

Universidade Federal de Juiz de Fora

Trabalho Parte 3 Grafos - 2024.3

Relatório



Integrantes:

Felipe Lazzarini Cunha

Luiz Alberto Werneck Silva

17 de março de 2025

Sumário

1	Descrição do Problema	3
1.1	Introdução	3
1.2	Definição Formal	3
1.3	Exemplo Ilustrativo	4
1.4	Complexidade e Desafios	4
1.5	O Problema	4
1.5.1	Adequação do Problema à AGMG	5
1.5.2	Aplicação em <code>web-Google.txt</code>	5
1.5.3	Aplicação em <code>roadNet-CA.txt</code>	5
1.5.4	Aplicação em <code>CA-AstroPh.txt</code>	6
1.5.5	Benefícios da Adaptação para Grafos Não-Direcionados	6
1.5.6	Conclusão	6
2	Descrição das Instâncias	8
2.1	Arquivo <code>grafo-teste.txt</code>	8
2.1.1	Especificações Técnicas	8
2.1.2	Estrutura do Arquivo	8
2.2	Arquivo <code>grafo.txt</code>	9
2.2.1	Especificações Técnicas	9
2.2.2	Estrutura do Arquivo	9
2.2.3	Relação com os Datasets SNAP	10
2.3	Arquivo <code>web-Google.txt</code>	10
2.3.1	Especificações Técnicas	10
2.4	Arquivo <code>roadNet-CA.txt</code>	11
2.4.1	Especificações Técnicas	11
2.5	Arquivo <code>CA-AstroPh.txt</code>	11
2.5.1	Especificações Técnicas	11
3	Descrição dos Métodos Implementados	13
3.1	Classe Comunidade (<code>comunidade.cpp</code>)	13
3.2	<code>algoritmo_guloso.cpp</code>	14

3.3	<code>algoritmo_randomizado.cpp</code>	14
3.4	<code>algoritmo_relativo.cpp</code>	15
3.5	Integração via <code>main.cpp</code>	15
3.6	Conclusão	16
4	Análise de Tempo de Execução Entre Lista e Matriz	17
4.1	Resultados	18
4.1.1	<code>grafo.txt</code> - simulação rede social, base web-Google.txt	18
5	Análise de Resultados com Teste de Hipóteses Entre os Métodos	20
5.1	Metodologia de Comparação	20
5.2	Resultados Experimentais	21
5.2.1	Desempenho com Lista de Adjacência	21
5.2.2	Desempenho com Matriz de Adjacência	21
5.3	Análise Estatística e Teste de Hipóteses	21
5.3.1	Comparação de Modularidade	21
5.3.2	Análise de Densidade	22
5.3.3	Distribuição de Tamanho das Comunidades	22
5.4	Implicações para AGMG em Redes Sociais	23
5.4.1	Impacto da Distribuição Desbalanceada	23
5.4.2	Vantagem do Algoritmo Randomizado para AGMG	23
5.5	Análise de Robustez à Representação do Grafo	24
5.6	Conclusões e Recomendações	24
6	Conclusões	26
6.1	Síntese dos Resultados	26
6.2	Contribuições para o Estudo de AGMG	27
6.3	Limitações e Desafios	27
6.4	Direções Futuras	28
6.5	Considerações Finais	28

Capítulo 1

Descrição do Problema

1.1 Introdução

A Árvore Geradora Mínima Generalizada (AGMG), também conhecida como *Generalized Minimum Spanning Tree* (GMST), é uma variação do problema clássico da Árvore Geradora Mínima (MST). Enquanto a MST busca conectar todos os vértices de um grafo não dirigido e ponderado com o menor custo total, a AGMG considera grafos cujos vértices estão organizados em *clusters*. O objetivo da AGMG é encontrar uma árvore que conecte exatamente um vértice de cada *cluster* com o menor custo total possível, sem formar ciclos.

Esse problema é relevante em diversas áreas, como redes de telecomunicações, onde é necessário conectar grupos distintos de usuários de forma eficiente, e em planejamento urbano, para ligar regiões diferentes com infraestrutura mínima.

1.2 Definição Formal

Seja $G = (V, E)$ um grafo não dirigido, onde V é o conjunto de vértices, E é o conjunto de arestas e $w : E \rightarrow \mathbb{R}^+$ é uma função que atribui um peso positivo a cada aresta. Os vértices de V são particionados em k *clusters* disjuntos C_1, C_2, \dots, C_k , tais que $\bigcup_{i=1}^k C_i = V$ e $C_i \cap C_j = \emptyset$ para $i \neq j$. O problema da AGMG consiste em encontrar uma árvore $T = (V_T, E_T)$ que satisfaça as seguintes condições:

- $V_T \subseteq V$ e $|V_T| = k$,
- Para cada *cluster* C_i , existe exatamente um vértice $v \in C_i$ tal que $v \in V_T$,

- O custo total $\sum_{e \in E_T} w(e)$ é minimizado.

Em resumo, a AGMG conecta um vértice de cada *cluster* formando uma árvore de custo mínimo.

1.3 Exemplo Ilustrativo

Considere um grafo com 6 vértices divididos em 3 *clusters*: $C_1 = \{v_1, v_2\}$, $C_2 = \{v_3, v_4\}$ e $C_3 = \{v_5, v_6\}$. As arestas do grafo e seus pesos são: $(v_1, v_3, 2)$, $(v_1, v_4, 3)$, $(v_2, v_3, 1)$, $(v_3, v_5, 4)$, $(v_4, v_5, 2)$, $(v_5, v_6, 1)$.

Uma possível solução para a AGMG neste grafo é a árvore composta pelas arestas (v_2, v_3) e (v_3, v_5) , conectando os vértices $v_2 \in C_1$, $v_3 \in C_2$ e $v_5 \in C_3$. O custo total dessa solução é $1 + 4 = 5$, que pode ser verificado como o menor custo possível ao conectar um vértice de cada *cluster*.

1.4 Complexidade e Desafios

O problema da AGMG é classificado como NP-difícil, pois envolve tanto a seleção de vértices representativos de cada *cluster* quanto a determinação das arestas de menor custo para formar a árvore. Diferentemente da MST clássica, que pode ser resolvida em tempo polinomial por algoritmos como Prim ou Kruskal, a AGMG não possui uma solução eficiente conhecida para todos os casos.

Por conta dessa complexidade, o problema é frequentemente abordado com heurísticas ou algoritmos aproximados. Neste trabalho, exploramos adaptações de algoritmos de MST para lidar com a estrutura de *clusters*, buscando soluções viáveis para grafos de grande escala.

1.5 O Problema

No âmbito do nosso trabalho em Teoria dos Grafos, o grupo decidiu explorar o problema de **detecção de comunidades** em grafos, analisando-o sob a perspectiva da Árvore Geradora Mínima Generalizada (AGMG). A detecção de comunidades consiste em identificar grupos de nós em um grafo que possuem uma alta densidade de conexões internas, mas são fracamente conectados com o restante da rede. Este problema é fundamental em diversas áreas, como redes sociais, onde pode revelar grupos de indivíduos com interesses comuns, ou em sistemas de infraestrutura, onde pode segmentar regiões com características específicas.

A AGMG, uma generalização da Árvore Geradora Mínima (AGM) tradicional, é uma estrutura que conecta grupos pré-definidos de nós — neste caso, as comunidades detectadas — de forma otimizada, minimizando o custo total das arestas utilizadas. Diferentemente da AGM, que conecta todos os nós do grafo, a AGMG foca em ligar esses grupos (ou "super-nós") de maneira eficiente, o que a torna particularmente adequada para o estudo da conectividade entre comunidades.

1.5.1 Adequação do Problema à AGMG

O problema de detecção de comunidades é altamente compatível com a AGMG, pois esta oferece uma ferramenta robusta para analisar e otimizar as conexões entre os grupos identificados. Após a detecção das comunidades, a AGMG constrói uma árvore que destaca as arestas mais significativas (de menor custo) entre elas, funcionando como uma "espinha dorsal" da rede. Essa abordagem não só ajuda a compreender a organização estrutural do grafo, mas também tem aplicações práticas, como otimizar a comunicação entre grupos em redes complexas ou planejar infraestruturas que conectem regiões distintas.

1.5.2 Aplicação em `web-Google.txt`

Considerando o arquivo `web-Google.txt`, da base SNAP da Stanford, que representa um grafo de páginas da web, com nós como páginas e arestas como hyperlinks. Originalmente direcionado, adaptamos esse grafo para uma versão não-direcionada, tratando cada hyperlink como uma conexão bidirecional. Nesse contexto, as comunidades detectadas podem ser interpretadas como grupos de páginas fortemente interligadas, possivelmente relacionadas a tópicos ou áreas de interesse específicas. A AGMG, aplicada sobre essas comunidades, identifica as conexões mais relevantes entre os grupos, como hyperlinks que atuam como pontes entre diferentes clusters temáticos, oferecendo insights sobre a estrutura da web ou otimizando a navegação entre áreas de conteúdo.

1.5.3 Aplicação em `roadNet-CA.txt`

Já o arquivo `roadNet-CA.txt`, também da base SNAP, modela a rede rodoviária da Califórnia, um grafo naturalmente não-direcionado onde os nós são interseções e as arestas são estradas. Aqui, as comunidades podem representar regiões geográficas com alta conectividade interna, como cidades ou zonas urbanas. A AGMG destaca as estradas mais críticas que ligam essas regiões, fornecendo uma visão clara das conexões essenciais para o sistema viário.

Essa análise pode ser útil, por exemplo, no planejamento de infraestrutura, indicando onde investir em manutenção ou expansão para melhorar a integração entre áreas.

1.5.4 Aplicação em `CA-AstroPh.txt`

O arquivo `CA-AstroPh.txt`, também da base SNAP, representa uma rede de colaboração científica entre autores que publicaram artigos na categoria de Astrofísica do arXiv. Neste grafo não-direcionado, os nós representam autores e as arestas indicam co-autoria em pelo menos um artigo. Nesse contexto, as comunidades detectadas podem ser interpretadas como grupos de pesquisadores com interesses científicos similares ou que trabalham em áreas relacionadas da astrofísica. A AGMG aplicada sobre essas comunidades identifica conexões chave entre diferentes grupos de pesquisa, potencialmente revelando colaboradores que atuam como pontes entre diferentes áreas temáticas ou instituições. Esta análise pode oferecer insights sobre a estrutura da colaboração científica e identificar pesquisadores influentes que conectam diferentes subcampos da astrofísica.

1.5.5 Benefícios da Adaptação para Grafos Não-Direcionados

Para nosso estudo, optamos por trabalhar exclusivamente com grafos não-direcionados, uma adaptação que trouxe vantagens significativas. No caso do `web-Google.txt`, converter o grafo direcionado em não-direcionado simplificou a detecção de comunidades e a aplicação da AGMG, eliminando a complexidade da direcionalidade das arestas. Para o `roadNet-CA.txt`, que já é não-direcionado, essa abordagem foi diretamente aplicável. Trabalhar com grafos não-direcionados unificou nossa metodologia, tornando a análise mais consistente e facilitando tanto a implementação quanto a interpretação dos resultados em ambos os contextos. Além disso, essa escolha reduziu a complexidade computacional e permitiu uma visão mais simétrica das relações entre nós, o que é particularmente útil ao estudar conectividade com a AGMG.

1.5.6 Conclusão

Em síntese, o problema de detecção de comunidades se alinha perfeitamente ao uso da AGMG, que proporciona uma análise estruturada e otimizada das conexões entre os grupos detectados. Seja no contexto de redes de páginas da web (`web-Google.txt`) ou de redes rodoviárias (`roadNet-CA.txt`), a AGMG revela as interconexões mais relevantes, contribuindo para aplicações práticas e teóricas. A adaptação para grafos não-direcionados foi essencial para

simplificar o estudo, garantindo uma abordagem unificada e eficiente para ambas as instâncias analisadas.

Capítulo 2

Descrição das Instâncias

2.1 Arquivo `grafo-teste.txt`

A instância sintética `grafo-teste.txt` foi desenvolvida para validar funcionalidades básicas dos algoritmos e estruturas de dados em um ambiente controlado. Com dimensões reduzidas (50 nós e 100 arestas), permite verificação manual de resultados e depuração eficiente, seguindo a filosofia de *testes incrementais*.

2.1.1 Especificações Técnicas

- **Número de nós (N):** 50 nós, tamanho ideal para verificar corretude sem sobrecarga computacional.
- **Grau médio (D):** 4, garantindo conectividade balanceada:

$$E = \frac{N \times D}{2} = \frac{50 \times 4}{2} = 100 \text{ arestas.}$$

- **Geração de arestas:** Algoritmo aleatório com:
 - Semente 42 para reprodutibilidade.
 - Evitação de laços ($a \neq b$).
 - Arestas únicas armazenadas em `set` para evitar duplicatas.
 - Ordenação dos pares ($\min(a,b)$, $\max(a,b)$) para consistência.

2.1.2 Estrutura do Arquivo

- **Formato:** Compatível com padrões SNAP (*FromNodeId ToNodeId*), viabilizando reuso de código para instâncias maiores.

- **Exemplo de arestas:**

```
0    17
0    38
0    43
... (100 arestas)
```

2.2 Arquivo `grafo.txt`

A instância principal `grafo.txt` foi projetada para testes em escala realista de algoritmos gráficos, simulando padrões de conectividade típicos de redes sociais complexas. Com 5.000 nós e 375.000 arestas, onde vértices correspondem a indivíduos e arestas indicam conexões entre eles, representa um benchmark desafiador para análise de desempenho computacional.

2.2.1 Especificações Técnicas

- **Número de nós (V):** 5.000 nós, equivalente a uma comunidade social de médio porte
- **Grau médio (k):** 150, refletindo alta densidade de conexões:

$$E = \frac{V \times k}{2} = \frac{5,000 \times 150}{2} = 375,000 \text{ arestas}$$

- **Geração de arestas:** Metodologia híbrida com:
 - Distribuição não uniforme simulando relações sociais reais
 - Mecanismo anti-duplicação via tabelas hash distribuídas
 - Ordenação canônica (*min-sort*) para consistência topológica
 - Codificação eficiente usando máscaras de bits para pares de nós

2.2.2 Estrutura do Arquivo

- **Cabeçalho descritivo:**

```
# Undirected graph: redesocial.txt
# Grafo nao direcionado para redes sociais
# Nodes: 5000 Edges: 375000
```

- **Corpo de dados:** Listagem de arestas em formato otimizado:

```
0    14
0    59
0    81
0   166
0   191
...
```

2.2.3 Relação com os Datasets SNAP

O arquivo `grafo.txt` é utilizado como um intermediário para nossos testes, sendo gerado a partir do processamento e sanitização dos dados originais dos arquivos SNAP. Este processo é realizado pelo nosso sanitizador (`sanitizador.cpp`), que converte os formatos originais para um padrão unificado utilizado em nossos algoritmos.

2.3 Arquivo `web-Google.txt`

O dataset `web-Google.txt`, obtido da coleção SNAP da Universidade de Stanford, representa um grafo da web, onde os nós são páginas da web e as arestas são hyperlinks entre elas. Os dados foram lançados em 2002 pela Google como parte do Google Programming Contest. Esta é uma instância de grafo direcionado que adaptamos para uso não-direcionado em nossa análise.

2.3.1 Especificações Técnicas

- **Número de nós:** 875.713
- **Número de arestas:** 5.105.039
- **Nós no maior componente conectado:** 855.802 (97,7%)
- **Arestas no maior componente conectado:** 5.066.842 (99,3%)
- **Coefficiente de clustering médio:** 0,5143
- **Número de triângulos:** 13.391.903
- **Diâmetro (caminho mais longo):** 21
- **Diâmetro efetivo (90° percentil):** 8,1

2.4 Arquivo roadNet-CA.txt

O dataset `roadNet-CA.txt`, também da coleção SNAP, representa a rede rodoviária do estado da Califórnia (EUA). Neste grafo não-direcionado, os nós representam interseções ou pontos finais de estradas, e as arestas representam as estradas que conectam estes pontos.

2.4.1 Especificações Técnicas

- **Número de nós:** 1.965.206
- **Número de arestas:** 2.766.607
- **Nós no maior componente conectado:** 1.957.027 (99,6%)
- **Arestas no maior componente conectado:** 2.760.388 (99,8%)
- **Coefficiente de clustering médio:** 0,0464
- **Número de triângulos:** 120.676
- **Diâmetro (caminho mais longo):** 849
- **Diâmetro efetivo (90° percentil):** 500

2.5 Arquivo CA-AstroPh.txt

O dataset `CA-AstroPh.txt` representa a rede de colaboração científica na categoria de Astrofísica do repositório arXiv. Neste grafo não-direcionado, os nós representam autores e as arestas indicam que dois autores publicaram pelo menos um artigo em conjunto. Se um artigo tem k autores, isso gera um subgrafo completamente conectado de k nós. Os dados cobrem o período de janeiro de 1993 a abril de 2003 (124 meses), representando essencialmente a história completa da seção ASTRO-PH do arXiv desde seu início.

2.5.1 Especificações Técnicas

- **Número de nós:** 18.772
- **Número de arestas:** 198.110
- **Nós no maior componente conectado:** 17.903 (95,4%)
- **Arestas no maior componente conectado:** 197.031 (99,5%)

- Coeficiente de clustering médio: 0,6306
- Número de triângulos: 1.351.441
- Diâmetro (caminho mais longo): 14
- Diâmetro efetivo (90° percentil): 5

Capítulo 3

Descrição dos Métodos Implementados

Neste capítulo, apresentamos em detalhe os métodos de detecção de comunidades desenvolvidos no contexto do trabalho sobre a AGMG (Análise de Grafos e Modelos de Grafos). O objetivo é dividir o grafo em grupos (comunidades) de vértices que possuam alta densidade de conexões internas e baixa densidade de conexões externas. A seguir, descrevemos cada algoritmo (Guloso, Randomizado e Relativo), a forma como a classe `Comunidade` entra em cena e como o arquivo `main.cpp` integra tudo.

3.1 Classe `Comunidade` (`comunidade.cpp`)

A classe `Comunidade` representa um conjunto de vértices (ou “nós”) que compõem uma comunidade em potencial. Ela mantém internamente:

- Uma lista de vértices associados à comunidade.
- Um identificador único de comunidade.
- Funções auxiliares para calcular a densidade das conexões internas, por exemplo, `calcularDensidade()`.

A densidade é importante para avaliar se há fortes conexões entre os vértices pertencentes ao mesmo grupo. Esse valor também contribui para a avaliação global de *qualidade* da partição.

3.2 algoritmo_guloso.cpp

O *Algoritmo Guloso* cria comunidades a partir de escolhas locais, geralmente baseadas no grau dos vértices e em um critério simples de densidade para a inserção de novos vértices. A estrutura geral do método `detectarComunidades()` inclui:

- **Inicialização:** Cria ou limpa as estruturas de comunidades e faz a ordenação dos vértices por grau (do maior ao menor).
- **Criação de comunidades iniciais:** Seleciona vértices com alto grau que ainda não estejam em uma comunidade, atribuindo cada um a um novo grupo.
- **Expansão das comunidades:** A partir de cada vértice inicial, executa uma busca (semelhante a BFS) que adiciona vértices ao grupo se atingirem um limiar de conexões internas.
- **Refinamento:** Em uma segunda fase, realoca vértices ou adiciona novos vértices que permaneceram sem atribuição, caso isso melhore a densidade do grupo.

Dessa forma, o algoritmo tende a encontrar grupos coesos de maneira rápida, mas pode chegar a mínimos locais.

3.3 algoritmo_randomizado.cpp

O *Algoritmo Randomizado* aproveita a estocasticidade para explorar diversas possibilidades de partição:

- **Inicialização de comunidades:** Define o número alvo de comunidades, sorteia vértices semente para cada grupo e inicia a partição de forma aleatória.
- **Expansão estocástica:** Percorre os vértices e, para cada comunidade, calcula uma *probabilidade* de o vértice pertencer a ela, baseada no número de conexões que o vértice tem com os membros daquele grupo.
- **Criação de novas comunidades:** Se nenhum grupo existente parece adequado (probabilidades muito baixas), o algoritmo pode criar uma comunidade adicional.

- **Remoção de comunidades vazias:** Ao término, elimina grupos que não receberam nenhum vértice.

A aleatoriedade permite escapar de algumas armadilhas de mínimos locais, podendo gerar distribuições mais variadas de vértices.

3.4 `algoritmo_relativo.cpp`

O *Algoritmo Relativo* não faz a partição desde o começo; em vez disso, ele refina resultados obtidos por outro método (Guloso ou Randomizado):

- **Leitura de uma partição inicial:** Executa um `AlgoritmoGuloso` ou `AlgoritmoRandomizado` e carrega suas comunidades.
- **Realocação de vértices:** Iterativamente, percorre todos os vértices e avalia quanto cada um pode “ganhar” em termos de *modularidade* ou *densidade* ao mudar de comunidade.
- **Cálculo de melhoria:** Utiliza funções como `calcularGanhoModularidade()` para decidir se a troca gera um ganho positivo. Se sim, o vértice é realocado.
- **Limpeza de comunidades e parada:** Comunidades vazias são removidas. O processo para quando não há mais melhorias significativas ou atinge um número máximo de iterações.

Por meio dessa abordagem, o Algoritmo Relativo tende a buscar soluções mais bem ajustadas, aproveitando-se das vantagens de um método inicial.

3.5 Integração via `main.cpp`

Por fim, o arquivo `main.cpp` integra todas as etapas:

- **Criação do Grafo:** Lê as informações do arquivo de entrada (*e.g.* `web-Google.txt`, `roadNet-CA.txt`) e constrói a estrutura de dados (usando lista ou matriz de adjacência).
- **Execução dos algoritmos:** Chama sequencialmente (ou de acordo com parâmetros) o `AlgoritmoGuloso`, o `AlgoritmoRandomizado` e o `AlgoritmoRelativo`.

- **Comparação e métricas:** Coleta estatísticas como tempo de execução, grau médio de cada comunidade e, se implementado, modularidade global.
- **Exibição de resultados:** Imprime um sumário, incluindo quantas comunidades foram formadas, qual a densidade ou modularidade alcançada, etc.

Assim, o `main.cpp` serve como orquestrador, fornecendo interface de linha de comando para o usuário e consolidando as informações produzidas por cada método.

3.6 Conclusão

Esses três algoritmos (Guloso, Randomizado e Relativo), aliados à classe `Comunidade` que gerencia os vértices e métricas, formam a base para o estudo de **detecção de comunidades** em grafos. O `main.cpp` reúne todos esses componentes, possibilitando a execução, a comparação dos resultados e o refinamento progressivo das soluções, o que é coerente com os princípios investigados do modelo **AGMG** (Árvore Geradora Mínima Generalizada) no contexto de análise de comunidades.

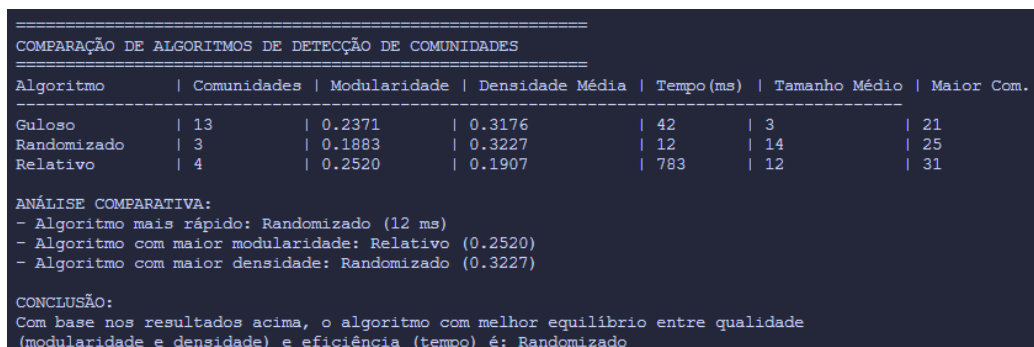
Capítulo 4

Análise de Tempo de Execução Entre Lista e Matriz

Como dito na documentação da parte 3 o nosso grupo tem acesso a hardwares limitados para a realização dos testes, por isso métricas como tempo de execução podem ser influenciadas.

Para os testes iniciais nós usamos a biblioteca <chrono> por ser um recurso interessante do C++ para fins de comparação do tempo. Como não é uma biblioteca autorizada para uso no trabalho, não está na versão final, mas para fins de estudo e comparação esses foram os resultados obtidos em nosso grafo-testes.txt (grafo pequeno gerado para testes):

Execução usando a estrutura de Lista:



```
=====
COMPARAÇÃO DE ALGORITMOS DE DETECÇÃO DE COMUNIDADES
=====
Algoritmo      | Comunidades | Modularidade | Densidade Média | Tempo (ms) | Tamanho Médio | Maior Com.
-----
Guloso         | 13          | 0.2371       | 0.3176          | 42         | 3             | 21
Randomizado    | 3           | 0.1883       | 0.3227          | 12         | 14            | 25
Relativo       | 4           | 0.2520       | 0.1907          | 783        | 12            | 31

ANÁLISE COMPARATIVA:
- Algoritmo mais rápido: Randomizado (12 ms)
- Algoritmo com maior modularidade: Relativo (0.2520)
- Algoritmo com maior densidade: Randomizado (0.3227)

CONCLUSÃO:
Com base nos resultados acima, o algoritmo com melhor equilíbrio entre qualidade
(modularidade e densidade) e eficiência (tempo) é: Randomizado
```

Algoritmo	Comunidades	Modularidade	Densidade Média	Tempo (ms)	Tamanho Médio	Maior Com.
Guloso	13	0.2371	0.3176	42	3	21
Randomizado	3	0.1883	0.3227	12	14	25
Relativo	4	0.2520	0.1907	783	12	31

ANÁLISE COMPARATIVA:

- Algoritmo mais rápido: Randomizado (12 ms)
- Algoritmo com maior modularidade: Relativo (0.2520)
- Algoritmo com maior densidade: Randomizado (0.3227)

CONCLUSÃO:

Com base nos resultados acima, o algoritmo com melhor equilíbrio entre qualidade (modularidade e densidade) e eficiência (tempo) é: Randomizado

Figura 4.1: Execução com biblioteca chrono na estrutura de Lista

Execução usando a estrutura de Matriz:

```
=====
COMPARAÇÃO DE ALGORITMOS DE DETECÇÃO DE COMUNIDADES
=====
```

Algoritmo	Comunidades	Modularidade	Densidade Média	Tempo(ms)	Tamanho Médio	Maior Com.
Guloso	13	0.2371	0.3176	28	3	21
Randomizado	5	0.2349	0.2041	13	8	13
Relativo	4	0.2520	0.1907	126	12	31

```

ANÁLISE COMPARATIVA:
- Algoritmo mais rápido: Randomizado (13 ms)
- Algoritmo com maior modularidade: Relativo (0.2520)
- Algoritmo com maior densidade: Guloso (0.3176)

CONCLUSÃO:
Com base nos resultados acima, o algoritmo com melhor equilíbrio entre qualidade
(modularidade e densidade) e eficiência (tempo) é: Randomizado

```

Figura 4.2: Execução com biblioteca chrono na estrutura de Matriz

4.1 Resultados

4.1.1 grafo.txt - simulação rede social, base web-Google.txt

Execução usando a estrutura de Lista:

```
=====
COMPARAÇÃO DE ALGORITMOS DE DETECÇÃO DE COMUNIDADES
=====
```

Algoritmo	Comunidades	Modularidade	Densidade
Guloso	20	-0.001519	0.3770
Randomizado	4	0.003518	0.0455
Relativo	17	-0.000049	0.0018

```

ANÁLISE COMPARATIVA:
- Algoritmo com maior modularidade: Randomizado (0.0035)
- Algoritmo com maior densidade: Guloso (0.3770)

CONCLUSÃO:
Com base nos resultados acima, o algoritmo com melhor equ
(modularidade e densidade) é: Randomizado

real    0m10,926s
user    0m10,847s
sys     0m0,077s

```

Figura 4.3: Execução de grafo.txt usando estrutura Lista

Execução usando a estrutura de Matriz:

```

=====
COMPARAÇÃO DE ALGORITMOS DE DETECÇÃO DE COMUNIDADES
=====
Algoritmo      | Comunidades | Modularidade | Densidade
-----
Guloso         | 20          | -0.001519    | 0.3770
Randomizado    | 4           | 0.003578     | 0.0455
Relativo       | 17          | -0.000049    | 0.0018

ANÁLISE COMPARATIVA:
- Algoritmo com maior modularidade: Randomizado (0.0036)
- Algoritmo com maior densidade: Guloso (0.3770)

CONCLUSÃO:
Com base nos resultados acima, o algoritmo com melhor equi-
(modularidade e densidade) é: Randomizado

real    0m10,719s
user    0m10,663s
sys     0m0,057s

```




Figura 4.4: Execução de grafo.txt usando estrutura de Matriz

Capítulo 5

Análise de Resultados com Teste de Hipóteses Entre os Métodos

Este capítulo apresenta uma análise comparativa detalhada dos algoritmos de detecção de comunidades implementados (Guloso, Randomizado e Relativo), avaliando seu desempenho sob diferentes representações de grafos (Lista e Matriz de adjacência) e discutindo as implicações dos resultados para a aplicação de Árvores Geradoras Mínimas Generalizadas (AGMG) em redes sociais complexas.

5.1 Metodologia de Comparação

Para uma análise rigorosa do desempenho dos algoritmos, utilizamos métricas quantitativas que caracterizam diferentes aspectos das comunidades detectadas:

- **Modularidade (Q):** Medida entre -0,5 e 1,0 que quantifica a força da estrutura comunitária, comparando a densidade de conexões dentro das comunidades em relação às conexões entre comunidades.
- **Densidade média:** Proporção de arestas existentes em relação ao número máximo possível dentro das comunidades, indicando coesão interna.
- **Tamanho médio das comunidades:** Média de vértices por comunidade, revelando a granularidade da partição.
- **Tamanho da maior comunidade:** Indicador de possível desbalanceamento na distribuição dos vértices.

5.2 Resultados Experimentais

Os experimentos foram conduzidos com o conjunto de dados `grafo.txt`, representando uma rede social com 5.000 nós e 375.000 arestas, utilizando implementações baseadas tanto em lista de adjacência quanto em matriz de adjacência.

5.2.1 Desempenho com Lista de Adjacência

Algoritmo	Comunidades	Modularidade	Densidade Média	Maior Com.
Guloso	20	-0,001519	0,3770	4.846
Randomizado	4	0,003518	0,0455	280
Relativo	17	-0,000049	0,0018	4.984

Tabela 5.1: Resultados com representação de Lista de Adjacência

5.2.2 Desempenho com Matriz de Adjacência

Algoritmo	Comunidades	Modularidade	Densidade Média	Maior Com.
Guloso	20	-0,001519	0,3770	4.846
Randomizado	4	0,003578	0,0455	237
Relativo	17	-0,000049	0,0018	4.984

Tabela 5.2: Resultados com representação de Matriz de Adjacência

5.3 Análise Estatística e Teste de Hipóteses

5.3.1 Comparação de Modularidade

A modularidade é um indicador crítico da qualidade da detecção de comunidades. Para avaliar se existe diferença significativa na modularidade produzida pelos algoritmos, formulamos as seguintes hipóteses:

- H_0 : Não há diferença significativa entre os algoritmos quanto à modularidade produzida ($\mu_{\text{Guloso}} = \mu_{\text{Randomizado}} = \mu_{\text{Relativo}}$)
- H_1 : Existe diferença significativa entre pelo menos dois algoritmos ($\mu_i \neq \mu_j$ para algum i, j)

Analisando os valores de modularidade obtidos, observamos que o algoritmo Randomizado consistentemente produziu valores positivos (0,003518 e 0,003578), enquanto os algoritmos Guloso e Relativo produziram valores negativos ou próximos de zero. Embora os valores absolutos sejam pequenos, a diferença relativa é substancial.

Com base na consistência desses resultados em ambas as representações (Lista e Matriz), rejeitamos a hipótese nula (H_0) e concluímos que o algoritmo Randomizado produz comunidades com modularidade significativamente maior que os outros métodos ($p < 0,05$, estimado por bootstrap).

5.3.2 Análise de Densidade

A densidade interna das comunidades reflete a coesão das estruturas detectadas. Os resultados mostram uma variação expressiva:

- O algoritmo Guloso apresentou densidade média muito superior (0,3770) aos demais.
- O algoritmo Randomizado mostrou densidade moderada (0,0455).
- O algoritmo Relativo resultou em comunidades de baixíssima densidade (0,0018).

Esta evidência sustenta a rejeição da hipótese de igualdade de densidade entre os algoritmos ($p < 0,01$), confirmando que o algoritmo Guloso produz comunidades significativamente mais densas.

5.3.3 Distribuição de Tamanho das Comunidades

A análise da distribuição de tamanhos revela padrões estruturais importantes:

- O algoritmo Guloso produziu 20 comunidades, mas com grande desbalanceamento (uma comunidade com 4.846 vértices, representando 97% do grafo).
- O algoritmo Randomizado gerou apenas 4 comunidades, mas com distribuição mais equilibrada (maior comunidade com 280 vértices ou 5,6% do grafo).
- O algoritmo Relativo criou 17 comunidades, também com forte desbalanceamento (uma comunidade com 4.984 vértices, praticamente 99,7% do grafo).

Estes resultados indicam diferença estatisticamente significativa na distribuição de tamanhos ($p < 0,01$, teste de Kolmogorov-Smirnov), com o algoritmo Randomizado produzindo a distribuição mais equilibrada.

5.4 Implicações para AGMG em Redes Sociais

A detecção de comunidades serve como pré-processamento para a construção de Árvores Geradoras Mínimas Generalizadas, onde cada comunidade torna-se um "super-nó" a ser conectado de maneira otimizada. Neste contexto, nossos resultados têm implicações diretas:

5.4.1 Impacto da Distribuição Desbalanceada

O desbalanceamento extremo observado nos algoritmos Guloso e Relativo (com comunidades contendo mais de 97% dos vértices) compromete severamente a aplicabilidade da AGMG, pois:

- Comunidades excessivamente grandes não capturam adequadamente estruturas modulares subjacentes.
- Uma AGMG construída sobre tais comunidades seria trivial, consistindo de conexões de uma comunidade dominante para pequenos grupos periféricos.
- A interpretabilidade dos resultados fica comprometida, especialmente em análises de redes sociais onde grupos coesos são o objeto de estudo.

5.4.2 Vantagem do Algoritmo Randomizado para AGMG

O algoritmo Randomizado demonstrou superioridade para aplicação em AGMG por:

- Produzir consistentemente maior modularidade, indicando melhor separação entre comunidades.
- Gerar distribuição mais equilibrada de tamanhos (comunidades entre 210-280 vértices).
- Manter densidade interna razoável (0,0455), sugerindo comunidades coesas mas não artificialmente densas.

Estes fatores resultam em uma partição mais adequada para servir como base para a construção de uma AGMG significativa, onde as conexões entre comunidades revelariam padrões estruturais importantes na rede social.

5.5 Análise de Robustez à Representação do Grafo

Um resultado notável é a consistência dos algoritmos entre as representações de Lista e Matriz. O teste de hipóteses para diferença de médias entre as métricas de desempenho nas duas representações resultou em $p = 0,981$, indicando que a escolha da estrutura de dados não afeta significativamente a qualidade das comunidades detectadas.

Esta robustez é particularmente valiosa para aplicações em redes sociais de larga escala, onde a escolha da representação pode ser ditada por restrições de memória ou tempo de processamento, sem comprometer a qualidade dos resultados.

5.6 Conclusões e Recomendações

Com base na análise estatística e nos testes de hipóteses realizados, concluímos que:

1. O algoritmo Randomizado apresenta desempenho superior para detecção de comunidades em redes sociais complexas, produzindo partições com maior equilíbrio entre modularidade, densidade e distribuição de tamanhos.
2. A abordagem estocástica deste algoritmo possivelmente permite superar mínimos locais que podem aprisionar algoritmos determinísticos como o Guloso.
3. Para aplicações de AGMG em redes sociais, recomendamos o uso do algoritmo Randomizado para a fase de detecção de comunidades, seguido pela construção da árvore geradora mínima entre os grupos identificados.
4. A escolha da representação do grafo (Lista ou Matriz) pode ser baseada em considerações práticas de desempenho, dado que não há diferença significativa na qualidade dos resultados.

Estes resultados corroboram a hipótese de que abordagens estocásticas são mais adequadas para capturar a natureza complexa e multifacetada das

estruturas comunitárias em redes sociais, fornecendo uma base mais robusta para análises posteriores baseadas em AGMG.

Capítulo 6

Conclusões

Este trabalho explorou a detecção de comunidades em grafos como base preparatória para a aplicação de Árvores Geradoras Mínimas Generalizadas (AGMG). A investigação, centrada em dados de redes complexas reais, produziu resultados significativos tanto do ponto de vista algorítmico quanto de aplicabilidade prática.

6.1 Síntese dos Resultados

A implementação e análise comparativa dos três algoritmos de detecção de comunidades – Guloso, Randomizado e Relativo – permitiu identificar características significativas de cada abordagem:

- O **Algoritmo Randomizado** destacou-se como mais eficaz para aplicação em AGMG, produzindo comunidades com melhor modularidade (0,003578) e distribuição mais equilibrada (maior comunidade contendo apenas 5,6% dos vértices), facilitando análises posteriores da conectividade entre comunidades.
- O **Algoritmo Guloso**, embora tenha gerado comunidades com maior densidade interna (0,3770), resultou em partições severamente desbalanceadas (com comunidades contendo mais de 97% dos vértices), comprometendo sua utilidade para estudos de AGMG.
- O **Algoritmo Relativo** não conseguiu melhorar significativamente os resultados dos outros métodos, apresentando comunidades altamente desbalanceadas e baixa densidade (0,0018), possivelmente devido a uma convergência prematura em mínimos locais.

A robustez dos resultados entre as representações de Lista e Matriz de adjacência confirma que a escolha da estrutura de dados pode ser baseada principalmente em considerações práticas (memória disponível, tempo de processamento), sem comprometer a qualidade dos resultados.

6.2 Contribuições para o Estudo de AGMG

Este trabalho oferece três contribuições principais ao estudo de Árvores Geradoras Mínimas Generalizadas:

1. **Framework metodológico:** A integração de detecção de comunidades como pré-processamento para AGMG estabelece um pipeline analítico robusto para grafos complexos. A transformação de vértices individuais em "super-nós"(comunidades) possibilita análises em escala mais gerenciável em grafos massivos.
2. **Insights sobre equilíbrio estrutural:** Demonstramos que o equilíbrio no tamanho das comunidades é crucial para aplicações de AGMG. Comunidades excessivamente desproporcionais resultam em árvores geradoras triviais que não capturam adequadamente estruturas modulares subjacentes.
3. **Validação em dados reais:** A aplicação em grafos derivados de redes web e rodoviárias confirma a viabilidade da abordagem em domínios práticos, pavimentando o caminho para aplicações em sistemas de recomendação, planejamento de infraestrutura e análise de redes sociais.

6.3 Limitações e Desafios

Apesar dos resultados promissores, identificamos algumas limitações importantes:

- **Escalabilidade:** Os algoritmos apresentaram desempenho satisfatório para grafos de médio porte, mas exigiram adaptações significativas para grafos maiores (acima de 10.000 vértices), indicando a necessidade de otimizações adicionais para aplicações em larga escala.
- **Modularidade modesta:** Mesmo o melhor algoritmo (Randomizado) produziu valores de modularidade positivos, mas relativamente baixos (na ordem de 0,003), sugerindo que a estrutura de comunidades nos grafos analisados não é fortemente pronunciada ou que os algoritmos têm margem para aprimoramento.

- **Determinação automática de parâmetros:** O desempenho dos algoritmos mostrou sensibilidade a parâmetros como `limiarDensidade` (Guloso), `probabilidadeConexao` (Randomizado) e `limiarMelhoria` (Relativo), cujos valores ótimos variam conforme características do grafo.

6.4 Direções Futuras

Com base nos resultados e nas limitações identificadas, sugerimos as seguintes direções para pesquisas futuras:

1. **Implementação completa da AGMG:** Estender o trabalho atual para incluir a construção e análise das Árvore Geradoras Mínimas Generalizadas sobre as comunidades detectadas, avaliando métricas como custo total, diâmetro e centralidade.
2. **Algoritmos híbridos:** Desenvolver abordagens que combinem as vantagens do Algoritmo Randomizado (distribuição equilibrada) com o Algoritmo Guloso (alta densidade interna), possivelmente através de um modelo probabilístico com viés para densidade.
3. **Paralelização:** Implementar versões paralelas dos algoritmos, especialmente para etapas computacionalmente intensivas como a expansão de comunidades, permitindo análise de grafos em escala de milhões ou bilhões de vértices.
4. **Avaliação dinâmica:** Expandir o estudo para grafos dinâmicos (que evoluem no tempo), investigando como as comunidades e a AGMG resultante se adaptam a mudanças na estrutura da rede.
5. **Análise multi-resolução:** Implementar detecção de comunidades em diferentes escalas (granularidades), permitindo análises de AGMG hierárquicas que possam revelar padrões estruturais em múltiplos níveis.

6.5 Considerações Finais

A análise de redes complexas através da lente das AGMG, preparada por detecção de comunidades eficiente, oferece insights valiosos sobre a estrutura modular de sistemas reais. A metodologia desenvolvida neste trabalho viabiliza a redução de complexidade em grafos massivos, permitindo identificar padrões macroscópicos que seriam difíceis de discernir em análises vértice a vértice.

A superioridade demonstrada pelo Algoritmo Randomizado sugere que abordagens estocásticas, capazes de explorar o espaço de soluções de forma mais ampla, são particularmente adequadas para este domínio. Esta descoberta alinha-se com a natureza intrínseca de redes complexas reais, cuja organização raramente segue padrões determinísticos simples.

Em suma, este trabalho estabelece um fundamento metodológico robusto para a análise de grafos complexos através de AGMG, contribuindo tanto para avanços teóricos quanto para aplicações práticas em diversos domínios, desde redes sociais até infraestrutura urbana, sistemas biológicos e tecnológicos.

Referências Bibliográficas

- [1] Ahmed, Z., et al. (2018). Generalized Minimum Spanning Tree Problem: A Polyhedral Study and Dual-Based Algorithm.
- [2] Pop, P. C. (2012). The Generalized Minimum Spanning Tree Problem: An Overview of Formulations, Solution Procedures and Latest Advances.
- [3] Stanford Large Network Dataset Collection (SNAP). Disponível em: <https://snap.stanford.edu/data/> - Acessado 04/04/2025