Utilizando um Algoritmo Guloso para o Problema do Clique

Matheus Melo, Natália Almada
Departamento de Cência da Computação, Universidade Federal de Roraima
Boa Vista, Roraima, Brasil
matheus.syn.eco@gmail.com
salmada.compsci@gmail.com

***Resumo*— Este artigo propõe a implementação de um algoritmo guloso um algoritmo guloso desenvolvido para aproximar a solução do problema do clique máximo em um grafo. O algoritmo funciona de forma iterativa, e inicia pela seleção do vértice com o maior número de adjacências, utilizando uma heurística local que assume que vértices com mais conexões têm maior probabilidade de fazer parte de um grande clique. Após a escolha do vértice inicial, o algoritmo expande o clique, incluindo outros vértices que são adjacentes a todos os membros do clique atual. Mostramos também sua complexidade, tempos de execução e *benchmarks*, além da discussão dos resultados.**

I. INTRODUÇÃO

O problema do clique máximo em grafos é um dos problemas clássicos em teoria dos grafos e otimização combinatória, possuindo diversas diversas aplicações práticas, tais como análise de redes sociais, bioinformática e redes de comunicação[1]. Dado um grafo não dirigido, o problema consiste em encontrar um subconjunto de vértices em que todos estão conectados entre si, ou seja, formar um subgrafo completo. Este problema é conhecido por sua complexidade NP-difícil, o que torna inviável a aplicação de algoritmos exatos para instâncias grandes devido ao crescimento exponencial do tempo de execução[3].

Nesse contexto, algoritmos gulosos se destacam como uma solução aproximada, pois oferecem uma maneira usualmente rápida e eficiente de obter uma solução, mesmo que não seja a ótima[7]. A abordagem gulosa se baseia em uma estratégia local, onde o algoritmo toma decisões iterativas que parecem ser as melhores no momento, escolhendo o vertice com maior quantidade de arestas, sem considerar o impacto dessas escolhas no futuro. Essas características tornam os algoritmos gulosos adequados para problemas de grande escala, como aqueles encontrados em redes sociais e grafos biológicos.

Neste trabalho, propomos a implementação de um algoritmo guloso para aproximar a solução do problema do clique máximo. A estratégia utilizada se baseia na escolha do vértice com o maior número de conexões (ou adjacências) e, em seguida, na expansão do clique incluindo outros vértices que mantêm a condição de adjacência com todos os membros do clique atual. A simplicidade dessa abordagem permite que ela seja executada em tempo polinomial, tornando-a adequada para grafos de tamanho moderado a grande, onde algoritmos exatos seriam inviáveis.

O algoritmo desenvolvido foi testado com benchmarks DIMACS[4], demonstrando eficiência em termos de tempo de execução, apesar de não garantir a solução ótima. Esta abordagem é embasada em conceitos presentes na literatura clássica de algoritmos, como descritos por Cormen et al. em "Introduction to Algorithms", além de heurísticas desenvolvidas para problemas combinatórios.

II. REFERENCIAL TEÓRICO

O termo "Clique" foi introduzido por Luce e Perry, em 1949[8] em um artigo publicado na área de Ciências Sociais, onde associaram subgrafos completos a cliques sociais. Mas apenas na década de 1970, Harary e Ross [5]propuseram uma solução para o problema do clique, definindo que um clique em um grafo é um conjunto de vértices em que cada par de vértices está conectado por uma aresta, ou seja, um subgrafo completo dentro do grafo. O tamanho de um clique corresponde ao número de vértices que ele contém. Por exemplo, um clique de tamanho 4, chamado de 4-clique, é formado por quatro vértices, todos conectados entre si por arestas.

A. Clique Máximo

Determina-se clique máximo quando o clique de maior tamanho dentro de um grafo nos quais todos os vertices tem conexçao entre si [7].

B. NP-Completude

Um artigo publicado em 1972 por Richard M. Karp[6] colocou o Problema do cliquem entre 21 algoritmos NP-Completos, provando através de redução do problema SAT para fórmulas na forma normal conjuntiva, utilizando o teorema de Stephen Cook[12] de 1971.

C. Benchmark

São essenciais na avaliação do desempenho dos algoritmos propostos, pois adronizam as comparações em dominios diferentes, auxiliando e ajudando a enxergar a idedentificação de vantagens e limitações entre diferentes algortmos para o resolver o mesmo problema. Garantindo dessa forma a confiabilidade nos experimentos e a possibilidade de serem reproduzidos por outras pessoas[10].

III. METODOLOGIA

A presente pesquisa adotou uma metodologia que incluiu tanto a revisão da solução proposta por Bernardo e Ferreira[2] em seu trabalho final sobre o problema do clique, realizado em 2023, quanto a busca por referências adicionais que pudessem enriquecer e embasar a abordagem adotada por McCaffrey[9]. A revisão foi conduzida por meio de uma análise dos repositório, associado à adaptação para o método requerido para a realização do trabalho.

Os benchmarks foram escolhidos com base na consolidação da

qualidade da biblioteca do Centro de Matemática Discreta e Ciência da Computação Teórica (DIMACS)[4], pois além de dados seguros, as características deles, tais como tamanho e desidade foram prmordiais para comparação do funcionamento do algoritmo.

Durante a execução dos benchmarks, foram coletados dados de tempo de execução e quantidade de soluções encontradas. Esses dados foram analisados e comparados com os resultados esperados, permitindo assim uma avaliação da solução proposta em termos de eficiência, eficácia e robustez para a identificação de cliques máximos nos grafos.

O Dataset foi escolhido a partir da disponibilização do Professor Dr. Herbert Oliveira Rocha, para entradas grandes de dados provenientes da rede social X(Twitter), buscando oferecer uma análise aplicável a probelmas reais, trazendo possíveis otimizações ou limitações na abordagem proposta.

A. Algoritmo Guloso

O algoritmo proposto para encontrar o clique máximo em grafos é baseado Algoritmos Gulosos(*Greedy*), que se baseia na escolha do vértice com o maior número de conexões (ou adjacências) e, em seguida, na expansão do clique incluindo outros vértices que mantêm a condição de adjacência com todos os membros do clique atual. Dessa forma, ele não funciona prevendo o futuro[3].

1. Inicialmente, o algoritmo guloso seleciona o vértice com o maior número de adjacências, assumindo que os vértices que possuem mais conexões têm maior probabilidade de fazer parte de um grande clique.
2. Em seguida, o algoritmo verifica, iterativamente, quais outros vértices podem ser adicionados ao clique já formado, garantindo que cada novo vértice selecionado seja adjacente a todos os vértices do clique atual.
3. O processo continua até que não seja mais possível adicionar novos vértices ao clique, ou seja, quando a inclusão de qualquer novo vértice violaria a condição de adjacência total.
4. Ao final, o algoritmo retorna o maior clique encontrado durante o processo de expansão.

A complexidade do algoritmo guloso é consideravelmente mais eficiente do que a de algoritmos exatos. Contudo, ele ainda tem uma complexidade de $O(n^2)$ no pior caso, onde n é o número de vértices. Isso ocorre porque o algoritmo precisa verificar os vizinhos de cada vértice e garantir que eles formem um clique válido, o que envolve comparações entre os vértices[3].

B. Vantagens e Desvantagens

As **vantagens** de um algoritmo guloso estão na facilidade de entender e implementar, o que o torna uma solução prática para muitos problemas de clique máximo. Oferece uma solução aproximada para o problema do clique máximo com tempo de execução $O(n^2)$, o que é viável para grafos de tamanho moderado.E, Comparado a algoritmos exatos, o algoritmo guloso consome menos tempo e memória, sendo ideal para casos onde uma solução rápida é mais importante do que a exatidão.

Entretando, em suas **desvantagens**, temos sua inconsistência na execução, a dependência da heurística, desemprenho ruim quando se trata de grafos esparsos e a não garantia de solução ótima.
Por isso, a escolha de um algoritmo guloso para solucionar o problema do clique máximo vai depender das restrições e necessidades da aplicação de cada grafo[1].

C. Testes

Os testes do algoritmo foram realizados com 7 entradas diferentes, cujos dados foram coletados e plotados em gráficos. As Bases fornecidas pelo DIMACS[4], 2 grafos de nome C; BROCK, que contém 3 grafos e foi feita pelos pesquisadores Mark Brockington e Joe Culberson, e GEN que contém 2 grafos e foi desenvolvida por Laura Sanchis:

Nome do Grafo	Número de Nós	Número de Arestas	Clique Máximo Conhecido	Solução Aproximada (Guloso)
brock200_2	200	9876	21	8
brock200_4	200	13089	17	12
brock800_2	800	208166	24	14

Tabela 1: Tabela de informações sobre os 3 grafos do dataset de Brock.

Nome do Grafo	Número de Nós	Número de Arestas	Clique Máximo Conhecido	Solução Aproximada (Guloso)
gen200_p0.9_44	200	17977	44	30
gen400_p0.9_55	400	71956	55	39

Tabela 2: Tabela de informações sobre os 2 grafos do dataset de Gen.

Nome do Grafo	Número de Nós	Número de Arestas	Clique Máximo Conhecido	Solução Aproximada (Guloso)
C125_9	125	6963	34	26
C500_9	500	235824	57	33

Tabela 3: Tabela de informações sobre os 2 grafos do dataset C.

B. Complexidade

1. Construção da Matriz de Adjacência: Se o grafo for fornecido como uma lista de arestas ou pares de vértices, construir a matriz de adjacência levaria $O(n^2)$, já que há possíveis pares de vértices a serem considerados.
2. Para cada vértice, contar o número de arestas incidentes (grau do vértice) leva $O(n)$ operações. Fazer isso para todos os vértices leva $O(n^2)$ no pior caso (em uma matriz

de adjacência, onde todas as operações de verificação de adjacências entre vértices são O(1).

3. Expansão do clique:Para cada vértice, o algoritmo verifica se ele é adjacente a todos os vértices do clique atual. Isso envolve verificar até O(n) vértices, e fazer isso para todos os vértices expande o clique até $O(n^2)$.

O cálculo de custo seria representado por $O(n^2)$, sem a necessidade de somatórios aninhados complexos, já que o algoritmo não tenta explorar todas as combinações possíveis de vértices, mas sim toma decisões localmente.[3, 7,11]

IV. RESULTADOS E DISCUSSÃO

Nesta seção,discutiremos os resultados obtidos na avaliação da solução adaptada a partir daproposta de Bernardo e Ferreira[2], e de McCaffrey[9]. Também discutiremos as limitações identificadas na solução e possíveis melhorias a serem consideradas.

Durante a avaliação experimental, aplicamos o algoritmo proposto nos grafos de teste disponibilizados no benchmark de grafos, de acordo com o que já foi mencionado ems eções anteriores, e observamos resultados imprecisos na detecção dos cliques máximos em relação aos valores esperados.

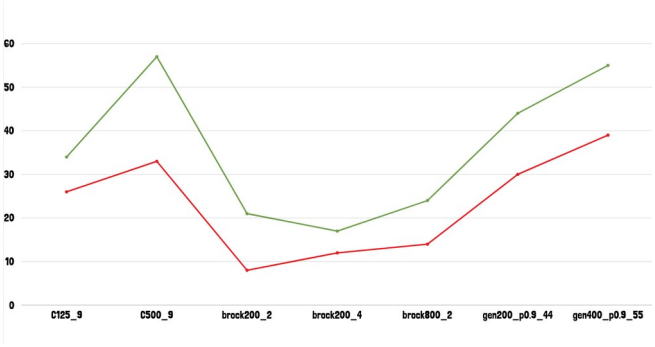

Gráfico 1: Gráfico de comparação entre quantidade de cliques máximos conhecidos(verde) e quiche máximo encontaado pelo algoritmo gulos (vermelho) no teste com os datasets.

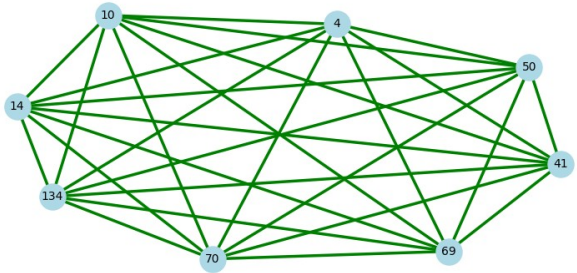

Figura 1: brock200_2, com 8 arestas encontradas em Tempo total Clique 0.000357 seg

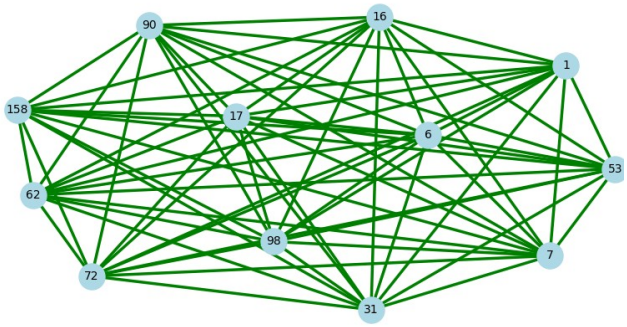

Figura 2: brock200_4, com 12 arestas encontradas em Tempo total Clique 0.000404 seg

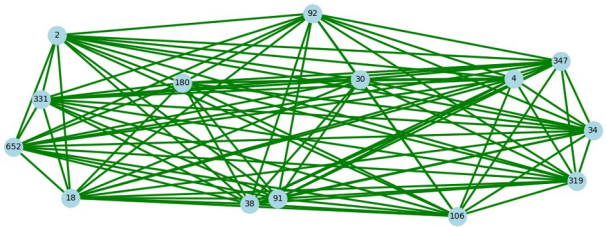

Figura 3:brock800_2 com 14 arestas encontradas em Tempo total Clique 0.004152 seg

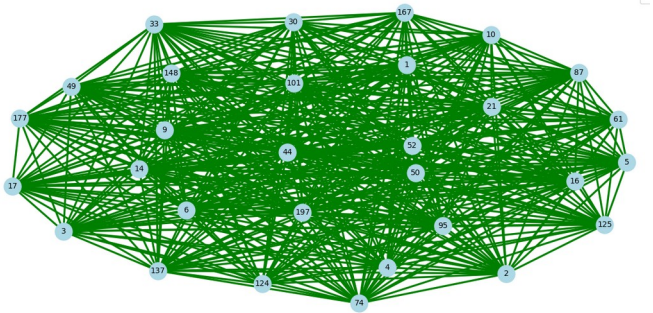

Figura 4: gen200_p0.9_44, com 30 arestas encontradas em Tempo total Clique 0.000232 seg

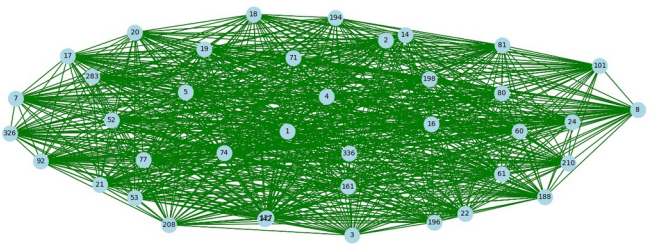

Figura 5: gen400_p0.9_55, com 39 arestas encontradas em Tempo total Clique 0.000523 seg

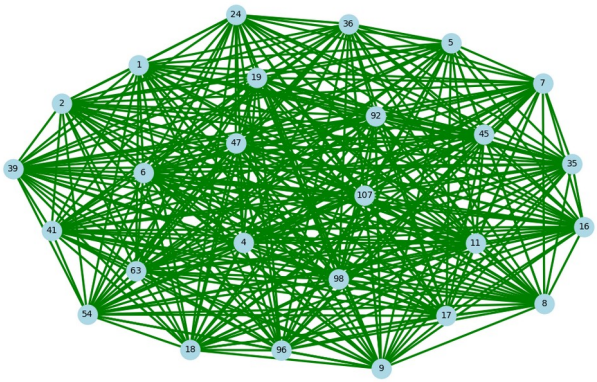

Figura 6: C125.9, com 26 arestas encontradas em Tempo total Clique 0.000072 seg

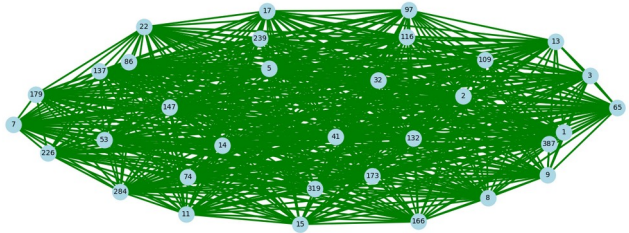

Figura 7: C500.9, com 33 arestas encontradas em Tempo total Clique 0.000834 seg

Conforme mostram os graficos e imagens plotadas, podemos perceber que o algoritmo ainda satisfaz resultados do benchmark, mesmo nãochegando aos resultados máximos já encontrados, perceber, em especial aproximação na execução do dataset brook200_4, que demonstra uma diferença de -5 vertices encontrados, um resultado ainda dentro do considerado esperado. Já o c500_9 demonstrou um desemprenho bastnate ruim se comparado o resultado esperado versus o resultado obtido, com diferença de -24 vertices.

O dataset proveniente do Kaggle requisitado pelo Professor não foi executado, por uma limitação tanto de hardware quanto do código implementado, apresentando erro fatal na execução.

Esses resultados indicam que a solução proposta possui algumas limitações na detecção dos cliques máximos, especialmente em grafos com maior densidade de arestas. Essas imprecisões podem comprometer a validade dos resultados obtidos e a correta identificação dos cliques máximos. A não leitura de um dataset robusta como a do Kaggle tabém apresenta uma falha na implementação do algoritmo.

Em alguns casos, a diferença diminui, indicando a possível existência de comprometimento a otimização e performance do algoritmo, que durante a execução e estudo dele não foi percebido e identificado para correção.

Em cocnlusão, percebemos que a solução proposta nesse trabalho apresenta muitas limitações na detecção dos cliques máximos e recomendamos a investigação dos algoritmos propostos para corrigir e otimizar os resultados, em especial a entrada de datassets mais robustos, como o do Kaggle.

V. CONCLUSÃO

Neste artigo, abordamos o clássico problema de teoria dos grafos. O problema do clique tem aplicalçao em diversas áreas e ainda é um problema que requer melhores soluções. Esta pesquise teve intenção de propor uma soluçãopara o problema, apresentando um algoritmo guloso para encontar os ciques maximos de grafos.

Apesar de os resultados não terem saído conforme o esperado, ainda forneceu uma análise crítica da solução proposta e se mostra promissor para futuras pesquisas na área. Para futuras pesquisas, sugerimos a otimização dessa abordagem ou a utilização de outros algoritmos para resolver esse problema de forma mais eficaz.

VI. REFERÊNCIAS

[1] BEHAR, Rachel; COHEN, Sara. Finding All Maximal Connected s-Cliques in Social Networks. In: International Conference on Extending Database Technology, 2018.

[2] BERNARDO, Guilherme; Ferreira, Kelvin. GuilhermeLucasKelvinAraujo_ProblemaDoClique_AA_RR_2023. 2023. Disponível em: https://github.com/GuilhermeBn198/GuilhermeLucasKelvinAraujo_ProblemaDoClique_AA_RR_2023. Acesso em: 11/09/2024

[3] CORMEN, Thomas H. et al. Algoritmos: Teoria e Prática. Tradução de Arlete Simille Marques. Rio de Janeiro: Elsevier, 2012.

[4] DIMACS Benchmark Set. The First DIMACS International Algorithm Implementation Challenge: The Maximum Clique, Graph Coloring, and Satisfiability Problems. Rutgers University, 1993. Disponível em: https://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark. Acesso em: 17/09/2024.

[5] HARARY, Frank; ROSS, Ivan C. "A procedure for clique detection using the group matrix." Sociometry, v. 20, n. 3, p. 205-215, 1957. Disponível em: <https://www.jstor.org/stable/2785673>. doi: 10.2307/2785673.

[6] KARP, M. Richard. Reducibility among combinatorial problems. In: MILLER, R. E.;

[7] KORTE, Bernhard; VYGEN, Jens. Combinatorial Optimization: Theory and Algorithms. 5. ed. Berlin, Heidelberg: Springer, 2018.

[8] LUCE, R. Duncan; PERRY, Albert D. A method of matrix analysis of group structure. Psychometrika, v. 14, n. 2, p. 95-116, 1949. Disponível em: https://www.jstor.org/stable/18152948. doi: 10.1007/BF02289146.

[9] MCCAFFREY, James. Test Run - Greedy Algorithms and Maximum Clique. MSDN Magazine, v. 26, n. 11, novembro 2011.

Disponível em: https://learn.microsoft.com/en-us/archive/msdn-magazine/2011/november/test-run-greedy-algorithms-and-maximum-clique. Acesso em: 11/09/2024.

[10] SAAVEDRA, Rafael H.; SMITH, Alan J. Analysis of Benchmark Characteristics and Benchmark Performance Prediction. ACM Transactions on Computer Systems, v. 14, n. 4, p. 344-384, Nov. 1996. Disponível em: https://doi.org/10.1145/235543.235545.

[11] THATCHER, J. W. (Eds.). Complexity of Computer Computations. New York: Plenum, [s.d.]. p. 85-103.

[12] COOK, Stephen. The complexity of theorem proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing. [S.l.: s.n.], 1971. p. 151-158.

Repositório GitHub:https://github.com/nataliaalmada/ProjetoFinalAA2024_Natalia_Matheus