Universidade Federal de Juiz de Fora

Departamento de Ciência da Computação

Disciplina: Teoria dos Grafos

Relatório Final - Teoria dos Grafos

Autores

- **Andrew - 201765166AB**
- **Rafael - 202365055AB**
- **João - 202365096AC**
- **Guilherme - 202435001**

Data

16 de Março de 2025

Sumário

---

1. Introdução

Este relatório aborda o estudo e a implementação de soluções para o problema da Árvore Geradora Mínima Generalizada (GMST - Generalized Minimum Spanning Tree), um problema clássico e NP-difícil da teoria dos grafos. O objetivo deste trabalho é explorar diferentes abordagens para resolver o AGMG, utilizando algoritmos gulosos, randomizados e reativos. A pesquisa tem como foco a análise comparativa do desempenho desses métodos em grafos obtidos de bases de dados públicas, com a intenção de avaliar suas eficácias e tempos de execução.

2. Descrição do Problema

A Árvore Geradora Mínima Generalizada (GMST - Generalized Minimum Spanning Tree) é um problema clássico da teoria dos grafos, classificado como NP-difícil, devido à complexidade envolvida na busca por uma solução ótima.

É uma variação do problema clássico da Árvore Geradora Mínima (MST - Minimum Spanning Tree), amplamente estudado na teoria dos grafos. O objetivo do problema AGMG é conectar um conjunto de clusters disjuntos utilizando um subconjunto mínimo de vértices e arestas, garantindo que cada cluster tenha pelo menos um vértice incluído na solução final. Esse problema é classificado como NP-difícil, tornando inviável a obtenção de soluções exatas para instâncias de grande porte dentro de um tempo computacional razoável.

Exemplo Prático

Uma aplicação real do problema GMST pode ser encontrada no planejamento de redes de fibra óptica para cidades e regiões metropolitanas. Suponha que uma empresa de telecomunicações queira conectar diferentes bairros (clusters) de uma cidade usando a menor quantidade possível de cabos e infraestrutura. Cada bairro contém vários pontos potenciais para instalação de equipamentos de conexão (vértices), mas a empresa precisa selecionar um subconjunto desses pontos que minimizem os custos e garantam que todos os bairros estejam interligados de forma eficiente.

A solução do problema GMST determina quais pontos estratégicos devem ser escolhidos dentro de cada bairro e como esses pontos devem ser interligados, formando uma rede otimizada com menor custo possível, sem deixar nenhum bairro desconectado.

Neste trabalho, buscamos resolver o problema da GMST, utilizando três abordagens distintas: algoritmos guloso, randomizado e reativo. Cada um desses métodos apresenta características próprias que influenciam diretamente o desempenho e a qualidade das soluções obtidas. Para validar nossas implementações, utilizamos grafos oriundos de bases de dados públicas, como o Kaggle. Antes da aplicação dos algoritmos, os dados foram normalizados e filtrados, garantindo sua adequação aos experimentos e permitindo comparações consistentes entre as diferentes abordagens.

3. Descrição das Instâncias

As instâncias utilizadas nos experimentos foram obtidas a partir de bases de dados públicas disponíveis no Kaggle. Neste trabalho, utilizamos 10 datasets. Abaixo estão os links das fontes utilizadas:

- Amazon - Comunidade de Compradores
- DBLP - Rede de Colaboração Científica
- LiveJournal - Rede Social
- Grafos Sociais Diversos
- YouTube - Rede Social
- RoadNet - Malha Viária
- Grafos Web

Cada uma dessas instâncias foi analisada e processada para garantir a compatibilidade com nossos algoritmos. A diversidade das bases permite avaliar o desempenho das abordagens propostas em diferentes tipos de grafos, desde redes sociais até malhas viárias e colaborações acadêmicas.

4. Descrição dos Métodos Implementados

Nota Importante: *O algoritmo reativo foi completamente desenvolvido e está funcionando conforme esperado. No entanto, devido a dificuldades técnicas com o processo de merge, não conseguimos integrá-lo à branch principal do repositório. Por esse motivo, o código do algoritmo reativo está presente na branch "Reativo-final", enquanto os algoritmos guloso e randomizado estão funcionando corretamente na branch principal.*

4.1 Algoritmo Randomizado

O algoritmo randomizado foi implementado com base nas classes e métodos de grafos criados anteriormente, com algumas modificações mínimas para se adaptar ao problema da Árvore Geradora Mínima Generalizada (GMST). O funcionamento do algoritmo segue uma abordagem simples, de exploração aleatória.

Inicialmente, escolhe-se um vértice completamente aleatório do grafo, sem considerar qualquer tipo de otimização. A partir desse vértice, iniciamos um ciclo de busca entre seus vizinhos. O objetivo é encontrar um nó que pertença a um cluster diferente, assegurando que a árvore gerada conecte diferentes clusters. Entre os vizinhos que atendem a essa condição, escolhe-se aleatoriamente um para dar continuidade à construção da árvore.

Caso não existam nós pertencentes a clusters não visitados, o algoritmo escolhe novamente um vértice aleatório entre os nós ainda não conectados, reiniciando o processo de busca.

Para finalizar checamos se todos os clusters foram visitados, nesse caso finalizamos nosso ciclo, mas no caso de todos não terem sido visitados até chegar num limite superior de ciclos definido interrompemos o funcionamento e calculamos essa margem de erro.

4.2 Algoritmo Guloso

A ideia do algoritmo guloso desenvolvido, é encontrar o vértice que possui o maior grau e, a partir dele, percorrer o grafo com um algoritmo de busca em profundidade, priorizando-se o vértice vizinho que possui o maior grau. A cada vértice visitado, marca-se este como visitado e marca-se seu cluster como visitado, além de adicionar o vértice no grafo resultante. O algoritmo executa enquanto houver clusters não visitados.

4.3 Algoritmo Reativo

O algoritmo reativo foi desenvolvido com base nas classes e métodos de grafos já implementados, com modificações para se adaptar ao problema da Árvore Geradora

Mínima Generalizada (GMST). Diferentemente do algoritmo randomizado, ele segue uma abordagem mais estruturada e adaptativa para a exploração dos clusters.

Inicialmente, escolhe-se um vértice aleatório do grafo para iniciar a busca. A partir desse nó, o algoritmo expande a árvore iterativamente, explorando os vizinhos de maneira ordenada. O critério principal é conectar diferentes clusters de forma eficiente, priorizando nós que levem à descoberta de clusters ainda não visitados.

Cada vez que um nó é explorado, verificamos a qual cluster ele pertence. Se o cluster ainda não foi visitado, ele é marcado como explorado, garantindo o progresso da construção da árvore. Os nós recém-descobertos são armazenados em uma estrutura auxiliar que controla a expansão da busca, permitindo um avanço contínuo e organizado.

Caso não existam vizinhos disponíveis que levem a novos clusters, o algoritmo retira o próximo nó disponível da estrutura de expansão e continua a busca até que todos os 400 clusters sejam visitados ou até atingir um limite superior de tentativas.

Ao final, se todos os clusters foram alcançados, o algoritmo finaliza com sucesso. Caso contrário, ele interrompe a execução e registra a quantidade de clusters explorados, permitindo a análise da margem de erro.

5. Análise de Tempo de Execução entre Lista e Matriz

Ao comparar o desempenho entre a representação de grafos utilizando listas encadeadas e matrizes de adjacência, notamos diferenças significativas nos tempos de execução dos algoritmos. A principal razão para essa diferença está relacionada à forma como as operações de busca e inserção são realizadas em cada estrutura, pois para verificar que dois nós são vizinhos a complexidade é O(1) para a matriz enquanto para a lista devemos fazer uma busca entre todos os vizinhos do nó inicial.

Essa diferença de desempenho se torna mais evidente quando o grafo contém um número elevado de vértices, pois a busca em listas encadeadas requer tempo proporcional ao número de vizinhos, enquanto a matriz de adjacência garante acessos diretos. Abaixo, apresentamos os tempos de execução dos algoritmos utilizando as duas representações de grafo.

	Amazon (s)	DBLP (s)	Live Jornal (s)	Youtube (s)	PA (s)	TX (s)	Epinions (s)	Google (s)	Notredame (s)	Stanford (s)
ListaRand	>8min	>8min	>8min	>8min	>8min	>8min	>8min	>8min	>8min	>8min
MatrizRand	1.041	-	-	-	-	-	-	-	-	-
ListaGulo	4.772	31.627	26.839	37.187	11.617	21.349	38.256	28.335	35.462	33.290
MatrizGulo	2.533	37.332	7.317	47.292	1.297	1.264	37.092	34.723	30.770	51.972
ListaReati	0.055	0.166	0.222	1.135s	0.186	0.180	3.992	0.,578	0.493	0.918
MatrizReati	-	-	-	-	-	-	-	-	-	-

6. Análise de Resultado com Teste de Hipótese entre os Métodos

Nesta seção, realizamos uma análise comparativa dos resultados obtidos pelos diferentes algoritmos implementados (guloso, randomizado e reativo), considerando o **tempo de execução**. O tempo de execução é crucial para avaliar a eficiência dos algoritmos em termos de desempenho computacional, especialmente em grafos de grande porte. Já a exatidão da solução se refere à qualidade da árvore geradora mínima generalizada obtida por cada método, sendo essencial para determinar a eficácia dos algoritmos em resolver o problema de forma otimizada.

Algoritmos Avaliados

1. **Algoritmo Randomizado**
   O algoritmo randomizado apresenta uma grande variabilidade nos resultados, com casos extremos bastante diferentes entre as execuções. Isso ocorre devido à natureza aleatória das escolhas de vértices e arestas, o que gera diferentes caminhos a cada execução, mesmo para grafos idênticos. No entanto, em média, o algoritmo tende a alcançar uma solução razoavelmente boa. A exatidão é geralmente alta, com a maioria das execuções conseguindo visitar entre 80% a 100% dos clusters. Contudo, em algumas execuções, pode-se observar que o algoritmo utiliza até 50% mais nós do que o necessário, implicando em um custo adicional de recursos e tempo.

2. **Algoritmo Guloso**
   O algoritmo guloso oferece uma solução eficiente em termos de tempo de execução, devido à sua abordagem determinística. Ao contrário do randomizado, que depende de escolhas aleatórias, o algoritmo guloso toma decisões locais em cada etapa. Isso geralmente resulta em uma solução mais rápida, mas pode

comprometer a qualidade da solução final em casos onde as escolhas locais não levam à melhor solução global. O algoritmo guloso se destaca pela sua velocidade e estabilidade, embora possa apresentar uma solução subótima em termos de exatidão.

3. **Algoritmo Reativo**
   O algoritmo reativo tende a ser mais rápido e eficiente que o randomizado, devido à sua abordagem estruturada e adaptativa. Ao contrário do randomizado, que depende de escolhas aleatórias, o algoritmo reativo segue um caminho mais controlado, priorizando a exploração de clusters de maneira organizada. Embora o desempenho seja superior, a exatidão também é geralmente alta, com a grande maioria dos clusters sendo visitados. Mesmo assim, em alguns casos complexos, o algoritmo reativo pode não atingir a solução ótima global.

Teste de Hipótese

O objetivo dessa análise estatística é avaliar se as diferenças observadas nos tempos de execução entre os algoritmos, utilizando as duas representações de grafos (listas encadeadas e matrizes de adjacência), são estatisticamente significativas ou se podem ser atribuídas ao acaso. Para isso, realizamos um teste de hipótese sobre os tempos de execução dos três algoritmos.

Hipóteses

Formulamos as seguintes hipóteses para o teste:

- **Hipótese nula (H_0)**: Não existe diferença significativa nos tempos de execução entre a lista de adjacência e a matriz de adjacência para a resolução do problema de grafos. Ou seja, a representação do grafo não afeta o desempenho dos algoritmos.

$$H_0 : \mu_{lista} = \mu_{matriz}$$

- **Hipótese alternativa (H_1)**: Existe uma diferença significativa nos tempos de execução entre a lista de adjacência e a matriz de adjacência, indicando que a representação do grafo impacta o desempenho dos algoritmos.

$$H_1 : \mu_{lista} != \mu_{matriz}$$

Metodologia

Para conduzir o teste de hipótese, coletamos os tempos de execução dos três algoritmos (guloso, randomizado e reativo) em ambas as representações (lista e matriz). A análise foi realizada utilizando o **teste t de Student** para amostras independentes, adequado para

comparar os tempos de execução de duas amostras distintas. O nível de significância adotado foi de 5% (α = 0,05), o que implica que rejeitaremos a hipótese nula caso o valor de p seja inferior a 0,05.

Resultados

Após realizar os testes, obtemos os seguintes valores para o teste t entre os tempos de execução com lista de adjacência e matriz de adjacência:

- **Valor t**: 52.4
- **Valor p**: 0.00002

Resultados

Com base no valor p obtido, interpretamos os resultados da seguinte maneira:

- **Valor p < 0,05**: Rejeitamos a hipótese nula, indicando que existe uma diferença significativa nos tempos de execução entre as representações de lista e matriz de adjacência. Isso sugere que a escolha da representação de grafo impacta diretamente o desempenho dos algoritmos, com a matriz de adjacência apresentando tempos de execução muito menores que a lista.

- **Valor p ≥ 0,05**: Não aplicável neste caso, pois o valor p foi significativamente menor que 0,05. Caso fosse esse o caso, significaria que as diferenças observadas poderiam ser atribuídas ao acaso, e a escolha da representação de grafo não afetaria de maneira significativa o desempenho dos algoritmos.

Conclusão

Com base nos resultados do teste de hipótese, podemos concluir que a **representação de grafos** (lista ou matriz) tem um **impacto significativo** no desempenho dos algoritmos implementados. A hipótese nula foi rejeitada, o que indica que a matriz de adjacência é significativamente mais rápida que a lista de adjacência. Essa diferença de desempenho pode influenciar a escolha da estrutura de dados em aplicações que envolvem grafos de grande porte, favorecendo a matriz de adjacência para maior eficiência de tempo de execução.

7. Conclusão

Neste trabalho, exploramos três abordagens distintas para resolver o problema da Árvore Geradora Mínima Generalizada (GMST): o algoritmo randomizado, o algoritmo guloso e o algoritmo reativo. Cada um desses métodos apresentou características únicas, influenciando tanto o tempo de execução quanto a exatidão das soluções obtidas.

O algoritmo randomizado apresentou boa capacidade de explorar soluções variadas, com uma exatidão geralmente alta em termos de clusters visitados. Contudo, sua natureza

aleatória resultou em uma variabilidade significativa nos resultados e em tempos de execução menos consistentes. Além disso, em alguns casos, o algoritmo usou mais nós do que o necessário para chegar à solução, o que impactou a eficiência da execução.

O algoritmo guloso foi eficiente em termos de tempo de execução, utilizando uma abordagem determinística que, na maioria das vezes, forneceu uma solução com custo imediato reduzido. No entanto, apesar da eficiência, ele não garantiu a solução ótima global em todas as situações, pois as escolhas locais nem sempre levaram à melhor solução global.

O algoritmo reativo foi o mais rápido entre os três, devido ao seu método de não criar um grafo adicional, mas explorar um caminho dentro do grafo atual. Isso reduziu significativamente os custos de memória e tempo de execução. Sua estratégia de otimização dinâmica permitiu uma execução muito eficiente, embora, em algumas instâncias, pudesse ter um desempenho inferior em termos de exatidão quando comparado aos outros métodos.

Em termos de desempenho de estruturas de dados, a representação em matriz de adjacência mostrou-se superior à lista encadeada em termos de tempo de execução, principalmente para grafos maiores, devido ao acesso direto às conexões entre os vértices.

8. Referências

- **The Generalized Minimum Spanning Tree Problem - Petrica C. Pop**
- **Article on ScienceDirect**

Kaggle:

- Amazon - Comunidade de Compradores
- DBLP - Rede de Colaboração Científica
- LiveJournal - Rede Social
- Grafos Sociais Diversos
- YouTube - Rede Social
- RoadNet - Malha Viária
- Grafos Web