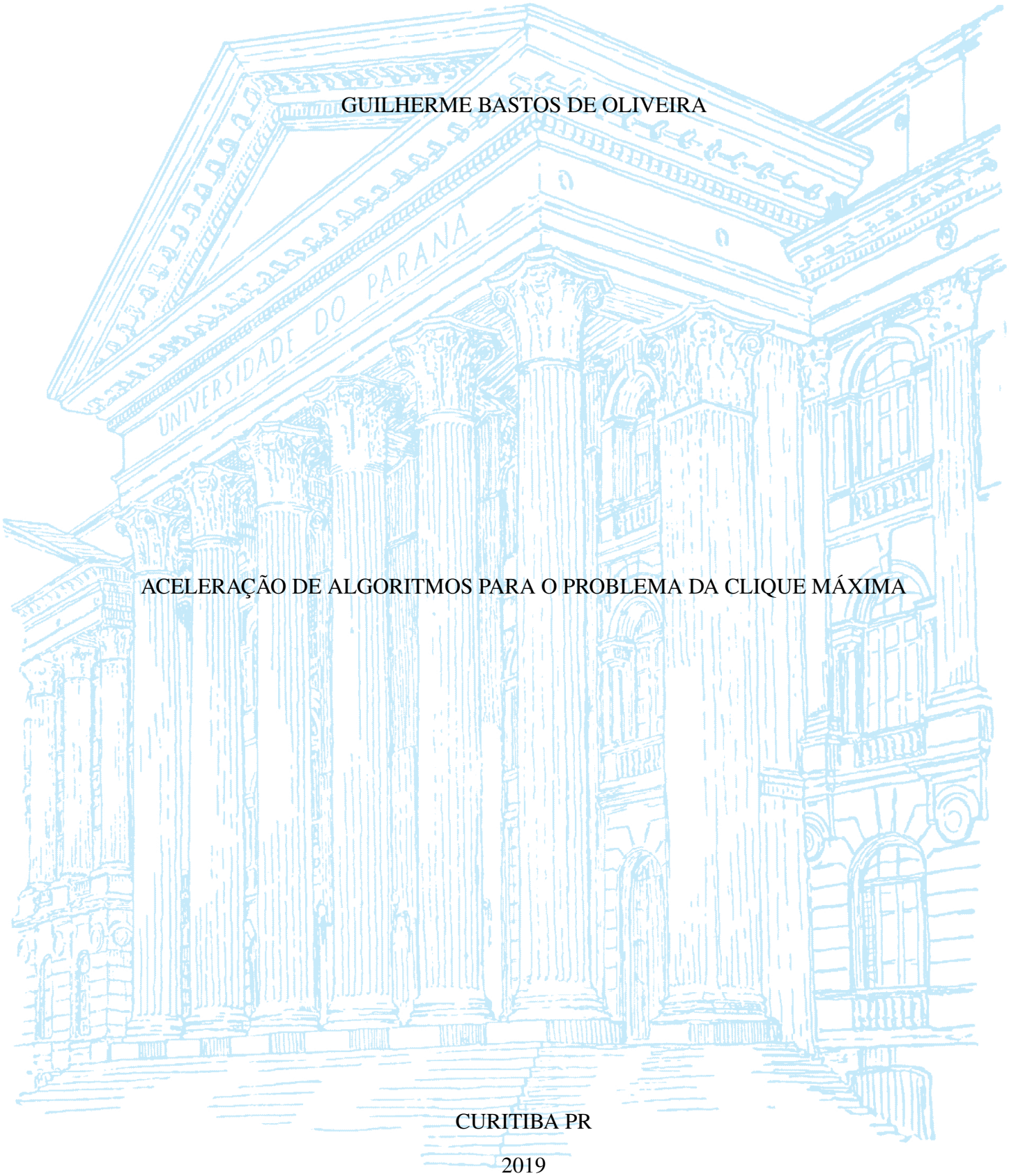
UNIVERSIDADE FEDERAL DO PARANÁ

GUILHERME BASTOS DE OLIVEIRA



ACELERAÇÃO DE ALGORITMOS PARA O PROBLEMA DA CLIQUE MÁXIMA

CURITIBA PR

2019

GUILHERME BASTOS DE OLIVEIRA

ACELERAÇÃO DE ALGORITMOS PARA O PROBLEMA DA CLIQUE MÁXIMA

Trabalho apresentado como requisito parcial à conclusão do Curso de Bacharelado em Ciência da Computação, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Prof. Dr. Renato Carmo.

CURITIBA PR

2019

Universidade Federal do Paraná
Setor de Ciências Exatas
Curso de Ciência da Computação

Ata de Apresentação de Trabalho de Graduação II

Título do Trabalho: Aceleração de Algoritmos Para o Problema da Clique Máxima

Autor:
GRR: 20167818 Nome: Guilherme Bastos de Oliveira

Apresentação: Data: 19/12/2019 Hora:16h00 Local: Auditório Alexandre Direne

Orientador: Renato Carmo

Membro da banca 1 André Guedes

Membro da banca 2 Murilo Silva

(nome) (assinatura)

AVALIAÇÃO – Produto escrito		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Conteúdo	(00-40)				
Referência Bibliográfica	(00-10)				
Formato	(00-05)				

AVALIAÇÃO – Apresentação Oral		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Domínio do Assunto	(00-15)				
Desenvolvimento do Assunto	(00-05)				
Técnica de Apresentação	(00-03)				
Uso do Tempo	(00-02)				

AVALIAÇÃO – Desenvolvimento		ORIENTADOR	MEMBRO 1	MEMBRO 2	MÉDIA
Nota do Orientador	(00-20)		**********	**********	
NOTA FINAL		**********	**********	**********	90

Pesos indicados são uma sugestão.

Dedico este trabalho àqueles que vieram antes de mim e tornaram isso possível

AGRADECIMENTOS

Agradeço à minha mãe, Renata, por todo o suporte, incentivo e confiança, essenciais durante esta jornada. Ao prof. Dr. Renato Carmo, pelos incontáveis conselhos, inúmeras horas de atenção e grande apoio, fundamentais para o desenvolvimento deste trabalho. À minha família, pelo carinho e preocupação com o meu desenvolvimento e sucesso. E aos meus amigos, por todos os momentos de companheirismo e amparo, que fizeram o caminho ser mais agradável.

RESUMO

Na teoria dos grafos, dois problemas clássicos são: o problema de decisão da clique e o problema da clique máxima (um problema de otimização). O problema da clique máxima é um problema NP-Difícil e portanto não são esperados algoritmos de tempo polinomial para obter soluções exatas para este problema. Mesmo assim o problema é interessante e possui usos diversos, como na área de redes e na biotecnologia, portanto é necessário a busca por métodos de encontrar soluções eficientes. Entre estes métodos um de destaque é uma abordagem do tipo *Branch & Bound*, que tenta reduzir o espaço de busca através de heurísticas. Neste trabalho serão apresentadas as definições formais para os problemas e estudadas possíveis abordagens.

Palavras-chave: Clique, Limitante, Teoria dos Grafos

ABSTRACT

In graph theory, two classic problems are: the clique decision problem and the maximum clique problem (an optimization problem). The maximum clique problem is NP-Hard and therefore polinomial time algorithms for the exact solution for this problem are not expected. Nevertheless the problem is interesting and have a diverse range of applications, such as networks and biotechnology, therefore is useful the search for methods to find efficient solutions. Between these methods there is one that stand out, an approach using Branch & Bound, that tries to reduce the search space through heuristics. In this work we will present formal definitions for the problem and study possible approaches.

Keywords: Clique, Bound, Graph Theory

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

DINF	Departamento de Informática
UFPR	Universidade Federal do Paraná
CM	Clique Máxima
CIM	Conjunto Independente Máximo
CMV	Cobertura Mínima por Vértices

LISTA DE SÍMBOLOS

$\Gamma_G(v)$	vizinhança de v em G
$\delta(v)$	grau de v
$\delta(G)$	grau mínimo de G
$\Delta(G)$	grau máximo de G
$\chi(G)$	número cromático de G
$\omega(G)$	tamanho de uma clique máxima de G

SUMÁRIO

1 INTRODUÇÃO

A teoria dos grafos é a área de estudo de grafos, estruturas de dados que podem ser utilizadas para representar relações entre elementos de um conjunto. Na teoria dos grafos, clique é o nome dado a um conjunto de vértices de um grafo $G = (V, E)$ onde todos os vértices estão conectados por arestas de G. O problema da clique máxima é encontrar o maior conjunto de vértices que forma uma clique em um determinado grafo. Neste campo de estudo existem diversos problemas. Neste trabalho o foco será em problemas relacionados a cliques de um grafo, principalmente o problema da clique máxima, que consiste em, dado um grafo, obter uma clique de maior tamanho possível. O objetivo principal é investigar possíveis maneiras de acelerar os algoritmos para estes problemas e possíveis diferenças entre algoritmos para os problemas de decisão e otimização.

O termo "clique" pode gerar confusão para os falantes nativos do português, uma vez que o termo significa uma coisa completamente distinta na língua inglesa, onde teve sua origem. Clique, na área de estudos sociais na língua inglesa denomina um grupo de indivíduos que interagem um com outros e compartilham interesses.

Foi utilizado por Luce e Perry para referenciar um conjunto de indivíduos onde todos possuem uma relação de amizade com todos os outros membros do conjunto. (Luce e Perry, 1949). A pesquisa apresentou um métodos de matrizes aplicados à análise de relações sociais. O modelo, além de estudar um objeto facilmente representado por grafos, apresenta uma representação matricial dos dados de maneira quase idêntica à representação de grafos por matrizes.

Outros problemas relevantes na teoria dos grafos podem ser reduzidos em tempo polinomial para o problema da clique máxima, como o problema do conjunto independente máximo e o problema da cobertura de vértices mínima.

Tais problemas são NP-Difíceis e portanto o problema da clique máxima é também NP-Difícil. Sendo assim não é esperada a existência de algoritmos polinomiais para a solução deste problema (Garey e Johnson, 1979). Além disso, aproximar o tamanho da clique máxima também é NP-Díficil para um fator de aproximação $n^{1-\varepsilon}$ para todo $\varepsilon > 0$ (Håstad, 1999).

1.1 ORGANIZAÇÃO DO TEXTO

Na seção seguinte serão apresentadas definições que serão utilizadas ao longo do trabalho. No Capítulo 2 são apresentados os conceitos de algoritmos *Branch & Bound* e limitantes. No Capítulo 3 algumas das principais soluções propostas para obter a clique máxima de um grafo são abordadas, assim como grafos para *benchmark* e um limitante exato para cliques. No Capítulo 4 são apresentados dois problemas conhecidos na teoria dos grafos e a relação deles com o problema da clique máxima. No Capítulo 5 é apresentada uma breve descrição sobre programação linear e uma maneira de resolver o problema da clique máxima com uma modelagem em programação linear. No Capítulo 6 são feitos comentários sobre a relação entre coloração e clique e apresentado um algoritmo de coloração gulosa. Finalmente no Capítulo 7 são feitas considerações finais.

1.2 DEFINIÇÕES

Um grafo simples não direcionado G é um par $(V(G), E(G))$ onde $V(G)$ é um conjunto finito e $E(G) \subseteq \binom{V(G)}{2}$. A Figura 1.1 mostra um exemplo de representação visual de um grafo.

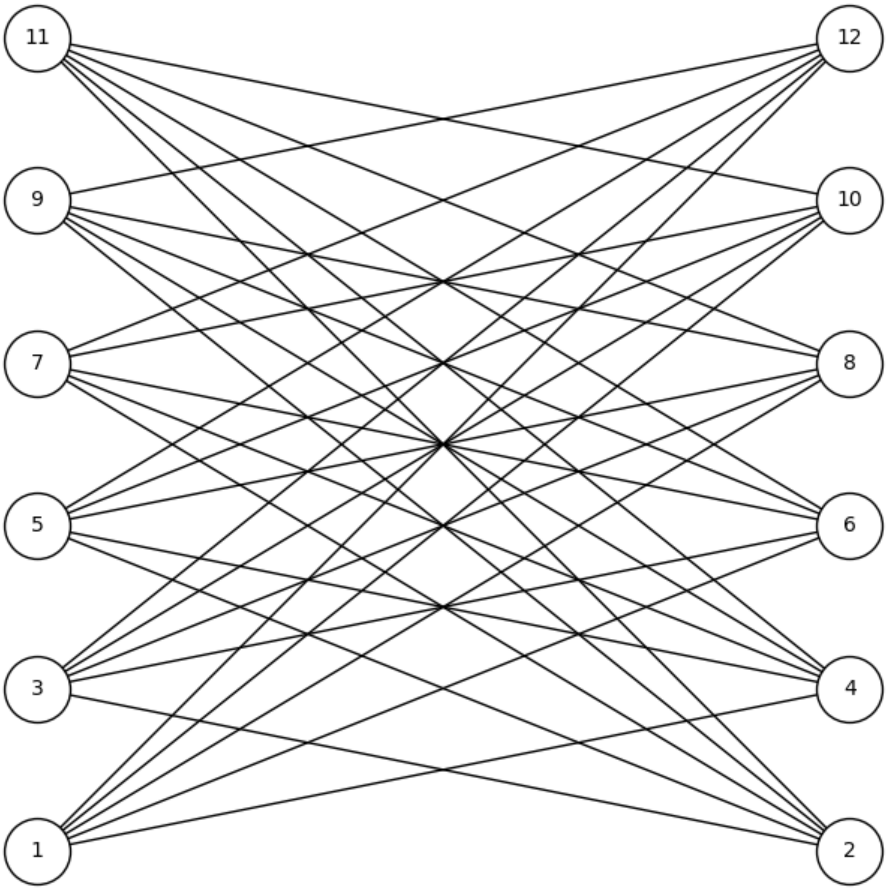


Figura 1.1: um grafo de 12 vértices

Os elementos de $V(G)$ são chamados de vértices de G e os elementos de $E(G)$ são chamados de arestas de G.

Se $a = \{u, v\}$ é uma aresta em $E(G)$, dizemos que os vértices u e v são pontas de a. Neste caso dizemos também que os vértices u e v são vizinhos (ou adjacentes) em G, e a é incidente em u e em v.

A vizinhança de um vértice v no grafo G é o conjunto dos vértices vizinhos de v em G, ou seja, $\Gamma_G(v) = \{u \in V(G) \text{ tal que } \{u, v\} \in E(G)\}$.

O grau de um vértice v, denotado por $\delta(v)$ é o número de arestas que incidem em V, ou seja, $\delta(v) = |\{e \in E \text{ tal que } v \in e\}|$.

O grau de um vértice com o menor grau em G é conhecido como o grau mínimo de G, denotado por $\delta(G)$.

O grau de um vértice com o maior grau em G é conhecido como o grau máximo de G, denotado por $\Delta(G)$.

O grafo complementar (ou complemento) de um grafo G é denotado por $\overline{G}$ e definido como o grafo que possui os mesmos vértices de G, onde dois vértices são vizinhos se e somente se não são vizinhos em G. Isto é: $V(\overline{G}) = V(G)$ e $E(\overline{G}) = \binom{V(G)}{2} - E(G)$.

Um conjunto independente I em um grafo é um subconjunto de vértices em que nenhum par é vizinho, ou seja, $I \subseteq V(G)$ onde $\{u, v\} \in E(G)$ para todo par $u, v \in C$.

Uma clique C em um grafo é um subconjunto de vértices em que todo par é vizinho, ou seja, $C \subseteq V(G)$ onde para todo par $u, v \in C$ então $\{u, v\} \in E(G)$.

Dois problemas relacionados a cliques serão discutidos neste trabalho. Um problema de decisão e um problema de otimização.

O problema de decisão consiste em, dado um inteiro k e um grafo G, decidir se existe clique de tamanho igual a k em G, que pode ser resolvido ao encontrar o tamanho de uma clique máxima de G.

Clique (CLIQUE)	
Instância:	Um grafo G Um inteiro k
Pergunta:	Existe clique de tamanho k em G?

Já o problema de otimização, conhecido com o problema da clique máxima, consiste em, dado um grafo G, encontrar uma clique máxima de G, ou seja, uma clique com o maior tamanho possível em G.

Clique Máxima (CM)	
Instância :	Um grafo G.
Resposta:	Uma clique de tamanho máximo em G

A Figura 1.2 demonstra um grafo com a clique máxima destacada.

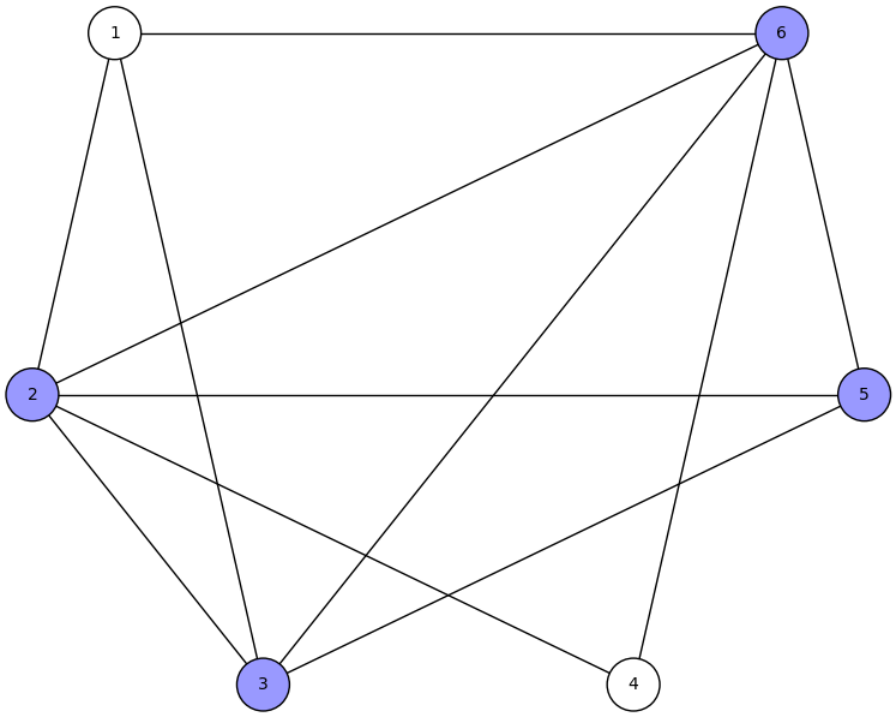


Figura 1.2: um grafo com clique máxima de tamanho 4 (vértices 2, 3, 5 e 6)

Denotamos o tamanho de uma clique máxima de G por $\omega(G)$.

2 ALGORITMOS *BRANCH & BOUND*

Algoritmos *Branch & Bound* são geralmente utilizados para resolver problemas combinatórios (para uma explicação da técnica ver (Kreher e Stinson, 1999)), fazendo proveito de limitantes para acelerar o processo de encontrar uma solução ótima para um problema. O método foi proposto primeiramente em (Land e Doig, 1960) com a intenção de resolver problemas de otimização discreta.

Os problemas de otimização discreta são aqueles onde as variáveis possuem valores discretos, como valores inteiros ou valores binários. No caso do problema de otimização da clique as variáveis são binárias, cada uma representando se um determinado vértice pertence ou não a clique.

As soluções mais robustas para o problema da clique máxima atualmente utilizam algoritmos *Branch & Bound* ou variações (como a Busca por Bonecas Russas, descrita na Seção 2.1) e geralmente utilizam limitantes baseados na coloração de grafos.

2.1 BUSCA POR BONECAS RUSSAS

Existe uma variação de algoritmos *Branch & Bound* conhecida como Busca por Bonecas Russas (em inglês, *Russian Dolls Search*).

O termo é utilizado para descrever a ideia de substituir uma busca em um problema por n buscas em subproblemas cada vez maiores, até atingir o problema original, onde n é o número de variáveis. O primeiro subproblema contém apenas a última variável, o $i - ésimo$ problema contém da $(n - i + 1) - ésima$ variável até a última. No caso do problema da clique máxima cada vértice tem uma variável binária associada, que indica se o vértice pertence ou não a clique.

O método pode ser menos eficiente quando comparado de maneira direta com um algoritmo de *Branch & Bound*, porém existem ordenações das variáveis onde é possível explorar de maneira mais eficiente os limitantes, diminuindo o espaço de busca (Verfaillie et al., 1996).

No caso do problema da clique máxima são considerados como variáveis os vértices do grafo de entrada, tornando necessário o desenvolvimento de mecanismos de ordenação que tornem a estratégia de Bonecas Russas mais eficiente.

2.2 LIMITANTES

Um limitante é uma função que representa o máximo (limitante superior) ou mínimo (limitante inferior) que determinada função pode assumir.

O estudo sobre limitantes para o problema da clique máxima é uma área de pesquisa ativa e existem diversos usos para tais limitantes.

Para algoritmos *Branch & Bound* são de interesse limitantes superiores, uma vez que a operação de poda faz uso de limitantes para descartar sub-árvores de busca.

2.2.1 Limitante Simples

Existem limitantes que, embora não forneçam limites justos o suficiente para serem aproveitados por técnicas de *Branch & Bound*, podem ser utilizados para exemplificar o possível funcionamento de um limitante. Nesta seção será explicado um limitante simples, baseado nas definições de clique.

É imediato notar que pela definição de clique que todos os vértices v que fazem parte de uma clique C de tamanho k possuem grau pelo menos igual a $k-1$, pois caso contrário é impossível que ele seja vizinho de todos os vértices em C.

Um limitante superior para o problema da clique máxima é então o grau do vértice com maior grau em G mais 1, ou seja:

$$\Delta(G) + 1 \geq \omega(G)$$

Note que não basta que exista um vértice com grau $k-1$, mas para existir uma clique de tamanho k em G são necessários k vértices com ao menos $k-1$ vizinhos. Portanto podemos definir um limitante mais justo que o anterior:

$$\omega(G) \leq max\{1 \leq t \leq n, \text{ tal que } \exists S \subseteq G \text{ onde } |S| \geq t \text{ e } \forall\, v \in S\ \delta(v) \geq t-1\}$$

É possível computar o limitante óbvio em tempo linear ($O(n+m)$) utilizando o seguinte algoritmo:

```
LimitanteSimples()
    Para cada v ∈ V(G)
        V[δ(v)] ← V[δ(v)] + 1
    lim ← 0
    Para i de n − 1 até 0
        lim ← lim + V[i]
        Se lim ≥ i + 1
            Devolva lim
```

3 REVISÃO BIBLIOGRÁFICA

Os problemas de decisão da clique e da clique máxima são problemas que geram uma área de pesquisa ativa, como resultado diversos algoritmos foram propostos para resolvê-los. Neste capítulo serão apresentados alguns algoritmos que resolvem o problema de maneira exata e um limitante exato para o problema.

3.1 BIBLIOTECA **CLIQUER**

Patric Östergård apresentou em 2002 um algoritmo que utiliza a técnica de Bonecas Russas (explicada na Seção 2.1), um subtipo de *Branch & Bound* com uma nova poda para resolver o problema da clique máxima (Östergård, 2002).

Um resultado deste trabalho foi o desenvolvimento da biblioteca **Cliquer** (Niskanen e Östergård, 2003), que implementa algoritmos para problemas relacionados a cliques, entre eles clique máxima e o problema de decisão da clique.

3.1.1 Problema de Otimização e Decisão na Biblioteca Cliquer

Em parte dos algoritmos *Branch & Bound* podemos ter um gasto de tempo considerável após encontrar uma clique máxima, testando as combinações restantes para verificar que não existe uma clique maior que a encontrada. Caso exista um algoritmo mais rápido para o problema de decisão, seria possível executar este algoritmo para momentos específicos do *Branch & Bound* e potencialmente evitar percorrer o espaço de busca inteiro.

O software SageMath (SageMath, 2017) fornece um interface com a biblioteca **Cliquer**, disponibilizando funções para solucionar o problema da clique máxima e o problema de decisão da clique, definidos na Seção 1.2.

Especulamos que os algoritmos das funções para resolver o problema de decisão (`clique_number()`) e o de decisão (`max_clique()`) fossem diferentes na biblioteca.

Com uma análise do código fonte do SageMath (disponível em `https://github.com/sagemath/sage`) e da biblioteca **Cliquer** (disponível em `https://users.aalto.fi/~pat/cliquer.html`) foi constatado que ambas as funções utilizam o mesmo algoritmo, implementado na função `clique_unweighted_find_single()`.

As funções `clique_number()` e `max_clique()` do SageMath fornecem implementações de algoritmos que resolvem os problemas de decisão e otimização, respectivamente. Ambas as funções realizam um cadeia de chamadas de procedimentos que inclui a função `clique_unweighted_find_single()`, que realiza um procedimento para encontrar os elementos de uma clique máxima.

Sendo assim a função que resolve o problema de decisão é apenas um "wrapper" para a do problema de otimização e utiliza o tamanho do conjunto encontrado para calcular a resposta. Portanto na implementação de (Östergård, 2002) não existe diferença entre os algoritmos para resolver os diferentes problemas da clique.

Existe a possibilidade de outras implementações para algoritmos destes problemas possuirem diferenças entre o problema de otimização e o de decisão, porém os algoritmos estudados se concentram no problema de otimização e não foram investigadas mais implementações para o problema de decisão.

3.2 LIMITANTES EXATOS PARA CLIQUE MÁXIMA

Um fator importante para o desempenho dos algoritmos de *Branch & Bound* é o limitante que será utilizado para realizar a poda do espaço de busca. Sendo assim existem diversos limitantes propostos. Um dos limitantes estudados foi apresentado por Budinich (Budinich, 2003) e infelizmente não apresentou resultados significativamente positivos considerando os testes apresentados e a complexidade de tempo para o cálculo dos limitantes.

3.2.1 Limitantes Comparados

O autor propõe um limitante superior e um inferior, nesta seção iremos abordar apenas os limitantes superiores, que possuem mais utilizada para algoritmos de *Branch & Bound*. São comparados 3 limitantes já existentes e o proposto pelo autor.

O Limitante 3.1 é originário de (Amin e Hakimi, 1972), onde n é o número de vértices e m o número de arestas.

$$\omega \leq \frac{3 + \sqrt{9 - 8(n - m)}}{2} \tag{3.1}$$

O Limitante 3.2, onde A é a matriz de adjacência de um grafo e $\rho(A)$ é o raio espectral (o maior valor absoluto de seus autovalores) de A, foi apresentado em (Wilf, 1967).

$$\omega \leq \rho(A) + 1 \tag{3.2}$$

O Limitante 3.3 surge de uma prova de (Amin e Hakimi, 1972), onde N_{-1} é o número de autovalores de A maiores ou iguais a -1 e $\text{rank}(A)$ é a dimensão do espaço coluna da matriz A.

$$\omega \leq N_{-1} + 1 < \text{rank}(A) + 1 \tag{3.3}$$

O limitante superior proposto em (Budinich, 2003) é mostrado na equação 3.4, onde n é o número de vértices e $\overline{A}$ é a matriz de adjacência do grafo complementar.

$$\omega \leq n - \frac{\text{rank}(\overline{A})}{2} \tag{3.4}$$

3.2.2 Resultados

Os resultados da comparação do desempenho dos limitantes é descrito na Tabela 3.1. É possível notar que para a maioria dos casos o método proposto encontra limitantes mais justos, porém a diferença nos testes, exceto para o limitante inferior com $n = 100$ e $0.60 \leq \delta \leq 0.95$, e $n = 200$ e $\delta = 0.90$, é inferior a 1. Para a utilização em algoritmos de *Branch & Bound* essa melhora não gera impactos significativos na poda.

O custo computacional para calcular os limitantes propostos é $\Theta(n^3)$ [1]. Algoritmos *Branch & Bound* irão realizar diversas vezes o cálculo dos limitantes para subconjuntos distintos do problema, portanto é indesejado que tenham custo computacional alto.

Outros limitantes exatos para o problema da clique máxima podem sofrer dos mesmos impedimentos para utilização com algoritmos *Branch & Bound*. Seriam necessários limitantes mais justos e com custo computacional menor para que fosse possível obter resultados próximos aos obtidos com limitantes por coloração.

[1] No artigo a complexidade afirmada é $O(n^3)$ para o limitante superior e ao longo do texto é comentado um custo mínimo de n^3, portanto conclui-se custo $\Theta(n^3)$.

Tabela 3.1: Comparação de limitantes obtidos para grafos aleatórios de n vértices e densidade de aresta δ, compara resultados anteriores (3.1, 3.2, 3.3) e o limitante proposto pelo autor. Adaptado de (Budinich, 2003)

n	δ	$\bar{\omega}$	2.1	2.2	2.3	2.4
						novo
100	0.05	3.10	6.06	**2.25**	11.58	16.25
	0.10	3.98	7.48	**2.99**	10.26	12.58
	0.20	5.00	8.75	**4.33**	8.84	10.00
	0.30	6.10	8.91	**5.17**	7.52	8.21
	0.40	7.54	8.38	**5.49**	6.24	6.66
	0.50	9.12	7.72	5.58	**5.19**	5.49
	0.60	11.56	6.71	5.28	**4.16**	4.34
	0.70	14.58	5.74	4.85	**3.33**	3.44
	0.80	19.98	4.48	4.03	**2.45**	2.51
	0.90	30.68	3.10	2.94	**1.61**	1.63
	0.95	43.52	2.24	2.19	**1.16**	1.16
200	0.10	4.16	14.79	**5.25**	20.97	24.20
	0.50	11.00	12.84	9.19	**8.71**	9.10
	0.90	-	189.69	180.17	**99.10**	100.00

3.3 GRAFOS DE *BENCHMARKS*

Para realizar comparação entre diferentes algoritmos para o problema da clique máxima é útil um conjunto de grafos de teste (ou *benchmark*) para verificar a corretude e comparar o desempenho entre as distintas abordagens.

Nesta seção serão apresentadas as complicações que grafos aleatórios podem gerar e um conjunto de grafos para a realização de *benchmarks* do problema da clique máxima.

3.3.1 Grafos Aleatórios

Para grande parte dos *benchmarks* realizados com grafos aleatórios, como em (Budinich, 2003) e (Tomita et al., 2017), o modelo é o seguinte: um grafo aleatório $G_{n,p}$ possui n vértices e probabilidade de aresta $0 < p < 1$, ou seja, as arestas são escolhidas independentemente e com probabilidade p.

Para grafos do tipo $G_{n,p}$ a distribuição do tamanho de uma clique máxima para grafos com mesmo n e p é altamente concentrada (Bollobás, 2001). Com a falta de variação de tamanhos de cliques máximas em grafos com a mesma densidade de arestas é possível que diferenças de desempenho de um algoritmo para diferentes categorias de grafos não sejam detectadas.

3.3.2 Grafos de *benchmark* DIMACS

O Center for Discrete Mathematics and Theoretical Computer Science - DIMACS realizou em 1992 o Second DIMACS Implementation Challenge (Segundo Desafio de Implementação DIMACS, em tradução livre), tratando dos seguintes problemas NP-Díficeis: clique máxima, coloração de grafos e satisfazibilidade.

Junto ao desafio foram disponibilizados grafos para *benchmark* relacionados ao problema da clique máxima (disponíveis em `http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmark/clique/`). Desde então este conjunto de grafos é utilizado para realizar comparações entre algoritmos.

3.4 ALGORITMOS QUE UTILIZAM COLORAÇÃO COMO LIMITANTES

O algoritmo MCLIQ (Tomita et al., 1988) para a resolução do problema da clique máxima originou uma família de algoritmos que utiliza uma abordagem *Branch&Bound* com coloração de grafos (explicada no Capítulo 6) como técnica para gerar limitantes.

Melhorias realizadas no algoritmo MCLIQ deram origem ao algoritmo MCQ (Tomita e Seki, 2003). A principal modificação é uma ordenação inicial dos vértices baseada nos graus dos vértices.

O algoritmo MCR (Tomita e Kameda, 2007) altera a ordenação inicial dos vértices do algoritmo MCQ para uma ordenação baseada no grau de vértices em subgrafos, como descrito em (Carraghan e Pardalos, 1990).

A adição de um procedimento chamado de ***Re-NUMBER*** e uma modificação no esquema de coloração empregada no algoritmo MCR resultaram no algoritmo MCS (Tomita et al., 2010).

O algoritmo mais recente desta família é o MCT (Tomita et al., 2017), que traz ainda mais melhorias sobre o MCS, principalmente pela adição de uma solução inicial aproximada como limitante inferior e uma modificação no sistema de ordenação e numeração de vértices.

3.5 PARALELIZAÇÃO

Existem soluções que fazem uso de paralelização para encontrar uma solução. O algoritmo BB-MaxClique, proposto em (San Segundo et al., 2011) e uma proposta mais recente (Corrêa et al., 2014) fazem uso de paralelismo a nível de bits (em inglês, *bit-level parallelism*). Esta paralelização parte da codificação de grafos como *bitmaps* e utiliza instruções de hardware que operam bits de maneira paralela.

3.6 COMPARATIVO ENTRE SOLUÇÃO

Uma comparação entre os principais algoritmos realizada numa revisão feita em (Wu e Hao, 2015) mostrou que os algoritmos que utilizam coloração para expansão e poda (como MCQ, MCR, MCS, MCT e BB-MaxClique) são os mais eficientes dos comparados.

4 PROBLEMAS EM GRAFOS

Existem diversos problemas na teoria dos grafos e alguns deles possuem relações interessantes com o problema da clique máxima, neste capítulo discutiremos dois problemas, a cobertura mínima por vértices e o conjunto independente máximo, assim como a relação destes com clique.

4.1 COBERTURA MÍNIMA POR VÉRTICES

Uma cobertura de vértices de um grafo G é um conjunto $C \subseteq (G)$ de maneira que todas as arestas do grafo incidam em algum vértice de C.

O problema da cobertura mínima por vértices é, dado um grafo G, encontrar o menor conjunto que seja uma cobertura de vértices do grafo G.

Cobertura Mínima por Vértices (CMV)
Instância : Um grafo G.
Resposta : Uma cobertura de vértices de tamanho mínimo em G

Seja G um grafo, C uma clique de tamanho k em G e $\overline{G}$ o grafo complementar de G. Para toda aresta $a = (u, v) \in E(\overline{G})$ é verdade que pelo menos um dos vértices, u ou v, está em $V(G) - C$ pois, caso contrário, ou seja, nenhum dos vértices pertence a $V(G) - C$, ambos fariam parte da clique e a aresta a pertenceria a G e não $\overline{G}$.

Portanto todas as arestas de $V(\overline{G})$ são cobertas pelo conjunto $V(G) - C$.

Assumindo que exista uma cobertura D, de tamanho $n - k$ em $\overline{G}$. Então todas as arestas de $E(\overline{G})$ incidem algum vértice de D. Não é possível que dois vértices, u e v não estejam em D, para todo $(u, v) \in E(\overline{G})$.

Portanto para todos os vértices $u, v \notin D$ existe uma aresta $(u, v) \in V(G)$, tais vértices formam uma clique.

Como existem k vértices fora da cobertura, existe uma clique de tamanho k em G.

Com isso temos que um grafo G, com n vértices, possui clique de tamanho k se e somente se existe uma cobertura de vértices de tamanho $n - k$ em $\overline{G}$.

4.2 CONJUNTO INDEPENDENTE MÁXIMO

Um conjunto independente I é máximo se possui tamanho máximo em G, ou seja, não existe um conjunto independente maior que I em G.

O problema do conjunto independente máximo é, dado um grafo G encontrar um conjunto independente máximo em G.

Conjunto Independente Máximo (CIM)
Instância : Um grafo G.
Resposta : Um conjunto independente de tamanho máximo em G

Existe uma relação forte entre o problema do conjunto independente máximo e o problema da clique máxima.

Seja G um grafo não direcionado e $\overline{G}$ o grafo complementar de G. Uma clique máxima de G é um conjunto independente máximo no grafo $\overline{G}$ e um conjunto independente máximo em $\overline{G}$ é uma clique máxima em G.

A redução de CM para CIM e de CIM para CM podem ser feitas em tempo polinomial.

5 CLIQUE E PROGRAMAÇÃO LINEAR

Programação linear é uma área de otimização. Problemas de programação linear são problemas de otimização descritos por sistemas de inequações lineares.

Um programa linear é definido pelo problema de maximizar uma função linear restrita ao conjunto de todos os vetores que satisfazem um sistema de equações e inequações. Considere o seguinte exemplo de um programa linear.

Maximizar

$$x_1 + x_2$$

Sujeito a

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_2 - x_1 &\leq 1 \\ x_1 + 6x_2 &\leq 15 \\ 4x_1 - x_2 &\leq 10 \end{aligned}$$

Este programa tem como solução $x_1 = 3, x_2 = 2$, uma representação visual do problema é mostrada na Figura 5.1.

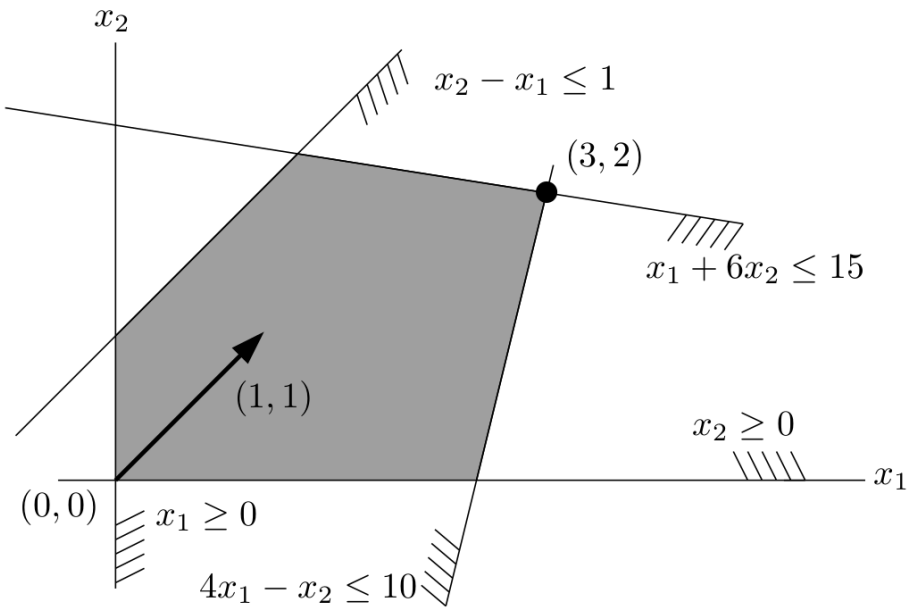


Figura 5.1: Representação do programa linear exemplo. Imagem de (Matoušek e Gärtner, 2007)

Um programa linear possui uma solução ótima ou é insolúvel.

5.1 PROGRAMAÇÃO INTEIRA E RELAXAMENTO

Em muitos casos são de interesse apenas soluções inteiras, os problemas de otimização com esta característica são conhecidos como programas inteiros (Matoušek e Gärtner, 2007).

Um subtipo de programação inteira conhecido como Programação Inteira Binária (em inglês chamada de *0-1 Programming*) é definido para problemas em que as variáveis representam situações dicotômicas, de modo que seus valores são 0 ou 1. Encontrar uma solução ótima para um programa inteiro é um problema NP-Difícil (Lenstra e Kan, 1979) e portanto não é esperado que exista um algoritmo capaz de resolvê-los em tempo polinomial.

Existe uma modificação denominada de relaxamento em programas inteiros binários onde o valor atribuído a uma variável x na solução não obedece $x \in \{0, 1\}$ mas sim $0 \leq x \leq 1$.

Ao resolver o programa relaxado ou encontramos uma solução ótima para o mesmo ou é insolúvel. Caso seja insolúvel temos que o programa original também é insolúvel. Caso uma solução seja encontrada, temos um limitante superior para o programa original (Matoušek e Gärtner, 2007).

5.2 PROGRAMAÇÃO LINEAR INTEIRA E CLIQUE MÁXIMA

É possível reduzir o problema CM para um programa linear, de maneira que uma solução ótima para o programa pode ser mapeada para uma solução do problema da clique máxima.

A maneira mais direta de realizar isto é utilizando a relação entre clique e conjunto independente no grafo complementar, obtendo o seguinte programa linear:

Seja $G = (V(G), E(G))$ um grafo e $\overline{G} = (V(G), E(\overline{G}))$ o grafo complementar de G

Maximizar

$$\sum_{v \in V} x_v$$

Sujeito a

$$x_u + x_v \leq 1 \text{ para toda aresta } \{u, v\} \in E(\overline{G})$$
$$x_v \in \{0, 1\} \text{ para todo } v \in V(G)$$

Portanto é possível utilizar resolvedores de programas lineares para encontrar cliques máximas. Pode ser possível, através de relaxamento obter limitantes utilizando programação linear, porém o programa descrito anteriormente encontra um sério problema para este uso: o resultado ótimo será sempre maior ou igual a $\frac{n}{2}$ onde n é o número de vértices. Uma solução onde $x_v = \frac{1}{2}$ para $v \in V(G)$ será sempre viável e terá um resultado da função objetivo igual a $\frac{n}{2}$, portanto a solução ótima é pelo menos $\frac{n}{2}$.

Além disso os algoritmos específicos para cliques possuem heurísticas mais precisas e ajustes finos, gerando resultados ótimos em menos tempo que resolvedores genéricos de programação linear (Wu e Hao, 2015).

5.3 IMPLEMENTAÇÃO DE PROGRAMAS LINEARES

Existem diversos resolvedores de programas lineares. Neste trabalho foram utilizados os fornecidos pelo SageMath, um software livre com o intuito de facilitar o acesso a ferramentas matemáticas livres (SageMath, 2017) com integração a linguagem Python.

O código a seguir implementa o programa linear binário descrito na seção anterior para encontrar $\omega(G)$ de um grafo G lido na entrada padrão. O código utiliza a biblioteca SageMath e uma biblioteca desenvolvida em (Züge, 2011).

```
1 #!/usr/bin/env sage
2
```

```
from sage.all import *
import graph

# leitura do grafo
G = graph.read_graph()

# geração do grafo complementar
G = G.complement()

# montando programa linear
lp = MixedIntegerLinearProgram()
X = lp.new_variable(binary=True)
lp.set_objective(sum(X[v] for v in g.vertices()))

for e in g.edges(labels=False):
   lp.add_constraint(X[e[0]] + X[e[1]] <= 1)

# chamada para o resolvedor
lp.solve()

# exibe resultado obtido
X = lp.get_values(X)
total = sum(X.values())
print(total)
```

6 CLIQUE E COLORAÇÃO DE GRAFOS

Uma coloração própria de um grafo $G = (V(G), E(G))$ é uma rotulação dos vértices de G tal que dois vértices vizinhos possuam rótulos diferentes.

A coloração de um grafo é definida por uma função $c : V \rightarrow \mathbb{N}$ onde para todo par de vértices u, v tal que $\{u, v\} \in E$ então $c(u) \neq c(v)$.

Seja k um inteiro, uma k-coloração é uma coloração própria definida com k rótulos.

Um grafo G é dito k-colorível se existe uma k-coloração em G.

Um problema de decisão relacionado à coloração de grafos é o problema da k-Colorabilidade, onde é desejado saber se um grafo G é k-colorível.

k-Colorabilidade (kC)	
Instância :	Um grafo G Um inteiro k
Pergunta :	Existe uma k-coloração em G?

O problema da k-Colorabilidade é um problema NP-Completo (Garey e Johnson, 1979).

O número cromático de um grafo G é o menor inteiro m tal que G é k-colorível, e é denotado por $\chi(G)$, ou seja

$$\chi(G) = min\{k \in N \mid G \text{ é } k\text{-colorível}\}$$

Uma coloração é dita ótima se possui exatamente $\chi(G)$ rótulos. A Figura 6.1 mostra uma coloração ótima do grafo da Figura 1.1.

O problema da coloração ótima é um problema de otimização onde o objetivo é, dado um grafo G, encontrar uma coloração ótima de G.

Coloração Ótima (CO)	
Instância :	Um grafo G.
Resposta :	Uma coloração ótima de G

Como o problema da k-Colorabilidade é NP-Completo, o problema da Coloração Ótima é NP-Difícil.

6.1 LIMITANTE PARA CLIQUE COM COLORAÇÃO

Existe uma relação entre coloração de vértices e cliques em um grafo.

Seja $\chi(G)$ o número cromático do grafo G temos que:

$$\chi(G) \geq \omega(G)$$

Sendo k a quantidade de rótulos utilizada em uma coloração própria de um grafo G, temos que:

$$k \geq \chi(G) \geq \omega(G)$$

Portanto, com toda coloração, mesmo que não ótima, temos um limitante superior para o tamanho da clique máxima.

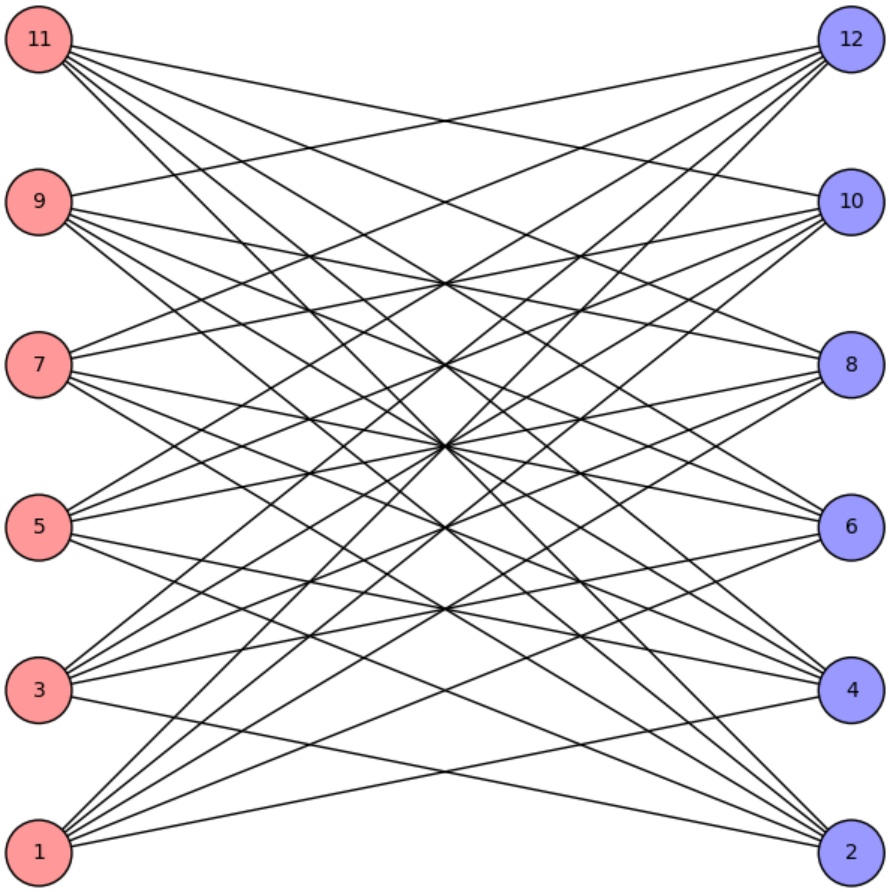


Figura 6.1: uma coloração ótima do grafo da Figura 1.1

6.2 COLORAÇÃO GULOSA

Um algoritmo é dito Guloso (ou em inglês *greedy*) quando se baseia na ideia de a cada passo escolher a melhor opção, sem realizar *backtracking* ou buscas para encontrar a solução ótima. Muitas vezes algoritmos gulosos não encontram a solução ótima para problemas de otimização, uma vez que decidir qual o melhor passo em um dado momento pode ser difícil ou impossível.

Como já foi dito, o problema de coloração é NP-Difícil e portanto não se acredita que existam algoritmos capazes de encontrar soluções ótimas em tempo polinomial. Porém para obter um limitante de uma coloração não é necessário que esta seja ótima, apenas é necessário que a coloração seja própria.

O Algoritmo Guloso computa uma coloração gulosa de um grafo G, com n vértices, onde Cor é um vetor indexado pelos vértices de G, e possui na posição i o rótulo atribuído ao vértice v_i, ou 0, caso ainda não tenha sido atribuído rótulo algum. A coloração atribui a um vértice o menor rótulo possível considerando sua vizinhança. A Figura 6.2(a) mostra como seria o resultado da coloração gulosa do grafo de exemplo da Figura 1.1.

O resultado obtido pelo algoritmo Guloso pode ser distante do ótimo, no grafo da Figura 6.2(a) é possível perceber uma caso desse, a coloração gulosa utiliza 6 rótulos mas uma coloração ótima (como mostrada na Figura 6.1) contém apenas de 2 rótulos.

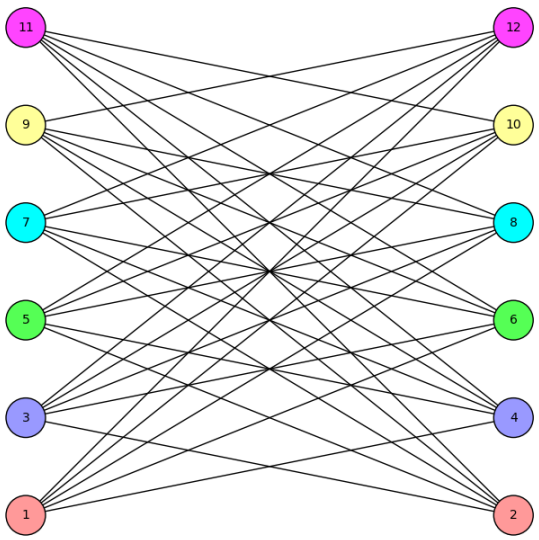
Guloso(G)

$Cor \leftarrow []$
Para *cada* $v \in V(G)$
  $Cor[v] = 0$
Para *cada* $v \in V(G)$
  $usadas \leftarrow \{\}$
  Para *cada vizinho* $\in \Gamma_G(v)$
    $usadas \leftarrow usadas \bigcup \{Cor[vizinho]\}$
  $rot \leftarrow min\{1 \leq k \leq n \text{ tal que } k \notin usadas\}$
  $Cor[v] \leftarrow rot$

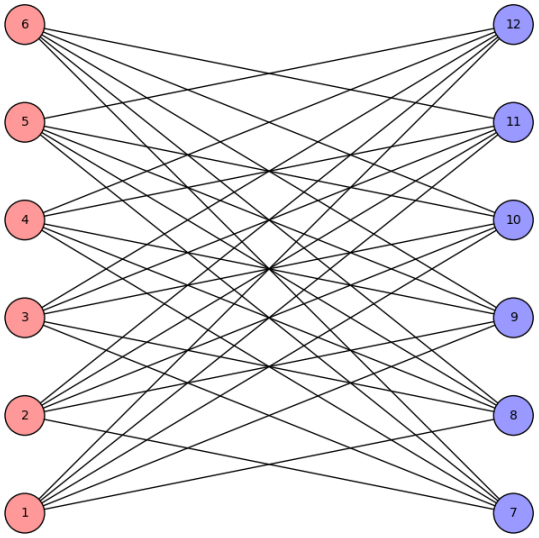
Seja $c_o : V(G) \rightarrow \mathbb{N}$ uma coloração ótima de um grafo G, qualquer ordenação de vértices em que $c_o(v_i) \leq c_o(v_{i+1})$ irá resultar em uma coloração ótima com o algoritmo Guloso. Portanto para qualquer grafo existe uma ordenação de vértices em que o algoritmo guloso irá gerar uma coloração ótima, entretanto encontrar uma ordenação com tal característica é NP-Difícil.

A Figura 6.2(b) mostra uma ordenação para o grafo de exemplo (Figura 1.1) para a qual a coloração gulosa é ótima. O exemplo foi escolhido para demonstrar que existem casos em que o algoritmo guloso é arbitrariamente ruim.

O grafo de exemplo pode ser incrementado de modo que sempre possua número cromático igual a 2 e a coloração obtida pelo algoritmo guloso utilize $\frac{n}{2}$ rótulos onde n é o número de vértices.



(a) coloração gulosa do grafo apresentado na figura 1.1



(b) coloração gulosa do grafo apresentado na figura 1.1 com reordenação de vértices

Figura 6.2: comparação do resultado de uma coloração goluso com diferentes ordenações de vértices

Se a coloração gulosa encontrar colorações que utilizam uma quantidade de rótulos próximo ao ótimo os algoritmos de *Branch & Bound* terão resultados mais eficientes. A coloração obtida com o algoritmo Guloso é dependente da ordem em que colore os vértices, portanto parte do esforço para acelerar algoritmos deste tipo é direcionado em encontrar heurísticas para a ordenação dos vértices.

Os algoritmos da Seção 3.4 seguem neste direção, realizando etapas de pré processamento. Como visto em (Wu e Hao, 2015) parte dos melhores algoritmos comparados também realizam

procedimentos similares, indicando que a coloração é um bom limitante para ser aplicado com *Branch & Bound* no problema da clique máxima.

7 CONCLUSÃO

O estudo de algoritmos para encontrar soluções exatas para o problema da clique máxima e o problema de decisão da clique é uma área extremamente ativa e possui diversas abordagens. Atualmente as abordagens que conseguem os melhores resultados em questão de eficiência são aquelas baseadas em *Branch & Bound* utilizando limitantes baseados na coloração gulosa de grafos e possuem alguma heurística para realizar a ordenação dos vértices. Outras abordagens podem trazer resultados interessantes, porém ainda não atingem os mesmos níveis de desempenho. Os avanços nesta área podem se dar através de aprimoramentos nos algoritmos, na utilização de novas abordagens ou novos métodos para o cálculo de limitantes.

REFERÊNCIAS

Amin, A. T. e Hakimi, S. L. (1972). Upper bounds on the order of a clique of a graph. *SIAM Journal on Applied Mathematics*, 22(4):569–573.

Bollobás, B. (2001). *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge.

Budinich, M. (2003). Exact bounds on the order of the maximum clique of a graph. *Discrete Applied Mathematics*, 127(3):535–543.

Carraghan, R. e Pardalos, P. M. (1990). An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6):375–382.

Corrêa, R. C., Michelon, P., Le Cun, B., Mautor, T. e Delle Donne, D. (2014). A Bit-Parallel Russian Dolls Search for a Maximum Cardinality Clique in a Graph. *ArXiv e-prints*.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco.

Håstad, J. (1999). Clique is hard to approximate within n1-ϵ. *Acta Mathematica*, 182(1):105–142.

Kreher, D. L. e Stinson, D. R. (1999). *Combinatorial algorithms: generation, enumeration, and search*. Discrete Mathematics and Its Applications. CRC Press, Boca Raton, Florida.

Land, A. H. e Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497.

Lenstra, J. e Kan, A. R. (1979). Computational complexity of discrete optimization problems. Em *Discrete Optimization I, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium*, páginas 121–140. Elsevier.

Luce, R. D. e Perry, A. D. (1949). A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116.

Matoušek, J. e Gärtner, B. (2007). *Understanding and using linear programming*. Springer.

Niskanen, S. e Östergård, P. R. J. (2003). *Cliquer User's Guide, Version 1.0*. Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48.

Östergård, P. R. J. (2002). A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1):197–207.

SageMath (2017). *SageMath, the Sage Mathematics Software System (Version 8.1)*. The Sage Developers. `http://www.sagemath.org`.

San Segundo, P., Rodríguez-Losada, D. e Jiménez, A. (2011). An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research*, 38(2):571–581.

Tomita, E. e Kameda, T. (2007). An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization*, 37(1):95–111.

Tomita, E., Kohata, Y. e Takahashi, H. (1988). A simple algorithm for finding a maximum clique. Relatório Técnico 1, University of Electro-Communications, Tokyo.

Tomita, E., Matsuzaki, S., Nagao, A., Ito, H. e Wakatsuki, M. (2017). A much faster algorithm for finding a maximum clique with computational experiments. *Journal of Information Processing*, 25:667–677.

Tomita, E. e Seki, T. (2003). An efficient branch-and-bound algorithm for finding a maximum clique. Em Calude, C. S., Dinneen, M. J. e Vajnovszki, V., editores, *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science (DMTCS)*, volume 2731 de *Lecture Notes in Computer Science*, páginas 278–289. Springer, Berlin, Heidelberg.

Tomita, E., Sutani, Y., Higashi, T., Takahashi, S. e Wakatsuki, M. (2010). A simple and faster branch-and-bound algorithm for finding a maximum clique. Em Rahman, M. e Fujita, S., editores, *Proceedings of the 4th International Workshop Algorithms and Computation (WALCOM)*, volume 5942 de *Lecture Notes in Computer Science*, páginas 191–203. Springer, Berlin, Heidelberg.

Verfaillie, G., Lemaître, M. e Schiex, T. (1996). Russian doll search for solving constraint optimization problems. Em *Proceedings AAAI*, páginas 181–187.

Wilf, H. S. (1967). The eigenvalues of a graph and its chromatic number. *Journal of the London Mathematical Society*, s1-42(1):330–332.

Wu, Q. e Hao, J.-K. (2015). A review on algorithms for maximum clique problems. *European Journal of Operational Research*, 242(3):693–709.

Züge, A. P. (2011). Solução exata do problema da clique máxima. Dissertação de Mestrado, Universidade Federal do Paraná.