011	0,800
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
leilD76.2	3521793	8,110	4011	0,010	6335	0,030	1356	< 0,001	1090	0,010
mzzv11	102254	0,180	6789	0,020	5605	0,030	6429	0,020	3292	0,030
neos-1440225	449390	0,770	42722	0,110	46112	0,120	46220	0,120	22512	0,070
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
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

UFOP

Conclusões e Trabalhos Futuros

- Apesar de exponencial no pior caso, o algoritmo de Bron e Kerbosch é uma alternativa bastante rápida se utilizado com pivotamento.

Conclusões e Trabalhos Futuros

- Apesar de exponencial no pior caso, o algoritmo de Bron e Kerbosch é uma alternativa bastante rápida se utilizado com pivotamento.
- A estratégia de estimar o peso da clique mesclada com pivotamento pelo máximo grau modificado teve o melhor desempenho em relação às outras tentativas de melhoria.

Conclusões e Trabalhos Futuros

- Apesar de exponencial no pior caso, o algoritmo de Bron e Kerbosch é uma alternativa bastante rápida se utilizado com pivotamento.
- A estratégia de estimar o peso da clique mesclada com pivotamento pelo máximo grau modificado teve o melhor desempenho em relação às outras tentativas de melhoria.
- Em relação aos problemas de Programação Inteira, mesmo para execuções limitadas por tempo ou por chamadas recursivas o algoritmo permite que sejam encontradas desigualdades violadas nos primeiros estágios de busca.

Conclusões e Trabalhos Futuros

- Apesar de exponencial no pior caso, o algoritmo de Bron e Kerbosch é uma alternativa bastante rápida se utilizado com pivotamento.
- A estratégia de estimar o peso da clique mesclada com pivotamento pelo máximo grau modificado teve o melhor desempenho em relação às outras tentativas de melhoria.
- Em relação aos problemas de Programação Inteira, mesmo para execuções limitadas por tempo ou por chamadas recursivas o algoritmo permite que sejam encontradas desigualdades violadas nos primeiros estágios de busca.
- Experimentos dentro do contexto de *Branch-and-Cut* estão sendo realizados com resultados promissores.

UFOP

Referências

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

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

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

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.

Horaud, R. and Skordas, T. (1989).
Stereo correspondence through feature grouping and maximal cliques.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 11:1168–1180.

Mitchell, E. M., Artymiuk, P. J., Rice, D. W., and Willett, P. (1990).
Use of techniques derived from graph theory to compare secondary structure motifs in proteins.
Journal of Molecular Biology, 212(1):151 – 166.