UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E BIOLÓGICAS
DEPARTAMENTO DE COMPUTAÇÃO

VINÍCIUS DA SILVA GOMES

UM COMPARATIVO ENTRE ALGORITMOS DE AGRUPAMENTO DE DADOS PARA DETECÇÃO DE COMUNIDADES

Ouro Preto, MG
2025

VINÍCIUS DA SILVA GOMES

UM COMPARATIVO ENTRE ALGORITMOS DE AGRUPAMENTO DE DADOS PARA DETECÇÃO DE COMUNIDADES

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal de Ouro Preto como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Jadson Castro Gertrudes

Ouro Preto, MG
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO

FOLHA DE APROVAÇÃO

Vinícius da Silva Gomes

Um comparativo entre algoritmos de agrupamento de dados para detecção de comunidades

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 02 de Setembro de 2025.

Membros da banca

Jadson Castro Gertrudes (Orientador) - Doutor - Universidade Federal de Ouro Preto
Vander Luis de Souza Freitas (Examinador) - Doutor - Universidade Federal de Ouro Preto
Ícaro Luiz Lage Vasconcelos (Examinador) - Bacharel - Programa de Pós-Graduação em Ciência da Computação - UFOP

Jadson Castro Gertrudes, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 02/09/2025.

Documento assinado eletronicamente por **Jadson Castro Gertrudes**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 31/08/2025, às 20:05, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0965553** e o código CRC **3CFD3A52**.

Agradecimentos

Agradeço primeiramente aos meus pais, pois sem eles eu não estaria aqui. Agradeço também ao meu irmão, que em momentos de dificuldade sempre esteve ao meu lado para me dar suporte. Por fim, agradeço ao meu orientador, Jadson, pelos ensinamentos transmitidos.

Resumo

O agrupamento de dados é uma tarefa fundamental do aprendizado de máquina não supervisionado, cujo objetivo é organizar um conjunto de dados em grupos com base na similaridade entre os elementos. No contexto de análise de redes complexas, essa técnica é aplicada para a detecção de comunidades, que consiste em identificar subgrupos de nós densamente conectados em um grafo. Este trabalho se dedica a comparar o desempenho de algoritmos de agrupamento na detecção de comunidades, com ênfase na avaliação de uma abordagem recente: o framework FOSC (Framework for Optimal Extraction of Clusters from Hierarchies), proposto por (CAMPELLO et al., 2013) e aprimorado por Anjos et al. (2019). Neste estudo, realizamos uma comparação do aprimoramento proposto por Anjos et al. (2019), que utiliza a métrica de modularidade Q no FOSC, contra algoritmos de detecção de comunidade consolidados. Utilizando o benchmark sintético LFR, o trabalho observou se a incorporação do FOSC a esses algoritmos tradicionais melhora a qualidade das comunidades detectadas. Os experimentos mostraram que o FOSC não apresentou melhorias significativas em relação aos algoritmos clássicos, embora também não tenha demonstrado desempenho inferior. Em diferentes configurações, o FOSC mostrou-se competitivo, alcançando valores de NMI próximos aos melhores métodos para baixos valores de μ. Conclui-se que, apesar de não superar os algoritmos tradicionais no benchmark LFR, o FOSC apresenta potencial e merece ser explorado em outros cenários e bases de dados, a fim de avaliar de forma mais ampla sua eficácia na detecção de comunidades.

Palavras-chave: Aprendizado de máquina. Aprendizado não supervisionado. Agrupamento de dados. Detecção de Comunidades.

Abstract

Data clustering is a fundamental task in unsupervised machine learning, whose goal is to organize a dataset into groups based on the similarity among its elements. In the context of complex network analysis, this technique is applied to community detection, which aims to identify subgroups of densely connected nodes within a graph. This work focuses on comparing the performance of clustering algorithms in community detection, with an emphasis on evaluating a recent approach: the FOSC framework (Framework for Optimal Extraction of Clusters from Hierarchies), proposed by (CAMPELLO et al., 2013) and later improved by Anjos et al. (2019). In this study, we compare the enhancement proposed by Anjos et al. (2019), which incorporates modularity Q into FOSC, against well-established community detection algorithms. Using the synthetic LFR benchmark, we investigated whether the integration of FOSC into traditional algorithms improves the quality of the detected communities. The experiments showed that FOSC did not provide significant improvements over classical algorithms, although it did not perform worse either. In different configurations, FOSC proved competitive, achieving NMI values close to the best-performing methods for low values of μ. We conclude that, although FOSC did not clearly outperform traditional algorithms in the LFR benchmark, it shows potential and should be further explored in other scenarios and datasets to better assess its effectiveness in community detection.

Keywords: Machine learning. Unsupervised learning. Data clustering. Community detection.

Lista de Ilustrações

Lista de Tabelas

Sumário

1 Introdução

1.1 Introdução e Justificativa

A teoria dos grafos é uma área fundamental na Ciência da Computação, com aplicações que abrangem desde a teoria da computação até estruturas de dados e representações de árvores binárias. Embora seja essencial para a modelagem de problemas computacionais, a teoria dos grafos é uma disciplina muito mais antiga que a própria Computação. Seus primeiros registros remontam ao século XVIII, com o trabalho de Euler, que resolveu o problema das pontes de Königsberg ao introduzir o conceito de caminho euleriano (EULER, 1741). Desde então, a teoria dos grafos tem se expandido e sido aplicada em diversas áreas do conhecimento, como ciências sociais (HARARY; NORMAN, 1953), biologia (PAVLOPOULOS et al., 2011), química (BALABAN, 1985), engenharia elétrica (DEO, 2016), entre outras.

Um dos ramos de estudo mais relevantes dentro da teoria dos grafos é a detecção de comunidades, que visa identificar grupos de nós altamente conectados entre si em comparação com outros nós da rede (BARABÁSI; PÓSFAI, 2016). Essa técnica tem sido amplamente aplicada em diversos campos, como criminologia, saúde pública e política, destacando sua importância e versatilidade (KARATAŞ; ŞAHIN, 2018). Por exemplo, na criminologia, a detecção de comunidades é utilizada para identificar grupos envolvidos em atividades criminosas, como fraudes e terrorismo (SARVARI et al., 2014). Na saúde pública, auxilia na identificação de pacientes com alto risco de desenvolver câncer de pulmão (BECHTEL et al., 2005), enquanto na política, é empregada para analisar a influência de ideologias e políticos dentro de grupos específicos (BIENKOV, 2012).

A detecção de comunidades está intimamente relacionada ao problema de agrupamento de dados, uma técnica de aprendizado não supervisionado que busca organizar dados em grupos com base em suas similaridades (JAIN; DUBES, 1988). Essa abordagem tem ganhado destaque em áreas como biologia, medicina, marketing e engenharia, sendo fundamental para a análise de redes complexas. Um exemplo notável ocorreu na biologia, onde a análise de redes metabólicas permitiu identificar módulos funcionais responsáveis por processos celulares específicos (BARABÁSI; PÓSFAI, 2016). O trabalho pioneiro de Ravasz et al. (2002) aplicou algoritmos de detecção de comunidades para identificar esses módulos em redes metabólicas, revelando padrões funcionais importantes.

Ao longo dos anos, diversos algoritmos têm sido desenvolvidos para a detecção de comunidades, cada um com suas particularidades e aplicações. Algoritmos hierárquicos, como o de Ravasz (RAVASZ et al., 2002) e o de Girvan-Newman (BAGROW; BOLLT, 2005), são amplamente utilizados, assim como métodos baseados em otimização de métricas, como o

algoritmo de Louvain (BLONDEL et al., 2008) e o Infomap (ROSVALL; BERGSTROM, 2008; ROSVALL; BERGSTROM, 2010). Além disso, algoritmos como o Link Clustering (AHN; BAGROW; LEHMANN, 2010) e o Clique Percolation (DERÉNYI; PALLA; VICSEK, 2005) foram projetados para lidar com aspectos específicos, como a sobreposição de comunidades. No entanto, a eficiência desses algoritmos varia conforme o contexto, e estudos comparativos, como o de Lancichinetti e Fortunato (2009), têm sido realizados para avaliar seu desempenho em diferentes cenários.

Nos últimos anos, novas abordagens têm emergido, integrando técnicas de detecção de comunidades com métodos de agrupamento de dados. Um exemplo relevante é o estudo conduzido por Anjos et al. (2019), que introduziu uma medida de qualidade denominada *Modularity Q*, projetada para ser compatível com o *Framework* para extração ótima de grupos a partir de hierarquia de grupos (FOSC[1]) (CAMPELLO et al., 2013). O FOSC se destaca por sua habilidade de realizar cortes locais em uma árvore de grupos e poderia oferecer uma maior flexibilidade e adaptabilidade ao lidar com grafos, em contraste com algoritmos que se baseiam em cortes horizontais fixos. A métrica *Modularity Q* foi aplicada com sucesso no framework HDBSCAN* (CAMPELLO et al., 2015), demonstrando desempenho superior em comparação com a métrica original implementada no modelo (ANJOS et al., 2019).

Embora essa abordagem não tenha sido aplicada diretamente à detecção de comunidades no estudo original, sua potencial aplicação nesse contexto nos levou a seguinte hipótese: *"A aplicação do framework FOSC, otimizando a medida de qualidade Modularity Q, resulta em uma detecção de comunidades mais precisa em comparação com algoritmos hierárquicos tradicionais, como o de Girvan-Newman e o de Ravasz."*

Neste contexto, este trabalho tem como premissa explorar a aplicação de algoritmos de agrupamento de dados, em particular o FOSC com a medida de qualidade *Modularity Q*, na detecção de comunidades. O FOSC é considerado estado da arte para a técnica de extração de grupos a partir de hierarquia de grupos.

A relevância deste estudo reside na necessidade contínua de aprimorar técnicas de detecção de comunidades, dada sua ampla aplicação em diversos campos. Além disso, a integração de métodos de agrupamento de dados com a detecção de comunidades pode abrir novas perspectivas para a análise de redes complexas, contribuindo para o avanço tanto da teoria dos grafos quanto do aprendizado não supervisionado.

1.2 Objetivos

O objetivo geral do presente trabalho será desenvolver e avaliar a aplicação do *framework* FOSC utilizando a medida de qualidade *Modularity Q* para a detecção de comunidades em redes complexas, comparando seu desempenho com algoritmos hierárquicos tradicionais, como o de

[1] Termo original em Inglês: *Framework for the Optimal Extraction of Clusters from Hierarchies*

Girvan-Newman e o de Ravasz, a fim de verificar se essa abordagem oferece maior precisão na identificação de comunidades.

Para que o objetivo geral seja alcançado, precisamos delimitar os seguintes objetivos específicos:

- **Implementação do FOSC com *Modularity Q***: Adaptar o framework FOSC para utilizar a métrica *Modularity Q* como critério de qualidade na extração de comunidades.
- **Análise Comparativa:** Comparar o desempenho do *framework* FOSC com a medida *Modularity Q* em relação a algoritmos hierárquicos tradicionais, como o de Girvan-Newman e o de Ravasz, utilizando o conhecido benchmark LFR (LANCICHINETTI; FORTUNATO, 2009).
- **Análise de Resultados:** Analisar os resultados obtidos, identificando as vantagens e limitações da aplicação do FOSC com a medida *Modularity Q* na detecção de comunidades, e compará-los com os resultados de algoritmos tradicionais.

1.3 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: no Capítulo 2, são apresentados os conceitos fundamentais sobre os algoritmos abordados. No Capítulo 3, descreve-se a configuração experimental adotada. No Capítulo 4, são expostos os resultados parciais obtidos, acompanhados de discussões e análises críticas. Por fim, no Capítulo 5, são apresentadas as conclusões do estudo, bem como um cronograma para atividades futuras.

2 Revisão Bibliográfica

A detecção de comunidades em uma rede consiste em identificar grupos de nós que são densamente conectados entre si, mas esparsamente ligados a outros grupos. Essa identificação pode ser realizada por meio de algoritmos, que se dividem principalmente em abordagens particionais e hierárquicas. No capítulo 2 será apresentado os conceitos teóricos fundamentais sobre o tema, incluindo a definição de comunidade, a distinção entre comunidades fortes e fracas, e uma análise dos principais algoritmos utilizados para detectá-las.

2.1 Comunidades

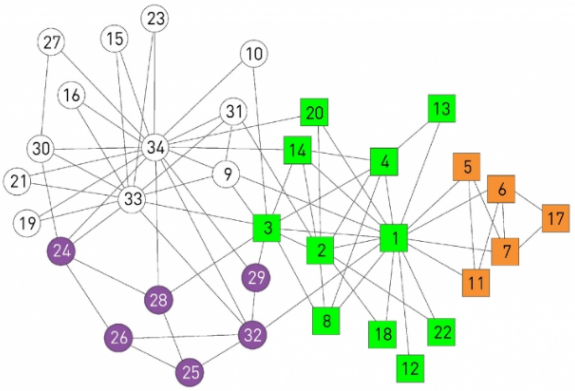
Uma comunidade, no contexto de redes complexas, pode ser definida como um subconjunto de vértices (ou nós) em um grafo que estão mais fortemente conectados entre si do que com os vértices externos a esse subconjunto. Em outras palavras, os vértices de uma comunidade possuem uma densidade de conexões internas significativamente maior do que suas conexões com o restante da rede. Essa definição é fundamental para a detecção de comunidades, que busca identificar esses grupos coesos dentro de uma estrutura de rede.

Um exemplo clássico frequentemente utilizado para ilustrar o conceito de comunidades é o problema do *Zachary's Karate Club*. Nesse caso, temos um clube de caratê composto por exatamente 34 membros. Por ser um grupo relativamente pequeno, todos os membros se conhecem. Para compreender melhor as relações entre os integrantes do clube, o sociólogo Wayne Zachary estudou, ao longo de três anos (de 1970 a 1972), as interações entre os 78 pares de conexões entre os membros que se relacionavam fora do clube. O destaque dessa história está em uma reviravolta ocorrida no grupo: um conflito entre o presidente e o instrutor do clube fez com que os membros se dividissem em dois grupos, metade apoiando o presidente e a outra metade seguindo o instrutor (BARABÁSI; PÓSFAI, 2016).

Na Figura 2.1, é possível visualizar a divisão do grupo em dois por meio do formato dos vértices. Os vértices circulares representam uma comunidade, e os vértices quadrados representam outra. As cores capturam a melhor partição da comunidade prevista por um algoritmo que otimiza o coeficiente de modularidade.

A representação de uma comunidade na literatura geralmente ocorre por meio de um grafo, no qual é possível identificar subgrafos cujos vértices apresentam similaridade entre si, conforme um padrão previamente estabelecido (BARABÁSI; PÓSFAI, 2016). No primeiro grafo da Figura 2.2, observa-se uma pequena representação de comunidades, em que os vértices verdes e laranjas correspondem a duas comunidades distintas. Já no segundo grafo, identifica-se uma única comunidade, representada na cor roxa, conectada a outros vértices cinzas que não pertencem

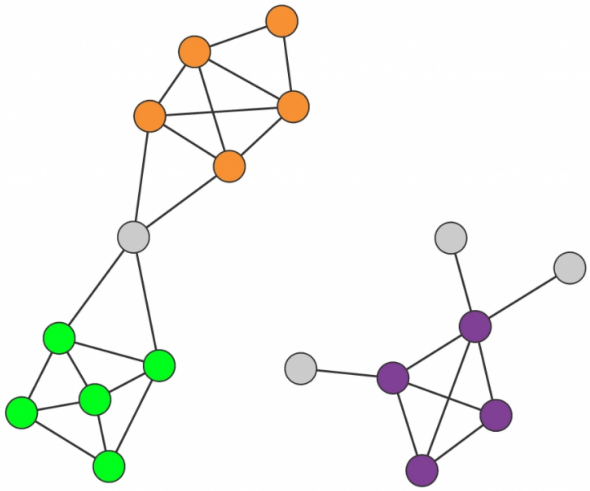
Figura 2.1 – Representação das comunidades dentro do conjunto de dados *Zachary's Karate Club*.



Fonte: Barabási e Pósfai (2016).

a essa comunidade.

Figura 2.2 – Representação de comunidades por meio de grafos.



Fonte: Barabási e Pósfai (2016).

Para identificar e validar comunidades em redes, a literatura se baseia em um conjunto de quatro hipóteses principais que definem desde a sua existência até a sua qualidade. Primeiramente, a *hipótese fundamental* assume que a rede possui uma estrutura de comunidades a ser descoberta. Em seguida, as propriedades dessa estrutura são definidas pela *hipótese de conectividade* (uma comunidade não pode ser desconexa) e pela *hipótese de densidade* (mais conexões internas do que externas), conforme discutido em (BARABÁSI; PÓSFAI, 2016). Para garantir que essas estruturas não são acidentais, a *hipótese aleatória* serve como um teste de significância estatística, comparando a rede com um modelo nulo aleatório. Finalmente, como um critério para encontrar a melhor partição possível, a *hipótese da modularidade* máxima apresenta que a divisão ótima da rede é aquela que maximiza a função de modularidade Q, fornecendo um objetivo claro para os algoritmos de otimização.

Contudo, mesmo que as hipóteses de conectividade e densidade contribuam para a

detecção de comunidades, elas ainda não são suficientes para defini-las de forma completa (BARABÁSI; PÓSFAI, 2016). Em Luce e Perry (1949), é apresentada uma das primeiras definições de comunidades, na qual os autores propõem identificar comunidades em um grafo com base na ideia de cliques. Ou seja, se for possível encontrar subgrafos completos dentro do grafo que atendam a uma definição específica, esses subgrafos podem ser considerados comunidades. Um exemplo disso é o subgrafo roxo mostrado na Figura 2.2. Porém, o próprio autor reconhece algumas limitações em sua definição. Uma delas é que, dentro do grupo, certos subgrafos altamente coesos, embora não atendam à definição formal de clique devido à omissão de algumas métricas, ainda poderiam ser informalmente chamados de "cliques".

Portanto, considerando que encontrar grandes cliques em um grafo extenso representa uma restrição significativa, é aceita atualmente (BARABÁSI; PÓSFAI, 2016) uma abordagem menos rígida para identificar comunidades. Suponha um subgrafo conectado $\mathbf{C}$ com N_c vértices em uma rede. Definem-se dois graus para um vértice i: o grau interno k_i^{int}, que representa as conexões que ligam i a outros vértices dentro de $\mathbf{C}$, e o grau externo k_i^{ext}, que representa o número de conexões que ligam i ao restante da rede.

Com base nessas definições podemos inferir duas situações:

- $k_i^{\text{int}} = 0$, o vértice i não possui conexões internas no subgrafo $\mathbf{C}$, indicando uma comunidade fraca, já que i não está conectado a nenhum outro vértice da comunidade $\mathbf{C}$.
- $k_i^{\text{ext}} = 0$, o vértice i não possui conexões externas ao subgrafo $\mathbf{C}$, indicando uma boa comunidade, pois todas as conexões de i estão ligadas a outros vértices pertencentes à mesma comunidade $\mathbf{C}$.

Agora, dada as definições anteriores, é possível definirmos o que diferencia uma comunidade forte para uma fraca:

- **Comunidade forte**: $\mathbf{C}$ é uma comunidade forte se cada nó dentro do subgrafo $\mathbf{C}$ tiver mais conexões internas no subgrafo do que externas ao restante do grafo. Formalmente, $\mathbf{C}$ representa uma comunidade forte se, para cada nó $i \in \mathbf{C}$ (2.1):

$$k_i^{\text{int}}(\mathbf{C}) > k_i^{\text{ext}}(\mathbf{C}). \tag{2.1}$$

- **Comunidade fraca**: $\mathbf{C}$ é uma comunidade fraca se o grau interno total de um subgrafo exceder seu grau externo total. Assim, $\mathbf{C}$ pode ser representado pela condição (2.2):

$$\sum_{i \in \mathbf{C}} k_i^{\text{int}}(\mathbf{C}) > \sum_{i \in C} k_i^{\text{ext}}(\mathbf{C}). \tag{2.2}$$

2.1.1 Modularidade

O conceito de modularidade na detecção de comunidades tem como objetivo mensurar a qualidade de cada partição dentro de uma rede. Assim, por meio da modularidade, é possível

avaliar o quão boa uma partição é em relação a outra. Esse conceito, portanto, é amplamente aplicado em diversos algoritmos com o intuito de otimizá-los para a detecção de comunidades (BARABÁSI; PÓSFAI, 2016).

Para ilustrar o uso da modularidade em uma rede, considere uma rede composta por N nós e L arestas, dividida em n_c comunidades. Cada comunidade é formada por N_c nós, que estão conectados entre si por L_c arestas, com $c = 1, \ldots, n_c$. Quando o número de arestas L_c dentro de uma comunidade é superior ao número esperado em uma rede aleatória (considerando o grau da rede), isso sugere que os nós pertencentes ao subgrafo C_c formam uma comunidade real. A modularidade permite medir a diferença entre a matriz de adjacência real da rede A_{ij} e o número esperado de arestas entre os nós i e j em uma rede aleatória, representado por p_{ij}, conforme a seguinte equação:

$$M_c = \frac{1}{2L} \sum_{(i,j) \in C_c} (A_{ij} - p_{ij}). \tag{2.3}$$

Na equação (2.3), p_{ij} é calculado após a aleatorização da rede original, preservando o grau esperado de cada nó. Usando um modelo nulo que mantém a distribuição de graus, temos:

$$p_{ij} = \frac{k_i k_j}{2L}. \tag{2.4}$$

A partir da equação (2.3), podemos tirar as seguintes conclusões: se M_c for positivo, o subgrafo C_c possui mais conexões do que o esperado, indicando que ele pode representar uma comunidade real. Se M_c for zero, isso significa que a conectividade entre os nós de C_c pode ser explicada apenas pela distribuição de graus da rede, sugerindo que não há uma estrutura comunitária clara. Se M_c for negativo, os nós de C_c não formam uma comunidade, já que o número de conexões é menor do que o esperado em uma rede aleatória.

A partir da equação (2.4), podemos derivar uma expressão mais simples para a modularidade de uma comunidade C_c:

$$M_c = \frac{L_c}{L} - \left(\frac{k_c}{2L}\right)^2, \tag{2.5}$$

onde L_c é o número total de arestas dentro da comunidade C_c, e k_c é o grau total dos nós dessa comunidade.

Quando generalizamos para a rede inteira, considerando uma partição que divide a rede em n_c comunidades, a modularidade total da rede é definida como a soma das contribuições de cada comunidade.

$$M = \sum_{c=1}^{n_c} \left[\frac{L_c}{L} - \left(\frac{k_c}{2L}\right)^2\right]. \tag{2.6}$$

A partir das fórmulas e conceitos apresentados, é possível identificar três propriedades principais da modularidade:

- **Alta modularidade:** Um valor alto de modularidade indica que a partição da rede resulta em uma estrutura de comunidades bem definida e mais distinta.

- **Modularidade igual a 0:** Se a modularidade for igual a 0, isso sugere que a rede não apresenta uma estrutura de comunidade evidente, possuindo essencialmente uma única comunidade.

- **Modularidade negativa:** Quando a modularidade é negativa, isso indica que os nós estão distribuídos em várias comunidades pequenas, mas sem uma verdadeira coesão entre elas, o que não é ideal para a detecção de comunidades.

2.2 Algoritmos Hierárquicos

Os algoritmos hierárquicos fazem parte da classe de algoritmos não supervisionados, ou seja, não dependem de rótulos pré-definidos para realizar a classificação. Esses algoritmos são amplamente empregados em problemas de agrupamento de dados, cujo objetivo é identificar e agrupar objetos que compartilhem um padrão específico, associando-os a outros objetos que exibam características semelhantes (JAIN; DUBES, 1988).

A relação de similaridade entre esses objetos é estabelecida por meio de uma matriz de similaridade, na qual as linhas e colunas representam os objetos analisados, e os elementos da matriz correspondem às distâncias entre esses objetos (JAIN; DUBES, 1988). No contexto de comunidades, utiliza-se x_{ij} para indicar a distância entre os vértices i e j, sendo que o grau de similaridade entre dois objetos é determinado com base na posição relativa dos vértices dentro da rede (BARABÁSI; PÓSFAI, 2016).

Após a construção da matriz de similaridade, o próximo passo é a aplicação de algoritmos hierárquicos, cujo objetivo é identificar grupos de vértices que apresentem um alto grau de similaridade (BARABÁSI; PÓSFAI, 2016). Nesse contexto, os algoritmos hierárquicos podem ser divididos em duas subclasses: aglomerativos e divisivos.

Nos algoritmos aglomerativos, cada vértice é inicialmente tratado como um grupo individual (ou *singleton*). A cada iteração, quando um alto grau de similaridade é detectado entre determinados vértices, eles são agrupados em um mesmo grupo. Por outro lado, os algoritmos divisivos seguem uma abordagem oposta. Inicialmente, todos os vértices são alocados em um único grupo, e, a cada iteração, os vértices com baixo grau de similaridade são removidos desse grupo principal e realocados para formar novos grupos, de acordo com suas similaridades (JAIN; DUBES, 1988).

Vale destacar que nessas abordagens, dentro do problema de detecção de comunidade, é gerada uma estrutura hierárquica conhecida como dendrograma, que tem como objetivo representar a formação de grupos e prever novas comunidades que possam surgir (BARABÁSI; PÓSFAI, 2016). Nas próximas subseções, serão apresentados alguns dos principais algoritmos hierárquicos utilizados para a identificação de comunidades.

2.2.1 Ravasz

O primeiro algoritmo a ser apresentado para a detecção de comunidades será o algoritmo aglomerativo de Ravasz, criado com o propósito de identificar módulos funcionais em redes metabólicas (RAVASZ et al., 2002; BARABÁSI; PÓSFAI, 2016).

O primeiro passo do algoritmo é a criação da matriz de similaridade, na qual os nós que pertencem à mesma comunidade apresentam um grau maior de similaridade em comparação aos nós de comunidades diferentes. No contexto de comunidades, nós que estão conectados e compartilham os mesmos vizinhos possuem maior probabilidade de pertencerem à mesma comunidade (BARABÁSI; PÓSFAI, 2016). A sobreposição topológica pode ser expressa por:

$$x_{ij}^{0} = \frac{J(i,j)}{\min(k_i, k_j) + 1 - \Theta(A_{ij})}, \tag{2.7}$$

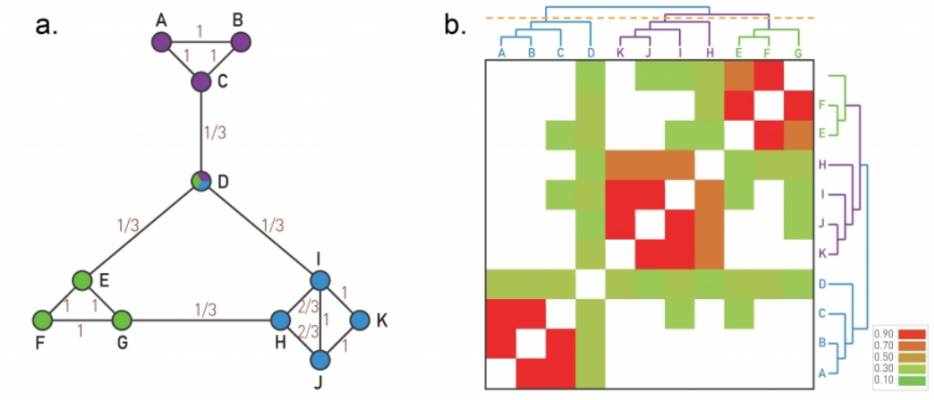
onde $\Theta(x)$ representa a função degrau de Heaviside, que assume o valor 0 para $x \leq 0$ e 1 para $x > 0$. $J(i,j)$ indica o número de vizinhos comuns entre os nós i e j, sendo incrementado $(+1)$ caso exista uma conexão direta entre eles. E por fim, $min(k_i, k_j)$ representa o menor dos graus k_i e k_j.

Da equação (2.7), é possível inferir as duas condições:

- $x_{ij}^{0} = 1$ ocorre quando os nós i e j possuam uma conexão entre si e compartilhem os mesmos vizinhos.
- $x_{ij}^{0} = 0$ ocorre quando os nós i e j não possuem vizinhos em comum e não estão conectados.

Tais condições são apresentadas na Figura 2.3(a).

Figura 2.3 – Exemplo dos passos do algoritmo de Ravasz.



Fonte: Barabási e Pósfai (2016).

No segundo passo, será determinado o grau de similaridade entre os grupos formados no primeiro passo. O algoritmo de Ravasz realiza essa tarefa utilizando a média dos clusters, ou seja, a média de x_{ij} entre os nós i e j que pertencem a comunidades diferentes (RAVASZ et al., 2002).

No terceiro passo temos a aplicação do algoritmo hierárquico, que é dividido nas seguintes etapas:

1. Será atribuído uma comunidade própria para cada nó. Em seguida, será calculado o valor de x_{ij} para a criação da matriz de similaridade.
2. Encontrar um par de comunidades ou de nós com maior similaridade, e uní-los em uma única comunidade.
3. Calcular a similaridade entre a nova comunidade e as demais comunidades.
4. Repetir as etapas 2 e 3 até que todos os nós formem uma única comunidade.

No quarto e último passo, será necessário analisar o dendrograma gerado a partir da aplicação do algoritmo. Pois um aspecto do algoritmo hierárquico aglomerativo é que, ao final do terceiro passo, todos os nós pertencem a uma única comunidade. No entanto, por meio do dendrograma, é possível visualizar a ordem na qual os nós foram se aninhando. Por exemplo, na Figura 2.3(b), ao visualizarmos apenas os nós folhas, podemos observar as primeiras comunidades formadas pelo algoritmo devido ao grau de similaridade de x_{ij}. Nesse caso, os nós A e B se uniram na parte azul da árvore; os nós K e J se uniram na parte roxa da árvore; e os nós E e F se uniram na parte verde da árvore. Após essas três primeiras comunidades serem formadas, elas foram iterativamente se juntando com os demais nós até que fosse possível observar todas as comunidades em uma única árvore, os chamados dendrogramas (BARABÁSI; PÓSFAI, 2016).

Para que seja possível extrair as comunidades, será necessário fazer cortes horizontais na árvore. No entanto, o algoritmo de Ravasz não especifica onde esses cortes devem ser feitos para extrair as melhores comunidades (BARABÁSI; PÓSFAI, 2016).

2.2.2 Girvan-Newman

Um exemplo de algoritmo hierárquico divisivo é o de Girvan-Newman. Esse tipo de algoritmo remove as conexões entre comunidades que apresentam baixa similaridade, até que cada nó pertença a uma comunidade distinta (BARABÁSI; PÓSFAI, 2016).

No primeiro passo do algoritmo, denominado centralidade, o objetivo é selecionar os pares de x_{ij} que pertencem a comunidades diferentes. Assim, o valor buscado na matriz de similaridade será alto apenas se os nós i e j pertencerem a comunidades distintas, e baixo caso contrário (BAGROW; BOLLT, 2005).

Além disso, assim como no algoritmo de Ravasz, será necessário determinar como será calculado o grau de similaridade entre os nós e as comunidades. Uma das métricas mais comumente utilizadas é a *Link Betweenness*, que define x_{ij} como o número de caminhos mais curtos que passam pela conexão (i, j) (BARABÁSI; PÓSFAI, 2016).

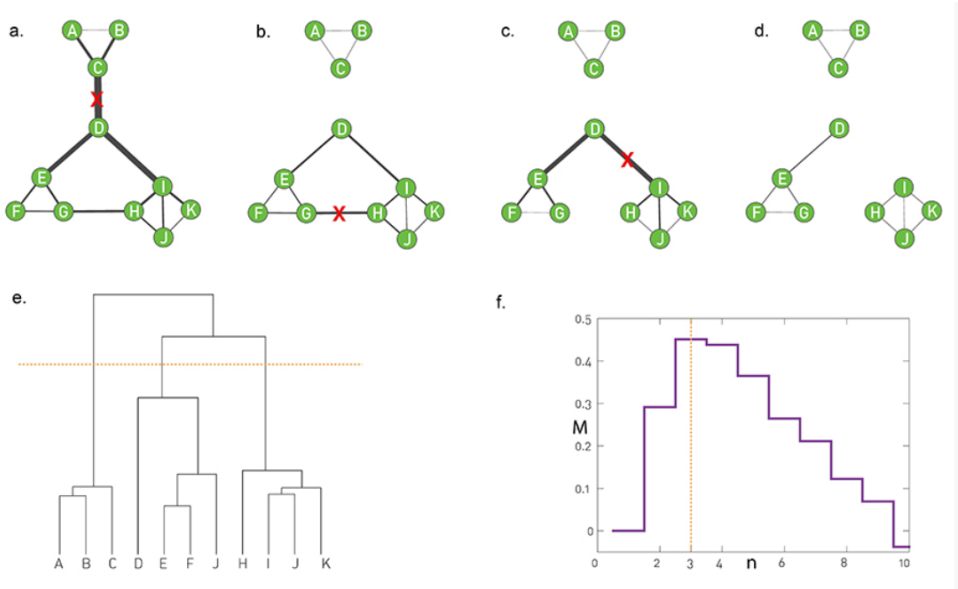
No passo dois será aplicado o algoritmo de agrupamento, que serão definidos em quatro etapas:

1. Calcular a centralidade x_{ij} de cada conexão.
2. Remover as conexões com o maior valor de centralidade. Em caso de empate, escolher umas das conexões aleatoriamente.
3. Recalcular a centralidade de cada conexão na rede alterada.
4. Repetir os passos 2 e 3 até que todas as conexões sejam removidas.

É possível visualizar cada iteração do algoritmo na Figura 2.4. Na etapa (a), o algoritmo remove a conexão que possui o maior valor de centralidade: a conexão entre C e D. Em seguida, logo após o corte de uma conexão, é necessário recalcular os valores de centralidade da rede, pois a remoção de uma conexão impacta todos os valores de centralidade de x_{ij}. Prosseguindo com as interações, ocorrerão mais dois cortes: entre as conexões G e H, e entre as conexões D e L.

Além disso, como o algoritmo de Girvan-Newman só para quando cada nó pertence a exatamente uma comunidade, ele também gera um dendograma, permitindo a identificação das comunidades na rede por meio de cortes horizontais, como mostrado na Figura 2.4(e).

Figura 2.4 – Exemplo dos passos do algoritmo de Girvan-Newman.



Fonte: Barabási e Pósfai (2016).

2.2.3 Greedy Modularity

O algoritmo Greedy Modularity, proposto por Newman (2004), é um método hierárquico aglomerativo que utiliza a modularidade como critério para a detecção de comunidades. A sua

premissa é construir iterativamente partições da rede, mesclando, a cada passo, as comunidades que resultam no maior aumento de modularidade. O objetivo final é identificar a partição que alcança o valor máximo de modularidade ao longo de todo o processo de aglomeração (BARABÁSI; PÓSFAI, 2016).

Os passos do algoritmo podem ser descritos da seguinte forma:

1. **Inicialização:** No início, cada nó da rede constitui sua própria comunidade.
2. **Mesclagem Iterativa:** A cada passo, calcula-se a variação de modularidade, denotada por ΔQ, que resultaria da fusão de cada par de comunidades conectadas. O par cuja união proporciona o maior aumento de modularidade (o maior $\Delta Q > 0$) é então mesclado.
3. **Construção do Dendrograma:** O passo 2 é repetido até que todos os nós pertençam a uma única comunidade. Esse processo gera um dendrograma onde cada nível de fusão está associado a um valor de modularidade total da rede.
4. **Seleção da Partição Ótima:** A estrutura de comunidades final não é necessariamente a última etapa. Ao invés disso, o algoritmo retorna à partição (ao nível do dendrograma) que registrou o maior valor de modularidade global durante todo o processo de execução.

Em relação à complexidade do algoritmo, levando em consideração que L representa o número de arestas, e N o número de nós em um grafo. O cálculo para o passo 2 do algoritmo pode ser feito em $O(L)$. Após selecionar as comunidades que irão mesclar, no pior caso para atualizar a matriz, teremos um $O(N)$. Como o algoritmo irá fazer $N - 1$ operações de mescla até unir todos os nós em uma única comunidade, a complexidade final será $O[(L + N)N]$, ou $O(N^2)$ em um grafo esparso (BARABÁSI; PÓSFAI, 2016).

2.2.4 Louvain

O algoritmo de Louvain, muito conhecido pela sua capacidade de trabalhar com redes complexas que chegam na casa dos bilhões de conexões, foi primeiro proposto no trabalho de Blondel et al. (2008).

O algoritmo é baseado em dois passos:

1. Inicialmente, cada nó da rede está conectado em uma rede ponderada além de, cada um dos nós também fazer parte de uma comunidade. Em seguida, será calculado o grau de modularidade de um nó caso ele seja adicionado em uma comunidade vizinha. Caso o ganho de modularidade seja positivo, então o nó é mesclado nessa comunidade. Porém, essa junção apenas irá ocorrer caso o ganho de modularidade seja positivo. Caso não haja ganho de modularidade, então as comunidades permanecem sem modificação. O processo

acaba até não ser mais possível obter melhorias. A fórmula matemática da modularidade calculada no primeiro passo pode ser expressa por:

$$\Delta M = \left[\frac{\sum_{\text{in}} + 2k_{i,\text{in}}^2 W - \left(\sum_{\text{tot}} + k_i^2 W\right)^2}{2W}\right] - \left[\frac{\sum_{\text{in}}^2 W - \left(\sum_{\text{tot}}^2 W\right)^2 - \left(k_i^2 W\right)^2}{2W}\right], \quad (2.8)$$

onde em uma comunidade C, $\sum_{\text{in}}$ representa a soma dos pesos das conexões dentro da comunidade C; $\sum_{\text{tot}}$ será a soma dos pesos de todas as conexões de todos os nós em C; k_i será a soma dos pesos das conexões incidentes ao nó i; $k_{i,in}$ será a soma dos pesos das conexões de i para todos os nós em C; e W será a soma dos pesos de toda as conexões da rede.

2. No segundo passo, a rede é reconstruída (ou agregada), tratando cada comunidade encontrada na fase anterior como um único nó. O peso da aresta entre dois desses novos nós é definido pela soma dos pesos de todas as conexões que existiam entre as comunidades correspondentes. Em seguida, os passos 1 (otimização de modularidade) e 2 (agregação da rede) são repetidos sobre esta nova rede. O algoritmo termina quando nenhuma nova fusão de comunidades resulta em um aumento da modularidade geral.

Em relação à sua complexidade, o algoritmo do Louvain depende muito mais da sua capacidade de memória do que do seu tempo computacional. Como a cada iteração do algoritmo, o número de vértices e arestas é diminuído, a complexidade não passa de $O(L)$ (BARABÁSI; PÓSFAI, 2016).

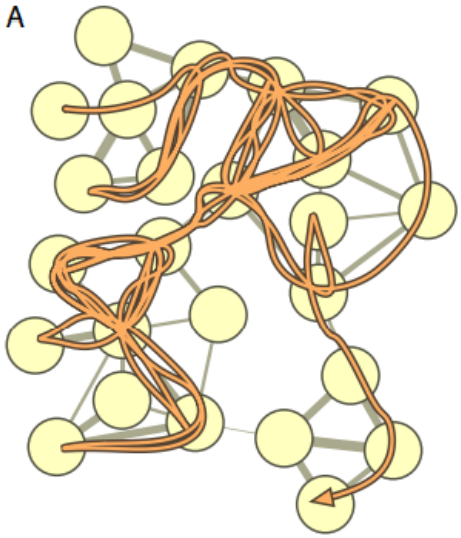
2.2.5 Infomap

O algoritmo Infomap é um algoritmo de detecção de comunidades que utiliza compressão de dados e uma função de qualidade como forma de otimizar a identificação de comunidades (BARABÁSI; PÓSFAI, 2016; ROSVALL; BERGSTROM, 2008).

O primeiro passo do algoritmo ocorre a partir de um caminho aleatório na rede, como mostrado na Figura 2.5, onde, partindo de um nó aleatório, é realizado um percurso aleatório até alcançar um ponto de destino. O objetivo desse percurso é descrevê-lo utilizando o menor número de símbolos possível. Para isso, as estruturas mais frequentemente utilizadas serão nomeadas com nomes curtos e únicos, usados para identificar as comunidades (ROSVALL; BERGSTROM, 2008).

O método utilizado para nomear os vértices será o método de Huffman, que atribuirá um código como nome aos nós com base em uma estimativa da probabilidade de que o passeio anterior visitará esse nó. No exemplo da Figura 2.6, utiliza-se 314 bits para fazer a nomeação, onde o nó de partida é representado pelo código 1111100, localizado no canto superior esquerdo. A partir dele, os vértices pelos quais o caminho passa são nomeados até alcançar o vértice de

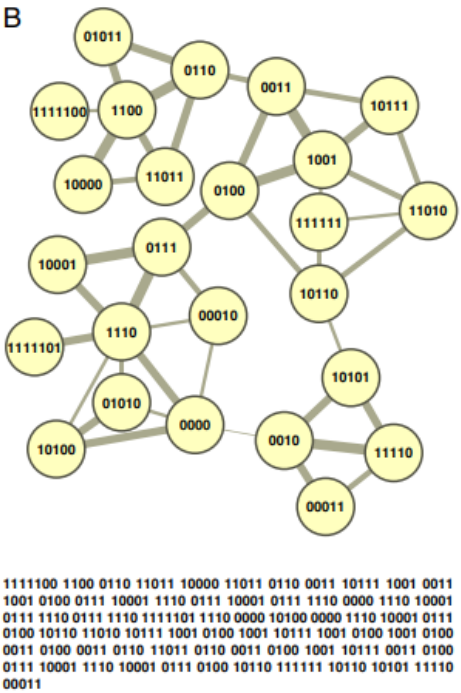
Figura 2.5 – Primeiro passo do algoritmo Infomap: caminho aleatório na rede.



Fonte: Rosvall e Bergstrom (2008).

destino, localizado no canto superior direito e representado pelo código 00011 (ROSVALL; BERGSTROM, 2008).

Figura 2.6 – Segundo passo do algoritmo Infomap: algoritmo de Huffman para nomeação de nós.

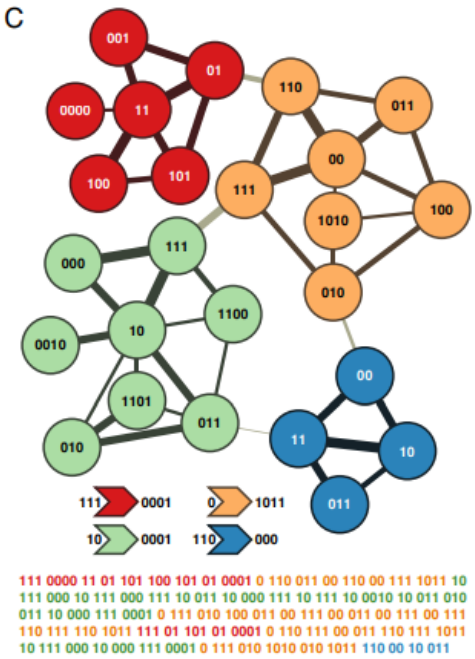


Fonte: Rosvall e Bergstrom (2008).

No terceiro passo do algoritmo, teremos uma representação de codificação em dois níveis do passeio aleatório. As comunidades irão receber nomes únicos em relação aos demais nós dentro da comunidade, porém, nós de diferentes comunidades poderão ter nomes iguais. Esse processo faz com que os nomes dos nós diminuam cerca de 32% em tamanho. Os códigos que

nomeiam os módulos e os códigos usados para indicar a saída de cada módulo são mostrados à esquerda e à direita abaixo da rede na Figura 2.7. A partir desse código, é possível descrever o passeio mostrado na Figura 2.5 (A) com 243 bits, apresentados abaixo da rede em (Figura 2.7). Os três primeiros bits, 111, indicam que o passeio começa no módulo vermelho, o código 0000 especifica o primeiro nó do passeio, e assim por diante (ROSVALL; BERGSTROM, 2008).

Figura 2.7 – Terceiro passo do algoritmo Infomap: codificação em dois níveis do passeio aleatório.



Fonte: Rosvall e Bergstrom (2008).

No quarto e último passo, ao seguir os passos do algoritmo e relatar apenas os nomes das comunidades, sem considerar sua localização, é possível obter uma granularidade eficiente da rede, que corresponde à estrutura da comunidade (ROSVALL; BERGSTROM, 2008), como mostrado na Figura 2.8.

O código de otimização é obtido achando o mínimo do *map equation*:

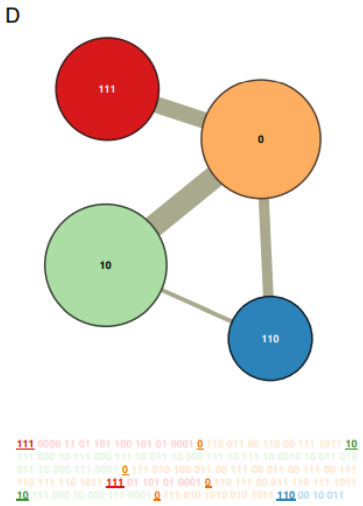
$$L = qH(Q) + \sum_{c=1}^{n_c} p_c \odot H(P_c). \tag{2.9}$$

A equação do código de otimização pode ser dividida em dois termos (ROSVALL; BERGSTROM, 2008): o primeiro representa a entropia do movimento entre módulos, e o segundo termo representa a entropia dos movimentos dentro dos módulos.

2.2.6 Framework for Optimal Selection of Clusters (FOSC)

No trabalho de Campello et al. (2013), é apresentado um *framework* para a seleção otimizada de *clusters*, denominado FOSC, utilizado para a extração de *clusters* em uma árvore hierárquica simplificada. Essa abordagem é implementada por meio do paradigma de programação

Figura 2.8 – Quarto passo do algoritmo Infomap: algoritmo de Huffman para nomeação de nós.



Fonte: Rosvall e Bergstrom (2008).

dinâmica *bottom-up*, que utiliza uma métrica de qualidade para avaliar os valores dos clusters e armazenar esses valores ao longo da árvore.

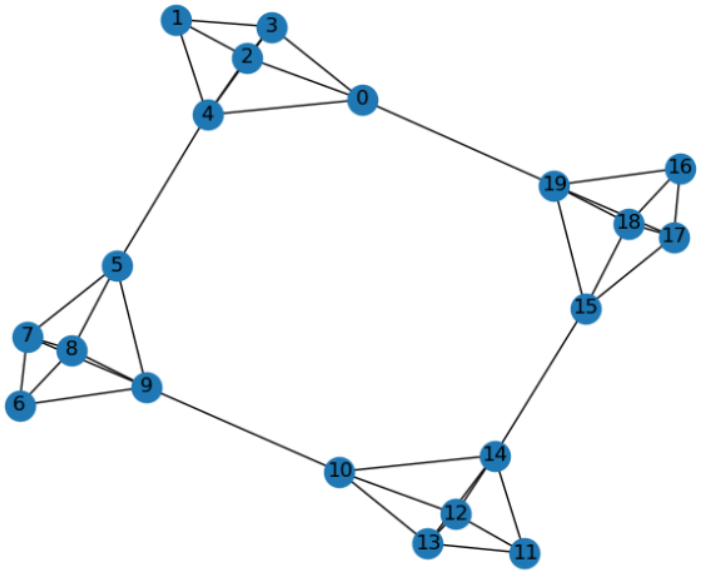
No entanto, é necessário criar a árvore simplificada para realizar a extração dos clusters. No contexto de detecção de comunidades, essa árvore pode ser obtida a partir do dendrograma gerado pelos algoritmos hierárquicos. Além disso, durante o processo de obtenção da árvore simplificada, utiliza-se um parâmetro chamado m_{ClSize}, que representa o número mínimo de nós necessários para a formação de um cluster.

Para um melhor entendimento dos passos do algoritmo, será apresentado um exemplo utilizando a rede *Connected Caveman Graph*, disponibilizada pelo pacote *NetowrkX*: <NetworkX>. Uma particularidade dessa rede é que é possível escolher tanto a quantidade de cliques quanto o tamanho de cada clique que serão gerados. Neste exemplo, foram utilizados os valores 4 para o número de cliques e 5 para o tamanho de cada clique. A rede pode ser visualizada na Figura 2.9.

Em seguida, foi aplicado o algoritmo hierárquico *Average Linkage* para gerar o dendrograma, como pode ser visto na Figura 2.10. A partir desse dendrograma, será possível gerar a árvore simplificada. Esse processo começa na raiz do dendrograma, onde, a partir do valor m_{ClSize}, descemos na árvore verificando se, naquele nível, é possível formar clusters que respeitem o valor mínimo de m_{ClSize} especificado. Para este exemplo, utilizou-se o valor 4. Caso, em determinado nível da árvore, não seja possível criar um novo cluster por não atender ao valor mínimo de m_{ClSize}, os nós correspondentes são considerados ruídos. A hierarquia de grupos gerada pode ser visualizada na tabela 2.1.

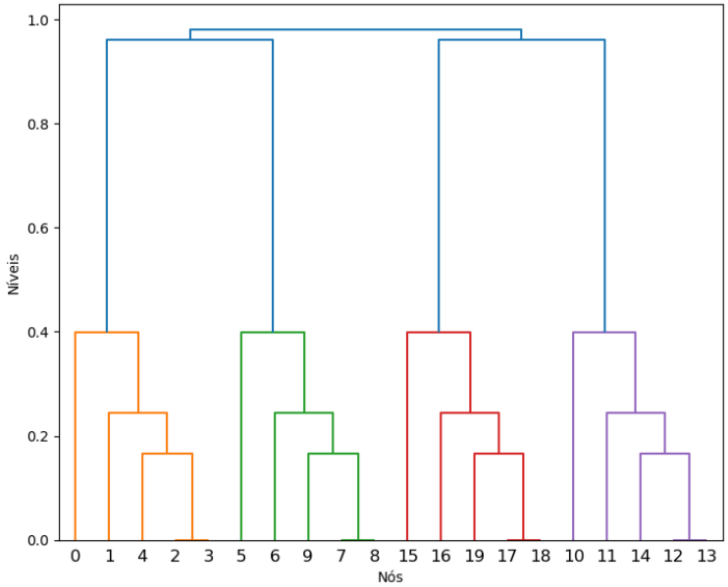
A partir da hierarquia de clusters gerada, é possível utilizarmos medidades de qualidade como forma de quantificar a qualidade de um cluster. Originalmente, no trabalho de Campello et al. (2013), a medida de qualidade utilizada para a comparação entre os clusters foi a medida de estabilidade (*cluster lifetime*). No entanto, na detecção de comunidades, como o foco está no

Figura 2.9 – Rede do Connected Caveman Graph.



Fonte: Elaborado pelo autor.

Figura 2.10 – Dendrograma gerado a partir do *Average Linkage*



Fonte: Elaborado pelo autor.

particionamento de vértices de um grafo, a medida de qualidade empregada precisa ser capaz de realizar essa divisão para identificar as comunidades.

Dessa forma, no trabalho de Anjos et al. (2019), que se baseou na detecção de comunidades para a criação de seu algoritmo, os autores utilizaram a métrica *Modularity Q*, originalmente proposta em Newman e Girvan (2004), como a métrica de qualidade para a detecção de comunidades:

$$QT(P) = \sum_{i=1}^{k} \left[\frac{IS(C_i)}{TS} - \left(\frac{DS(C_i)}{TS} \right)^2 \right]. \tag{2.10}$$

Tabela 2.1 – Hierarquia de clusters produzida a partir do dendrograma gerado pelo método de Average Linkage.

Scale	0,98	0,96	0,40	0,24	0,17	0,00
x_0	1	2	6	0	0	0
x_1	1	2	6	6	0	0
x_2	1	2	6	6	0	0
x_3	1	2	6	6	0	0
x_4	1	2	6	6	0	0
x_5	1	2	7	0	0	0
x_6	1	2	7	7	0	0
x_7	1	2	7	7	0	0
x_8	1	2	7	7	0	0
x_9	1	2	7	7	0	0
x_{10}	1	3	5	0	0	0
x_{11}	1	3	5	5	0	0
x_{12}	1	3	5	5	0	0
x_{13}	1	3	5	5	0	0
x_{14}	1	3	5	5	0	0
x_{15}	1	3	4	0	0	0
x_{16}	1	3	4	4	0	0
x_{17}	1	3	4	4	0	0
x_{18}	1	3	4	4	0	0
x_{19}	1	3	4	4	0	0

Fonte: Elaborado pelo autor.

Nesse contexto, $P = \{C_i, \ldots, C_k\}$ representa as partições dos vértices k em grupos disjuntos C_i; $IS(C_i)$ representa a soma dos pesos das conexões dentro do grupo C_i; $DS(C_i)$ corresponde à soma dos pesos das conexões que possuem pelo menos um vértice em C_i; e, por fim, TS é a soma dos pesos das conexões no grafo (ANJOS et al., 2019).

A ideia por trás do uso da *Modularity Q* como métrica de qualidade está no fato de que comunidades consideradas fortes estarão particionadas em grupos que possuem um grande número de conexões internas (BARABÁSI; PÓSFAI, 2016). Portanto, levando em consideração que essa métrica é compatível com o algoritmo FOSC, conforme mostrado por Anjos et al. (2019), podemos questionar se, caso o algoritmo de extração de grupos FOSC fosse aplicado em comunidades utilizando a métrica *Modularity Q*, ele poderia extrair melhores comunidades em comparação com os algoritmos já existentes na literatura.

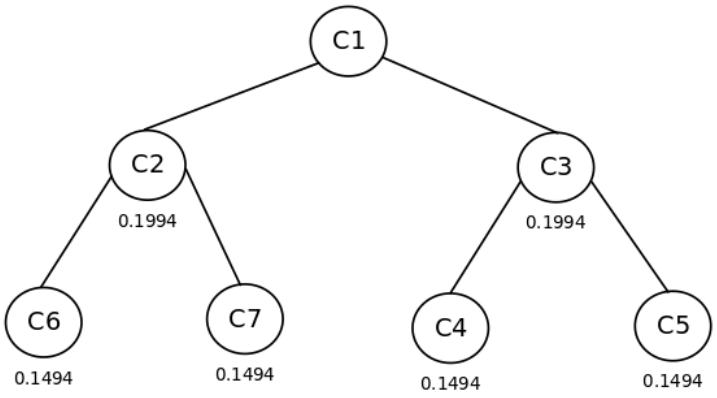
Com a medida de qualidade estabelecida e aplicada na árvore de clusters, é agora necessário entender os passos do algoritmo para a seleção ótima dos clusters (CAMPELLO et al., 2013):

1. Se a soma dos valores de qualidade dos nós filhos for superior ao valor do nó pai, os nós filhos são adicionados ao conjunto solução, e o nó pai é desconsiderado.

2. Caso contrário, se o valor de qualidade do nó pai for superior à soma dos valores de qualidade dos nós filhos, então o nó pai é incluído na solução, e os filhos são desconsiderados.

Esses passos podem ser melhor visualizados na Figura 2.11. Começando pelos nós folhas mais profundos da árvore, $\mathbf{C}_6$ e $\mathbf{C}_7$, a soma da modularidade desses dois clusters resulta em $0,2988$. Comparando esse valor com o do nó pai, $\mathbf{C}_2$, observa-se que $0,2988$ é superior a $0,1994$. Logo, os grupos $\mathbf{C}_6$ e $\mathbf{C}_7$ sobem de nível na árvore, e o nó $\mathbf{C}_2$ é descartado. Seguindo os mesmos passos para os demais clusters, a soma de $\mathbf{C}_4$ e $\mathbf{C}_5$ resulta em $0,2988$, valor superior ao do seu pai, $\mathbf{C}_3$, que é $0,1994$. Portanto, como o próximo nó pai nesse caso seria o nó raiz $\mathbf{C}_1$, o algoritmo para aqui, e o conjunto solução será composto pelos clusters $\mathbf{C}_6$, $\mathbf{C}_7$, $\mathbf{C}_4$ e $\mathbf{C}_5$.

Figura 2.11 – Exemplo de uma árvore de clusters. Os valores abaixo de cado nó representam a qualidade do grupo.



Fonte: Elaborado pelo autor.

No exemplo apresentado foi possível visualizar na prática a aplicação de dois princípios do algoritmo:

1. **Aditividade:** a comparação entre os valores de qualidade dos nós folhas e seu respectivo pai será realizada até se chegar à raiz da árvore.

2. **Localidade:** os valores dos grupos em diferentes regiões da árvore não influenciam os grupos que estão sendo testados no momento.

2.3 LFR Benchmark

O LFR é um benchmark conhecido para detecção de comunidades que acaba aparecendo nos trabalhos mais conhecidos que buscam testar a eficiência dos algoritmo de detecção de

comunidades, como em Lancichinetti e Fortunato (2009), Yang, Algesheimer e Tessone (2016) e Barabási e Pósfai (2016).

Uma das limitações de outros benchmarks conhecidos está na geração de redes que não condizem com a realidade. Em Lancichinetti e Fortunato (2009), por exemplo, os autores expõem que um benchmark amplamente utilizado, o GN benchmark, gera um grafo no qual todos os nós possuem graus semelhantes e todas as comunidades têm o mesmo tamanho. No entanto, em redes reais, a distribuição de grau geralmente apresenta uma cauda longa, assim como a distribuição do tamanho das comunidades (BARABÁSI; PÓSFAI, 2016). Dessa forma, um algoritmo que apresenta um bom desempenho no GN benchmark não necessariamente terá um bom desempenho em redes reais. Para corrigir essa limitação, o LFR benchmark surge como uma alternativa, construindo redes cujas distribuições tanto dos graus dos nós quanto dos tamanhos das comunidades pré-definidas seguem leis de potência (LANCICHINETTI; FORTUNATO, 2009; BARABÁSI; PÓSFAI, 2016).

O benchmark é construído da seguinte forma:

- O algoritmo começa com N nós isolados.
- Em seguida, cada nó é atribuído a uma comunidade de tamanho N_c, onde N_c segue a distribuição de lei de potência $P_{N_c} \sim N_c^{-\zeta}$, com o expoente da comunidade ζ. Além disso, cada nó i recebe um grau k_i, selecionado a partir da distribuição de lei de potência $p_k \sim k^{-\gamma}$, com expoente de grau γ.
- Cada nó i dentro de uma comunidade recebe um grau interno $(1-\mu)k_i$, representado por links cuja cor corresponde à cor do nó. Os graus restantes, (μk_i), representados por links pretos, conectam-se a nós de outras comunidades.
- Todas as *stubs* (pontas de arestas) dos nós dentro da mesma comunidade são conectadas aleatoriamente entre si até que não restem pontas de arestas livres. Dessa forma, mantém-se a sequência de graus internos de cada nó dentro de sua comunidade. Os *stubs* μk_i restantes são conectadas aleatoriamente a nós de outras comunidades.

Na Tabela 2.2 é apresentado o que cada parâmetro do LFR representa.

O benchmark LFR, tornou-se uma ferramenta de grande importância para a avaliação de algoritmos de detecção de comunidades, como mostrado nos trabalhos de Yang, Algesheimer e Tessone (2016) e no livro de Barabási e Pósfai (2016). Entretanto, ele também apresenta algumas limitações, conforme exposto em Le et al. (2017), especialmente em relação ao parâmetro de mistura.

Uma das limitações destacadas por Le et al. (2017) está na homogeneidade do parâmetro μ para as comunidades geradas. Muitos trabalhos utilizam esse parâmetro como base para realizar testes, ajustando-o de modo a aumentar as conexões externas e reduzir as internas, o que torna a

Tabela 2.2 – Descrição dos parâmetros do LFR.

Símbolo	Parâmetro	Descrição
N	n	Número de nós na rede.
$\langle k \rangle$	$average\_degree$	Grau médio da rede.
γ	$tau1$	Expoente da distribuição de grau.
$k_{\max}$	max_degree	Grau máximo permitido para qualquer nó.
ζ	$tau2$	Expoente da distribuição do tamanho das comunidades.
-	$min\_community$	Tamanho mínimo de uma comunidade.
-	$max\_community$	Tamanho máximo de uma comunidade.
-	μ	Parâmetro de mistura.

Fonte: Elaborado pelo autor.

tarefa de detecção de comunidades mais desafiadora. Contudo, no LFR esse ajuste é aplicado de forma fixa a todas as comunidades da rede, o que pode não representar adequadamente o comportamento de redes reais, já que a variação nos parâmetros de mistura pode afetar de maneira distinta o desempenho dos algoritmos de detecção de comunidades (LE et al., 2017).

2.4 Trabalhos Relacionados

O trabalho de Lancichinetti e Fortunato (2009) apresenta uma análise comparativa de algoritmos de detecção de comunidades, avaliando seu desempenho em diferentes tipos de redes. Um diferencial do estudo é a crítica dos autores ao fato de que muitos algoritmos, embora desenvolvidos para essa finalidade, raramente são submetidos a testes rigorosos. Além disso, quando testados, a avaliação costuma se limitar a redes pequenas com comunidades conhecidas ou a grafos artificiais com estruturas simplificadas, que não representam fielmente a complexidade das redes do mundo real.

Nesse estudo, os autores testaram diversos algoritmos, como Girvan-Newman, *Greedy Modularity Optimization*, CFinder e Infomap. A comparação foi realizada em três *benchmarks* distintos: o GN, com comunidades bem definidas; o LFR, com heterogeneidade no tamanho das comunidades e no grau dos nós; e grafos aleatórios, que serviram como *baseline* por não possuírem uma estrutura de comunidade inerente.

Os resultados indicaram que o desempenho dos algoritmos varia consideravelmente conforme o *benchmark* utilizado. Alguns se mostraram eficazes em tipos específicos de rede, enquanto outros apresentaram dificuldades, levando à conclusão de que nenhum algoritmo é universalmente superior e que sua eficácia depende fortemente da estrutura da rede analisada. O estudo também revelou fraquezas específicas, como o baixo desempenho de métodos baseados em *modularity optimization* em redes com sobreposição de comunidades, evidenciando suas limitações.

Já em Yang, Algesheimer e Tessone (2016), os autores realizam uma análise comparativa

de oito algoritmos do estado-da-arte, utilizando exclusivamente o *benchmark* LFR. O trabalho teve três objetivos centrais: propor critérios para selecionar o algoritmo mais adequado com base em propriedades observáveis da rede; usar o parâmetro de mistura (μ) como um indicador de confiabilidade; e estudar como o tamanho da rede impacta a acurácia e o tempo de computação.

Como principal conclusão, os autores observaram que a escolha do algoritmo ideal depende de um balanço entre acurácia e eficiência. Para redes menores, onde o tempo de processamento é menos crítico, algoritmos como Infomap, *Label Propagation*, *Walktrap* e *Spin-glass* são recomendados pela alta acurácia. Em redes maiores, a eficiência se torna crucial, favorecendo Infomap, Label Propagation e Multilevel. Ao ponderar ambos os critérios, o algoritmo *Multilevel* destacou-se como a opção mais robusta na maioria dos cenários.

A análise também confirmou que o parâmetro de mistura (μ) é um indicador eficaz do desempenho esperado (YANG; ALGESHEIMER; TESSONE, 2016). Para valores baixos de μ, a maioria dos algoritmos apresenta bons resultados. Contudo, à medida que μ aumenta – indicando comunidades menos definidas–, a performance de todos os algoritmos diminui, especialmente em redes de grande porte.

Em suma, enquanto Lancichinetti e Fortunato (2009) demonstrou que não existe um "melhor algoritmo" para todos os casos, Yang, Algesheimer e Tessone (2016) avançou ao fornecer um *framework* prático para escolher o algoritmo mais adequado, utilizando o tamanho da rede e o parâmetro μ como guias.

Este trabalho se alinha às abordagens de Lancichinetti e Fortunato (2009), Yang, Algesheimer e Tessone (2016) ao empregar o *benchmark* LFR para uma avaliação rigorosa em redes que simulam cenários realistas. O diferencial da presente proposta reside na sua análise focada: serão avaliados diversos algoritmos hierárquicos, conforme a fundamentação teórica, e seus resultados serão comparados diretamente com a abordagem baseada no FOSC, proposta por Anjos et al. (2019), a fim de investigar a eficácia deste novo método no contexto da detecção de comunidades.

3 Configuração experimental

Em relação à implementação dos algoritmos que foram testados neste trabalho, grande parte deles já possui implementações prontas no pacote NetworkX, disponível em Hagberg, Schult e Swart (2008), ou em repositórios que contêm suas respectivas documentações.

Além disso, nas etapas iniciais foi utilizado o benchmark LFR disponibilizado pelo pacote NetworkX [1]. Porém, uma das dificuldades encontradas em seu uso foi o fato de essa implementação ser muito rígida na escolha de parâmetros, o que inviabilizou a replicação daqueles parâmetros comumente utilizados na literatura. Por conta disso, tornou-se necessário recorrer a outro gerador de redes LFR. Assim, o gerador do benchmark utilizado nos testes é disponibilizado por Santo Fortunato em seu site oficial [2].

Os algoritmos que já possuem implementação no NetworkX, em suas documentações, ou em repositórios de terceiros são:

1. Girvan-Newman, que pode ser utilizado por meio da função *nx.girvan_newman()*, disponível no pacote NetworkX [3].
2. Louvain, acessível por meio da função *community_louvain.best_partition()*, que pode ser encontrada em seu repositório no GitHub [4].
3. Greedy Modularity, disponível por meio da função *nx.greedy_modularity_communities()*, também fornecida pelo pacote NetworkX [5].
4. Infomap, acessível por meio da função *infomap.Infomap()*, disponibilizada em seu próprio repositório no GitHub [6].

Em relação aos algoritmos que não possuem implementação própria e que precisarão ser desenvolvidos, destacam-se os algoritmos de Ravasz e o próprio algoritmo FOSC.

O algoritmo de Ravasz não possui uma implementação pronta, ao contrário da maioria dos algoritmos apresentados anteriormente. Portanto, sua implementação manual foi necessária, baseada no trabalho de Ravasz et al. (2002).

[1] <https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.LFR_benchmark_graph.html>
[2] <https://www.santofortunato.net/resources>
[3] <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html>
[4] <https://github.com/taynaud/python-louvain>
[5] <https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html>
[6] <https://github.com/mapequation/infomap>

Entretanto, algumas modificações foram realizadas para adequar a implementação às necessidades deste trabalho. Por exemplo, a implementação original utilizava rótulos de nós para iterar sobre a matriz de distâncias, o que poderia causar erros caso os nós não fossem representados por números inteiros. Para solucionar esse problema, foi utilizado um dicionário para mapear os nós a índices sequenciais.

Como mencionado na fundamentação teórica, o FOSC é um algoritmo que busca identificar os melhores *clusters* dentro de uma árvore simplificada (CAMPELLO et al., 2013). Neste trabalho, o objetivo é aplicar esse algoritmo a um dendrograma gerado a partir de um algoritmo hierárquico (Ravasz) e avaliar sua performance em relação à acurácia na detecção de comunidades, comparando-o com outros algoritmos hierárquicos.

A implementação do FOSC utilizada como base foi utilizada em trabalhos anteriores (SILVA, 2024). Foram necessárias algumas modificações nessa implementação, já que originalmente ela trabalhava com dados tabulares. Ou seja, foi necessária a mudança da entrada do algoritmo, que inicialmente aceitava um dendrograma no formato *linkage* e dados tabulares, passando a aceitar um dendrograma ainda no formato *linkage*, mas agora aplicado diretamente à própria rede gerada pelo benchmark LFR.

3.1 Especificação de parâmetros e validação

Os testes foram realizados sobre os algoritmos Ravasz, Girvan-Newman, Greedy Modularity, Infomap, Louvain e FOSC (Ravasz). Para o valor de m_{ClSize} do FOSC, foi utilizado o valor 4. Em relação aos parâmetros utilizados, foram adotados os mesmos parâmetros empregados nos testes descritos em Barabási e Pósfai (2016), com uma iteração para cada valor de μ. Os parâmetros do benchmark LFR são apresentados na Tabela 3.1.

Tabela 3.1 – Parâmetros do LFR

Símbolo	**Parâmetro**	**Valor**
N	n	1000
$\langle k \rangle$	$average\_degree$	20
$k_{\max}$	max_degree	50
γ	$tau1$	2
ζ	$tau2$	1
-	$min\_community$	20
-	$max\_community$	100
-	μ	$0,1 \leq \mu \leq 0,8$ (incremento de 0,1)

Fonte: Elaborado pelo autor.

Além de utilizar os mesmos parâmetros do benchmark para os seis algoritmos testados, também é necessário mensurar a qualidade das comunidades detectadas pelos algoritmos ao longo dos testes. Para isso, foi utilizada a função *Normalized Mutual Information* do pacote

sklearn [7]. Nessa função, uma das partições passadas representa o conjunto de comunidades reais da rede gerada pelo LFR, conhecido como *ground_truth*, enquanto a outra partição corresponde àquela identificada pelo algoritmo.

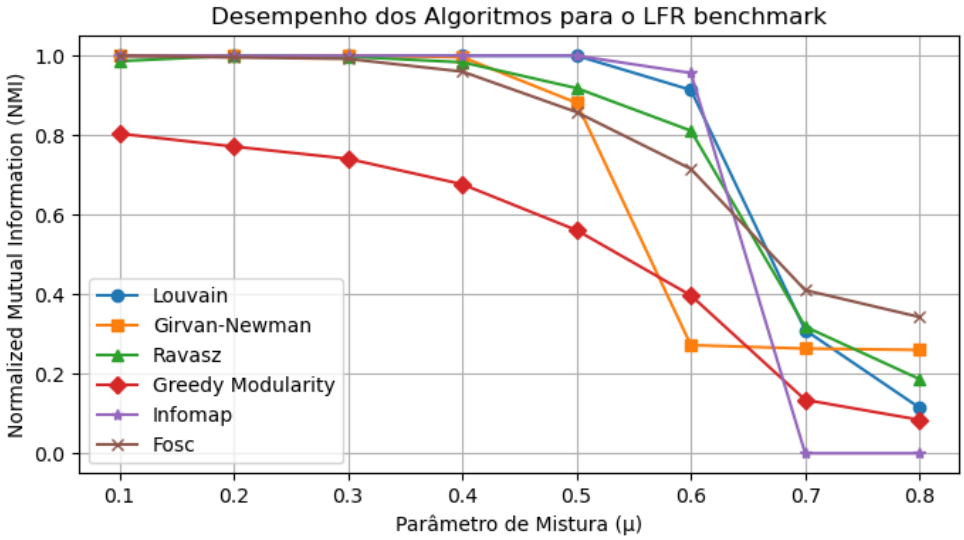
Com base nessas partições, a função retorna a informação mútua normalizada em uma escala de 0 (nenhuma informação mútua) a 1 (correlação perfeita), permitindo avaliar a qualidade da partição detectada em relação à partição real.

[7] <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html>

4 Resultados

Seguindo os mesmos procedimentos adotados nos trabalhos de Lancichinetti e Fortunato (2009), Yang, Algesheimer e Tessone (2016) e em Barabási e Pósfai (2016), os testes dos algoritmos foram conduzidos manipulando o valor do parâmetro de mistura (μ) a cada iteração do algoritmo, variando de $0,1$ a $0,8$, conforme ilustrado na Figura 4.1.

Figura 4.1 – Resultados da aplicação de métodos de detecção de comunidades a partir do Benchmark LFR.



Fonte: Elaborado pelo autor.

Os resultados obtidos indicaram que, para valores baixos de μ, a maior parte dos algoritmos atinge um NMI igual a 1, ou seja, identificam corretamente todas as comunidades, ou valores muito próximos a isso, como $0,99$ ou $0,98$. O único que não apresentou o mesmo desempenho foi o algoritmo Greedy Modularity, que obteve valor de $0,8$.

A queda de desempenho dos algoritmos ocorre, na maioria das vezes, quando o parâmetro de mistura μ é igual a $0,50$. Nessa situação, algoritmos como Girvan-Newman, Ravasz e FOSC começam a perder desempenho e se afastam do valor de NMI igual a $1,00$, enquanto Louvain e Infomap ainda alcançam o valor máximo, mas passam a decair em seguida, à medida que o parâmetro de mistura se aproxima de $0,60$.

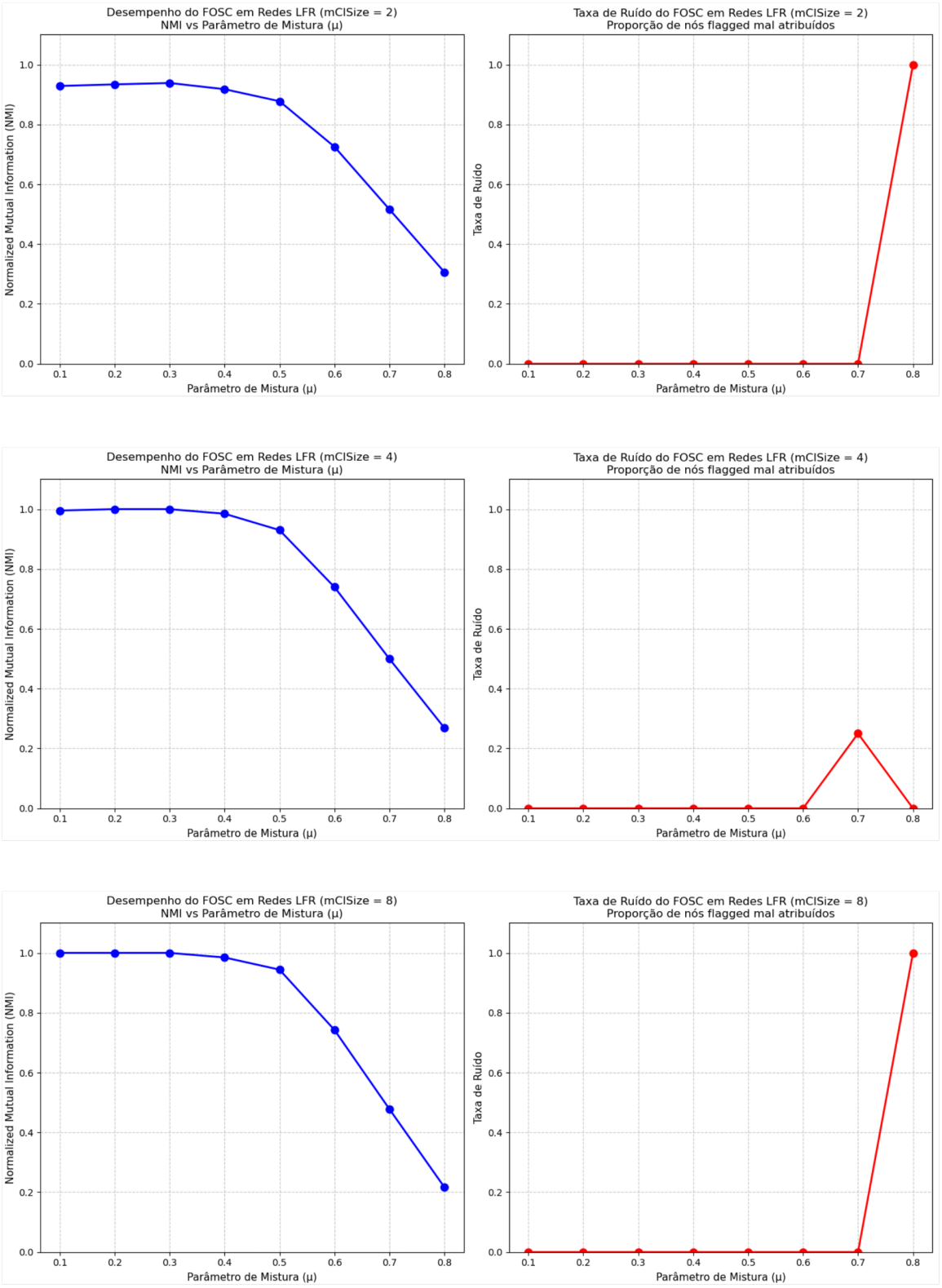
Para um parâmetro de mistura igual a $0,80$, em que a rede se torna bem mais desorganizada e a detecção de comunidades passa a ser uma tarefa mais complexa, todos os algoritmos apresentaram valores de NMI abaixo de $0,40$. Nesse caso, o FOSC mostrou um desempenho ligeiramente superior aos demais, alcançando um valor de NMI de $0,34$.

Além disso, um dos fatores que podem influenciar o cálculo do NMI no FOSC está relacionado à taxa de ruídos, isto é, nós que acabam sendo alocados em comunidades incorretas

pelo algoritmo. Conforme mencionado na revisão bibliográfica, durante o processo de criação de clusters, caso não seja possível formar um novo cluster porque os nós presentes naquele nível da árvore não ultrapassam o valor de m_{ClSize}, esses nós são rotulados como ruído. Após a execução do algoritmo FOSC, é possível verificar o destino de cada nó para analisar se esses ruídos foram corretamente absorvidos por suas comunidades de origem ou se permaneceram em comunidades equivocadas. Quando isso ocorre, há um impacto direto no cálculo do NMI, uma vez que a comparação é feita em relação às comunidades reais.

Como mostrado na Figura 4.2, até um parâmetro de mistura igual a $0,50$, os ruídos encontrados na rede nunca acabam em comunidades que correspondam exatamente às comunidades reais do *ground truth* do NMI. Porém, para níveis mais altos, como $0,60$ até $0,80$, já é possível observar que a taxa de ruídos alocados em comunidades incorretas é bastante elevada. Isso pode explicar a queda gradual do desempenho do algoritmo a partir do parâmetro de mistura $0,50$.

Figura 4.2 – Taxa de Ruídos do FOSC para uma iteração do Benchmark LFR com os valores de m_{ClSize}: 2, 4 e 8.



Fonte: Elaborado pelo autor.

5 Considerações finais do trabalho

5.1 Conclusões

Nesse trabalho, teve-se como objetivo a implementação e o teste do algoritmo de extração ótima FOSC, utilizando a modularidade Q como métrica de qualidade para a definição dos melhores clusters. O algoritmo foi testado a partir dos primeiros passos do método de Ravasz: criação da matriz de similaridade e aplicação do algoritmo hierárquico *Average Linkage*. Após essas etapas iniciais, partindo da rede gerada pelo benchmark LFR e do dendrograma produzido pelo algoritmo de Ravasz, o FOSC foi aplicado com o propósito de identificar os clusters que maximizassem a métrica de qualidade modularidade Q.

Os testes foram realizados utilizando um dos benchmarks mais comumente empregados na avaliação de algoritmos de detecção de comunidades, o LFR benchmark (LANCICHINETTI; FORTUNATO, 2009), com parâmetros iguais aos utilizados no Capítulo 9 do Barabási e Pósfai (2016). Nesses testes, podemos ver os desempenhos dos algoritmos testados ao se variar o parâmetro de mistura da rede. Além disso, os algoritmos comparados ao FOSC incluíram alguns dos métodos mais consolidados na literatura, como Girvan-Newman, Ravasz, Greedy Modularity, Louvain e Infomap.

Foram realizados testes para avaliar as taxas de ruído, ou seja, a ocorrência de nós alocados em comunidades diferentes das comunidades reais geradas pelo benchmark LFR. Nessas avaliações, realizaram-se três experimentos variando-se o valor do parâmetro m_{ClSize}, que define o tamanho mínimo dos agrupamentos, em cada etapa. A variação desse parâmetro impacta diretamente a alocação dos nós, o que, por sua vez, penaliza os valores de NMI e pode resultar em uma redução mais lenta da métrica.

Em suma, os resultados obtidos até esta etapa do trabalho mostram o desempenho de cinco algoritmos testados em comparação ao FOSC. A partir desses resultados, foi possível concluir que o FOSC não apresentou melhorias significativas em relação aos principais algoritmos da literatura para o benchmark LFR, mas também não demonstrou piora relevante em comparação com eles.

5.2 Trabalhos futuros

Para trabalhos futuros, como discutido por Le et al. (2017), o *benchmark* LFR apresenta certas características na criação de suas comunidades que não representam de forma fiel o que seria uma comunidade real, principalmente devido à aplicação do parâmetro de mistura de maneira fixa e homogênea para todas as comunidades da rede.

Dessa forma, surgem novas questões em relação ao algoritmo: no *benchmark* LFR, o FOSC mostrou-se competitivo em relação aos principais métodos, porém sem apresentar uma vantagem suficiente que justificasse seu uso. Mas como ele se comportaria em outros benchmarks? Seria possível que o FOSC alcançasse um desempenho superior?

Quanto ao próprio algoritmo FOSC, neste trabalho as etapas iniciais para a geração do dendrograma — que posteriormente seria utilizado como entrada para o FOSC — foram baseadas nos primeiros passos do algoritmo de Ravasz: construção da matriz de similaridade e aplicação de um algoritmo hierárquico (*average linkage*). Vale ressaltar que existem outros algoritmos que também operam sobre dendrogramas, como é o caso do Girvan-Newman. Assim, testar o FOSC com dendrogramas gerados por outros algoritmos constitui uma interessante possibilidade para trabalhos futuros, podendo contribuir para a melhoria dos resultados obtidos pelo método.

5.3 Uitilização de ferramentas de Inteligência Artificial

Neste trabalho, foram utilizadas Inteligências Artificiais para a correção gramatical e de concordância do texto, além da formatação das tabelas apresentadas. As ferramentas utilizadas foram o ChatGPT (para correção gramatical e de concordância) e o DeepSeek (para formatação das tabelas).

Referências

AHN, Y.-Y.; BAGROW, J. P.; LEHMANN, S. Link communities reveal multiscale complexity in networks. *nature*, Nature Publishing Group UK London, v. 466, n. 7307, p. 761–764, 2010.

ANJOS, F. d. A. R. dos; GERTRUDES, J. C.; SANDER, J.; CAMPELLO, R. J. A modularity-based measure for cluster selection from clustering hierarchies. In: SPRINGER. *Data Mining: 16th Australasian Conference, AusDM 2018, Bahrurst, NSW, Australia, November 28–30, 2018, Revised Selected Papers 16*. [S.l.], 2019. p. 253–265.

BAGROW, J. P.; BOLLT, E. M. Local method for detecting communities. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, APS, v. 72, n. 4, p. 046108, 2005.

BALABAN, A. T. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, ACS Publications, v. 25, n. 3, p. 334–343, 1985.

BARABÁSI, A.-L.; PÓSFAI, M. *Network science*. Cambridge: Cambridge University Press, 2016. ISBN 9781107076266 1107076269. Disponível em: <http://barabasi.com/networksciencebook/>.

BECHTEL, J. J.; KELLEY, W. A.; COONS, T. A.; KLEIN, M. G.; SLAGEL, D. D.; PETTY, T. L. Lung cancer detection in patients with airflow obstruction identified in a primary care outpatient practice. *Chest*, Elsevier, v. 127, n. 4, p. 1140–1145, 2005.

BIENKOV, A. Astroturfing: what is it and why does it matter. *The Guardian*, v. 8, p. 2012, 2012.

BLONDEL, V. D.; GUILLAUME, J.-L.; LAMBIOTTE, R.; LEFEBVRE, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, IOP Publishing, v. 2008, n. 10, p. P10008, 2008.

CAMPELLO, R. J.; MOULAVI, D.; ZIMEK, A.; SANDER, J. A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies. *Data Mining and Knowledge Discovery*, Springer, v. 27, p. 344–371, 2013.

CAMPELLO, R. J.; MOULAVI, D.; ZIMEK, A.; SANDER, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, ACM New York, NY, USA, v. 10, n. 1, p. 1–51, 2015.

DEO, N. *Graph theory with applications to engineering and computer science*. [S.l.]: Courier Dover Publications, 2016.

DERÉNYI, I.; PALLA, G.; VICSEK, T. Clique percolation in random networks. *Physical review letters*, APS, v. 94, n. 16, p. 160202, 2005.

EULER, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, p. 128–140, 1741.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

HARARY, F.; NORMAN, R. Z. Graph theory as a mathematical model in social science. University of Michigan, Institute for Social Research Ann Arbor, 1953.

JAIN, A. K.; DUBES, R. C. *Algorithms for clustering data*. [S.l.]: Prentice-Hall, Inc., 1988.

KARATAŞ, A.; ŞAHIN, S. Application areas of community detection: A review. In: IEEE. *2018 International congress on big data, deep learning and fighting cyber terrorism (IBIGDELFT)*. [S.l.], 2018. p. 65–70.

LANCICHINETTI, A.; FORTUNATO, S. Community detection algorithms: a comparative analysis. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, APS, v. 80, n. 5, p. 056117, 2009.

LE, B.-D.; NGUYEN, H. X.; SHEN, H.; FALKNER, N. Glfr: A generalized lfr benchmark for testing community detection algorithms. In: IEEE. *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. [S.l.], 2017. p. 1–9.

LUCE, R. D.; PERRY, A. D. A method of matrix analysis of group structure. *Psychometrika*, Springer, v. 14, n. 2, p. 95–116, 1949.

NEWMAN, M. E. Fast algorithm for detecting community structure in networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, APS, v. 69, n. 6, p. 066133, 2004.

NEWMAN, M. E.; GIRVAN, M. Finding and evaluating community structure in networks. *Physical review E*, APS, v. 69, n. 2, p. 026113, 2004.

PAVLOPOULOS, G. A.; SECRIER, M.; MOSCHOPOULOS, C. N.; SOLDATOS, T. G.; KOSSIDA, S.; AERTS, J.; SCHNEIDER, R.; BAGOS, P. G. Using graph theory to analyze biological networks. *BioData mining*, Springer, v. 4, p. 1–27, 2011.

RAVASZ, E.; SOMERA, A. L.; MONGRU, D. A.; OLTVAI, Z. N.; BARABÁSI, A.-L. Hierarchical organization of modularity in metabolic networks. *science*, American Association for the Advancement of Science, v. 297, n. 5586, p. 1551–1555, 2002.

ROSVALL, M.; BERGSTROM, C. T. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 105, n. 4, p. 1118–1123, 2008.

ROSVALL, M.; BERGSTROM, C. T. Mapping change in large networks. *PloS one*, Public Library of Science San Francisco, USA, v. 5, n. 1, p. e8694, 2010.

SARVARI, H.; ABOZINADAH, E.; MBAZIIRA, A.; MCCOY, D. Constructing and analyzing criminal networks. In: IEEE. *2014 IEEE security and privacy workshops*. [S.l.], 2014. p. 84–91.

SILVA, P. H. O. d. Um comparativo entre a modularidade knn e a medida de estabilidade para extração ótima de grupos a partir de hierarquia de grupos. 2024.

YANG, Z.; ALGESHEIMER, R.; TESSONE, C. J. A comparative analysis of community detection algorithms on artificial networks. *Scientific reports*, Nature Publishing Group UK London, v. 6, n. 1, p. 30750, 2016.