PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e de Informática

Trabalho final - Arborescência*

Projeto e Analise de Algoritmos - ICEI - Puc Minas

Andrei Gonçalves Rohlfs Massaini[1]
Gabriel de Oliveira Barbosa[2]
Gustavo Pereira Cristofaro[3]
João Gabriel Polonio Teixeira[4]
Luiz Fernando Carneiro Rodrigues[5]
Victor Lustosa[6]

Resumo

O cálculo de uma árvore geradora mínima direcionada, chamada arborescência, é um problema algorítmico fundamental da computação, embora não seja tão comum quanto a sua contraparte não direcionada. Este trabalho apresenta uma análise comparativa entre dois algoritmos utilizados na resolução do problema de encontrar árvores geradoras mínimas em grafos direcionados: o Algoritmo de Edmonds e o GGST. A problemática da árvore geradora mínima desempenha um papel crucial em diversas aplicações, desde redes de comunicação até otimização de rotas.

O Algoritmo de Edmonds, conhecido por sua eficiência em grafos direcionados, foi examinado em detalhes, destacando suas características, princípios fundamentais e complexidade computacional. Da mesma forma, o GGST, uma alternativa contemporânea, foi explorado, revelando suas inovações e abordagens distintas para o mesmo desafio.A comparação entre os dois algoritmos foi realizada considerando diversos critérios, como desempenho computacional, escalabilidade, adaptabilidade a diferentes tipos de grafos e complexidade de implementação. Serão apresentados casos de estudo e experimentos práticos similares aos do (BöTHER et al., 2023) para ilustrar o desempenho relativo de cada algoritmo em cenários específicos.

*Artigo apresentado ao Instituto de Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais, como trabalho prático da disciplina de projeto e análise de algoritmos.

[1]Aluno do Programa de Graduação em Ciência da Computação, Brasil – andrei.massaini@hotmail.com.
[2]Aluno do Programa de Graduação em Ciência da Computação, Brasil – fabioleandro@pucminas.br.
[3]Aluno do Programa de Graduação em Ciência da Computação, Brasil – gcristofaro@sga.pucminas.br.
[4]Aluno do Programa de Graduação em Ciência da Computação, Brasil – joao.polonio@sga.pucminas.br.
[5]Aluno do Programa de Graduação em Ciência da Computação, Brasil – luizfernandotjk@gmail.com.
[6]Aluno do Programa de Graduação em Ciência da Computação, Brasil – victor.lustosa@sga.pucminas.br.

1 INTRODUÇÃO

O problema da árvore geradora mínima e o problema da arborescência geradora mínima são dois importantes desafios na área da computação. Ambos problemas são similares, mas o problema da arborescência geradora mínima é a versão direcionada do problema anterior.

Os dois problemas estão relacionados à busca por estruturas específicas em um grafo ponderado, com o objetivo de encontrar subgrafos que conectem todos os vértices com custo mínimo. A árvore geradora mínima refere-se à procura por uma árvore que conecte todos os vértices de um grafo ponderado, minimizando a soma dos pesos das arestas. Já o problema da arborescência geradora mínima, são usados grafos direcionados, onde se busca encontrar uma estrutura em forma de árvore que conecte todos os vértices a um vértice específico, chamado de raiz, minimizando o custo total das arestas. O problema da árvore geradora mínima é extensivamente estudado. Por outro lado, o problema da arborescência geradora mínima recebeu proporcionalmente menos atenção ao longo dos anos.

O algoritmo de Edmonds, também reconhecido como o método para encontrar uma arborescência geradora mínima, foi descoberto de forma independente por Edmonds (EDMONDS, 1967), Chu (CHU, 1965) e Bock (BOCK, 1971). Karp (KARP, 1971) foi o pioneiro em fornecer uma prova combinatorial de sua correção. Esse algoritmo, chamado de Algoritmo de Edmonds, com uma complexidade de O(nm), serviu como fundamento para variações mais sofisticadas que se seguiram. Entre elas estão as versões desenvolvidas por Tarjan e Gabow (GABOW et al., 1986), as quais exibem complexidades temporais de O(min(n^2, m log n)) e O(n log n + m), respectivamente. A versão desenvolvida por Gabow (GABOW et al., 1986), é uma versão refinada do algoritmo de Tarjan, que iremos chamar de GGST.

Nesse trabalho, iremos realizar uma análise experimental do comportamento dos algoritmos de Edmonds e o GGST. Os experimentos realizados serão baseados nos experimentos abordados em (BöTHER et al., 2023), onde avaliaremos o desempenho dos algoritmos, utilizando grafos de diferentes tamanhos, para realizar uma análise gráfica do custo computacional.

2 ALGORITMO DE EDMONDS

O algoritmo implementado no código[7] é uma versão do algoritmo de Edmonds para encontrar uma arborescência mínima em um grafo direcionado. O problema de encontrar uma arborescência mínima em um grafo, também conhecido como arborescência de custo mínimo, é um desafio fundamental em teoria dos grafos. Uma arborescência mínima é uma árvore direcionada que abrange todos os vértices do grafo, minimizando a soma dos pesos das arestas.

O algoritmo de Edmonds inicia a construção da arborescência mínima escolhendo, para cada vértice diferente do vértice de raiz r, a aresta de entrada mais barata $\pi(v)$. Se o conjunto dessas $n - 1$ arestas não contiver ciclos, temos uma arborescência mínima. Caso contrário, o

[7] https://github.com/spaghetti-source/algorithm

algoritmo contrai cada ciclo, escolhendo uma aresta específica para remover. Esta abordagem é crucial para garantir a otimalidade da solução, já que remove ciclos sem afetar a solução global.

A eficiência do algoritmo de Edmonds é evidenciada por sua análise de complexidade computacional, que é $\mathcal{O}(V \cdot A)$, onde V é o número de vértices e A é o número de arestas no grafo. Essa notação linear indica que o desempenho do algoritmo aumenta de maneira proporcional ao produto do número de vértices e arestas, demonstrando sua escalabilidade para problemas de diferentes tamanhos.

Para utilizar essa implementação em seus próprios projetos, você pode encontrar o código-fonte completo no GitHub[8] para o repositório. O código é escrito em C++ e pode ser integrado facilmente em outros projetos para resolver problemas relacionados a arborescências mínimas em grafos direcionados.

3 GGST

O GGST introduz uma abordagem estratégica ao processar vértices, formando um caminho de crescimento ao escolher consistentemente o vértice de origem da última aresta de entrada. Essa ordem deliberada é crucial para a otimização do algoritmo. A inserção de arestas fictícias com custo zero simplifica a descrição, mantendo um grafo totalmente conectado sem afetar o tempo de execução. O GGST gerencia ativamente as arestas por meio da floresta ativa, otimizando a construção dinâmica da árvore de abrangência mínima. Com uma complexidade de tempo de $O(n \log n + m)$, o GGST destaca-se como uma solução eficiente para o Problema de Árvore de Abrangência Mínima Direcionada. Sua estratégia de processamento e a manipulação cuidadosa de estruturas de dados, como a floresta ativa, contribuem para um desempenho superior em grafos de diversas densidades. A evolução do GGST a partir das bases de Edmonds representa uma conquista notável na eficiência dos algoritmos para grafos direcionados, sendo uma ferramenta valiosa.

O algoritmo GGST utilizado nesse trabalho, foi uma versão otimizada proposta pelo artigo (BöTHER et al., 2023)[9]. Primeiramente, não são inseridas arestas fictícias; ao invés disso, um novo caminho é iniciado sempre que a raiz é alcançada. Em segundo lugar, substituem-se listas encadeadas por arrays dinâmicos sempre que possível, modificando-se as listas de saída, listas passivas e o caminho de crescimento apenas na frente, permitindo o uso de arrays salvos de forma reversa. A terceira otimização consiste em eliminar a necessidade de referências cruzadas ao simplificar os padrões de exclusão, ocorrendo apenas durante a contração de um ciclo. A exclusão de arestas de saída é realizada limpando listas completas de saída e refletindo as exclusões nas listas passivas, enquanto múltiplas arestas de entrada são eliminadas limpando listas passivas completas e refletindo as exclusões nas listas de saída. Essas modificações sincronizadas facilitam a interação entre as listas de saída e passivas, restringindo as modificações à frente das listas. Além disso, propõe-se ignorar a remoção completa das listas passivas durante

[8] https://github.com/spaghetti-source/algorithm
[9] https://github.com/chistopher/arbok/

a exclusão de algumas arestas, mantendo entradas inválidas nelas, porém, estas se tornam auto laços que podem ser identificados e ignorados durante a contração, já que as listas passivas são limpas após essa identificação. Uma estratégia para consolidar multi-arestas é proposta: para cada aresta passiva no ciclo, compara-se as duas primeiras arestas na lista de saída da origem da aresta passiva e deleta-se a mais cara. Essa operação é repetida para cada origem que possui arestas passivas no ciclo, mantendo a aresta mais barata no início da lista de saída. Essa abordagem difere da proposta de Gabow et al., que deletam a primeira aresta ou a aresta passiva atualmente inspecionada, requerendo um método para obter o identificador de cada aresta passiva na lista de saída.

4 EXPERIMENTOS

Nesta seção, abordaremos a metodologia adotada para a comparação dos algoritmos na resolução do problema, descrevendo detalhadamente a condução dos testes, detalhando a propriedade da bases de dados utilizadas, assim como as especificações computacionais empregadas nos experimentos.

4.1 Setup

Os experimentos foram conduzidos em um computador equipado com um processador Intel Core i7 8700k, 16GB de memória RAM, e utilizando o sistema operacional Windows Subsystem for Linux (WSL). As implementações foram escritas em C++. Além disso, a versão de Edmonds foi adaptada para se ajustar à mesma interface proposta no artigo de Bother.

4.1.1 Base de dados

Para conduzir os experimentos, foram utilizados ao todo 755 instâncias de grafos das seguintes fontes. O número exato de cada uma delas se encontra nos parêntesis:

- **konect (319):** Todas as redes direcionadas com tamanho inferior a 5GB do projeto Konect.[10].
- **networkrepository (80):** Uma seleção de redes esparsas do projeto Network Repository[11]. O projeto consiste principalmente em redes não direcionadas e não rotula as que são direcionadas. Baixamos todas as redes (aproximadamente 3000) e mantivemos aquelas rotuladas como direcionadas em seus respectivos formatos de arquivo.

[10]https://konect.cc/
[11]http://networkrepository.com/

- **girgs (200):** Este conjunto de dados contém Grafos Geométricos Inhomogêneos Aleatórios (GIRGs), um modelo generativo de rede intimamente relacionado a grafos aleatórios hiperbólicos (KRIOUKOV et al., 2010) (BRINGMANN et al., 2019). Utilizamos o gerador eficiente proposto por (BLÄSIUS et al., 2019) com parâmetros padrão, exceto para n, deg e sementes. Definimos $n = 10^4$ e graus médios de 50 a 2000 em incrementos de 100, com 10 redes por configuração. A direção das arestas é gerada aleatoriamente.

- **antilemon (5):** Conjunto esparso de redes projetadas para serem desafiadoras para o problema de arborescência mínima. Elas exigem pelo menos $n/2$ contrações com pelo menos $n/2$ arestas apontando para cada ciclo contraído. Geramos redes com $n = 10^i$ para $i \in [2, 6]$.

- **fastestspeedrun (141):** Casos de teste de uma tarefa de programação da ICPC Northwestern Europe Regional Contest 2018[12]. Elas têm até 2500 vértices e são totalmente conectadas.

- **yosupo (10):** Casos de teste para o problema da Árvore de Spanning Direcionada no site Library Checker. As redes são grafos (ERDőS; RéNYI, 1959) com uma árvore de spanning aleatória a partir do vértice raiz como subgrafo. Pesos são amostrados uniformemente ao acaso.

Para instâncias que não eram ponderadas previamente, foi-se aplicada uma lógica para atribuir pesos às suas respectivas arestas de maneira uniforme e aleatória.

No caso de instâncias que não possuíam uma raiz especifica, restringimos o grafo com base em seu maior componente conexo e adicionamos um vértice raiz que se conecta com todos os demais vértices, e seu respectivo peso na aresta como infinito.

[12]https://2018.nwerc.eu/

5 RESULTADOS OBTIDOS

Nesta seção, abordaremos os resultados obtidos ao comparar a execução de ambos algoritmos no conjunto de dados mencionados anteriormente.

5.1 Tempo total de execução

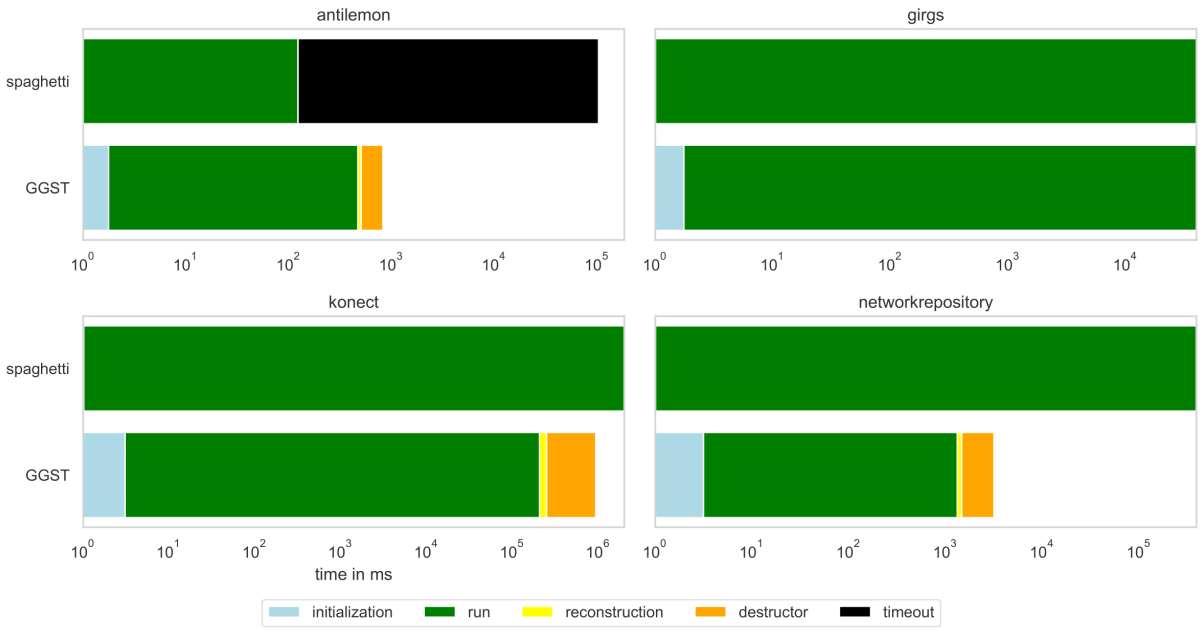

Figura 1 – Tempo de execução dos algoritmos por base de dados

Na figura 1, foram registrados os tempos de execução de cada algoritmo, incluindo as fases de inicialização, reconstrução e desconstrução de suas estruturas correspondentes. Importante observar que, na implementação específica do algoritmo de Edmonds utilizada, tais fases não foram disponibilizadas, uma vez que essa implementação foi concebida de maneira concisa e direta, focando exclusivamente na execução total do algoritmo.

Pode-se observar que no conjunto de dados *antelemon*, o algoritmo de Edmonds apresentou *timeout* em diversas instâncias desta base de dados. Esse comportamento pode ser atribuído à natureza esparsa dos grafos presentes nesse conjunto de dados, o que torna desafiante a execução eficiente do algoritmo não polinomial de Edmonds.

5.2 Proporção de empates

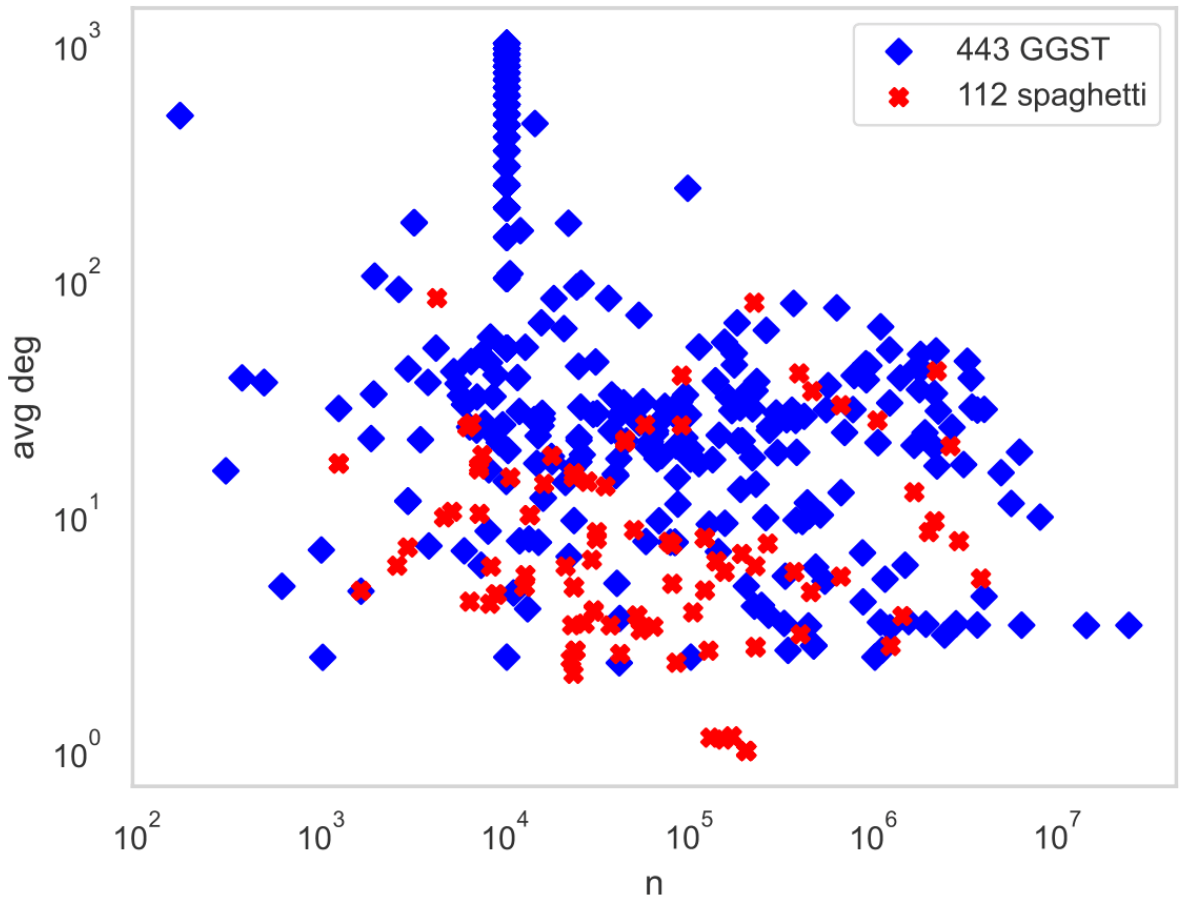

Figura 2 – Algoritmos mais rápidos e empates

A Figura 2 mostra a distribuição das proporções de empates entre os algoritmos avaliados. Entre as 755 instâncias testadas, a implementação *GGST* venceu em 443, enquanto o *Edmonds* foi superior em 112. Houve 200 empates, porém os mesmos não são relevantes uma vez que estes ocorreram em instâncias muito pequenas, onde houve uma diferença de menos de 1 milisegundo entre os dois algoritmos.

5.3 Análise de Escalabilidade

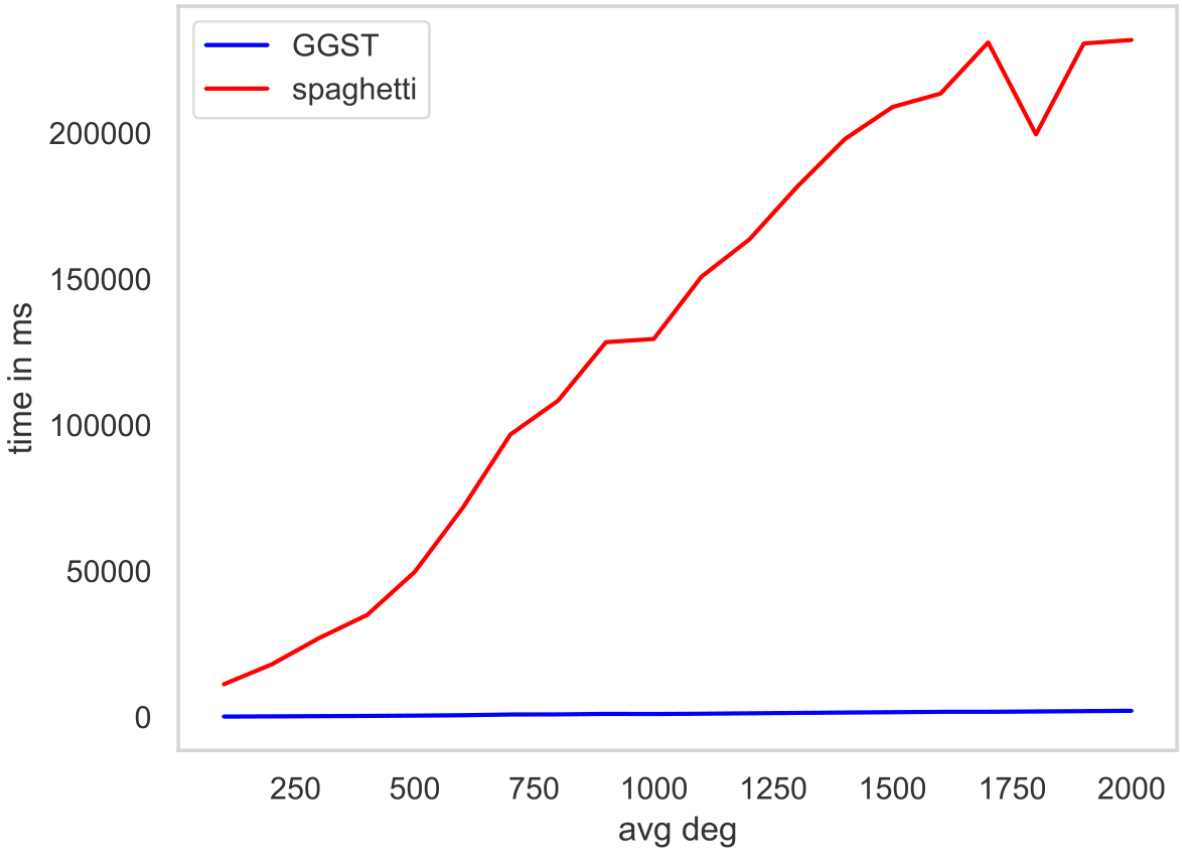

Figura 3 – Tempo de execução no conjunto de dados *Girgs* com base na densidade de arestas.

Na Figura 3, é apresentada a análise de escalabilidade para o tempo de execução em relação ao conjunto de dados *Girgs*, baseado na densidade de arestas.

A análise destaca o comportamento dos algoritmos em termos de escalabilidade, uma métrica crucial nos testes realizados. Observa-se que, na implementação *GGST*, o tempo de execução das instâncias mostrou-se linear. Em contraste, a implementação de *Edmonds* apresentou um crescimento exponencial, tornando o algoritmo inviável para execução em grafos muito grandes.

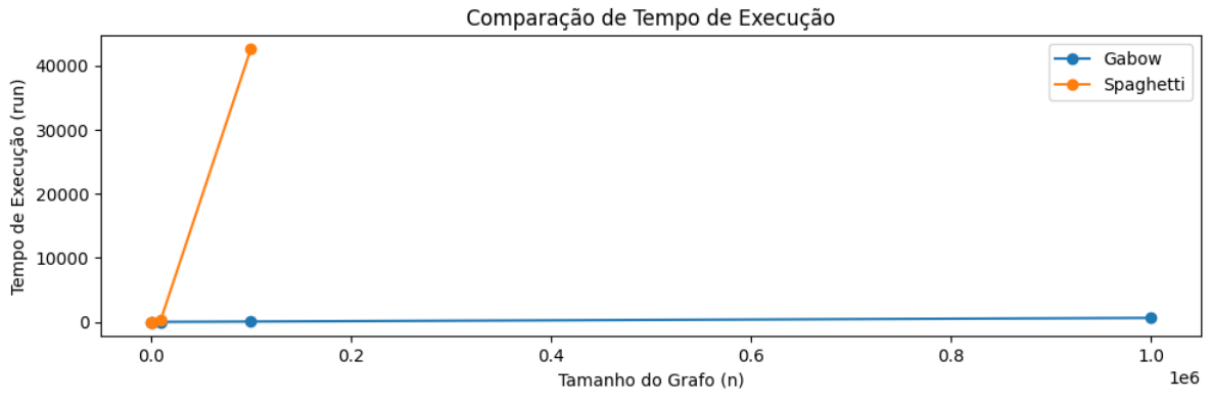

Figura 4 – Tempo de execução no conjunto de dados *antilemon* com base na densidade de arestas.

Na Figura 4, podemos observar o tempo de execução no conjunto de dados *antilemon*. Nota-se que o algoritmo de *Edmonds*, cresce exponencialmente e não chega nem a analisar todo o conjunto de dados, uma vez que ele excede o tempo estabelecido como *timeout* (30 minutos), em instâncias onde o número de arestas é consideravelmente maior, enfatizando ainda mais a sua ineficiência em resolver o problema em redes demasiadamente grandes.

6 CONCLUSÃO

A implementação do algoritmo GGST para o problema da árvore geradora mínima, apresenta uma notável superioridade em eficiência assintótica, destacando-se com uma complexidade de $O(n \cdot \log^2 n + m)$, enquanto o algoritmo de Edmonds, possui uma complexidade de $\mathcal{O}(V \cdot A)$, onde V é o número de vértices e A é o número de arestas. A abordagem inovadora do GGST de representação de ciclo persistente, a manutenção eficiente de informações sobre arestas de entrada durante o algoritmo principal e a incorporação de uma estrutura de DSU persistente, contribuem para um desempenho excepcional. Testes em redes do mundo real evidenciam a robustez do GGST, superando consistentemente a implementação de Edmonds. Além disso, o GGST exibe eficiência notável em grafos esparsos, consolidando sua versatilidade e preferência em diversas situações práticas, tornando-a uma escolha preferencial em comparação com a implementação original de Edmonds.

REFERÊNCIAS

BLÄSIUS, T. et al. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. In: **Proceedings of the Annual European Symposium on Algorithms (ESA)**. [S.l.: s.n.], 2019.

BOCK, F. C. An algorithm to construct a minimum directed spanning tree in a directed network. **Developments in Operations Research**, 1971.

BRINGMANN, K.; KEUSCH, R.; LENGLER, J. Geometric inhomogeneous random graphs. **Theoretical Computer Science**, v. 760, 2019.

BöTHER, Maximilian; KIßIG, Otto; WEYAND, Christopher. Efficiently computing directed minimum spanning trees. 2023.

CHU, Y.-J. On the shortest arborescence of a directed graph. **Scientia Sinica**, v. 14, 1965.

EDMONDS, J. Optimum branchings. **Journal of Research of the National Bureau of Standards**, v. 71B, p. 233, 1967.

ERDőS, P.; RéNYI, A. On random graphs, i. **Publicationes Mathematicae (Debrecen)**, v. 6, 1959.

GABOW, H. N. et al. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. **Combinatorica**, v. 6, 1986.

KARP, R. M. A simple derivation of edmonds' algorithm for optimum branchings. **Networks**, v. 1, 1971.

KRIOUKOV, D. et al. Hyperbolic geometry of complex networks. **Physical Review E**, v. 82, 2010.