Algoritmos de Clique Máximo em Redes Complexas

Guilherme Pontes Pinto[1], **Leonardo de Assis da Silva**[1]

[1]Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná (UTFPR)
Avenida Sete de Setembro – 3165 – 80.230-901 – Curitiba – PR – Brasil

Abstract. *Finding situations where polynomial time algorithms are able to find good solutions for NP-Hard problems is of great practical interest, in fact, verifying this enhancement on complex networks unfolds several possibilities of research, applications, and improvements on a diverse number of fields. Thus, this paper seeks to examine the advantage of exploring the power law distribution of complex networks using greedy algorithms for finding approximate solutions for the Maximum Clique problem.*

Resumo. *Encontrar situações onde algoritmos de tempo polinomial possam resolver de forma satisfatória problemas NP-difícil é de grande interesse prático, aliás, verificar este aprimoramento na área de redes complexas enseja várias possibilidades de pesquisas, aplicações e melhorias nos mais diversos domínios. Assim, este trabalho visa conferir os benefícios de explorar a distribuição de lei das potências em redes complexas, através de algoritmos gulosos, na resolução aproximada do problema de Clique Máximo.*

1. Introdução

As redes complexas estão frequentemente presentes no meio em que vivemos. Estas redes são representadas por grafos, que são objetos matemáticos que modelam a natureza e fenômenos em que existem conexões entre uma determinada quantidade de elementos. Embora grafos sejam estudados há mais de 200 anos, a teoria de redes complexas só se estabeleceu a partir da década de 90, em particular devido ao acesso a grandes volumes de dados que a internet passou a possibilitar. A teoria das redes complexas pode ser definida como o estudo empírico de grafos de grande escala presentes em diversos domínios.

Vários algoritmos que buscam a solução ótima para problemas modelados em grafos podem não apresentar os resultados em tempo polinomial, e, logo, surge cenário propício à busca de algoritmos que encontrem soluções "aproximadamente boas", mas que sejam capazes de executar em tempo polinomial [21]. Ainda neste contexto, é importante observar que, embora alguns algoritmos para problemas em grafos arbitrários sejam inviáveis por sua complexidade de tempo, os mesmos podem funcionar em casos particulares de grafos. No caso específico de muitas redes complexas, os grafos em questão possuem uma distribuição de vértices bastante particular que favorece a eficiência de algoritmos.

Este trabalho busca comparar os resultados encontrados através de algoritmos de aproximação gulosos com os resultados providos pelo algoritmo ótimo do problema de Clique Máximo em uma série de grafos retiradas de bases de grafos conhecidas [1][2][4]. Mais especificamente, foram implementados e analisados os resultados dos algoritmos

ótimo e gulosos em redes complexas, de forma a observar se a propriedade lei das potências (*power law*) pode proporcionar melhoria na aproximação.

Depois de efetuar experimentos com os algoritmos desenvolvidos para o problema, verificou-se que, embora este problema não possua garantia de aproximação em tempo polinomial [8], os resultados obtidos aproximavam-se com uma média de cerca de 0.98 vezes o valor da solução ótima, sendo o menor fator de aproximação encontrado cerca de 0.85. Esses resultados são apresentado detalhadamente na seção 4.

2. Fundamentação Teórica

Um grafo é uma estrutura matemática composta por um conjunto V de vértices e um conjunto E de arestas. As arestas, por sua vez, estão associadas a dois vértices, formando uma relação entre eles. Por terem diversas aplicações na modelagem de problemas reais, os grafos são amplamente aplicados e estudados na ciência da computação [17] [21].

Um grafo completo é um grafo que contém todas as arestas possíveis. Uma clique em um grafo G, por sua vez, é um subgrafo completo do grafo G. Ademais, a busca pela maior clique em um grafo é conhecido como o problema de clique máximo e possui aplicações em redes sociais e na área de bioinformática, por exemplo [7][20][5][12]. No grafo representado na figura 1, onde é exposto as conexões e o grau de cada vértice, é possível observar várias cliques, dentre eles a clique ABC de tamanho 3 e a máxima, representada por DEFGH, com tamanho 5.

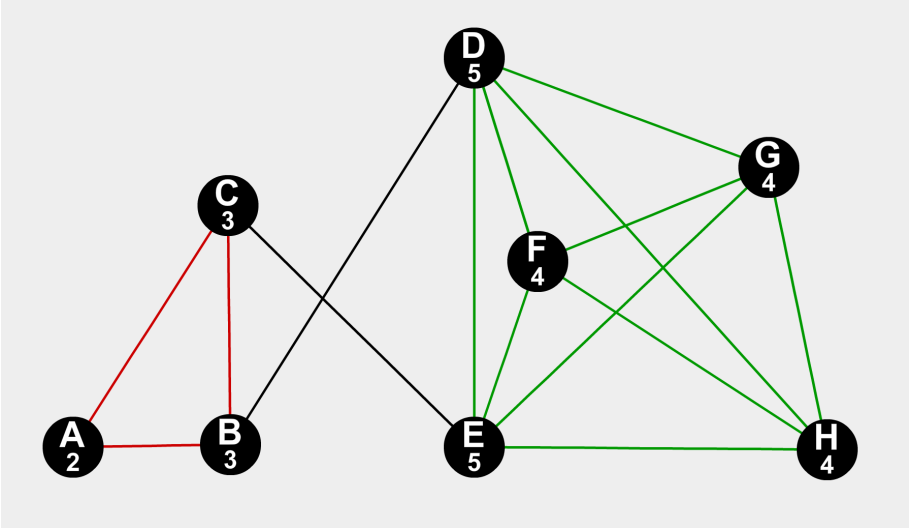

Figura 1. Um grafo que possui cliques de diferentes tamanhos.

2.1. Redes Complexas

Ao analisar casos práticos de grafos no mundo real, tal como a *World Wide Web*, sistema nervoso, padrões de citações de trabalhos científicos, entre outros, normalmente denominados como redes complexas, é possível observar que embora de áreas distintas, propriedades são compartilhadas por uma ampla quantidade destas grandes redes [13][14][18][19], das quais as mais estudadas são a *power law*, o efeito mundo pequeno e o coeficiente de alta aglomeração de vértices [17].

Neste trabalho, a propriedade de redes complexas a ser explorada será a *power law*, a qual pode ser definida como um expoente α, onde $2 \leq \alpha \leq 3$, que se manifesta na distribuição de grau dos vértices [17]. Para [6], grafos *power law* são peculiares por possuírem uma característica probabilística na qual os novos vértices têm maiores chances de se anexarem a um vértice de alto grau, o que resulta em uma distribuição específica independente do tamanho do grafo. Assim, esta propriedade pode ser empregada em algoritmos de otimização para aumentar sua aproximação do resultado ótimo.

2.2. Algoritmos de otimização

Um problema de otimização busca a melhor solução entre todo o conjunto de soluções factíveis. Assim, um problema de otimização pode maximizar ou minimizar uma dada função matemática. Da mesma forma, um algoritmo de otimização buscará o maior ou o menor valor possível de acordo com a natureza do problema a resolver. Assim, para fins de comparação, na elaboração deste estudo foi utilizado um algoritmo de solução ótima, além do algoritmo de aproximação guloso.

Algoritmos ótimos são algoritmos que buscam a solução ótima global, ou seja, a melhor solução possível para um determinado problema computacional. Esses tipos de algoritmos, no entanto, podem possuir grande complexidade computacional, como acontece com muitos dos problemas modelados em grafos. Dessa forma, pode não ser possível verificar uma solução ótima para um problema em tempo polinomial.

Segundo [9], em vários problemas de otimização, algoritmos que tentam encontrar as melhores escolhas dinamicamente exigem esforço em demasia, em contraste à algoritmos mais simples e eficientes que também seriam capazes de resolver tais problemas. Esta segunda classe é chamada de algoritmos gulosos e é caracterizada por realizar sempre a melhor escolha do momento, ou seja, escolhe a opção ótima local esperando que esta lhe guie à solução ótima global.

2.3. Complexidade

Diferentes algoritmos possuem diferentes complexidades de tempo e para se determinar qual o mais eficiente é necessário a análise de cada caso. No entanto há uma clara distinção entre dois tipos de algoritmos: polinomiais e não-polinomiais. Os polinomiais pode ser compreendido como a classe de algoritmos cujo o limite superior de sua notação assintótica é limitada por O($p(n)$), onde p é uma expressão polinomial [16].

Os algoritmos não-polinomiais são, em contraste, algoritmos que não possuem tal limitação. Os problemas para os quais não se conhecem algoritmos polinomiais que os solucionem podem ainda ser divididos em sub-categorias, dentre elas problemas NP-completos (problemas de decisão) e problemas NP-difíceis, que são problemas pelo menos tão difíceis quanto problemas NP-completos, mas que não são problemas de decisão [16].

Muitos dos problemas de otimização em grafos são NP-difíceis, no entanto, em alguns casos quando restritos à apenas grafos do mundo real [11][13][14][18], é possível encontrar ”aproximações boas”utilizando algoritmos polinomiais. Por exemplo, explorando a propriedade *power law* no problema de Miníma Cobertura de Vértices, foi desenvolvido um algoritmo guloso que apresenta uma aproximação de, em média, 1.02 o tamanho do subgrafo de solução ótima [10]. Similarmente, no problema de clique máximo,

também NP-difícil [16], surge a oportunidade de limitar os grafos *power law* como os únicos a serem analisados pelos algoritmos gulosos e observar se há benefícios advindos desta delimitação.

3. Metodologia

Dado o caráter experimental do projeto, será necessário utilizar estruturas de grafos já disponíveis em bases de dados relevantes [1][2][4] a fim de verificar o comportamento das adaptações dos algoritmos gulosos Best In e Worst Out[15], atualizando o grau dos vértices a cada operação, ou seja, utilizando a heurística New[22], e do algoritmo Cliquer de solução ótima, fornecido pela biblioteca SAGE [3]. A partir de tais recursos, os algoritmos serão executados, de forma a computar seus respectivos resultados para que, posteriormente, através do tabulamento desses dados, seja realizada a análise comparativa.

3.1. Algoritmo Worst Out

A estratégia principal deste algoritmo é conceder preferência aos vértice de grau baixo quando for necessário fazer uma remoção para obter uma clique, de forma que ao final de sua execução o próprio grafo seja completo e, consequentemente, o clique máximo seja o número de vértices restantes. Em redes complexas, este algoritmo tem o potencial para aproveitar-se da típica distribuição *power law* e massivo número de vértices pouco conectados.

Algorithm 1 Adaptação Worst Out

1: $clique \leftarrow$ número de vértices
2: **while** $clique > 1$ **and** $grafo$ não for uma clique **do**
3: $\quad menor\_vertice \leftarrow$ vértice de menor $grau$
4: $\quad$ remove $menor\_vertice$ do $grafo$
5: $\quad clique \leftarrow$ número de vértices
6: **end while**
7: **return** $clique$

Seguindo a heurística New, ao decorrer da execução, com as remoções, o grau de cada item é atualizado. Desta forma, o vértice a ser retirado é sempre aquele de menor interesse à clique, dado que o grau representa sua conectividade no grafo do momento. Devido à essas escolhas exclusivamente locais, não existe expectativas de resultados exatos, mas, em contrapartida, este algoritmo apresenta tempo de execução O(n^2).

3.2. Algoritmo Best in

Este algoritmo apresenta lógica semelhante à anterior, porém possui uma abordagem inversamente simétrica, na qual os cliques são encontrados através da seleção dos vértices mais conectados, e o maior dos cliques é retornado. Para isso, há uma iteração responsável por executar tal procedimento para cada vértice, enquanto o grau do vértice verificado for maior que o clique máximo já encontrado.

Algorithm 2 Adaptação Best in

Require: vetor de vértices $grafo$ ordenado por $grau$ de forma decrescente

1: $maximum \leftarrow 0$
2: **for** $i = 0$ to $grafo[i] > maximum$ **do**
3: $clique \leftarrow 0$
4: $tamanho\_potencial \leftarrow grafo[i].grau$
5: $clique\_potencial[0] \leftarrow grafo[i]$
6: $clique\_potencial[] \leftarrow$ lista de adjacência de $grafo[i]$
7: **for** $j = 1$ to $j > tamanho\_potencial$ **do**
8: atualizar o grau dos vértices em $clique\_potencial[]$
9: $maior\_vertice \leftarrow$ vértice de maior grau
10: remover vértices não adjacentes ao $maior\_vertice$ de $clique\_potencial$
11: $tamanho\_potencial \leftarrow$ número de vértices em $clique\_potencial[]$
12: $clique \leftarrow clique + 1$
13: **end for**
14: **if** $clique > maximum$ **then**
15: $maximum \leftarrow clique$
16: **end if**
17: **end for**
18: **return** $maximum$

Como o interesse está no maior clique em que cada vértice está inserido, a cada iteração são considerados apenas os vizinhos do vértice sendo verificado, ou seja, o restante do grafo é descartado temporariamente, de forma a ser necessário atualizar o grau dos vértices restantes e, a cada iteração de seleção do vizinho mais conectado, novas remoções ocorrem e, portanto, nova atualização de grau. Similarmente ao algoritmo anterior, o resultado ótimo não é garantido e sua complexidade é polinomial $O(n^3)$.

4. Resultados

Nesta seção serão apresentados os resultados obtidos para o problema de clique máximo, utilizando os algoritmos acima citados em 25 redes complexas oriundas de áreas diversas. Quanto à forma que os dados foram estruturados na tabulação dos resultados: os grafos foram ordenados em forma crescente pelo número de vértices, receberam um índice exclusivo, correspondente em ambas tabelas, e foram elencados os cliques encontrados em cada algoritmo. Através da Tabela 1 foi possível constatar que não há superioridade absoluta de um algoritmo guloso sobre o outro, assim como uma aparente irrelevância da quantidade de vértices e arestas em tal disputa.

Tabela 1. Cliques encontrados por algoritmos gulosos

Grafo	Nome da rede	Vértices	Arestas	Guloso	
				Worst Out	Best in
1	Airlines	235	1 298	11	10
2	USAir	332	2 126	22	19
3	Codeminer	724	1 015	3	4
4	CpanAuthors	839	2 112	7	6
5	EuroSis	1 285	6 462	11	13
6	Oclinks	1 899	13 838	6	6
7	YeastS	2 284	6 646	9	8
8	CA-GrQc	5 241	14 484	44	43
9	p2p-Gnutella08	6 301	20 777	4	5
10	Wiki-Vote1	7 115	8 346	2	3
11	p2p-Gnutella09	8 114	26 013	4	5
12	p2p-Gnutella06	8 717	31 525	3	4
13	p2p-Gnutella05	8 846	31 839	4	4
14	CA-HepTh	9 875	25 973	32	32
15	p2p-Gnutella04	10 876	39 994	2	4
16	CA-AstroPh	18 771	198 050	57	57
17	p2p-Gnutella25	22 687	54 705	2	4
18	CA-CondMat	23 133	93 439	26	22
19	p2p-Gnutella24	26 518	65 369	2	4
20	Cit-HepTh	27 769	352 285	20	22
21	p2p-Gnutella30	36 682	88 328	2	4
22	Email-Enron	36 692	183 831	17	20
23	Brightkite_edges	58 228	214 078	37	31
24	p2p-Gnutella31	62 586	147 892	2	4
25	soc-Epinions1	75 879	405 740	21	20

Sendo assim, não foi possível eleger um dos algoritmos para ser comparado diretamente com o resultado ótimo na Tabela 2, porém, dado a complexidade polinomial da heurística gulosa, é admissível a utilização de ambos algoritmos concomitantemente, descartando-se o pior resultado. Em termos gerais, dos 25 casos de estudo, 21 alcançaram o resultado ótimo e, em média, os testes obtiveram um fator de aproximação de 0.98, ou seja, apenas 2% menor que o maior clique ótimo. Faz-se relevante ainda apontar que nos pior caso absoluto houve apenas dois vértices de diferença no clique e, em porcentagem, 0.85 vezes o tamanho do clique máximo.

Pode-se observar, dessa forma, que as características que fariam com que o fator de aproximação obtido pelo algoritmo guloso fosse menor não estão comumente presentes em grafos *power law*. Consequentemente, a aproximação encontrada pela heurística gulosa, para esse problema, foi relativamente alta.

Tabela 2. Comparação da aproximação com o resultado ótimo

Grafo	OPT	MAX-Greedy	Aproximação
1	11	11	1.00
2	22	22	1.00
3	4	4	1.00
4	8	7	0.87
5	13	13	1.00
6	7	6	0.85
7	9	9	1.00
8	44	44	1.00
9	5	5	1.00
10	3	3	1.00
11	5	5	1.00
12	4	4	1.00
13	4	4	1.00
14	32	32	1.00
15	4	4	1.00
16	57	57	1.00
17	4	4	1.00
18	26	26	1.00
19	4	4	1.00
20	23	22	0.95
21	4	4	1.00
22	20	20	1.00
23	37	37	1.00
24	4	4	1.00
25	23	21	0.91
		Média	**0.98**

5. Conclusões

Ao estudar redes complexas é necessária a manipulação de grafos com grande número de elementos e conexões, tornando a resolução de problemas modelados nestes muito custosa ou até mesmo inviável computacionalmente. Com isso, a busca por resultados aproximados se torna atrativa pelas características compartilhadas por grafos reais.

Assim, ao efetuar o tabelamento dos resultados obtidos na seção 4, foi possível observar que os algoritmos gulosos utilizados para a busca do clique máximo nos grafos apresentados, chegaram a valores muito próximos aos encontrados com o algoritmo ótimo. Outro ponto importante à observar é que, mesmo ao levar em conta o caráter exponencial do algoritmo ótimo, o mesmo foi utilizado de forma satisfatória na totalidade dos grafos empregados no desenvolvimento deste artigo, chegando, em todos eles, aos seus respectivos valores de clique máximo. Tais fatos demonstram o comportamento diferenciado desses algoritmos em grafos *power law*. Em outros tipos de grafos, os resultados obtidos com os mesmos algoritmos poderiam ser insatisfatórios.

Como demonstrado experimentalmente neste artigo, e também em [10], é, portanto, possível explorar as características presentes nas redes complexas a fim de implementar algoritmos de otimização para a resolução de diversos problemas. Surge, dessa forma, a oportunidade para a utilização das redes complexas e suas propriedades, explorando-as em outros problemas, diferentes cenários e abordagens de estudo.

Referências

[1] Gephi datasets. `https://github.com/gephi/gephi/wiki/Datasets`. Accessed: 2016-03-31.

[2] Pajek datasets. `http://vlado.fmf.uni-lj.si/pub/networks/data`. Accessed: 2016-03-31.

[3] Sagemath library. `http://www.sagemath.org/library.html`. Accessed: 2016-03-31.

[4] Stanford large network dataset collec. `http://snap.stanford.edu/data`. Accessed: 2016-03-31.

[5] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59(1):133–142, 2011.

[6] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[7] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[8] Renato Carmo and Alexandre Züge. Branch and bound algorithms for the maximum clique problem under a unified framework. *Journal of the Brazilian Computer Society*, 18(2):137–151, 2012.

[9] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[10] Mariana O Da Silva, Gustavo A Gimenez-Lugo, and Murilo VG Da Silva. Vertex cover in complex networks. *International Journal of Modern Physics C*, 24(11):1350078, 2013.

[11] Erik D Demaine, Felix Reidl, Peter Rossmanith, Fernando Sanchez Villaamil, Somnath Sikdar, and Blair D Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *arXiv preprint arXiv:1406.2587*, 2014.

[12] John D Eblen, Charles A Phillips, Gary L Rogers, and Michael A Langston. The maximum clique enumeration problem: algorithms, applications, and implementations. *BMC bioinformatics*, 13(10):1, 2012.

[13] Stephen Eubank, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, and Nan Wang. Structural and algorithmic aspects of massive social networks. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '04, pages 718–727, Philadelphia, PA, USA, 2004. Society for Industrial and Applied Mathematics.

[14] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Conductance and congestion in power law graphs. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):148–159, 2003.

[15] Roland Kopf and Günther Ruhe. A computational study of the weighted independent set problem for general graphs. *Found. Control Eng.*, 12(4):167–180, 1987.

[16] R Garey Michael and S Johnson David. Computers and intractability: a guide to the theory of np-completeness. *WH Free. Co., San Fr*, 1979.

[17] Mark Newman. *Networks: an introduction*. OUP Oxford, 2010.

[18] Kihong Park and Heejo Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *ACM SIGCOMM computer communication review*, volume 31, pages 15–26. ACM, 2001.

[19] Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Transactions on Networking (TON)*, 11(4):514–524, 2003.

[20] Etsuji Tomita, Tatsuya Akutsu, and Tsutomu Matsunaga. Efficient algorithms for finding maximum and maximal cliques: Effective tools for bioinformatics.

[21] Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

[22] Patric R.J. Östergård. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics*, 120(1–3):197 – 207, 2002. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.