Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Carlos Pereira Lopes Filho

Agrupamento Profundo de Grafos Usando Redes Neurais de Grafos e Seleção de Sementes

São Carlos
2025

Carlos Pereira Lopes Filho

Agrupamento Profundo de Grafos Usando Redes Neurais de Grafos e Seleção de Sementes

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Alan Demétrius Baria Valejo

São Carlos
2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Carlos Pereira Lopes Filho, realizada em 04/06/2025.

Comissão Julgadora:

Prof. Dr. Alan Demétrius Baria Valejo (UFSCar)

Prof. Dr. Murilo Coelho Naldi (UFSCar)

Prof. Dr. Ricardo Marcondes Marcacini (USP)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Resumo

Tarefas de detecção de comunidades (agrupamento, do inglês *clustering*) desempenham um papel fundamental em grafos com atributos, os quais incorporam tanto a estrutura topológica quanto os atributos dos nós, sendo representados por vetores de características. Métodos de detecção de comunidades baseados em Redes Neurais em Grafos (*Graph Neural Networks* - (GNNs)) têm se mostrado eficazes na extração de padrões a partir desses dados. A maioria das abordagens existentes utiliza um algoritmo de agrupamento tradicional para identificar elementos representativos, os quais são posteriormente empregados no treinamento da GNN e na tarefa de detecção de comunidades. Entretanto, ao selecionar esses elementos, tais algoritmos consideram apenas o vetor de características de cada nó, negligenciando a informação topológica do grafo, o que impacta negativamente o processo de aprendizado da GNN. Para solucionar essa questão, propomos o *Deep Graph Clustering via Seed Selection* (DGCSS), um modelo composto por três módulos: (1) o módulo de seleção de sementes, que identifica os nós representativos; (2) o módulo de *embedding*, que utiliza mecanismos de Atenção em Grafos (*Graph Attention*) para capturar informações topológicas globais; e (3) o módulo de auto-supervisão, que utiliza os nós representativos para orientar a tarefa de detecção de comunidades. Uma vantagem do nosso algoritmo é que ele integra, em todos os módulos, tanto a estrutura topológica quanto os atributos dos nós para identificar os elementos representativos. Este é o primeiro algoritmo de detecção de comunidades baseado em GNN que incorpora a seleção de sementes, estabelecendo uma referência significativa para pesquisas futuras. A análise empírica realizada em grafos do mundo real demonstra que o uso de sementes é competitivo quando comparado a algoritmos tradicionais, como o K-Means combinado com GNNs.

Palavras-chave: Agrupamento, Seleção de Sementes, Grafos, Redes Neurais para Grafos.

Abstract

Clustering plays a fundamental role in attributed graphs, which incorporate both topological structure and node attributes represented as feature vectors. Deep clustering methods based on Graph Neural Networks (GNNs) have proven effective in extracting patterns from such data. Most existing approaches use a traditional clustering algorithm to identify representative elements, which are later employed in the training of the GNN and the clustering task. However, when selecting representative elements, these clustering algorithms consider only the feature vector of each instance, neglecting topological information. This limitation negatively impacts the GNN learning process. To address this issue, we propose Deep Graph Clustering via Graph Neural Network and Seed Selection (DGCSS), a model consisting of three modules: (1) the seed selection module, which identifies representative nodes; (2) the embedding module, which employs a graph attentional network to capture global topological information; and (3) the self-supervised module, which leverages the representative nodes to guide the clustering task. An advantage of our algorithm is that it integrates both the topological structure and node attributes across all modules to identify representative elements. This is the first GNN-based clustering algorithm that incorporates seed selection, establishing a significant reference for future research. The empirical analysis of real-world graphs provides evidence that the use of seeds is competitive when compared to traditional algorithms, such as K-Means combined with GNNs.

Keywords: Graph Clustering, Seed Selection, Graphs, Graph Neural Networks.

Lista de ilustrações

Lista de tabelas

Sumário

Capítulo 1

Introdução

Com o desenvolvimento de plataformas em rede, os grafos se tornaram ferramentas fundamentais para a representação de dados (XIA; SILVA; DOE, 2021). Assim, os grafos têm sido utilizados em diversos problemas reais para formular relações em redes sociais (NEWMAN; WATTS; STROGATZ, 2002), moléculas (GUO; CHEN; KARYPIS, 2021), tráfego de veículos (XIA; WANG; RODRIGUES, 2020), redes de citações (LIU; ZHANG; WANG, 2020), redes de comunicações (LIU; WANG; ZHANG, 2022), entre outros.

Além disso, atualmente, há um interesse crescente na modelagem desses sistemas reais por meio da apresentação de grafos com atributos, ou seja, grafos cujos nós incorporam características adicionais. Esses grafos podem ser analisados usando algoritmos de Aprendizado de Máquina (*Machine Learning)* para inúmeras tarefas. Nesse cenário, a tarefa de detecção de comunidades em grafos (comumente também referenciada como agrupamento de nós em grafos ou simplesmente agrupamento) tem sido alvo de intensa pesquisa em diversas áreas, incluindo análise de redes sociais (GIRVAN; NEWMAN, 2002), bioinformática (EISEN et al., 1998) e sistemas de recomendação (LINDEN; SMITH; YORK, 2003; JAIN, 2010). Neste trabalho, iremos mencionar este tipo de problema apenas como "detecção de comunidades".

Nos últimos anos, as Redes Neurais de Grafos (*Graph Neural Netoworks* - GNNs) ganharam destaque na comunidade científica (WU et al., 2021b) devido à sua capacidade de lidar com diversos paradigmas de aprendizagem. Embora as GNNs tenham sido tradicionalmente empregadas para classificação semissupervisionada de nós, elas estão sendo cada vez mais utilizadas também para detecção de comunidades, com o objetivo de agrupar nós semelhantes entre si e diferentes de nós em outros grupos (ou outras comunidades) (WU et al., 2021b).

Métodos de detecção de comunidades baseados em GNN são recentes e podem ser

amplamente categorizados em Modelos de Embedding de Grafos e Modelos Profundos (*Deep Models*) (LI; ZHU, 2024). Os métodos baseados em *embedding* geralmente seguem um processo em duas etapas. Na primeira, *Modelos de Embedding de Grafos*, uma Rede Neural de Grafos (GNN) é usada para aprender representações latentes de nós (comumente conhecidas como *Embeddings* de nós) que capturam informações estruturais e de atributos. Em seguida, esses *Embeddings* servem como entrada para um algoritmo de agrupamento externo, como o K-Means. Embora seja eficaz, essa abordagem separa o aprendizado de representação do processo de detecção de comunidades, fazendo com que os *Embeddings* sejam aprendidos sem considerar diretamente o objetivo de detecção de comunidades. Como limitação, o modelo não pode refinar os *Embeddings* para melhorar a qualidade do processo de detecção de comunidades, limitando o potencial de soluções verdadeiramente completas (ZHANG et al., 2023; JIAO; LI, 2025).

Em contraste, as abordagens de Modelos Profundos buscam incorporar uma função de perda que otimiza diretamente o processo de detecção de comunidades. Por exemplo, (MORADAN et al., 2023) otimiza a modularidade, uma conhecida função de qualidade de grafos. No entanto, conforme discutido em (FORTUNATO; BARTHéLEMY, 2007), medidas baseadas em modularidade falham em detectar comunidades de certos tamanhos. Além disso, definir uma função de perda efetiva para esse propósito continua sendo um grande desafio. Uma solução comum é empregar inicialmente um algoritmo de agrupamento externo para identificar um conjunto de nós representativos, que são então usados para treinar uma GNN, como no algoritmo DNENC (*Deep neighbor-aware embedding for node clustering*) proposto em (WANG et al., 2022). Essa estratégia elimina a necessidade de uma função de perda complexa, pois os nós selecionados naturalmente orientam o modelo GNN na formação de grupos. Especificamente, esses nós atuam como pontos de referência (denominados centróides), auxiliando o modelo GNN a diferenciar comunidades e a organizar nós semelhantes de acordo. Entretanto, uma limitação dessa estratégia é que o algoritmo de agrupamento externo considera apenas informações de atributos para selecionar nós representativos, negligenciando a informação topológica do grafo.

Para superar essas limitações, este estudo apresenta o **DGCSD** (*Deep Graph Clustering with Seed Detection*), um *framework* composto por três módulos principais. O primeiro módulo, o **Módulo de Detecção de Sementes**, identifica nós representativos usando um algoritmo de detecção de sementes que aproveita a estrutura topológica do grafo, incorporando métricas como Centralidade de Intermediação (*Betweenness Centrality* - BC), Centralidade de Proximidade (*Closeness Centrality* - CC) e Inicialização Aleatória (*Random Seeds*). O segundo módulo, o **Módulo de Embedding**, emprega um Autoencoder de Grafos (GAE) (KIPF; WELLING, 2016b) com um codificador que captura efetivamente as relações entre nós centrais (YANG et al., 2023), enquanto um decodificador reconstrói a estrutura do grafo. Por fim, o **Módulo de Semi-Supervisão** integra uma função de divergência de Kullback-Leibler (KL) para quantificar a discrepân-

cia entre as distribuições previstas e alvo, otimizando-a conjuntamente com a perda de reconstrução. Esse componente aprimora os *Embeddings* aprendidos, tornando-os mais adequados para detecção de comunidades.

1.1 Objetivos

O principal objetivo deste trabalho é avaliar a viabilidade e os benefícios da combinação de mecanismos de seleção de sementes com Redes Neurais em Grafos (GNN) em tarefas de detecção de comunidades, buscando investigar se a integração desses dois métodos pode oferecer um desempenho superior,

Para atingir esse objetivo principal, os seguintes objetivos específicos foram delineados:

- **Análise da Integração de Métodos:** Investigar a sinergia entre os mecanismos de seleção de sementes e GNN, avaliando como cada componente contribui para a melhoria dos resultados em tarefas de detecção de comunidades.
- **Avaliação de desempenho:** Estabelecer e aplicar métricas de comparação para mensurar o desempenho da abordagem combinada em relação a métodos tradicionais de detecção de comunidades.
- **Formulação e Implementação do Algoritmo DGCSS:** Como resultado desta pesquisa, foi formulado e implementado o algoritmo DGCSS (*Deep Graph Clustering with Seed Selection*), que integra de maneira inovadora os mecanismos de seleção de sementes com a capacidade de aprendizado das GNNs.
- **Validação Experimental:** Conduzir experimentos em conjuntos de dados diversos para avaliar a eficiência do DGCSS em termos de qualidade dos resultados obtidos, robustez e generalização, comparando-o com abordagens já estabelecidas.

Essa estrutura de objetivos visa não apenas comprovar a hipótese de que a combinação de métodos de seleção de sementes com GNN pode melhorar significativamente o desempenho na detecção de comunidades em grafos, mas também estabelecer uma base teórica e prática para futuras pesquisas na área de análise e processamento de grafos.

1.2 Contribuições

Este trabalho apresenta um conjunto de contribuições originais voltadas ao aprimoramento de tarefas de detecção de comunidades com o uso combinado de técnicas de seleção de sementes e arquiteturas de Redes Neurais em Grafos (GNN). Tais contribuições buscam fornecer um arcabouço teórico e prático robusto, capaz de ser reproduzido e estendido a diferentes domínios de aplicação. A seguir, listamos as principais contribuições de forma sistematizada:

1. **Integração de estratégias de seleção de sementes com Redes Neurais de Grafos (GNN):** Uma abordagem unificada que aproveita simultaneamente informações dos atributos dos nós e da topologia do grafo para aprimorar a qualidade dos resultados gerados. Essa integração eleva a capacidade de aprendizado das GNNs em cenários não supervisionados ao direcionar o modelo para regiões mais informativas do grafo. A formulação proposta oferece uma base teórica para sustentar avanços futuros em aprendizado de representação baseado em grafos. Além disso, é flexível e pode ser adaptada a outras tarefas não supervisionadas, como previsão de links e detecção de anomalias em grafos.

2. **Um novo modelo baseado em GNN chamado DGCSS (Deep Graph Clustering with Seed Selection):** O modelo combina estratégias de seleção de sementes com mecanismos de atenção para orientar o processo de aprendizado, promovendo a seleção de nós estruturalmente relevantes. A arquitetura incorpora uma função de perda baseada na divergência de Kullback–Leibler (KL), permitindo que a Rede de Atenção em Grafos (GAT) gere *embeddings* discriminativos e conscientes de detecção de comunidades. Esse design incentiva o modelo a produzir representações latentes que reflitam tanto padrões estruturais locais quanto globais do grafo.

3. **Um framework de código aberto para detecção de comunidades em grafos:** Disponibilizamos um código totalmente documentado e extensível que implementa a metodologia proposta. O *framework* foi projetado para favorecer a reprodutibilidade, a facilidade de experimentação e futuras pesquisas. Ele permite que usuários adaptem ou estendam as estratégias de seleção de sementes, as arquiteturas de GNN e os mecanismos de avaliação, tornando-o um recurso valioso para a comunidade de aprendizado de máquina em grafos.

4. **Análise paramétrica do modelo proposto:** Foram realizadas análises sobre como diferentes configurações afetam o desempenho do modelo, fornecendo uma compreensão clara acerca dos hiperparâmetros, da arquitetura da GNN e da escolha de métricas de avaliação, além de permitir ajustes finos ao cenário de aplicação.

5. **Avaliação de desempenho em comparação com algoritmos consolidados na literatura:** O modelo desenvolvido foi submetido a um conjunto de experimentos empíricos em múltiplos conjuntos de dados, incluindo benchmarks de uso recorrente na área. Os resultados foram confrontados com métodos do estado da arte, validando tanto a eficácia quanto a eficiência da solução proposta.

1.3 Organização do texto

O texto está organizado da seguinte forma:

- ❏ No Capítulo 2 são apresentados conceitos que fundamentam o trabalho, como aprendizado positivo, redes neurais, algoritmos para atualização de pesos da rede, etc;
- ❏ No Capítulo 3 são apresentados os materiais e os métodos, tais quais as abordagens propostas e as ferramentas para testar os algoritmos;
- ❏ No Capítulo 4 são apresentados resultados preliminares e testes paramétricos; e
- ❏ No Capítulo 5 são apresentadas as discussões e trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo estabelece a fundamentação necessária para a compreensão deste trabalho. Na seção 2.1 são apresentados os conceitos fundamentais de aprendizado de máquina; na seção 2.2 são abordados os princípios do aprendizado não supervisionado em grafos; e na seção 2.3 são descritas as arquiteturas de redes neurais tradicionais e as específicas para grafos, além do algoritmo *back-propagation* e dos modelos de GNN que serão utilizados.

2.1 Aprendizado de Máquina

A crescente complexidade dos problemas computacionais e o volume exponencial de dados gerados pela sociedade moderna têm impulsionado o desenvolvimento de ferramentas computacionais autônomas, capazes de resolver tarefas sem a intervenção de seres humanos (FACELI et al., 2011). Pode-se definir o Aprendizado de Máquina (AM) como uma subárea da Inteligência Artificial (IA) que se dedica à criação de modelos capazes de aprender a partir da experiência (MITCHELL, 1997).

Segundo a definição apresentada em Mitchell (1997), o AM consiste em um programa de computador que aprende a partir de uma experiência **E**, com respeito a uma classe de tarefas **T** e uma medida de performance **P**, de modo que a performance na tarefa **T**, avaliada por **P**, melhore à medida que a experiência **E** se acumula.

Deve-se também considerar a aprendizagem indutiva, que é uma abordagem amplamente utilizada no campo do Aprendizado de Máquina, onde o objetivo é inferir regras ou padrões com base em conjuntos de dados previamente observados. Essa abordagem pode ser subdividida em quatro principais tipos: supervisionado, semissupervisionado, não-supervisionado e por reforço (RANI; NAYAK; VYAS, 2015).

- **Aprendizado Supervisionado**: Baseia-se na utilização de um conjunto de dados rotulados, possibilitando que o modelo aprenda a identificar padrões e generalize para novos exemplos (KOTSIANTIS et al., 2007; HAYKIN, 2009). Nesta abordagem, um "professor externo" fornece o conhecimento do ambiente por meio de pares entrada-saída, resultando no treinamento de um algoritmo que mapeia entradas do conjunto X para os rótulos correspondentes em Y (ALPAYDIN, 2014; FACELI et al., 2011).

- **Aprendizado Não Supervisionado**: Neste paradigma, os algoritmos exploram as características intrínsecas dos dados sem o suporte de rótulos explícitos, com o objetivo de identificar estruturas ou agrupamentos naturais (ZHU; GOLDBERG, 2009; BISHOP, 2006). Técnicas como o *K-Means*, análise de componentes principais (PCA) e modelos de mistura (MCLACHLAN; PEEL, 2000) exemplificam essa abordagem, permitindo a extração de informação relevante de dados de alta dimensionalidade.

- **Aprendizado por Reforço**: Aqui, o modelo (ou agente) aprende a tomar decisões interagindo com o ambiente, ajustando suas ações com base em feedbacks recebidos na forma de recompensas ou punições (GARCIA; FERNÁNDEZ, 2015). Esse paradigma elimina a necessidade de um conjunto fixo de dados, pois o agente refina suas estratégias dinâmicas através de um processo iterativo de tentativa e erro.

- **Aprendizado Semissupervisionado**: Este método surge para mitigar a dependência do grande volume de dados rotulados exigidos pelo aprendizado supervisionado. Em cenários semissupervisionados, o conjunto de treinamento consiste em um pequeno número de exemplos rotulados combinado com uma grande quantidade de dados não rotulados (ZHU; GOLDBERG, 2009; SONG et al., 2022). Essa abordagem pode ser subdividida em:

  1. *Aprendizado Indutivo*: Dado um conjunto de treinamento $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$, o algoritmo aprende uma função $f : \mathcal{X} \to \mathcal{Y}$ com o intuito de obter um bom desempenho na classificação de dados futuros.
  2. *Aprendizado Transdutivo*: Considerando dois conjuntos distintos, $\{\mathbf{x}_i, y_i\}_{i=1}^{l}$ e $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, o objetivo é aprender uma função $f : \mathcal{X}^{l+u} \to \mathcal{Y}^{l+u}$ que classifique efetivamente os dados do conjunto não rotulado.

2.2 Aprendizado de Máquina em Grafos

Grafos são utilizados para representar estruturas de dados interconectadas e encontram aplicações em diversas áreas, como sistemas sociais, ecossistemas, redes biológicas

e sistemas de informação. De maneira geral, os modelos de aprendizado aplicados a grafos têm como objetivo converter as características estruturais presentes em um grafo em vetores correspondentes em um espaço latente. Muitos desses algoritmos fazem uso de técnicas de aprendizado profundo (*Deep Learning*). Assim, a representação vetorizada dos grafos pode ser empregada em tarefas como classificação, predição de links ou processos de *embedding* (XIA et al., 2021).

Neste capítulo, serão apresentados os conceitos fundamentais do aprendizado em grafos, bem como os paradigmas de aprendizado não supervisionado aplicados a essas estruturas.

2.2.1 Conceitos Fundamentais

Um grafo é formalmente definido por $G = (V, E, \mathcal{X}, \mathcal{Y})$, onde:

- $V = \{v_i\}_{i=1}^{n}$ é o conjunto de n vértices;
- $E = \{e_{i,j} = (v_i, v_j) \mid i, j = 1, \ldots, n\}$ representa o conjunto de arestas que estabelecem conexões entre os vértices;
- $\mathcal{X} = \{x_i \in \mathbb{R}^m\}_{i=1}^{n}$ é o conjunto de vetores de características associados a cada vértice; e
- $\mathcal{Y} = \{y_i \in \{0, 1\}\}_{i=1}^{n}$ contém os rótulos dos vértices para fins de tarefas classificatórias (WU et al., 2021a).

A estrutura relacional do grafo é comumente representada por uma matriz de adjacência $A \in \mathbb{R}^{n \times n}$, onde

$$A_{i,j} = \begin{cases} 1, & \text{se existe uma aresta entre } v_i \text{ e } v_j, \\ 0, & \text{caso contrário.} \end{cases}$$

Além disso, as características e os rótulos dos vértices podem ser organizados nas matrizes $X \in \mathbb{R}^{n \times m}$ e $Y \in \mathbb{R}^{n \times 1}$, respectivamente.

No contexto de detecção de comunidades em grafos, o objetivo é particionar o conjunto V em subconjuntos disjuntos $\{V_1, V_2, \ldots, V_k\}$ tais que:

$$V = \bigcup_{c=1}^{k} V_c \quad \text{e} \quad V_c \cap V_{c'} = \emptyset, \quad \forall\, c \neq c'.$$

Cada subconjunto V_c representa uma comunidade (ou *cluster*), onde se espera que a densidade de conexões entre os vértices dentro de V_c seja significativamente maior do que a densidade das conexões entre vértices pertencentes a comunidades diferentes.

Mais formalmente, para uma comunidade V_c, podemos definir:

$$\text{Densidade interna: } d_{\text{in}}(V_c) = \frac{2|E(V_c)|}{|V_c|(|V_c| - 1)},$$

onde $E(V_c)$ é o conjunto de arestas com ambas as extremidades em V_c. Em contrapartida, a densidade de conexões entre V_c e o conjunto complementar $V \setminus V_c$ é dada por:

$$d_{\text{out}}(V_c) = \frac{|E(V_c, V \setminus V_c)|}{|V_c|(n - |V_c|)},$$

sendo $E(V_c, V \setminus V_c)$ o conjunto de arestas que conectam vértices de V_c a vértices fora dele.

A identificação de comunidades envolve a procura por partições $\{V_1, V_2, \ldots, V_k\}$ que maximizem o critério de modularidade ou outras métricas que enfatizam a importância de conexões internas fortes em comparação com conexões externas. Essa segmentação do grafo permite tanto a descoberta de estruturas inerentes como a análise detalhada dos padrões de conexão presentes no grafo.

2.2.2 Aprendizado Não Supervisionado em Grafos

Tarefas de aprendizado não supervisionado possuem o objetivo de identificar informações relevantes diretamente dos dados, sem a necessidade de um supervisor externo (FACELI et al., 2011). Esse processo baseia-se na descoberta de propriedades intrínsecas, podendo ser encapsulandas, ou não, em um espaço vetorial de dimensão reduzida (conhecido como *embeddings*), que permitem construir representações versáteis para diferentes aplicações (SOUTO et al., 2003).

2.3 Redes Neurais

Uma Rede Neural Artificial (RNA) é um sistema computacional que simula de forma abstrata os processos de transmissão e processamento de informações observados no cérebro humano (HAYKIN, 2009; FACELI et al., 2011). Nesta seção, exploraremos diferentes arquiteturas, incluindo o *perceptron*, o *multilayer perceptron*, os *AutoEncoders* e as redes baseadas em atenção para grafos (*Graph Attention*), ressaltando suas características e aplicações.

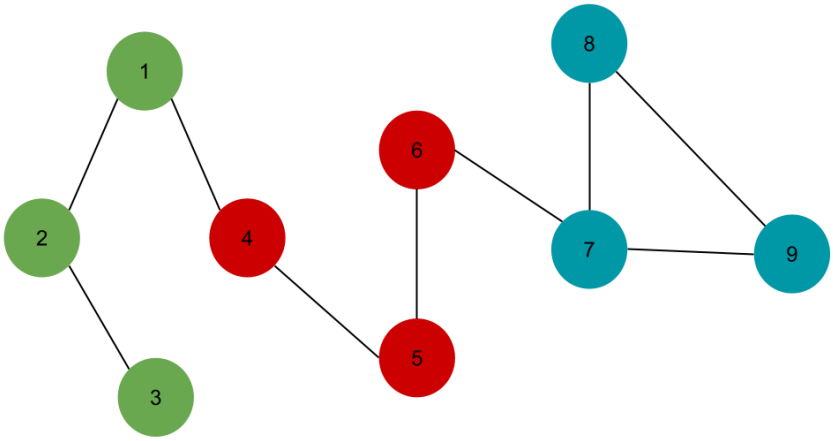
2.3.1 Perceptron

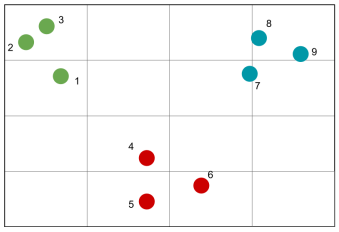
O modelo mais simples de rede neural é o *perceptron* (MCCULLOCH; PITTS, 1943), que é formado por um algoritmo de aprendizado supervisionado utilizado para classificação binária. Essa unidade é conhecida como neurônio artificial e aplica transformações lineares e não lineares nos dados.

Considere o vetor de entrada $x = (x_1, x_2, \ldots, x_m)$ e o vetor de pesos $w = (w_1, w_2, \ldots, w_m) \in \mathbb{R}^m$, além do termo de bias $b \in \mathbb{R}$ e uma função de ativação não linear, denotada por $\varphi(\cdot)$. A saída y do perceptron é calculada pela seguinte expressão:

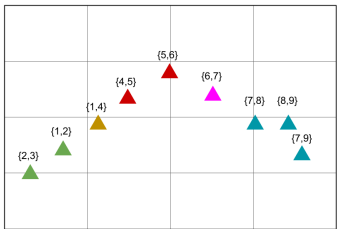
$$y = \varphi\left(\sum_{i=1}^{m} x_i\, w_i + b\right).$$

Figura 1 – Representação do *embedding* de um grafo

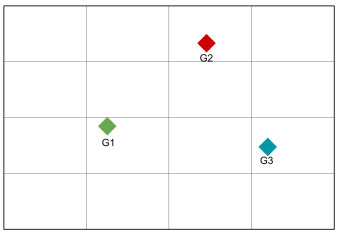




(a) *Embedding* Vértices



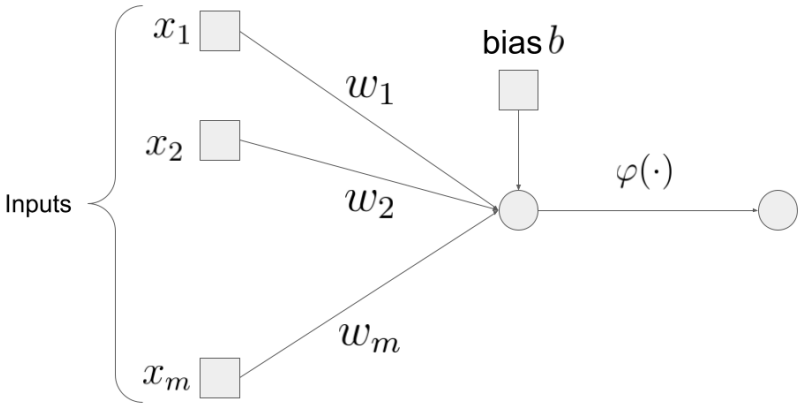
(b) *Embedding* Arestas



(c) *Embedding* Subgrafos

Fonte: (MESSIAS, 2024)

Figura 2 – Estrutura do funcionamento de um perceptron



Fonte: Adaptado de Haykin (2009)

Duas das principais funções de ativação utilizadas são: a função limiar (*threshold*)

$$\varphi(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \tag{1}$$

ou a função sigmoide

$$\varphi(x) = \frac{1}{1 + \exp(-ax)} \tag{2}$$

O perceptron, cujo funcionamento pode ser observado na Figura 2, é um modelo capaz de resolver exclusivamente problemas de classificação binária em que os dados são linearmente separáveis, isto é, onde existe um hiperplano que consegue dividir as classes de maneira clara. Mesmo com essa restrição, o perceptron serviu como fundamento para o desenvolvimento de arquiteturas de processamento mais avançadas, que ampliaram o escopo das aplicações em tarefas de aprendizado (HAYKIN, 2009).

Ao classificar um exemplo incorretamente, o modelo faz a adaptação dos pesos utilizando o seguinte algoritmo (HAYKIN, 2009):

1. Se o n-ésimo vetor de entrada $\mathbf{x}(n)$ é corretamente classificado pelo vetor de pesos $\mathbf{w}(n)$, então nenhuma alteração é feita no vetor $\mathbf{w}(n)$. Ou seja, supondo C_1 como a classe que representa $y = 1$ e C_2 a classe que representa $y = 0$, temos

   $$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ pois } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ para } \mathbf{x}(n) \text{ pertencente a classe } C_1$$

   $$\mathbf{w}(n+1) = \mathbf{w}(n), \text{ pois } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ para } \mathbf{x}(n) \text{ pertencente a classe } C_2,$$

2. Caso contrário, o vetor de pesos do perceptron é atualizado de acordo com a regra

   $$\mathbf{w}(n+1) = \mathbf{w}(n) - \eta(n)\mathbf{x}(n) \text{ se } \mathbf{w}^T(n)\mathbf{x}(n) > 0 \text{ e } \mathbf{x}(n) \text{ pertence a classe } C_2$$

   $$\mathbf{w}(n+1) = \mathbf{w}(n) + \eta(n)\mathbf{x}(n) \text{ se } \mathbf{w}^T(n)\mathbf{x}(n) \leq 0 \text{ e } \mathbf{x}(n) \text{ pertence a classe } C_1,$$

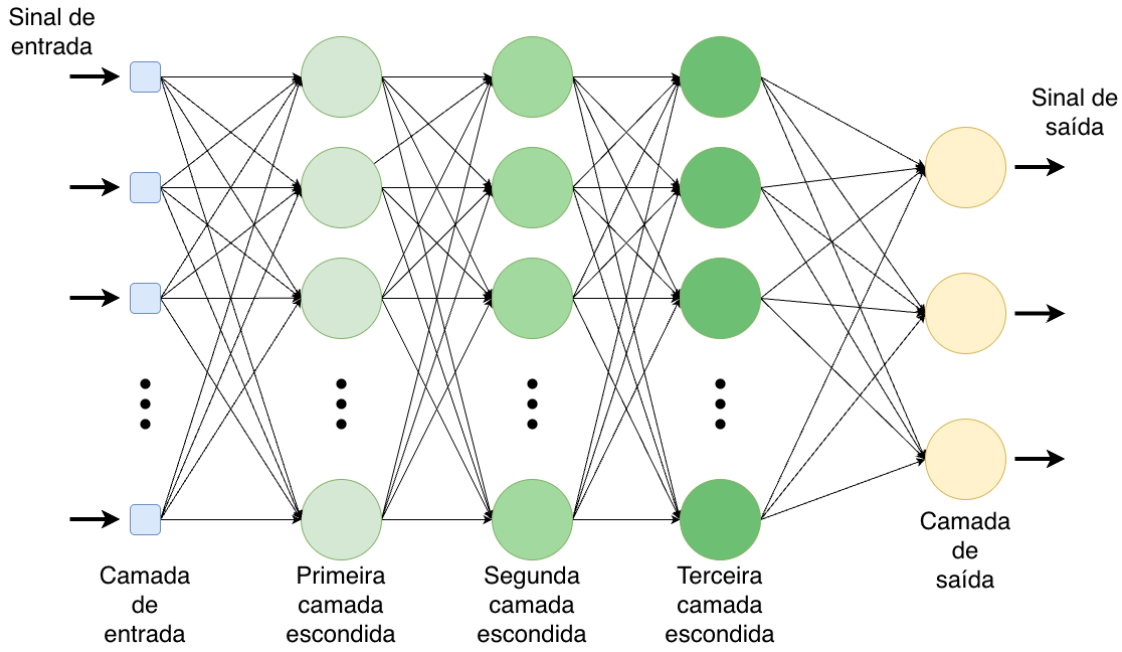
onde $\eta(n)$ representa uma taxa de aprendizado.

2.3.2 Redes Perceptron Multicamadas

Um modelo de rede perceptron multicamada (Multilayer Perceptron - MLP) se diferencia do perceptron simples por incluir camadas ocultas, ou seja, neurônios que não estão diretamente conectados às entradas nem atuam como neurônios de saída. Em cada neurônio de uma MLP, é utilizada uma função de ativação não linear diferenciável, permitindo ao modelo capturar relações complexas nos dados.

Além disso, a MLP conta com um alto grau de conectividade representado pelos pesos sinápticos, possibilitando o processamento eficiente de informações e a resolução de

Figura 3 – Arquitetura de uma Rede Neural Multicamada



Fonte: Produzido pelo autor.

problemas cujos dados não são linearmente separáveis. A Figura 3 ilustra o funcionamento desse tipo de rede.

O objetivo da MLP é aproximar uma função f^*. Por exemplo, para um classificador, $y = f^*(\mathbf{x})$ mapeia uma entrada x para uma categoria y. Uma rede *feedforward* define uma função $\mathbf{y} = f(\mathbf{x}; \theta)$ e aprende os valores dos parâmetros θ que resultam na melhor aproximação (GOODFELLOW; BENGIO; COURVILLE, 2016).

De modo geral, a MLP consiste em uma rede *feedforward* que aceita uma entrada $\mathbf{x}$ e produz uma saída $\hat{y}$. Durante a fase de treino, a propagação produz uma função de custo $j(\theta)$, onde θ são os pesos da rede. O algoritmo *back-propagation* (RUMELHART; HINTON; WILLIAMS, 1986), que será discutido adiante, permite que a função de custo seja retro propagada pela rede para computar o gradiente. Para atualizar os pesos, o algoritmo mais usado é o gradiente descendente (ROBBINS; MONRO, 1951), que utiliza o gradiente da função de custo para atualizar os pesos da rede.

2.3.3 Algoritmo *Back Propagation*

Nesta seção, apresenta-se uma análise matemática completa do algoritmo de *back-propagation* utilizado para ajustar os parâmetros de um *Multilayer Perceptron* (MLP). O método baseia-se no cálculo eficiente dos gradientes do erro em relação a todos os pesos e bias da rede, permitindo a aplicação de técnicas de otimização, como a descida de gradiente.

Suponha que a rede possua L camadas, onde a camada $l = 1$ representa a camada de

entrada, $l = 2, \ldots, L-1$ são as camadas ocultas e $l = L$ é a camada de saída. Para cada camada l, defina:

- $a^{(l)}$: vetor de ativações (saídas) da camada l,
- $z^{(l)}$: vetor resultante da combinação linear dos ativadores, dado por

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)},$$

- $W^{(l)}$ e $b^{(l)}$: matriz de pesos e vetor de bias da camada l, respectivamente,
- $\varphi(\cdot)$: função de ativação, aplicada de maneira elemento a elemento, de modo que

$$a^{(l)} = \varphi(z^{(l)}).$$

O processo de treinamento busca minimizar uma função de custo J que depende da saída da rede e dos rótulos desejados. Um exemplo clássico é o erro quadrático médio:

$$J(W, b) = \frac{1}{2} \|a^{(L)} - y\|^2,$$

onde $a^{(L)}$ é a saída da camada final (rede) e y é o vetor de rótulos.

Propagação para Frente (Forward Pass):

Na etapa de forward pass, calculamos as ativações de cada camada a partir da entrada:

$$\begin{aligned} z^{(l)} &= W^{(l)} a^{(l-1)} + b^{(l)}, \quad l = 2, 3, \ldots, L, \\ a^{(l)} &= \varphi(z^{(l)}). \end{aligned}$$

Dessa forma, obtemos a saída $a^{(L)}$ que é utilizada para a avaliação do erro pela função J.

Propagação para Trás (Backward Pass):

O algoritmo de back-propagation calcula os gradientes $\frac{\partial J}{\partial W^{(l)}}$ e $\frac{\partial J}{\partial b^{(l)}}$ para cada camada l utilizando a regra da cadeia. Definimos o erro local (ou "delta") na camada l como:

$$\delta^{(l)} = \frac{\partial J}{\partial z^{(l)}},$$

que pode ser calculado recursivamente, começando da camada de saída. Para a camada final, considerando a função de custo definida, temos:

$$\delta^{(L)} = (a^{(L)} - y) \circ \varphi'(z^{(L)}),$$

onde "$\circ$" denota o produto elemento a elemento (Hadamard) e $\varphi'(z^{(L)})$ é o vetor dos valores da derivada da função de ativação avaliados em $z^{(L)}$.

Para as camadas $l = L-1, L-2, \ldots, 2$, o erro é retropropagado conforme:

$$\delta^{(l)} = \left((W^{(l+1)})^\top \delta^{(l+1)}\right) \circ \varphi'(z^{(l)}).$$

Esta relação decorre diretamente da aplicação da regra da cadeia sobre a composição das funções.

Os gradientes dos parâmetros são então obtidos por:

$$\frac{\partial J}{\partial W^{(l)}} = \delta^{(l)}(a^{(l-1)})^\top, \quad \frac{\partial J}{\partial b^{(l)}} = \delta^{(l)},$$

para cada camada l. Estes gradientes são usados para atualizar os parâmetros por meio de um algoritmo de otimização (por exemplo, descida de gradiente):

$$W^{(l)} := W^{(l)} - \eta \frac{\partial J}{\partial W^{(l)}}, \quad b^{(l)} := b^{(l)} - \eta \frac{\partial J}{\partial b^{(l)}},$$

onde η é a taxa de aprendizado.

Resumo:

O back-propagation permite ajustar os parâmetros de uma MLP calculando recursivamente os deltas $\delta^{(l)}$ a partir da camada de saída até as camadas de entrada e, a partir destes, derivar as contribuições de cada peso e bias para o erro total. Essa abordagem sistemática, fundamentada na regra da cadeia, é crucial para o treinamento eficiente de redes neurais profundas e para a aprendizagem de representações complexas dos dados.

2.3.4 AutoEncoders

Um *AutoEncoder* (AE) é uma arquitetura de rede neural baseada em MLP treinada com o intuito de reproduzir seus dados de entrada na saída. O treinamento ocorre de maneira não supervisionada, visando extrair uma representação compacta e significativa dos dados. Formalmente, o problema consiste em identificar duas funções: o *encoder* $g_1 : \mathbb{R}^n \to \mathbb{R}^p$ e o *decoder* $g_2 : \mathbb{R}^p \to \mathbb{R}^n$, de forma que a combinação $g_2 \circ g_1$ minimize a discrepância entre a entrada $\mathbf{x}$ e sua reconstrução.

Essa tarefa pode ser expressa matematicamente como

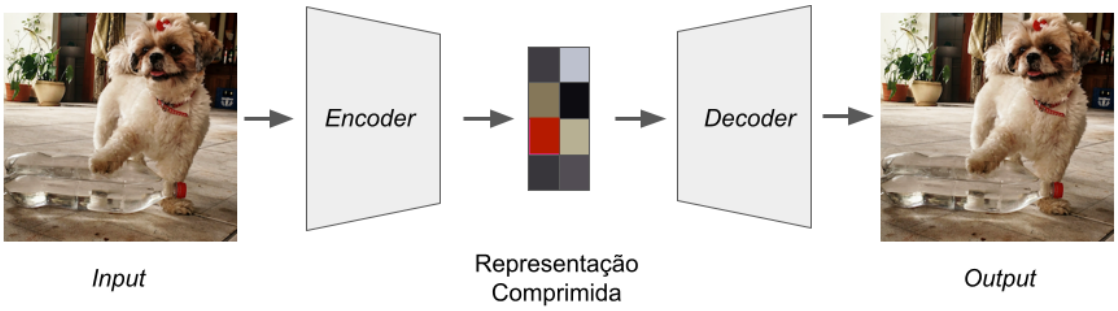
$$\arg\min_{g_1, g_2} E\left[\Delta\Big(\mathbf{x}, g_2 \circ g_1(\mathbf{x})\Big)\right], \tag{3}$$

onde E representa o valor esperado sobre a distribuição das amostras, e Δ é uma função de perda que mede a distância entre o *output* do *decoder* e o *input* (GOODFELLOW; BENGIO; COURVILLE, 2016; BANK; KOENIGSTEIN; GIRYES, 2021). Assim, a saída gerada pelo AutoEncoder é definida por

$$\tilde{\mathbf{x}} = f(\mathbf{x}) = g_2 \circ g_1(\mathbf{x}), \tag{4}$$

como ilustrado na Figura 4.

Figura 4 – AutoEncoder



Fonte: Adaptado de Bank, Koenigstein e Giryes (2021)

Na esquerda é possível verificar uma imagem que, ao passar pela rede, tem sua saída próxima à entrada. A sua versão comprimida é chamada de *bottleneck* e representa uma versão em menor dimensão da imagem. Para os propósitos deste trabalho, tanto a entrada quanto a saída serão de informações referentes aos nós de um grafo.

2.3.5 Redes Neurais em Grafos

Os avanços em redes neurais e técnicas de *deep learning* têm impulsionado expressivos progressos em tarefas de reconhecimento de padrões e mineração de dados. Embora esses algoritmos se destaquem em dados organizados em espaços euclidianos, muitas aplicações lidam com estruturas em forma de grafos, para as quais operações convencionais, como somas, produtos, cálculos de distância e convoluções, não são diretamente aplicáveis. Um desafio adicional está na variabilidade da matriz de adjacência, cuja ordenação pode variar conforme a aplicação, como ilustrado nas Figuras 5 e 6 (WU et al., 2021c).

Para endereçar tais limitações, diversas arquiteturas de redes neurais especializadas para dados em grafos foram desenvolvidas. Essas abordagens podem ser divididas em quatro categorias principais: *Recurrent Graph Neural Networks* (RecGNN), *Spatial Temporal Graph Neural Networks* (STGNN), *Graph AutoEncoders* (GAE) e *Graph Convolutional Networks* (GCN) (WU et al., 2021c). Em todas essas arquiteturas, a ideia central é a de *Transferência de Mensagem* (Message Passing, MP), que permite a cada nó $v \in V$ atualizar sua representação ao combinar seu vetor de características x_v com as informações agregadas dos seus vizinhos $\mathcal{N}(v)$. De forma geral, esse processo pode ser definido pelas seguintes etapas:

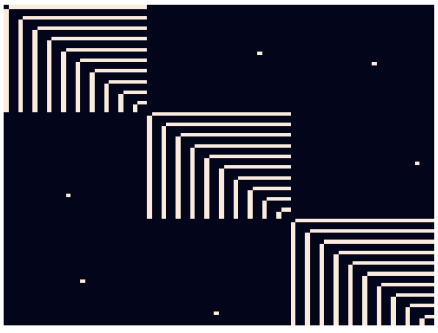
$$\textbf{Agregação:} \quad m_v = \mathrm{AGG}\Big(\{x_u : u \in \mathcal{N}(v)\}\Big),$$

$$\textbf{Atualização:} \quad x'_v = \mathrm{UPT}(x_v, m_v).$$

Essa formulação permite que os nós capturem tanto as informações locais quanto a estrutura topológica do grafo. Cada categoria de GNN explora essas etapas de forma específica:

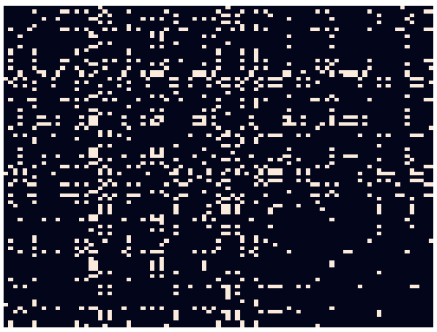
- ***Recurrent Graph Neural Networks***: Iterativamente atualizam as representações dos nós utilizando mecanismos recorrentes até que se atinja um estado de convergência (SCARSELLI et al., 2009).
- ***Spatial Temporal Graph Neural Networks***: Integram variações temporais dos atributos dos nós, considerando simultaneamente a dependência espacial e temporal (YU; YIN; ZHU, 2018).
- ***Graph AutoEncoders***: Empregam técnicas de aprendizado não supervisionado para aprender representações latentes dos nós ou arestas, facilitando tarefas como a geração de grafos e a extração de embeddings (BANK; KOENIGSTEIN; GIRYES, 2021).
- ***Graph Convolutional Networks***: Adaptam o conceito de convolução para grafos, onde a representação de um nó v é construída a partir do seu vetor de características x_v e dos vetores x_u dos nós na sua vizinhança $\mathcal{N}(v)$, conforme descrito em (KIPF; WELLING, 2016a).

Figura 5 – Matriz de adjacência ordenada em comunidades



Fonte: (MESSIAS, 2024)

Figura 6 – Matriz de adjacência ordenada de forma aleatória



Fonte: (MESSIAS, 2024)

2.3.5.1 Transferência de Mensagem

Para a construção de GNNs, o mecanismo de Transferência de Mensagem MP é o *framework* mais utilizado(GILMER et al., 2017).

Considerando um grafo $G = (V, E, X)$, o *framework* faz o *encoding* de cada vértice $v \in V$ com um vetor de representação $\mathbf{h}_v$ e continua atualizando aquela representação

iterativamente, coletando representações de seus vizinhos e aplicando uma camada de rede neural para realizar uma transformação não linear dessas representações.

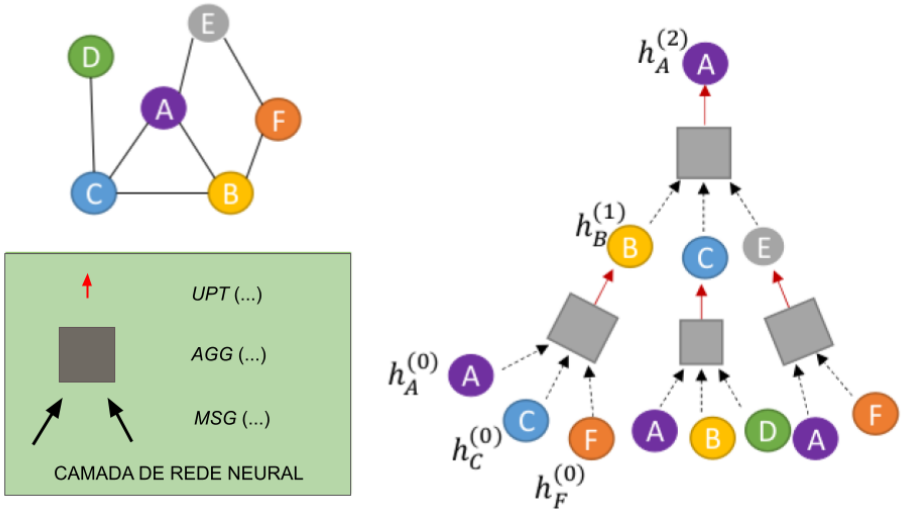
O modelo consiste basicamente em dois passos (WU et al., 2022):

1. Inicializa a representação de um vértice a partir do vetor $\mathbf{h}_v^{(0)} = \mathbf{x}_v, \forall v \in V$
2. Considerando a mensagem transmitida pela estrutura do grafo, atualiza cada vetor de representação. Na l-ésima camada, $l = 1, 2, \cdots, L$, performa os seguintes passos:

$$\begin{aligned} \text{Message:} \quad & \mathbf{m}_{uv}^{(l)} = \text{MSG}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_u^{(l-1)}\right), \forall (u,v) \in E \\ \text{Aggregation:} \quad & \mathbf{a}_v^{(l)} = \text{AGG}\left(\{\mathbf{m}_{uv}^{(l)} | u \in N_v\}\right), \forall v \in V \\ \text{Update:} \quad & \mathbf{h}_v^{(l)} = \text{UPT}(\mathbf{h}_v^{(l-1)}, \mathbf{a}_v^{(l)}), \forall v \in V \end{aligned} \tag{5}$$

Tipicamente, as funções MSG, AGG e UPT são implementadas como redes $feedforward$; entre elas, AGG precisa ser comutativa (WU et al., 2022). A Figura 7 mostra esse fluxo de Transferência de Mensagem para um vértice do grafo.

Figura 7 – Rede Neural em Grafo



Fonte: (WU et al., 2022)

Como todas as operações são diferenciáveis, utiliza-se retropropagação para ajustar os parâmetros das redes (por exemplo, via MSELoss$(\cdot,\cdot)$ ou cross-entropy). Por isso, o modelo de Transferência de Mensagem fornece a base para a maioria dos frameworks de GNN.

2.3.5.2 Frameworks

Para a correta compreensão deste trabalho, devemos dar destaque para os seguintes *frameworks*: Rede Neural Convolucional em Grafos (GCN) (KIPF; WELLING, 2016a),

Attributed Network AutoEncoder (ANAE) (CEN et al., 2020), *Relational Graph Convolutional Networks* (RGCN) (SCHLICHTKRULL et al., 2018). Sendo que a *Graph Convolutional Network* (GCN) foi proposta em Kipf e Welling (2016a) e é uma das mais populares e efetivas *baselines* de redes neurais em grafos (SONG et al., 2021).

Uma GCN pode ser construída considerando a matriz laplaciana e a operação de convolução. Suponha $L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ a matriz laplaciana normalizada de um grafo G onde D é a matriz diagonal dos vértices, $D_{i,i} = \sum_j A_{i,j}$. Essa matriz pode ser fatorada em $L = U\Lambda U^T$, onde $U = [u_0, u_1, \cdots u_{n-1}] \in \mathbb{R}^{n\times n}$ é a matriz dos autovetores ordenada pelos autovalores da matriz diagonal de autovalores Λ. Dado $X \in \mathbb{R}^{n\times m}$ uma matriz de características dos vértices do grafo, uma transformada de Fourier em grafos pode ser definida como $\mathcal{F}(x) = U^T x$, sendo sua inversa $\mathcal{F}^-1(x) = Ux$.

A transformada de Fourier em grafos projeta o grafo de entrada em um espaço vetorial ortonormal onde a base é formada pelos autovetores da matriz laplaciana normalizada. Agora, o sinal X de entrada de um grafo pode ser convolucionada em relação a um filtro $g \in \mathcal{R}^n$ através da equação

$$x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g)) = U(U^T x \odot U^T g), \tag{6}$$

onde $\odot$ representa o produto direto de dois vetores. A diferença principal entre as redes convolucionais encontra-se na escolha do filtro g.

A GCN (KIPF; WELLING, 2016a) utiliza a convolução da seguinte forma:

$$x *_G g_\theta = \theta(I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})x \tag{7}$$

Para habilitar múltiplas entradas de *inputs* e *outputs*, a GCN modifica a Equação 7 da seguinte forma:

$$H = X *_G G_\theta = f(\overline{A}X\theta) \tag{8}$$

onde $\overline{A} = I_n + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ e $f(\cdot)$ é uma função de ativação. Utilizando esse valor de $\overline{A}$ causa instabilidade a GCN. Para lidar com esse problema, é feito uma troca do valor de $\overline{A}$ para $\overline{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ com $\tilde{A} = A + I_n$ e $\tilde{D} = \sum_j \tilde{A}_{ij}$

Dessa forma, a GCN considera um modelo de rede neural $f(X, A)$ com a seguinte regra de propagação:

$$H^{(l+1)} = \varphi\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{9}$$

onde $\tilde{A} = A + I_N$ é a matriz de adjacência de um grafo não direcionado G com arestas de laço e I_N é a matriz identidade. $\tilde{D}_{i,i} = \sum_j \tilde{A}_{ij}$, $W^{(l)}$ é uma matriz de pesos treináveis de uma camada l, $\varphi(\cdot)$ se refere a uma função de ativação, como RELu$(\cdot)$ por exemplo. $H^{(l)} \in \mathbb{R}^{N\times D}$ é a matriz de ativação da l-ésima camada e $H^0 = X$, onde X é a matriz de características dos vértices no grafo.

Para lidar com grafos que representam diferentes relações, Schlichtkrull et al. (2018) apresentaram o modelo de *Ralational Graph Convolutional Network* (RGCN). Ao contrário dos GCNs padrão, que aplicam a mesma transformação a todas as arestas, os R-GCNs aplicam diferentes transformações com base no tipo de relação entre os nós. Isso significa que a mensagem passada de um vértice para outro é modulada pelo tipo específico de aresta (ou relação) que os conecta. A Equação 10 define a agregação em cada camada de uma RGCN

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}\right) \tag{10}$$

onde $\mathcal{N}_i^r$ denota o conjunto de índices de vizinhos do vértice i sob a relação $r \in \mathcal{R}$. $c_{i,r}$ é uma constante de normalização específica do problema que pode ser aprendida ou escolhida antecipadamente (como $c_{i,r} = |\mathcal{N}_i^r|$).

Esse modelo foi desenvolvido para lidar com grafos relacionais. Nesse trabalho, esse modelo será utilizado considerando os diferentes grafos gerados pelo modelo de reescrita.

O modelo do *Graph AutoEncoder* (GAE) (KIPF; WELLING, 2016c) é um modelo que aprende a representação dos dados. A arquitetura busca aprender representações compactas da estrutura de um grafo. Portanto, o GAE não aprende representações dos vértices individuais, apenas da estrutura do grafo.

Para lidar com essa lacuna, Cen et al. (2020) propôs a arquitetura *Attributed Network AutoEncoder* (ANAE), que consiste em um *framework* criado para lidar com problemas sobre grafos com características nos vértices. O objetivo da ANAE é aprender o conteúdo de um nó para um grafo com atributos, levando em consideração também a estrutura do grafo.

Similar ao modelo ANAE, Ng et al. (2019) propõe o uso da regra de propagação da Equação 9 na saída do *encoder* de um *AutoEncoder*, aplicando o algoritmo MP na representação latente. Nesse trabalho, consideraremos uma arquitetura *Graph AutoEncoder* (GAE) como o modelo apresentado em Ng et al. (2019).

Considerando X como a matriz de características formada pelos elementos $\mathbf{x}$, a Equação 4 pode ser escrita como

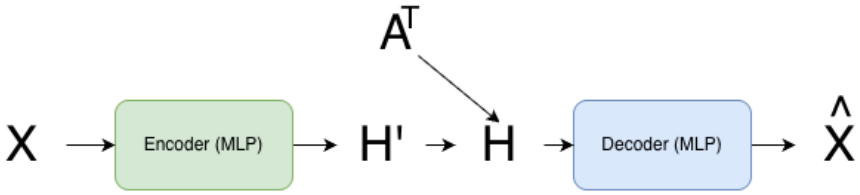
$$\tilde{X} = f(X, A) = g_2\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}g_1(X)\right) \tag{11}$$

Assim, o algoritmo consiste em aprender funções g_1 e g_2 que minimizem a expectativa do erro Δ, conforme a Equação 12:

$$\arg\min_{g_1,g_2} E\left[\Delta\left(X, g_2\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}g_1(X)\right)\right)\right] \tag{12}$$

A Figura 8 apresenta o esquema de funcionamento do *Graph AutoEncoder*

Figura 8 – Esquematização de um GAE



Fonte: Adaptado de Ng et al. (2019)

Dessa forma, a rede é capaz de aprender a representação dos nós considerando a estrutura topológica do grafo, ou seja, dos vizinhos diretos de um nó alvo.

2.3.5.3 Mecanismos de Atenção em Grafos

O mecanismo de atenção (*Attention*) emergiu inicialmente no contexto de modelos sequenciais para processamento de linguagem natural, como descrito em (VASWANI et al., 2017), e desde então tem se mostrado uma ferramenta poderosa para a modelagem de relações contextuais em dados de alta dimensionalidade. Em sua essência, o mecanismo de atenção permite que o modelo aprenda a ponderar de forma diferenciada as contribuições de diferentes entradas, destacando aquelas que são mais relevantes para a tarefa em questão.

Quando adaptado para o contexto de grafos, o mecanismo de atenção permite uma agregação ponderada das informações dos nós vizinhos, o que culminou no desenvolvimento dos *Graph Attention Networks* (GATs) (VELIčKOVIć et al., 2018). Em métodos tradicionais de agregação em grafos, como a média ou a soma simples, todos os vizinhos contribuem de forma uniforme para a representação de um nó. No entanto, essa abordagem ignora a heterogeneidade inerente às relações entre os nós, onde alguns podem fornecer informações muito mais relevantes do que outros.

A formulação básica do mecanismo de atenção aplicado a grafos consiste em calcular uma representação atualizada $\mathbf{h}'_i$ para cada nó i como uma combinação ponderada das representações dos seus vizinhos $j \in \mathcal{N}_i$:

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right), \tag{13}$$

onde:

- $\mathbf{h}_j$ denota o vetor de características do nó j;
- $\mathbf{W}$ é uma matriz de transformação aprendível que projeta as características para um novo espaço;
- $\sigma(\cdot)$ representa uma função de ativação não-linear, como ReLU ou LeakyReLU;

- α_{ij} é o coeficiente de atenção, que reflete a importância relativa do nó j na atualização da representação do nó i.

O coeficiente α_{ij} é calculado utilizando uma função de similaridade, seguida de uma normalização via *softmax* para garantir que a soma dos pesos dos vizinhos seja igual a 1. Uma forma comum de definir α_{ij} é:

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T \left[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_k\right]\right)\right)}, \tag{14}$$

onde:

- $\mathbf{a}$ é um vetor de parâmetros aprendíveis que atua como um *query* na avaliação da relevância dos pares de nós;
- $[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j]$ denota a concatenação dos vetores transformados dos nós i e j;
- A função LeakyReLU$(\cdot)$ introduz uma não-linearidade que ajuda na mitigação do problema de gradientes nulos.

A normalização *softmax* aplicada aos coeficientes α_{ij} não apenas estabiliza o treinamento, mas também permite uma interpretação probabilística da importância relativa dos vizinhos. Essa capacidade de diferenciar a contribuição dos nós vizinhos é crucial para tarefas como classificação de nós, predição de links e outras aplicações onde a estrutura do grafo desempenha um papel determinante.

Adicionalmente, a técnica de *multi-head attention* tem sido amplamente utilizada para aumentar a expressividade dos modelos (VASWANI et al., 2017). Em vez de utilizar um único mecanismo de *attention*, múltiplas cabeças independentes são empregadas para capturar diferentes aspectos das relações entre os nós. Matematicamente, a aplicação de K cabeças de atenção pode ser expressa como:

$$\mathbf{h}'_i = \Big\|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(k)} \mathbf{W}^{(k)} \mathbf{h}_j\right), \tag{15}$$

onde:

- $\alpha_{ij}^{(k)}$ e $\mathbf{W}^{(k)}$ são, respectivamente, os coeficientes de atenção e a matriz de transformação associados à k-ésima cabeça;
- $\Big\|$ denota a operação de concatenação das representações geradas por cada cabeça.

Essa abordagem permite que o modelo capte uma diversidade de padrões de interação, uma vez que cada cabeça pode focar em diferentes aspectos da vizinhança do nó, resultando em representações mais robustas e discriminativas.

Em síntese, o emprego do mecanismo de atenção em grafos, e sua extensão para *Graph Attention Networks*, representa um avanço significativo na modelagem de dados

relacionais. Essa técnica não só melhora a capacidade de representar a heterogeneidade das relações entre nós, mas também contribui para a interpretabilidade dos modelos, uma vez que os coeficientes de atenção oferecem informações sobre a importância relativa das conexões na rede. Tais avanços têm impulsionado o desempenho em uma variedade de tarefas e continuam a ser objeto de intensas pesquisas na comunidade acadêmica e industrial.

2.4 Detecção de comunidades em Grafos

A análise de grafos frequentemente envolve tarefas de detecção de comunidades, que são grupos de nós que possuem conexões intensas entre si e relativamente escassas com o restante da rede. Essas estruturas, também chamadas de comunidades (ou *clusters*), podem refletir padrões funcionais, sociais ou organizacionais presentes no grafo (FORTUNATO, 2010).

De forma formal, considere um grafo $G = (V, E)$, onde V é o conjunto de nós e E o conjunto de arestas. O problema de detecção de comunidades em grafos consiste em particionar V em subconjuntos $\{V_1, V_2, \ldots, V_k\}$ de modo que a medida de modularidade Q seja maximizada. Essa modularidade é dada por:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{16}$$

onde:

- A_{ij} representa o elemento (i, j) da matriz de adjacência, valendo 1 se houver uma aresta entre os nós i e j e 0 caso contrário;
- k_i e k_j denotam os graus dos nós i e j, respectivamente;
- m é o número total de arestas do grafo;
- c_i e c_j são os identificadores das comunidades a que pertencem os nós i e j;
- $\delta(c_i, c_j)$ é a função delta de Kronecker, que assume o valor 1 quando $c_i = c_j$ e 0 se $c_i \neq c_j$.

A formulação acima, inicialmente proposta em (NEWMAN, 2006), implica que partições com valores elevados de modularidade evidenciam uma maior densidade de conexões internas às comunidades do que o esperado por acaso.

Para resolver esse problema, diversos métodos têm sido propostos, abrangendo desde algoritmos de otimização direta da modularidade (NEWMAN, 2004), passando por técnicas espectrais (LUXBURG, 2007), até modelos estatísticos probabilísticos e abordagens inovadoras baseadas em Redes Neurais em Grafos (*Graph Neural Networks - GNNs*) (KIPF; WELLING, 2017a).

Um dos métodos clássicos é o algoritmo de Louvain, que adota uma estratégia iterativa e hierárquica para maximizar a modularidade (BLONDEL et al., 2008). Inicialmente, cada nó é considerado como uma comunidade separada. Em seguida, o algoritmo avalia a possibilidade de realocar nós para comunidades vizinhas, de forma que a modularidade global seja incrementada. A variação da modularidade decorrente da movimentação de um nó i para uma comunidade vizinha pode ser calculada por:

$$\Delta Q = \left[\frac{\Sigma_{in} + k_{i,in}}{2m} - \left(\frac{\Sigma_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{in}}{2m} - \left(\frac{\Sigma_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right] \tag{17}$$

onde:

- Σ_{in} é a soma dos pesos das arestas que interligam os nós dentro da comunidade alvo;
- Σ_{tot} representa a soma dos pesos das arestas incidentes em todos os nós da comunidade alvo;
- $k_{i,in}$ é a soma dos pesos das arestas que conectam o nó i aos nós da comunidade alvo;
- k_i é o grau do nó i;
- m continua representando o número total de arestas.

Além da maximização da modularidade, outras métricas e técnicas são utilizadas na literatura, como a minimização da condutância ou métodos espectrais baseados na decomposição da matriz Laplaciana. A matriz Laplaciana L é definida por:

$$L = D - A \tag{18}$$

na qual D é a matriz diagonal contendo os graus dos nós e A é a matriz de adjacência do grafo (CHUNG, 1997).

Os métodos espectrais tiram proveito das propriedades dos autovalores e autovetores da matriz Laplaciana, associando autovalores próximos de zero à existência de comunidades bem definidas no grafo (LUXBURG, 2007).

Mais recentemente, o uso de modelos de aprendizado de máquina, e em particular das GNNs, tem expandido as possibilidades para desafios de detecção de comunidades em grafos (KIPF; WELLING, 2017b). Tais abordagens permitem a incorporação de informações adicionais sobre os nós e arestas, bem como a exploração de estruturas locais e globais complexas, utilizando técnicas como *Attention Networks* (VELIčKOVIć et al., 2018), que são apresentadas na seção 2.3.5.3.

2.5 Mecanismos de seleção de sementes em Grafos

A seleção de sementes em grafos consiste em identificar k nós representativos que servirão como pontos âncora, ou centróides, em estratégias de detecção de comunidades em grafos e extração de informações estruturais. Matematicamente, esse processo pode ser descrito em duas etapas principais:

1. **Construção do Ranking dos Vértices:** Inicialmente, define-se uma função de importância $f : V \to \mathbb{R}$, onde V é o conjunto de vértices do grafo. Essa função pode ser baseada em diversas medidas de centralidade, como grau, centralidade de intermediação ou centralidade de proximidade. Após a aplicação dessa função, gera-se um ranking dos nós ordenado de forma decrescente, ou seja, os vértices com maiores valores de f são considerados mais representativos.

2. **Seleção dos k Nós Representativos:** A partir do ranking obtido, seleciona-se os k vértices superiores, onde k pode ser definido conforme a aplicação ou por meio de alguma heurística. Esses nós selecionados são considerados sementes e podem ser utilizados como ponto de partida para algoritmos de detecção de comunidades em grafos. Essa seleção garante que os pontos de partida para a execução do algoritmo reflitam de maneira adequada as propriedades estruturais do grafo (FORTUNATO, 2010).

A escolha adequada das sementes é fundamental, pois ela influencia decisivamente a qualidade das comunidades (ou *clusters*) formadas, impactando a capacidade do algoritmo em capturar corretamente as comunidades subjacentes no grafo.

Uma das abordagens clássicas para a identificação de sementes é o uso de medidas de centralidade, que quantificam a importância ou influência de um nó dentro da topologia do grafo. Entre essas medidas, destacam-se a **Centralidade de Intermediação** e a **Centralidade de Proximidade**.

2.5.1 Centralidade de Intermediação

A centralidade de intermediação (*Betweenness Centrality*) avalia o papel de um nó como intermediário na transmissão de informações entre pares de nós. Formalmente, a centralidade de intermediação $C_B(v)$ de um nó v é definida por:

$$C_B(v) = \sum_{\substack{s,t \in V \\ s \neq v \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{19}$$

onde σ_{st} representa o número total de caminhos mínimos entre os nós s e t, e $\sigma_{st}(v)$ é o número desses caminhos que passam por v (BRANDES, 2001). Esta métrica evidencia nós que, por estarem estrategicamente posicionados, podem influenciar fortemente a dinâmica e a comunicação do grafo, tornando-os candidatos naturais para a função de sementes.

2.5.2 Centralidade de Proximidade

A centralidade de proximidade (*Closeness Centrality*), por sua vez, mede o quão próximos um nó está de todos os demais nós do grafo. Ela é calculada como:

$$C_C(v) = \frac{1}{\sum_{u \in V} d(v,u)}, \tag{20}$$

onde $d(v,u)$ representa a distância mínima entre os nós v e u (SABIDUSSI, 1966). Nós com alta centralidade de proximidade possuem, em média, uma menor distância para os demais nós, sendo assim indicados como sementes capazes de refletir a conectividade global do grafo.

2.5.3 Seleção Aleatória de Sementes

Além dos métodos baseados em centralidade, a seleção aleatória (*Random Selection*) de nós pode ser empregada como uma estratégia de *baseline* ou em cenários onde a topologia não apresenta distinções marcantes. Embora menos informativa em termos de estrutura, essa abordagem possibilita a comparação do desempenho dos métodos baseados em centralidade com uma estratégia não dirigida, auxiliando na validação dos algoritmos propostos.

2.6 Revisão da Literatura

A detecção de comunidades em grafos é uma área que se dedica à segmentação de grafo em comunidades (ou grupos de nós), onde cada comunidade é formada por nós que compartilham características ou interações similares. Esses algoritmos têm como principal objetivo explorar tanto os atributos dos nós quanto as relações estruturais definidas pelas arestas, permitindo a descoberta de padrões ocultos e a redução da complexidade dos dados. Em outras palavras, o intuito é obter representações mais compactas e informativas que permitam identificar com precisão agrupamentos baseados em critérios como similaridade, densidade ou modularidade.

A evolução dos métodos de detecção de comunidades em grafos tem sido significativamente impulsionada pela incorporação de Graph Neural Networks (GNNs). Recentes avanços (GUO; DAI, 2021; LI; ZHU, 2024; WANG et al., 2022; WU et al., 2021b) mostram que as GNNs permitem uma extração eficiente das complexas relações topológicas dos grafos, integrando o aprendizado dos atributos dos nós com a estrutura relacional. Essa integração possibilita não apenas o aprimoramento na qualidade e coerência das comunidades, mas também a criação de frameworks onde a aprendizagem de representações e a otimização do processo de detecção de comunidades em grafos ocorrem de forma conjunta.

Nesta seção, serão exploradas as principais abordagens que se utilizam de técnicas clássicas e modernas para tarefas de detecção de comunidades em grafos, destacando como os algoritmos atuais aproveitam a capacidade das GNNs para superar desafios tradicionais e alcançar uma segmentação mais robusta e representativa dos grafos.

2.6.1 Modelos de *Graph Embedding*

Os modelos de *embedding* têm como objetivo aprender representações latentes dos nós que preservem tanto as informações dos atributos quanto as relações topológicas do grafo (CAI; ZHENG; CHANG, 2018). Tais abordagens frequentemente utilizam arquiteturas do tipo *Graph Autoencoder* (GAE), que operam de maneira não supervisionada para extrair as características intrínsecas dos grafos. Uma vez aprendidas, essas representações são geralmente submetidas a algoritmos de agrupamento em grafos em espaço vetorial, como o *K-Means* ou o *Spectral Clustering*, para a formação de comunidades relevantes.

Entre os desafios enfrentados, destaca-se o problema do *over-squashing*, que surge quando a compressão das informações topológicas resulta na perda de detalhes essenciais sobre a conectividade dos nós. Para mitigar essa dificuldade, alguns métodos propõem intervenções na estrutura original do grafo, como a remoção, adição ou modificação de arestas. Por exemplo, o DGEN (*Dual Graph Embedding Network*) (WANG et al., 2024) emprega operações de *pooling* em vizinhanças para selecionar nós centrais, enquanto outras abordagens ajustam os pesos das arestas utilizando algoritmos inspirados no *PageRank* (ZHANG et al., 2023) para aprimorar a detecção de comunidades em grafos.

Outra vertente importante consiste na incorporação de estratégias baseadas em Redes Gerativas Adversariais (GANs). No ARGA (*Adversarially Regularized Graph Autoencoder*) (PAN et al., 2020), o uso de técnicas adversariais é fundamental para moldar a distribuição latente, contribuindo para a preservação das propriedades estatísticas dos dados. Estratégias semelhantes foram adotadas por Jin et al. (2019), que incorporam *capsule learning* para a geração de *embeddings* mais robustos. Apesar desses avanços, a preservação da integridade dos dados, especialmente no caso de nós isolados, continua sendo um desafio, o que motivou propostas como a apresentada por (SHEN et al., 2023), que desenvolve uma arquitetura em duas partes: um módulo residual para reduzir a perda de informação e um módulo de normalização específico para os nós isolados.

Adicionalmente, novos enfoques têm sido propostos visando superar as limitações dos tradicionais GAEs e suas variantes, como as *Variational Graph Autoencoders* (VGAE). Modelos recentes, por exemplo, empregam técnicas lineares para construir representações latentes com maior escalabilidade (SALHA; HENNEQUIN; VAZIRGIANNIS, 2021). Outros esforços combinam filtros Laplacianos com redes perceptron multicamadas (MLP) para combater o fenômeno da sobre-suavização (*over-smoothing*), conforme exemplificado pelo GLASS (SUN et al., 2023). Além disso, mecanismos de atenção têm sido incorporados para codificar a centralidade dos nós, permitindo uma diferenciação mais acurada

entre as contribuições dos vizinhos, como ilustrado por (YANG et al., 2023). Apesar do sucesso desses métodos, a separação entre o processo de aprendizagem das representações e o algoritmo de agrupamento frequentemente resulta em uma integração subótima, limitando o desempenho final na definição das comunidades.

2.6.2 Modelos de aprendizado profundo

Em contraste com os modelos de *embedding*, os modelos de aprendizado profundo (*Deep Models*) buscam uma integração mais íntima entre a aprendizagem das representações latentes e o processo de detecção de comunidades em grafos. Esses métodos podem ser divididos em duas estratégias principais: (i) a otimização direta de uma medida de qualidade da formação das comunidades através da definição de funções-objetivo que incorporam o conceito de detecção de comunidades em grafos, e (ii) a utilização de algoritmos externos para identificar nós representativos que guiam a formação das comunidades, auxiliando o treinamento posterior de redes neurais, como as GNNs.

A primeira estratégia enfrenta o desafio intrínseco de definir funções-objetivo robustas em um cenário não supervisionado (GUO; DAI, 2021; LI; ZHU, 2024). Ao mesmo tempo, a integração desses objetivos com a aprendizagem hierárquica das estruturas dos grafos exige abordagens inovadoras que evitem a dependência excessiva de heurísticas. Por outro lado, a utilização de um procedimento preliminar de agrupamento, frequentemente realizado por algoritmos como *K-Means*, fornece um ponto de partida que pode ser refinado por meio de treinamento da GNN. Essa abordagem híbrida, representada na Figura 9, permite que as GNNs ajustem as fronteiras das comunidades, identificando partições mais refinadas e contextualmente coerentes (BO et al., 2020).

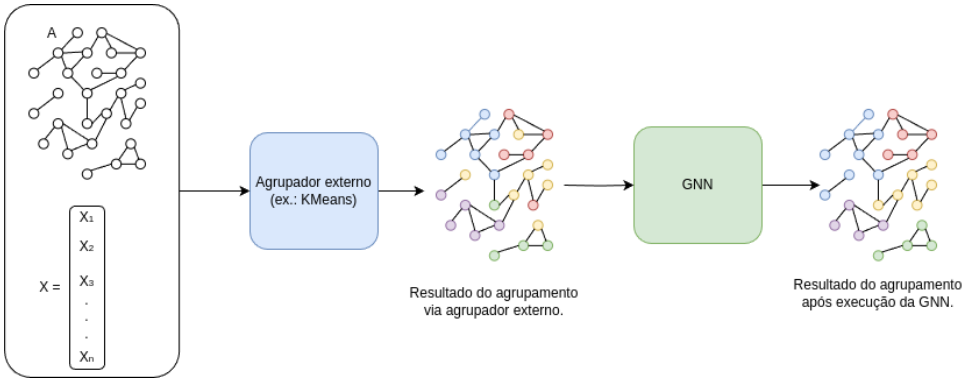


Figura 9 – Mecanismo onde um agrupador externo tradicional, como por exemplo o *K-Means*, é combinado com GNN para refinamento no processo de detecção de comunidades em grafos.

Fonte: Produzido pelo autor.

Um exemplo notório dessa integração é o framework DNENC (*Deep Neighbor-aware Embedded Node Clustering*) (WANG et al., 2022). Nele, um *Graph Autoencoder* é empregado para extrair representações latentes combinando atributos dos nós e a estrutura

do grafo, seja por meio de codificadores baseados em convolução ou mecanismos de atenção (*Attention*), seguido pela aplicação de *K-Means* para atribuir suavemente os nós às comunidades. Esse processo é então iterativamente refinado por um módulo de autotreinamento que ajusta tanto as representações quanto a própria formação das comunidades de maneira conjunta e integrada.

Outras abordagens, como o UCoDe (MORADAN et al., 2023), empregam GNNs para calcular métricas de modularidade de maneira eficiente, maximizando as conexões intracomunidades e minimizando as interações entre comunidades. De forma semelhante, o UDGC (JIAO; LI, 2024) adota uma estratégia de otimização mútua dentro de um framework, que permite a formação de comunidades mais coesass. Metodologias adicionais, como a proposta por (CHOONG; LIU; MURATA, 2020) e o EG-VGAE (CHENG et al., 2024), demonstram a eficácia da integração de informações evolutivas ou variantes probabilísticas no processo de detecção de comunidades em grafos, superando limitações inerentes às abordagens estáticas.

É seguro afirmar que os Modelos de aprendizado profundo representam uma evolução dos métodos tradicionais de detecção de comunidades em grafos ao incorporar de maneira conjunta a aprendizagem das representações e o refinamento das comunidades. Essa integração possibilita a descoberta de padrões mais complexos e a exploração de informações latentes que não seriam capturadas por outras abordagens. Contudo, a definição de funções de custo que reflete de forma satisfatória o conceito de detecção de comunidades em grafos continua sendo uma área ativa de pesquisa, exigindo o desenvolvimento de técnicas cada vez mais sofisticadas e adaptativas.

Alternativamente, muitas abordagens de *Deep Clustering* dependem de moldar suas funções de custo em torno de uma distribuição-alvo. Essa distribuição é definida por nós representativos para cada comunidade (ou *cluster*), tipicamente identificados por algoritmos convencionais de agrupamento, como o *K-Means*. Esses algoritmos tradicionais oferecem uma implementação direta, um desempenho robusto em agrupamento e apontam com precisão exemplos representativos (GUO; DAI, 2021). Além disso, impor uma distribuição auxiliar incentiva a formação natural de comunidades bem separadas, ao guiar os nós para agrupamentos coesos no espaço latente (LI; ZHU, 2024).

Capítulo 3

Agrupamento Usando Redes Neurais de Grafos e Seleção de Sementes

Esse capítulo apresenta o método proposto para uma nova abordagem em problemas de Detecção de Comunidades em Grafos. Como consequência, é proposto o **DGCSS** (*Deep Graph Clustering with Seed Selection* ou em português Agrupamento Profundo Usando Redes Neurais em Grafos e Seleção de Sementes), uma abordagem inovadora baseada em Redes Neurais em Grafos (GNNs) para a realização de detecção de comunidades em grafos. A originalidade do DGCSS reside em sua arquitetura composta por três módulos: (1) o módulo de seleção de sementes, que identifica os nós representativos (os quais, em tarefas de detecção de comunidades, podem fazer o papel de centroides das comunidades); (2) o módulo de *embedding*, que utiliza mecanismos de Atenção em Grafos (*Graph Attention*) para capturar informações topológicas globais; e (3) o módulo de auto-supervisão, que utiliza os nós representativos para orientar a tarefa de detecção de comunidades. A detecção de potenciais centróides das comunidades, denominados "sementes", é realizada por meio de um de três distintos mecanismos implementados: Centralidade de Intermediação (*Betweenness Centrality*), Centralidade de Proximidade (*Closeness Centrality*) e Sementes Aleatórias, este último utilizado como *baseline*. Cada método é avaliado de forma independente, permitindo investigar como as diferentes estratégias para a seleção de sementes influenciam a formação das comunidades. Ao fornecer pontos de referência que capturam diferentes aspectos da topologia do grafo, ou mesmo através da escolha de posições aleatórias, essas sementes orientam o processo subsequente conduzido pela GNN, de modo a definir fronteiras coerentes entre as comunidades. Todo o código da implementação do DGCSS está disponível no GitHub.

3.1 Algoritmo DGCSS (Deep Graph Clustering with Seed Selection)

A execução do DGCSS começa com a seleção das sementes, e em seguida inicia-se uma segunda etapa do DGCSS empregando um *Autoencoder* de Grafos (GAE) para mapear os nós para um espaço latente de dimensão inferior, preservando informações estruturais locais e globais, o que favorece um agrupamento mais eficaz. Em seguida, os *embeddings* preliminares são refinados por meio de um módulo dedicado de *Agrupamento Auto-Supervisionado*, que combina a função de reconstrução do GAE com uma função de perda específica para o agrupamento. Essa perda de agrupamento visa aproximar os nós de seus centróides atribuídos, evitando soluções triviais e aprimorando a interpretabilidade das comunidades finais. A otimização conjunta do GAE e do módulo de agrupamento permite equilibrar a reconstrução estrutural com a coesão das comunidades formadas, resultando em representações robustas e bem separadas no espaço latente. Em suma, o *DGCSS* é implementado em três módulos principais:

1. **Seleção de Sementes**
2. **Geração de *embeddings* via GAE**
3. **Agrupamento Auto-Supervisionado**

Essa estratégia, que se inicia ancorando os centróides no grafo, procede com o aprendizado dos *embeddings* orientado por esses centróides e, por fim, refina as definições das comunidades através da auto-supervisão. Tal abordagem integrada configura uma nova metodologia fundamentada e eficaz para o agrupamento de grafos, capaz de explorar de maneira aprofundada tanto a estrutura global quanto as particularidades locais dos dados. Uma visão completa da arquitetura proposta pode ser vista na Figura 10

3.1.1 Módulo de seleção de sementes

A seleção de sementes em algoritmos de detecção de comunidades em grafos consiste em identificar um conjunto inicial de nós influentes, frequentemente denominados centróides ou "sementes(KEMPE; KLEINBERG; TARDOS, 2003), que servem como ponto de partida para o processo de detecção de comunidades (ou *clustering*). A partir da identificação desses nós estratégicos, o algoritmo utiliza as sementes para orientar o aprendizado, facilitando a convergência para subestruturas coesas e representativas dentro do grafo.

Diversas estratégias podem ser empregadas para a seleção das sementes (FREEMAN, 1978), abrangendo desde a amostragem aleatória até abordagens baseadas em métricas de centralidade, tais como a Centralidade de Intermediação e a Centralidade de Proximidade. Tais métricas quantificam a influência ou relevância de um nó na topologia do

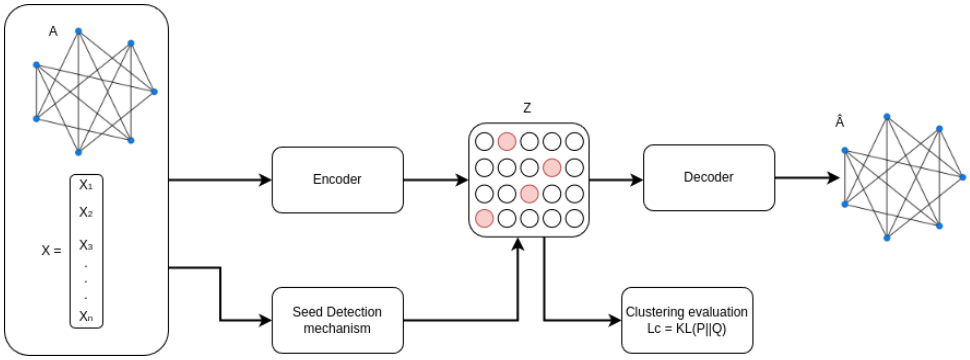


Figura 10 – Arquitetura do DGCSS. A entrada consiste em um grafo definido pela matriz de adjacência A e pelo vetor de atributos dos nós X. Com base na topologia original do grafo, emprega-se um mecanismo de seleção de sementes, que marca os nós selecionados como centróides na matriz de *embedding* Z. Além da computação convencional da função de perda do *Autoencoder* de Grafos (GAE), uma perda de agrupamento é calculada e incorporada à função de perda global da rede.

Fonte: Produzido pelo autor.

grafo, e sua escolha é determinante, pois impacta significativamente tanto a velocidade de convergência quanto a qualidade das comunidades finais.

Na arquitetura do DGCSS, os mecanismos de seleção de sementes são empregados para identificar os centróides iniciais que ancoram o processo de detecção de comunidades. Esses centróides servem como referência para as funções de perda do modelo, que avaliam as distâncias entre os nós e os centróides para calcular os valores de perda e orientar o treinamento da rede. Assim, a seleção adequada das sementes é imperativa para garantir que a aprendizagem da GNN seja guiada por informações estruturais pertinentes.

Para a seleção de sementes, propomos a utilização e a comparação de três mecanismos distintos:

1. **Centralidade de Intermediação** (FREEMAN, 1977);
2. **Centralidade de Proximidade** (SABIDUSSI, 1966);
3. **Sementes Aleatórias**.

Cada um desses métodos baseia-se em métricas diferentes para a seleção dos centróides, e a escolha entre eles influencia diretamente o treinamento do modelo e os resultados do agrupamento. A análise comparativa entre esses mecanismos, bem como seus impactos na qualidade do processo de detecção de comunidades, é discutida detalhadamente na seção de Experimentos 4.

A seguir, apresenta-se um resumo matemático do processo de seleção de sementes: seja $\mathcal{U} \subset V$ o conjunto de sementes selecionadas, onde V é o conjunto de nós do grafo. A seleção de cada semente $u \in \mathcal{U}$ pode ser formulada como a maximização de uma função

de relevância $R(v)$ definida sobre os nós $v \in V$, isto é:

$$\mu = \arg\max_{v \in V} R(v), \tag{21}$$

em que $R(v)$ pode representar, por exemplo, a centralidade de intermediação ou de proximidade. Essa formulação permite a integração de diferentes critérios na escolha das sementes, contribuindo para a robustez e eficácia do método proposto.

3.1.2 Módulo de *Embedding*

O processo de *embedding* desenvolvido neste trabalho é realizado por meio de um *Autoencoder* de Grafos (GAE), conforme os princípios apresentados na seção 2.3.4. Neste contexto, o objetivo é projetar os nós do grafo em um espaço latente de dimensionalidade reduzida, preservando ao máximo as relações de adjacência existentes na estrutura original. Para atingir esse objetivo, emprega-se uma rede neural capaz de codificar as informações do grafo, que incluem tanto os valores da matriz de adjacência quanto os atributos dos nós, em vetores latentes, com o intuito de capturar as estruturas intrínsecas do grafo e, simultaneamente, facilitar a execução de tarefas subsequentes, como o agrupamento e a predição de ligações.

Mais especificamente, a etapa de *Codificação* pode ser formalizada pela seguinte equação:

$$Z = \sigma\Big(\tilde{A}XW\Big), \tag{22}$$

onde:

- $\tilde{A}$ representa a matriz de adjacência normalizada;
- X é a matriz dos atributos dos nós;
- W denota os pesos treináveis do modelo;
- $\sigma(\cdot)$ é a função de ativação, por exemplo, $ReLU$.

Dessa forma, Z constitui a representação latente de cada nó, a qual é posteriormente submetida a um processo de decodificação. Nesta fase, o modelo busca reconstruir a estrutura original do grafo ou prever as relações existentes entre os nós. Essa abordagem integra o aprendizado de representações discriminativas à geração de estruturas plausíveis, alinhando-se com a proposta apresentada em (KIPF; WELLING, 2016b) e contribuindo para a eficácia nas tarefas subsequentes de análise de grafos.

3.1.3 Módulo de Auto-Supervisão e Função de Perda

No terceiro módulo da arquitetura do DGCSS, empregamos um módulo de auto-supervisão, que integra o processo de agrupamento diretamente ao treinamento do *Auto-*

encoder de Grafos (GAE). Enquanto as arquiteturas padrão de GAE baseiam-se exclusivamente na matriz de adjacência para as operações, o DGCSS implementa uma matriz M (ver Equação 23) que não apenas considera os vizinhos adjacentes, mas também abrange vizinhos a t saltos. Isso é alcançado por meio de uma matriz de transição B, definida por $B_{ij} = \frac{1}{d_i}$, onde d_i denota o grau do nó i. Assim, cada elemento B_{ij} representa a probabilidade de um passo de passeio aleatório do nó i para o nó j. Ao empregar M em substituição à matriz de adjacência original $\tilde{A}$ durante o treinamento da GNN, incorporamos a relevância dos vizinhos de múltiplos saltos ao processo de aprendizagem:

$$M = (B + B^2 + \cdots + B^t)\, t, \tag{23}$$

onde t é um parâmetro ajustável que indica quantos níveis de vizinhança serão considerados.

No que tange à função de perda, o DGCSS constrói seu objetivo L pela soma de dois componentes: o erro de reconstrução do *Autoencoder* de Grafos, L_r, e o erro de agrupamento, L_c, ponderado por um fator γ. Formalmente, a perda total é definida como:

$$L = L_r + \gamma \cdot L_c. \tag{24}$$

Enquanto L_r quantifica a discrepância entre a matriz de adjacência original e sua reconstrução, L_c orienta o agrupamento dos nós no espaço latente. Para cada nó representado por z_i, utiliza-se uma distribuição *t-Student* (Equação 25) para mensurar a proximidade entre z_i e os centróides μ_u de cada comunidade, resultando na matriz $Q[i][u]$:

$$q_{iu} = \frac{(1 + \|z_i - \mu_u\|^2)^{-1}}{\sum_k (1 + \|z_i - \mu_k\|^2)^{-1}}. \tag{25}$$

Com base em Q, calcula-se uma distribuição de probabilidade P (Equação 26), que representa a probabilidade de o nó i pertencer ao comunidade u:

$$p_{iu} = \frac{\frac{q_{iu}^2}{\sum_i q_{iu}}}{\sum_k \left(\frac{q_{ik}^2}{\sum_i q_{ik}} \right)}. \tag{26}$$

Por fim, o erro de agrupamento L_c é obtido por meio da divergência de Kullback–Leibler (KULLBACK; LEIBLER, 1951) entre as distribuições P e Q:

$$L_c = KL(P \parallel Q). \tag{27}$$

Este mecanismo de auto-supervisão garante que o processo de agrupamento evolua em conjunto com o treinamento do GAE, orientando continuamente a rede para refinar suas representações internas. Ao monitorar a distribuição de comunidades através de L_c, o modelo desenvolve representações latentes não apenas altamente discriminativas, mas também sensíveis às estruturas de vizinhança de múltiplos saltos, culminando em comunidades mais coesos e em um desempenho preditivo aprimorado.

3.1.4 Algoritmo completo

O Algoritmo 1 sumariza o DGCSS. Abaixo a explicação linha a linha do seu funcionamento:

Algoritmo 1 Algoritmo DGCSS

Require:
- ❑ Grafo $G = (V, E)$ e atributos dos nós X
- ❑ Número de sementes k
- ❑ Métrica de centralidade $c(v)$ (ex.: *Betweenness* ou *Closeness*)
- ❑ Hiperparâmetro γ (para balancear as perdas de reconstrução e formação das comunidades (*clustering*))

Ensure: *embeddings* dos nós aprendidos Z e atribuições das comunidades.

1: **Definição:** Seja $Z \in \mathbb{R}^{N \times d'}$ a matriz de *embeddings* latentes obtida através de um *Graph Autoencoder* (GAE).
2: Defina a divergência de Kullback-Leibler entre uma distribuição alvo P e uma distribuição prevista Q como:

$$\mathcal{L}_{\mathrm{KL}} = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

3:
4: **Algoritmo:**
5: 1. **Seleção de Sementes:** Calcule a pontuação de centralidade $c(v)$ para cada nó $v \in V$ e selecione os u nós com os maiores valores como sementes.
6: 2. **Criação dos *Embeddings*:** Aplique o GAE a X e G para obter as representações latentes:

$$Z = \mathrm{GAE}(X, G)$$

7: 3. **Marcação das Sementes:** Marque os nós sementes na matriz de *embeddings* Z.
8: 4. **Decodificação e Reconstrução:** Decodifique Z para reconstruir o grafo e compute a perda de reconstrução $\mathcal{L}_{\mathrm{rec}}$.
9: 5. **Atribuição de Clusters:** Atribua cada nó a um cluster, utilizando a distribuição prevista Q (derivada de Z).
10: 6. **Cálculo de P:** Calcule a distribuição P.
11: 7. **Perda de Clustering:** Calcule a perda de clustering $\mathcal{L}_{\mathrm{clust}}$:

$$\mathcal{L}_{\mathrm{clust}} = \mathcal{L}_{\mathrm{KL}}(P \,\|\, Q)$$

12: 8. **Perda Total:** Calcule a perda total:

$$\mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \gamma \cdot \mathcal{L}_{\mathrm{clust}}$$

13: 9. **Atualização da Rede:** Atualize os parâmetros da rede minimizando $\mathcal{L}$.
14: 10. **Iteração:** Repita os passos 2 a 8 até a convergência.
15: **return** *embeddings* finais Z e atribuições finais de clusters.

Linhas 1 e 2: Define a matriz de *embeddings* $Z \in \mathbb{R}^{N \times d'}$, que é obtida por meio de um *Autoencoder* para Grafos (GAE). Também apresenta a fórmula da divergência de

Kullback-Leibler ($\mathcal{L}_{\mathrm{KL}}$) que mede a discrepância entre a distribuição alvo P e a prevista Q.

Linha 3: Inicia a parte operacional do algoritmo, separando as definições iniciais das etapas de processamento.

Linha 5: Realiza a seleção de sementes. Aqui, calcula-se a pontuação de centralidade $c(v)$ para cada nó $v \in V$ e, a partir desse ranking, selecionam-se os k nós com os maiores valores, que atuarão como centróides para o processo de detecção de comunidades.

Linha 6: Executa a etapa de criação dos *embeddings* aplicando o GAE sobre os atributos dos nós X e a estrutura do grafo G, gerando as representações latentes armazenadas em Z.

Linha 7: Marca os nós que foram selecionados como sementes na matriz de *embeddings* Z, destacando-os para serem utilizados nas etapas subsequentes de atribuição das comunidades.

Linha 8: Decodifica os *embeddings* Z para reconstruir a estrutura original do grafo e calcula a perda de reconstrução $\mathcal{L}_{\mathrm{rec}}$. Essa etapa é fundamental para garantir que os *embeddings* mantenham as informações estruturais do grafo.

Linha 9: Atribui cada nó à uma comunidade. Essa atribuição é feita através do cálculo da distribuição prevista Q, derivada dos *embeddings*.

Linha 10: Calcula a distribuição P, para que posteriormente a divergência de Kullback–Leibler seja calculada.

Linha 11: Calcula a perda de *clustering* (ou *clustering loss*) $\mathcal{L}_{\mathrm{clust}}$ (Equação 27), que é definida como a divergência de Kullback-Leibler entre as distribuições P e Q, evidenciando discrepâncias na atribuição das comunidades.

Linha 12: Define a perda total $\mathcal{L}$ como a soma ponderada da perda de reconstrução e da perda no processo de detecção de comunidades, ou simplesmente a perda de *clustering*, sendo que o hiperparâmetro γ controla a influência relativa da perda de *clustering*.

Linha 13: Atualiza os parâmetros do modelo minimizando a perda total $\mathcal{L}$, geralmente por meio de métodos de otimização como o gradiente descendente.

Linha 14: Repete-se iterativamente as etapas de criação dos *embeddings*, atribuição das comunidades e atualização dos parâmetros, até que ocorra uma convergência do modelo e a perda se estabilize.

Linha 15: Por fim, o algoritmo retorna os *embeddings* finais Z e as atribuições definitivas dos comunidades.

Capítulo 4

Análise Experimental

A aplicação de Redes Neurais em Grafos (GNN) para problemas de detecção de comunidades tem sido amplamente estudada na literatura atual, evidenciando a relevância e o potencial dessa abordagem (WU et al., 2021c). Nesse contexto, o novo algoritmo DGCSS foi comparado com outros métodos conhecidos na área, selecionados dentre os diversos algoritmos emergentes. Para a avaliação comparativa, foram utilizados conjuntos de dados amplamente reconhecidos em *benchmarks*, além de tecnologias consolidadas, que serão detalhadamente descritas nas seções subsequentes.

4.1 Ferramentas

Durante a execução dos experimentos foi utilizada a linguagem de programação *Python*, com as bibliotecas *Numpy* (HARRIS et al., 2020), *PyTorch Geometric* (FEY; LENSSEN, 2019) e *Networkx* (HAGBERG; SCHULT; SWART, 2008). A Tabela 1 mostra as ferramentas utilizadas em cada etapa da pesquisa.

Etapa da pesquisa	**Ferramenta Utilizada**	**Bibliotecas**
Elaboração dos Algoritmos	*Python* 3.10.16	*Numpy, Pytorch Geometric, NetworkX, Networkit*
Testes paramétricos	*Python* 3.10.16	*Numpy, Pytorch Geometric, NetworkX, Pandas, Matplotlib, Networkit*
Avaliação dos resultados	*Python* 3.10.16	*Numpy, Pytorch Geometric, NetworkX, Matplotlib, Networkit*

Tabela 1 – Ferramentas

4.2 Conjuntos de dados e *Baselines*

A avaliação experimental foi conduzida seguindo uma metodologia que buscou garantir a fidelidade da implementação e a relevância comparativa dos resultados obtidos. Inicialmente, a implementação do algoritmo DGCSS foi realizada por completo e encontra-se disponível no GitHub, proporcionando total transparência e reprodutibilidade dos experimentos. Para assegurar a correção e a consistência do código, foram conduzidos centenas de testes unitários e integrados, os quais verificaram tanto o comportamento funcional dos módulos individuais quanto a interação entre eles.

Posteriormente, foram executados extensivos testes paramétricos, totalizando centenas de execuções, com o objetivo de explorar diversas configurações e combinações de hiperparâmetros. Essa fase foi fundamental para identificar os ajustes ideais e para compreender a sensibilidade do algoritmo a variações nos parâmetros, reconhecendo, contudo, que o espaço de busca é vasto e que inúmeras combinações ainda podem ser investigadas.

Em paralelo aos testes de robustez e ajustes, o desempenho do DGCSS foi avaliado por meio de comparações diretas com algoritmos conhecidos e amplamente avaliados neste campo de pesquisa. Esses *baselines* foram escolhidos com base em sua relevância e reconhecimento na literatura, garantindo assim uma análise crítica e contextualizada dos resultados.

Para todos os três experimentos foram utilizados os conjuntos de dados Cora, CiteSeer e Amazon Photo (YANG; COHEN; SALAKHUTDINOV, 2016; BOJCHEVSKI; GüNNEMANN, 2018; MCAULEY et al., 2015). Por serem conjuntos de dados com tamanhos, formas e complexidades diferentes, a utilização desses conjuntos de dados possibilita uma avaliação robusta e comparável dos métodos, fornecendo evidências empíricas substanciais acerca da eficácia e do potencial do DGCSS. Além disso, os três conjuntos de dados são amplamente usados em *benchmarks* para este campo de pesquisa.

Tabela 2 – Estatísticas dos Conjuntos de Dados

Dataset	**Vértices**	**Arestas**	**Densidade**	**Características**	**Labels**
Cora	2708	5278	0.0007	1433	7
CiteSeer	3327	4552	0.0004	3703	6
Amazon Photo	7650	238162	0.0002	745	8

Para fazer uma comparação justa do modelo proposto com os outros algoritmos, comparamos o nosso modelo com algoritmos já conhecidos em tarefas de detecção de comunidades, descritos abaixo.

- **K-Means**

  O K-Means é um dos algoritmos de particionamento mais clássicos e amplamente utilizados em tarefas de detecção de comunidades. Baseado em uma abordagem iterativa, ele visa minimizar a variância intra-comunidades ao designar pontos de

dados aos centros mais próximos. Apesar de sua simplicidade e eficiência computacional, especialmente em grandes volumes de dados, sua performance pode ser sensível à inicialização dos centróides e à definição prévia do número de comunidades (MACQUEEN, 1967).

❑ ***FastGreedy***

O algoritmo *FastGreedy* é uma técnica hierárquica para detecção de comunidades, frequentemente empregada em contextos de análise de grafos. Este método busca otimizar a modularidade de forma rápida, fundindo iterações de nós e comunidades de maneira gananciosa. Sua aplicabilidade se destaca em cenários onde a estrutura subjacente dos dados pode ser modelada como um grafo, possibilitando a identificação de agrupamentos densos de forma eficiente (CLAUSET; NEWMAN; MOORE, 2004).

❑ ***SpectralClustering***

O *SpectralClustering* utiliza a decomposição espectral da matriz de similaridade dos dados para realizar a redução dimensional e identificar estruturas de comunidades não-lineares. Ao explorar os autovalores e autovetores da matriz Laplaciana, esse algoritmo consegue capturar a estrutura geométrica dos dados, oferecendo uma abordagem robusta para problemas complexos onde as comunidades podem não ser convexas ou claramente separadas em termos euclidianos (NG; JORDAN; WEISS, 2002).

❑ **GAE + *K-Means***

Nesta abordagem, um Graph Autoencoder (GAE) é empregado para aprender representações latentes dos nós a partir da estrutura do grafo, possibilitando a preservação das características topológicas e semânticas dos dados. Posteriormente, essas representações são submetidas ao *K-Means*, combinando o poder do aprendizado não supervisionado com a eficácia do particionamento tradicional. Esta integração facilita a extração de padrões mais refinados, demonstrando melhorias na segmentação dos dados em comparação com métodos que operam diretamente no espaço original (KIPF; WELLING, 2016d).

❑ **VGAE + *K-Means***

Similar à abordagem anterior, o VGAE (*Variational Graph Autoencoder*) estende o modelo de *AutoEncoder* ao incorporar uma modelagem probabilística que captura a incerteza inerente às representações latentes. Essa abordagem variacional permite uma melhor generalização dos padrões aprendidos, especialmente em contextos onde a variabilidade dos dados é significativa. A posterior aplicação do *K-Means* sobre essas representações latentes propicia uma segmentação mais robusta, beneficiando-se

tanto da modelagem probabilística quanto da eficiência do particionamento clássico (KIPF; WELLING, 2016e).

- **DNENC**
  O DNENC (WANG et al., 2022) combina um mecanismo tradicional de detecção de comunidades (*K-Means*) com um Graph Autoencoder (GAE) para refinar o processo. Inicialmente, a GAE extrai *embeddings* dos nós, preservando as características estruturais e dos atributos do grafo. Em seguida, o *K-Means* é aplicado para identificar os centróides correspondentes a cada comunidade. Por fim, uma função de perda é otimizada para aprimorar a definição das comunidades, refinando as representações dos nós e melhorando a qualidade da detecção das comunidades.

Essa seleção de *baselines* proporciona uma base sólida para a avaliação comparativa, abrangendo desde métodos tradicionais de *clustering* até técnicas que integram aprendizado de representações em grafos, possibilitando uma análise abrangente do desempenho do DGCSS em diversos cenários.

4.3 Avaliação Paramétrica

Para avaliar os parâmetros do modelo proposto foram realizados experimentos variando os valores em cada um dos conjuntos de dados e observando os resultados obtidos. O objetivo é entender quais são as influências dos parâmetros configuráveis no algoritmo nos resultados em cada um dos conjuntos de dados selecionados. Alguns outros parâmetros podem ser modificados a fim de se tentar melhorar ainda mais a performance do DGCSS em conjuntos de dados específicos, tais como a taxa de aprendizado (*learning rate*) inicial e sua variação ao longo das iterações do algoritmo. Entretanto, as documentações mais detalhadas são referentes a variação nos tamanhos da camada oculta da GNN *hidden layer* e camada de saída *output layer*, bem como a variação do *clustering loss gamma* (apresentado na equação 24).

4.3.1 Tamanho da camada oculta

Nesta subseção, serão explorados os efeitos da variação dos tamanhos da camada oculta (*hidden layer*) na performance da GNN proposta. Em nossa arquitetura, a *hidden layer* é responsável por extrair e transformar as características dos dados, realizando operações de agregação e propagação de informação entre os nós. Essa camada atua como o principal motor do aprendizado, permitindo a captura de padrões complexos e a incorporação das relações estruturais presentes no grafo. Já a camada de saída (*output layer*) traduz as representações latentes obtidas nas camadas intermediárias para um espaço de saída que pode ser interpretado em termos das tarefas de detecção de comunidades. Essa camada é

crucial para garantir que as estruturas aprendidas sejam expressas de forma que possibilite uma segmentação consistente e comparável dos dados.

Nas Figuras 11 e 12 é possível avaliar a influência do tamanho da camada oculta sobre a métrica de desempenho *Normalized Mutual Information* (NMI) para os conjuntos de dados utilizados Cora e CiteSeer. Nota-se que a mudança no tamanho da camada oculta pouco impacta no resultado geral do processo de detecção de comunidades considerando a métrica de NMI. Entretanto, é importante reforçar que o impacto no tempo de execução é significativo, visto que o aumento no tamanho da camada oculta aumenta a quantidade de processamento necessária para o treinamento da GNN. Em alguns testes executados, a mudança de uma camada oculta de tamanho 16 para 512 fez com o tempo de execução aumentasse em média 92%, por exemplo.

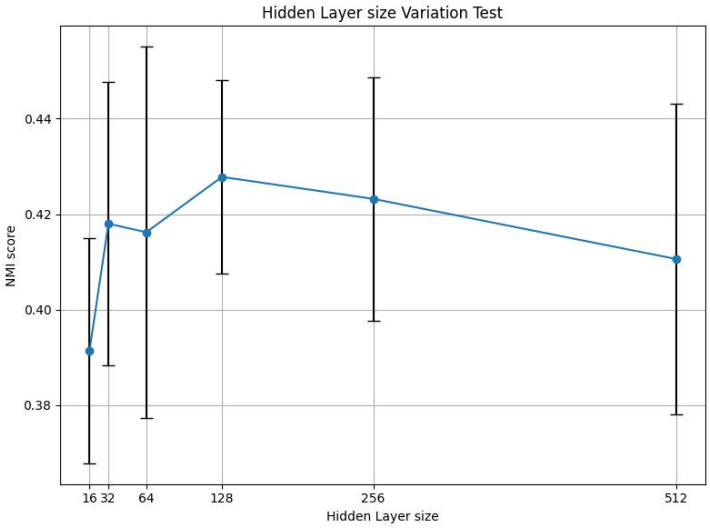


Figura 11 – Variação do tamanho da camada oculta na aplicação DGCSS + Closeness Centrality no conjunto de dados Cora. Considera-se a média de 5 execuções para cada configuração de parâmetros.

Fonte: Produzido pelo autor

4.3.2 Variação do *clustering loss gamma*

A variação do parâmetro *clustering loss gamma* (ver equação 27) tem papel fundamental no processo de otimização do algoritmo, influenciando diretamente a capacidade de aprendizado e a formação dos agrupamentos. Esse parâmetro atua como um fator de ponderação na função de perda, balanceando a importância entre a reconstrução das representações latentes e a incorporação da informação de detecção de comunidades durante o treinamento da GNN. Ao aumentar o valor de *gamma* (γ), o modelo é forçado a dar maior ênfase à minimização da distância intra-comunidade e à maximização da separabilidade entre comunidades, o que pode resultar em agrupamentos mais coerentes e robustos.

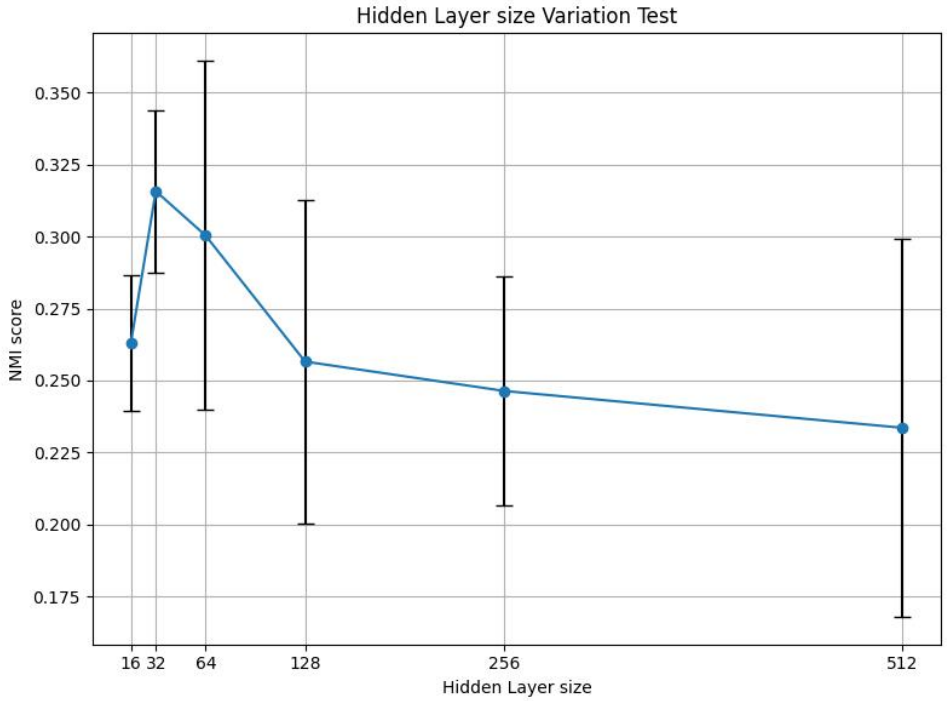


Figura 12 – Variação do tamanho da camada oculta na aplicação DGCSS + Closeness Centrality no conjunto de dados Citeseer. Considera-se a média de 5 execuções para cada configuração de parâmetros.

Fonte: Produzido pelo autor

Para investigar a sensibilidade do modelo à variação deste hiperparâmetro, foram conduzidos experimentos sistemáticos onde o valor de *clustering loss gamma* foi incrementado dentro de um intervalo pré-definido. Em cada configuração, observou-se o comportamento do desempenho do algoritmo através da métrica *Normalized Mutual Information* (NMI), que fornece uma medida quantitativa da similaridade entre os agrupamentos obtidos e as classes reais dos dados. Essa análise é essencial, pois valores inadequados de *gamma* podem levar a um *overfitting* do critério de detecção de comunidades ou, inversamente, a uma subutilização das informações de agrupamento, comprometendo a generalização do modelo.

Os resultados experimentais, que podem ser vistos nas Figuras 13 e 14, evidenciam como a variação de *clustering loss gamma* impacta significativamente a qualidade dos agrupamentos. Observa-se, por exemplo, que incrementos moderados nesse parâmetro tendem a melhorar a coesão interna das comunidades (*clusters*) e a clareza na separação entre eles. Entretanto, aumentos excessivos podem resultar em uma superespecialização do modelo para as configurações de detecção de comunidades, diminuindo a performance geral do algoritmo. Dessa forma, a escolha do valor ideal de γ (olhar Equação 24) é crucial para atingir um equilíbrio entre a fidelidade da representação dos dados e a eficácia dos agrupamentos. Essa investigação detalhada oferece subsídios para a correta parametrização do algoritmo, contribuindo para a construção de modelos de alta performance em

cenários de detecção de comunidades com grafos.

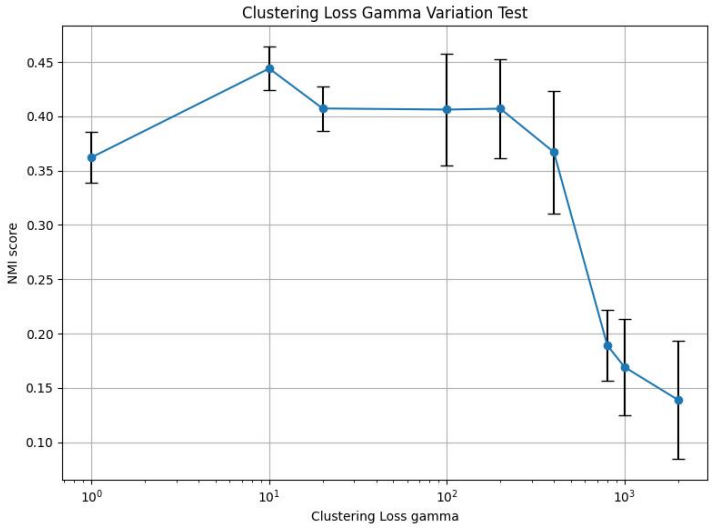


Figura 13 – Variação do *clustering loss gamma* na aplicação DGCSS + Closeness Centrality no conjunto de dados Cora. Considera-se a média de 5 execuções para cada configuração de parâmetros.

Fonte: Produzido pelo autor

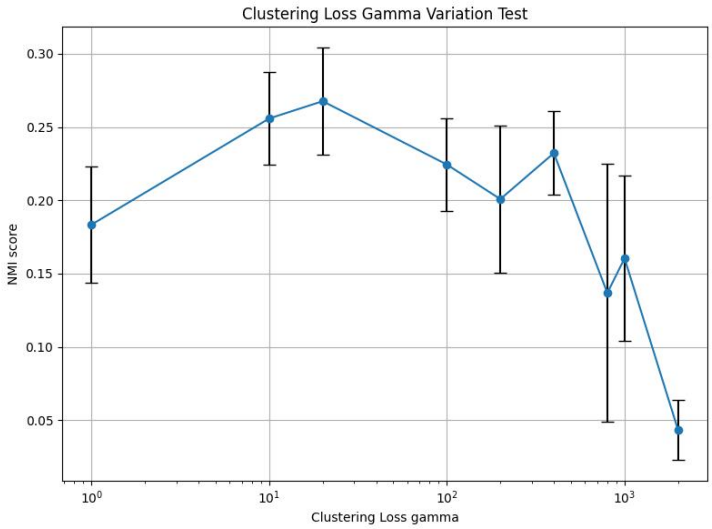


Figura 14 – Variação do *Clustering Loss gamma* na aplicação DGCSS + Closeness Centrality no conjunto de dados Citeseer. Considera-se a média de 5 execuções para cada configuração de parâmetros.

Fonte: Produzido pelo autor

Analisando a função de perda de *clustering* (*clustering loss*) em conjunto com a função de perda do GAE (*GAE loss*), podemos avaliar nas Figuras 15, 16 e 17 como a variação do *clustering loss gamma* afeta o comportamento da perda geral do algoritmo. Valores mais altos de *clustering loss gamma* conduzem o algoritmo para a obtenção de valores mais baixos de *clustering loss*, enquanto valores mais baixos de *clustering loss gamma*

priorizam a redução dos valores de função de perda do GAE. Nas análises conduzidas por este trabalho, entende-se que valores de *clustering loss gamma* entre 10 e 1000 são coerentes para testes e combinações com outros parâmetros e hiperparâmetros do DGCSS.

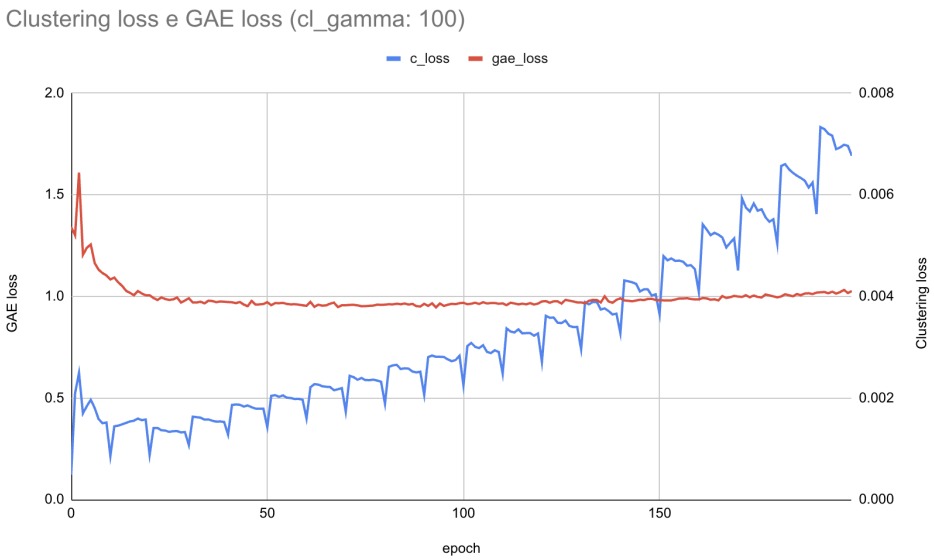


Figura 15 – Valores de GAE loss e Clustering loss durante a execução do DGCSS + Closeness Centrality no conjunto de dados Cora. O valor do *clustering loss gamma* para essa execução foi de 100. A linha vermelha (*gae_loss*) representa os valores de perda do GAE (GAE *loss*), enquanto a linha azul (*c_loss*) representa a função de perda *clustering loss* (equação 27)

.

Fonte: Produzido pelo autor

4.4 Avaliação de desempenho DGCSS

4.4.1 Métricas

Para avaliar a qualidade dos agrupamentos gerados pela nossa abordagem, empregamos três métricas amplamente utilizadas na avaliação de algoritmos de detecção de comunidades: *Normalized Mutual Information* **(NMI)**, *Adjusted Rand Index* **(ARI)** e Modularidade ***(Modularity)***. A NMI quantifica a quantidade de informação compartilhada entre as comunidades preditass e a classificação de referência, sendo normalizada de modo a restringir seus valores ao intervalo entre 0 e 1, onde pontuações mais elevadas indicam um alinhamento superior. O ARI, por sua vez, mensura a similaridade entre as atribuições das comunidades e os rótulos de referência, ajustando o índice para efeitos aleatórios; embora possa assumir valores negativos em cenários desfavoráveis, normalmente os resultados oscilam entre 0 e 1. Por fim, a Modularidade avalia a robustez da divisão de uma rede em comunidades, apresentando valores maiores para estruturas comunitárias

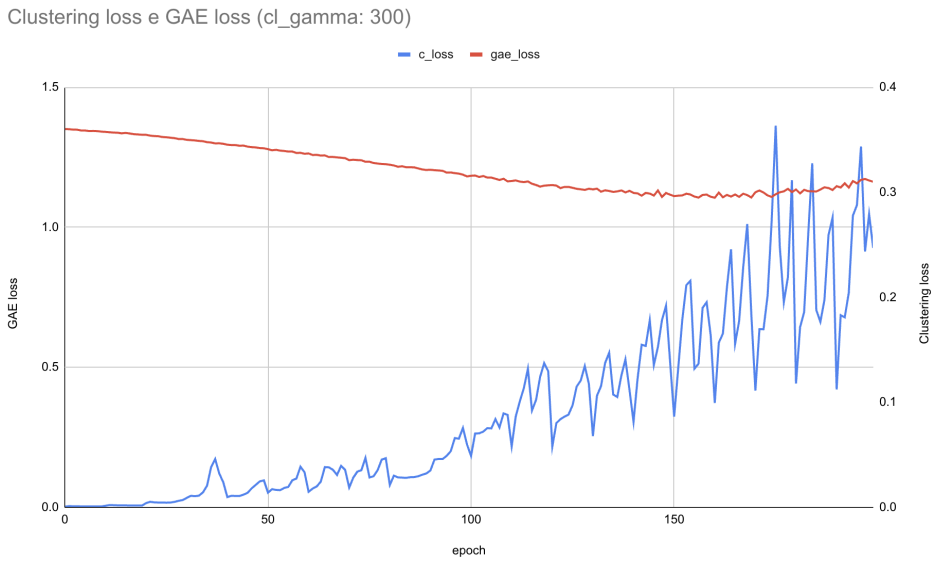


Figura 16 – Valores de GAE loss e Clustering loss durante a execução do DGCSS + Closeness Centrality no conjunto de dados Cora. O valor do *clustering loss gamma* para essa execução foi de 300. A linha vermelha (*gae_loss*) representa os valores de perda do GAE (GAE *loss*), enquanto a linha azul (*c_loss*) representa a função de perda *clustering loss* (equação 27).

Fonte: Produzido pelo autor

mais bem definidas. Os conjuntos de dados selecionados para a realização dos testes são detalhados na seção 4.2.

4.4.2 Parâmetros

Em todos os experimentos, empregamos os seguintes hiperparâmetros:

- **Intervalo de cálculo de P:** 10
- **Gama da perda de agrupamento:** 20
- **Tamanho da camada oculta:** 64 para Cora e Citeseer e 128 para Amazon Photo
- **Tamanho da camada de saída:** 16
- **Épocas:** 400

Esses valores constituem uma linha de base robusta para a exploração do modelo. Investigações preliminares sobre configurações alternativas revelaram diversas possibilidades de aprimoramento de desempenho. Por exemplo, o aumento do número de neurônios na camada oculta (*hidden layer)* demonstrou potencial para melhorar os resultados do agrupamento; entretanto, visando assegurar uma comparação rigorosa e equitativa com os algoritmos de última geração, optou-se por manter o tamanho da *hidden layer* fixo em todos os experimentos.

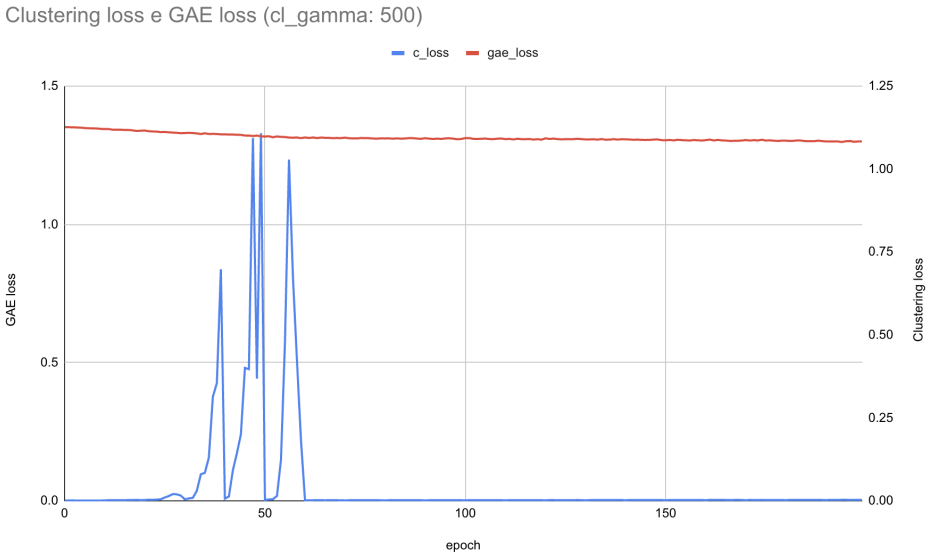


Figura 17 – Valores de GAE loss e Clustering loss durante a execução do DGCSS + Closeness Centrality no conjunto de dados Cora. O valor do *clustering loss gamma* para essa execução foi de 500. A linha vermelha (*gae_loss*) representa os valores de perda do GAE (GAE *loss*), enquanto a linha azul (*c_loss*) representa a função de perda *clustering loss*.

Fonte: Produzido pelo autor

De maneira semelhante, nossa análise indicou que o coeficiente de *clustering loss gamma* (ver Equação 24), γ, desempenha um papel fundamental na modulação dos resultados. Variações em γ podem influenciar significativamente o desempenho, evidenciando a necessidade de um ajuste mais refinado deste hiperparâmetro. Adicionalmente, experimentos com arquiteturas de rede com maior número de camadas ocultas não produziram melhorias consistentes, sugerindo que o aumento da complexidade do modelo não se traduz necessariamente em melhor performance. Ademais, a implementação de uma estratégia de decaimento da taxa de aprendizado se revelou uma abordagem promissora para a otimização, indicando que esquemas adaptativos de aprendizado podem oferecer benefícios adicionais.

Embora nossa configuração de referência tenha se mostrado eficaz, esses experimentos exploratórios ressaltam o vasto potencial existente no espaço de hiperparâmetros. Trabalhos futuros serão direcionados à investigação sistemática dessas configurações alternativas, com a expectativa de que tais refinamentos possam otimizar ainda mais o desempenho da abordagem proposta.

4.4.3 Resultados

Em nossos experimentos, avaliamos o desempenho de diversos algoritmos de agrupamento aplicados aos conjuntos de dados *Cora*, *Citeseer* e *Amazon - Photo* (detalhes

adicionais podem ser encontrados na Tabela 2). Testes iniciais com algoritmos tradicionais, como *K-Means*, *FastGreedy* e *SpectralClustering*, evidenciaram suas limitações na captura das estruturas complexas inerentes a esses conjuntos de dados. Tais métodos simples demonstraram dificuldades em gerar agrupamentos significativos, o que indica que as estruturas subjacentes das redes demandam abordagens mais sofisticadas para serem eficazmente delineadas.

Nosso algoritmo proposto, o DGCSS, demonstrou desempenho promissor. Embora não tenha superado em todas métricas de referência alcançadas por métodos como GAE e VGAE, o DGCSS obteve resultados consistentes, conforme ilustrado nos gráficos de clusterização apresentados na Figura 18 e nas Tabelas 3, 4 e 5. O algoritmo foi capaz de identificar grupos significativos em todos os conjuntos de dados, tanto em suas configurações BC (*Betweenness Centrality*) quanto CC (*Closeness Centrality*), demostrando indícios de sua aptidão para captar a estrutura intrínseca dos dados. Contudo, existe ainda possibilidade de melhorias futuras. Acreditamos que uma exploração mais abrangente das combinações de parâmetros e estratégias de ajuste mais refinadas poderá gerar melhores resultados, elevando o potencial competitivo do DGCSS.

No dataset **Cora**, a coesão temática e a alta densidade de links favorecem a detecção de sementes informativas, de modo que o método DGCSS + CC obteve o melhor equilíbrio entre *NMI* (0,458) e *ARI* (0,402), enquanto as variantes de autoencoder (GAE/VGAE + K-Means) maximizaram a modularidade (0,721) ao explorar bem a estrutura global do grafo.

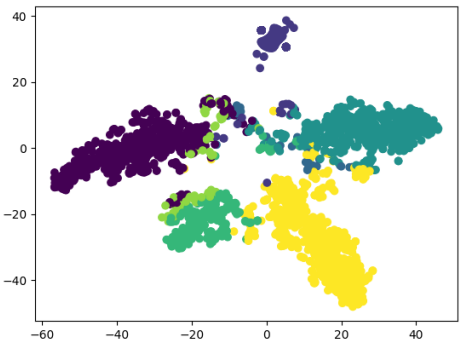
Já em **Citeseer**, a dispersão dos rótulos e o maior ruído textual dificultam a obtenção de sementes de alta qualidade. Por isso, nosso algoritmo DGCSS apresentou desempenho inferior (NMI máximo 0,245, ARI 0,215) em comparação com VGAE + K-Means (NMI 0,343, ARI 0,331). Uma explicação coerente, é a vantagem do autoencoder com sua capacidade de aprender representações latentes contínuas que suavizam a variabilidade dos atributos.

No caso do conjunto de dados **Amazon Photo**, a rede de co-compra exibe comunidades mais bem definidas por afinidade de interesse, beneficiando tanto a detecção de sementes quanto o mecanismo de detecção de comunidades: a variante DGCSS + Random alcançou o maior *NMI* (0,587), DGCSS + CC ficou em primeiro no *ARI* (0,457) e o DNENC liderou a modularidade (0,642). Isso indica que, em grafos onde as conexões refletem preferências fortes, até escolhas aleatórias de sementes podem produzir representações robustas.

Em resumo, cada algoritmo mostra pontos fortes distintos: sementes guiadas (DGCSS) elevam NMI/ARI em redes com comunidades bem separadas; autoencoders (GAE/VGAE) exploram melhor a modularidade em grafos ruidosos ou menos estruturados; e métodos clássicos de clustering falham quando não aproveitam informação de estrutura ou atributos.

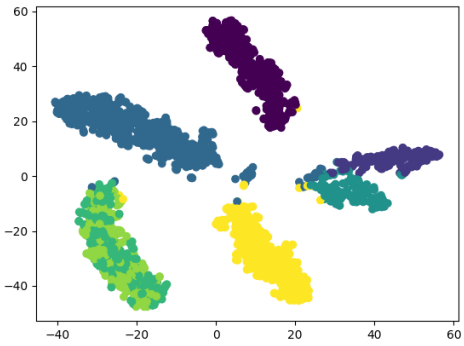
De forma resumida, os experimentos realizados evidenciam os desafios inerentes ao processo de detecção de comunidades em grafos, ressaltando a importância de um aprendizado que envolve um processo de representações robusto. O desempenho encorajador tanto dos métodos baseados em *Autoencoders* de grafos quanto da abordagem DGCSS sugere que, com otimizações adicionais, há um potencial significativo para aprimorar os resultados de agrupamento em pesquisas futuras.

Figura 18 – DGCSS + BC no conjunto Cora



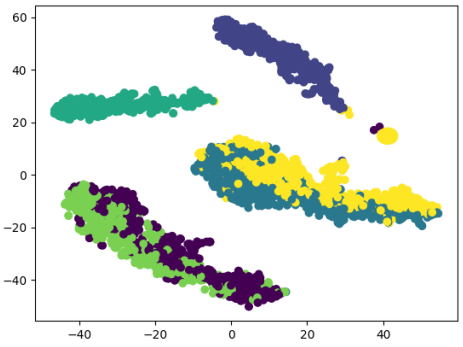
Fonte: Produzido pelo autor

Figura 19 – DGCSS + CC no conjunto Cora



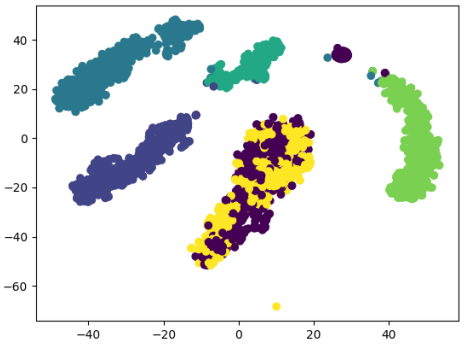
Fonte: Produzido pelo autor

Figura 20 – DGCSS + BC no conjunto Citeseer



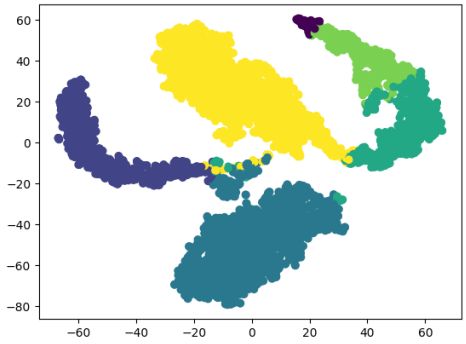
Fonte: Produzido pelo autor

Figura 21 – DGCSS + CC no conjunto Citeseer



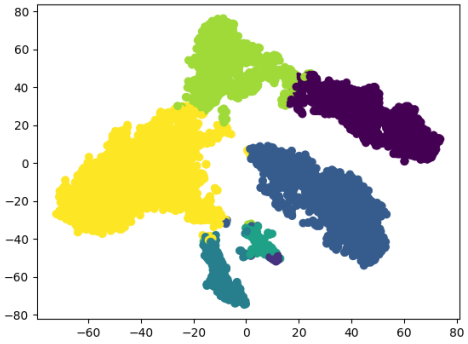
Fonte: Produzido pelo autor

Figura 22 – DGCSS + BC no conjunto Amazon Photo



Fonte: Produzido pelo autor

Figura 23 – DGCSS + CC no conjunto Amazon Photo



Fonte: Produzido pelo autor

Tabela 3 – Desempenho de agrupamento no conjunto de dados **Cora**. As métricas reportadas são a Informação Mútua Normalizada (NMI), o Índice Rand Ajustado (ARI) e a Modularidade (Mod.).

Método	**NMI**	**ARI**	**Mod.**
K-Means	0.077 (±0.06)	0.026 (±0.04)	0.128 (±0.08)
FastGreedy	0.007 (±0)	-0.002 (±0)	0.002 (±0)
SpectralClustering	0.141 (±0.01)	0.107 (±0.01)	0.190 (±0.01)
GAE + *K-Means*	0.454 (±0.03)	0.373 (±0.06)	**0.721** (±0.01)
VGAE + *K-Means*	0.450 (±0.02)	0.371 (±0.05)	0.720 (±0.01)
DNENC	0.428 (±0.07)	0.338 (±0.09)	0.693 (±0.03)
DGCSS + CC	**0.458** (±0.04)	**0.402** (±0.05)	0.670 (±0.03)
DGCSS + BC	0.325 (±0.03)	0.255 (±0.05)	0.569 (±0.02)
DGCSS + Random	0.296 (±0.05)	0.238 (±0.07)	0.486 (±0.05)

Tabela 4 – Desempenho de agrupamento no conjunto de dados **Citeseer**. As métricas reportadas são a Informação Mútua Normalizada (NMI), o Índice Rand Ajustado (ARI) e a Modularidade (Mod.).

Método	**NMI**	**ARI**	**Mod.**
K-Means	0.154 (±0.09)	0.107 (±0.08)	0.245 (±0.07)
FastGreedy	0.003 (±0)	0.000 (±0)	0.000 (±0)
SpectralClustering	0.208 (±0)	0.192 (±0)	0.291 (±0)
GAE + *K-Means*	0.312 (±0)	0.287 (±0)	**0.743** (±0)
VGAE + *K-Means*	**0.343** (±0.07)	**0.331** (±0.07)	0.740 (±0.07)
DNENC	0.279 (±0.02)	0.266 (±0.03)	0.686 (±0.04)
DGCSS + CC	0.235 (±0.02)	0.179 (±0.03)	0.611 (±0.04)
DGCSS + BC	0.237 (±0.01)	0.180 (±0.02)	0.557 (±0.06)
DGCSS + Random	0.245 (±0.03)	0.215 (±0.05)	0.702 (±0.04)

Tabela 5 – Desempenho de agrupamento no conjunto de dados **Amazon Photo**. As métricas reportadas são a Informação Mútua Normalizada (NMI), o Índice Rand Ajustado (ARI) e a Modularidade (Mod.).

Método	**NMI**	**ARI**	**Mod.**
K-Means	0.112 (±0)	0.045 (±0)	0.060 (±0)
FastGreedy	0.002 (±0)	0.000 (±0)	0.000 (±0)
SpectralClustering	0.421 (±0)	0.290 (±0)	0.315 (±0)
GAE + *K-Means*	0.524 (±0.11)	0.309 (±0.07)	0.577 (±0.13)
VGAE + *K-Means*	0.555 (±0.03)	0.376 (±0.05)	0.592 (±0.05)
DNENC	0.559 (±0.03)	0.411 (±0.03)	**0.642** (±0.02)
DGCSS + CC	0.570 (±0.06)	**0.457** (±0.06)	0.635 (±0.03)
DGCSS + BC	0.497 (±0.02)	0.364 (±0.02)	0.604 (±0.07)
DGCSS + Random	**0.587** (±0.04)	0.453 (±0.06)	0.624 (±0.02)

Capítulo 5

Conclusão

Nessa dissertação foi proposto o algoritmo DGCSS para tarefas de detecção de comunidades, que integra mecanismos de seleção de sementes com Redes Neurais em Grafos (GNN). Essa abordagem inovadora difere das estratégias convencionais, que geralmente combinam métodos tradicionais de detecção de comunidades com GNNs, aumentando a complexidade do sistema. Ao combinar a seleção de sementes com GNN, o DGCSS visa proporcionar uma identificação inicial mais assertiva dos agrupamentos, facilitando a propagação de informações nos grafos.

Os experimentos realizados evidenciaram que o algoritmo é capaz de gerar representações significativas dos dados e, consequentemente, de promover uma detecção de comunidades eficiente. Embora o DGCSS não tenha se destacado como o melhor em todas as métricas e cenários avaliados, os resultados apontam para seu potencial, especialmente considerando a vantagem de reduzir a complexidade associada às abordagens tradicionais. Essa constatação reforça a relevância da proposta e justifica o direcionamento para a investigação de melhorias e refinamentos futuros.

Ademais, os estudos indicam que há inúmeras possibilidades de combinações de parâmetros a serem exploradas, o que contribui para a dificuldade em testar todas as possíveis combinações de configurações. Essa variabilidade ressalta tanto a flexibilidade do DGCSS quanto o desafio inerente à busca por uma otimização global do desempenho. Assim, o trabalho aqui apresentado estabelece uma base sólida para futuras pesquisas que poderão explorar ajustes finos e alternativas metodológicas, contribuindo para o avanço das técnicas de detecção de comunidades em grafos.

5.1 Contribuição

Neste projeto de pesquisa, a principal contribuição foi a entrega do algoritmo DGCSS, desenvolvido integralmente durante o mestrado. O código-fonte do algoritmo está disponível de forma aberta no GitHub, permitindo a replicabilidade dos resultados e incentivando futuras melhorias pela comunidade.

O DGCSS opera em três fases de execução: o módulo de seleção de sementes (3.1.1), empregado para identificar nós representativos no grafo; o módulo de *embedding* (3.1.2), que utiliza mecanismos de atenção em grafos (*Graph Attention*) para capturar informações topológicas dos dados; e o módulo de auto-supervisão (3.1.3) e função de perda, que utiliza os nós representativos encontrados na primeira fase para orientar a tarefa de detecção de comunidades.

Adicionalmente, os experimentos mostram uma comparação do DGCSS com outros algoritmos conhecidos na área, utilizando conjuntos de dados amplamente utilizados para comparações e avaliações de técnicas de GNN em tarefas de detecção de comunidades. Essa comparação evidencia as potencialidades e limitações do algoritmo, contribuindo para a discussão e o aprimoramento das abordagens atuais no campo.

Vale ressaltar que a implementação do DGCSS no GitHub também conta com outros mecanismos de detecção de centróides já implementados, os quais podem ser avaliados em futuras pesquisas. Entre eles, destacam-se: WBC *Weighted Betweenness Centrality*), *PageRank*, *K-Means*, *FastGreedy*, *KCore*, *EigenV* (*Eigenvector Centrality*), CSC (*Cosine Similarity Centrality*) e CSD (*Cosine Similarity Density*). Embora tais mecanismos não tenham sido discutidos em profundidade neste trabalho, por fugirem um pouco do escopo principal desta pesquisa, sua inclusão amplia as possibilidades de experimentação e aprimoramento do algoritmo.

5.2 Limitação

Uma das principais limitações deste trabalho foi o tempo de execução do algoritmo em conjuntos de dados, e consequentemente grafos, maiores, o que torna inviável a avaliação completa do DGCSS em grafos com centenas de milhares de conexões entre nós. Além disso, embora o aumento do tamanho das camadas da GNN possa, em alguns casos, levar a resultados superiores, os requisitos de hardware necessários, tanto em termos de processamento quanto de memória, excederam as condições disponíveis durante este projeto de pesquisa. Por fim, a vasta quantidade de combinações possíveis de parâmetros impõe um desafio adicional, pois testar uma infinidade de configurações seria de grande interesse, mas não coube no tempo disponível para este projeto.

5.3 Trabalhos Futuros

Apesar das limitações encontradas, este projeto apresenta uma abordagem inovadora ao combinar seleção de sementes com GNN para tarefas de detecção de comunidades. Assim, pretende-se continuar explorando novas combinações de parâmetros e testar o algoritmo em outros conjuntos de dados, ampliando o leque de cenários avaliados.

Dado que o algoritmo foi implementado de forma modular, facilitando a troca de mecanismos e módulos, uma linha promissora de pesquisa é a avaliação de outros métodos de seleção de sementes. Essa modularidade não só permite a integração de novas técnicas, mas também possibilita uma análise detalhada do impacto de cada componente na performance geral do algoritmo.

Além disso, encontra-se em estudo a combinação da técnica de Similaridade de Cosseno com os mecanismos de seleção de sementes, com o objetivo de determinar centróides iniciais em grafos onde os nós representam dados textuais. Essa abordagem poderá contribuir para a melhoria dos agrupamentos e reforça a ideia de que, com um algoritmo modular, de código aberto e bem organizado, é possível explorar continuamente novas combinações de mecanismos para aprimorar o desempenho do algoritmo.

Referências

ALPAYDIN, E. **Introduction to Machine Learning**. 3. ed. Cambridge, MA: MIT Press, 2014. (Adaptive Computation and Machine Learning). ISBN 978-0-262-02818-9.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. **Autoencoders**. 2021.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006.

BLONDEL, V. D. et al. Fast unfolding of communities in large networks. **Journal of Statistical Mechanics: Theory and Experiment**, IOP Publishing, n. 10, p. P10008, 2008.

BO, D. et al. Structural deep clustering network. In: **Proceedings of The Web Conference 2020**. ACM, 2020. (WWW '20), p. 1400–1410. Disponível em: <http://dx.doi.org/10.1145/3366423.3380214>.

BOJCHEVSKI, A.; GüNNEMANN, S. **Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking**. 2018.

BRANDES, U. A faster algorithm for betweenness centrality. **Journal of Mathematical Sociology**, Springer, v. 25, n. 2, p. 163–177, 2001.

CAI, H.; ZHENG, V. W.; CHANG, K. C.-C. A comprehensive survey of graph embedding: Problems, techniques, and applications. **IEEE Transactions on Knowledge and Data Engineering**, v. 30, n. 9, p. 1616–1637, 2018.

CEN, K. et al. **ANAE: Learning Node Context Representation for Attributed Network Embedding**. 2020.

CHENG, J. et al. Unveiling community structures in static networks through graph variational bayes with evolution information. **Neurocomputing**, Elsevier BV, v. 576, p. 127349, abr. 2024. ISSN 0925-2312. Disponível em: <http://dx.doi.org/10.1016/j.neucom.2024.127349>.

CHOONG, J. J.; LIU, X.; MURATA, T. Optimizing variational graph autoencoder for community detection with dual optimization. **Entropy**, MDPI AG, v. 22, n. 2, p. 197, fev. 2020. ISSN 1099-4300. Disponível em: <http://dx.doi.org/10.3390/e22020197>.

CHUNG, F. R. K. **Spectral Graph Theory**. Providence, RI: American Mathematical Society, 1997.

CLAUSET, A.; NEWMAN, M. E. J.; MOORE, C. Finding community structure in very large networks. **Physical Review E**, APS, v. 70, n. 6, p. 066111, 2004.

EISEN, M. B. et al. Cluster analysis and display of genome-wide expression patterns. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 95, n. 25, p. 14863–14868, 1998.

FACELI, K. et al. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [S.l.]: LTC, 2011.

FEY, M.; LENSSEN, J. E. Fast graph representation learning with PyTorch Geometric. In: **ICLR Workshop on Representation Learning on Graphs and Manifolds**. [S.l.: s.n.], 2019.

FORTUNATO, S. Community detection in graphs. **Physics reports**, Elsevier, v. 486, n. 3-5, p. 75–174, 2010.

FORTUNATO, S.; BARTHéLEMY, M. Resolution limit in community detection. **Proceedings of the National Academy of Sciences**, v. 104, n. 1, p. 36–41, 2007. Disponível em: <https://www.pnas.org/doi/abs/10.1073/pnas.0605965104>.

FREEMAN, L. C. A set of measures of centrality based on betweenness. **Social Networks**, Elsevier, v. 1, n. 1, p. 215–239, 1977.

_____. Centrality in social networks: Conceptual clarification. **Social Networks**, Elsevier, v. 1, n. 3, p. 215–239, 1978.

GARCIA, J.; FERNÁNDEZ, F. A comprehensive survey on safe reinforcement learning. **Journal of Machine Learning Research**, v. 16, n. 1, p. 1437–1480, 2015.

GILMER, J. et al. **Neural Message Passing for Quantum Chemistry**. 2017.

GIRVAN, M.; NEWMAN, M. E. J. Community structure in social and biological networks. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 99, n. 12, p. 7821–7826, 2002.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GUO, L.; DAI, Q. End-to-end variational graph clustering with local structural preservation. **Neural Computing and Applications**, Springer Science and Business Media LLC, v. 34, n. 5, p. 3767–3782, out. 2021. ISSN 1433-3058. Disponível em: <http://dx.doi.org/10.1007/s00521-021-06639-7>.

GUO, Y.; CHEN, Z.; KARYPIS, G. Graph representation learning for molecular data. **Nature Machine Intelligence**, v. 3, n. 5, p. 455–465, 2021.

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.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

JAIN, A. K. Data clustering: 50 years beyond k-means. **Pattern Recognition Letters**, Elsevier, v. 31, n. 8, p. 651–666, 2010.

JIAO, Z.; LI, X. An end-to-end deep graph clustering via online mutual learning. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), p. 1–8, 2024. ISSN 2162-2388. Disponível em: <http://dx.doi.org/10.1109/TNNLS.2024.3353217>.

_____. An end-to-end deep graph clustering via online mutual learning. **IEEE Transactions on Neural Networks and Learning Systems**, v. 36, n. 2, p. 3847–3854, 2025.

JIN, D. et al. Adversarial capsule learning for network embedding. In: **2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)**. [S.l.: s.n.], 2019. p. 214–221.

KEMPE, D.; KLEINBERG, J.; TARDOS, É. Maximizing the spread of influence through a social network. In: ACM. **Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.], 2003. p. 137–146.

KIPF, T. N.; WELLING, M. **Semi-Supervised Classification with Graph Convolutional Networks**. arXiv, 2016. Disponível em: <https://arxiv.org/abs/1609.02907>.

_____. Variational graph auto-encoders. In: **Proceedings of the International Conference on Learning Representations (Workshop Track)**. [S.l.: s.n.], 2016. ArXiv preprint arXiv:1611.07308.

_____. **Variational Graph Auto-Encoders**. 2016. Disponível em: <https://arxiv.org/abs/1611.07308>.

_____. Variational graph auto-encoders. **arXiv preprint arXiv:1611.07308**, 2016.

_____. Variational graph auto-encoders. **arXiv preprint arXiv:1611.07308**, 2016.

_____. Semi-supervised classification with graph convolutional networks. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2017.

_____. Semi-supervised classification with graph convolutional networks. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2017.

KOTSIANTIS, S. B. et al. Supervised machine learning: A review of classification techniques. **Emerging artificial intelligence applications in computer engineering**, Amsterdam, v. 160, n. 1, p. 3–24, 2007.

KULLBACK, S.; LEIBLER, R. A. On Information and Sufficiency. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 22, n. 1, p. 79 – 86, 1951. Disponível em: <https://doi.org/10.1214/aoms/1177729694>.

LI, Z.; ZHU, C. Self-supervised graph clustering via attention auto-encoder with distribution specificity. **Multimedia Systems**, Springer Science and Business Media LLC, v. 30, n. 3, maio 2024. ISSN 1432-1882. Disponível em: <http://dx.doi.org/10.1007/s00530-024-01346-4>.

LINDEN, G.; SMITH, B.; YORK, J. Amazon.com recommendations: Item-to-item collaborative filtering. **IEEE Internet Computing**, IEEE, v. 7, n. 1, p. 76–80, 2003.

LIU, Q.; WANG, J.; ZHANG, T. Leveraging graph structures for communication network optimization. **IEEE Transactions on Communications**, v. 70, n. 3, p. 1856–1866, 2022.

LIU, Q.; ZHANG, T.; WANG, H. Graph-based citation network analysis for academic literature. **Scientometrics**, v. 125, n. 2, p. 789–808, 2020.

LUXBURG, U. v. A tutorial on spectral clustering. **Statistics and Computing**, Springer, v. 17, n. 4, p. 395–416, 2007.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. In: UNIVERSITY OF CALIFORNIA PRESS. **Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability**. [S.l.], 1967. p. 281–297.

MCAULEY, J. et al. **Image-based Recommendations on Styles and Substitutes**. 2015. Disponível em: <https://arxiv.org/abs/1506.04757>.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 127–147, 1943.

MCLACHLAN, G.; PEEL, D. **Finite Mixture Models**. [S.l.]: John Wiley & Sons, 2000.

MESSIAS, G. H. **Redes neurais em grafos para aprendizado positivo: uma abordagem utilizando reescrita**. Dissertação (Dissertação (Mestrado em Ciência da Computação)) — Universidade Federal de São Carlos, São Carlos, 2024. Disponível em: <https://repositorio.ufscar.br/handle/20.500.14289/20326>.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1.

MORADAN, A. et al. Ucode: unified community detection with graph convolutional networks. **Machine Learning**, Springer Science and Business Media LLC, v. 112, n. 12, p. 5057–5080, set. 2023. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/s10994-023-06402-0>.

NEWMAN, M. E. J. Fast algorithm for detecting community structure in networks. **Physical Review E**, American Physical Society, v. 69, n. 6, p. 066133, 2004.

_____. Modularity and community structure in networks. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 103, n. 23, p. 8577–8582, 2006.

NEWMAN, M. E. J.; WATTS, D. J.; STROGATZ, S. H. Random graph models of social networks. **Proceedings of the National Academy of Sciences**, v. 99, n. 1, p. 2566–2572, 2002.

NG, A. Y.; JORDAN, M. I.; WEISS, Y. On spectral clustering: Analysis and an algorithm. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2002. v. 14, p. 849–856.

NG, I. et al. **A Graph Autoencoder Approach to Causal Structure Learning**. 2019.

PAN, S. et al. Learning graph embedding with adversarial training methods. **IEEE Transactions on Cybernetics**, Institute of Electrical and Electronics Engineers (IEEE), v. 50, n. 6, p. 2475–2487, jun. 2020. ISSN 2168-2275. Disponível em: <http://dx.doi.org/10.1109/TCYB.2019.2932096>.

RANI, M.; NAYAK, R.; VYAS, O. An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. **Knowledge-Based Systems**, v. 90, p. 33–48, 2015. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705115003779>.

ROBBINS, H.; MONRO, S. A stochastic approximation method. **The Annals of Mathematical Statistics**, Institute of Mathematical Statistics, v. 22, n. 3, p. 400–407, 1951. ISSN 00034851. Disponível em: <http://www.jstor.org/stable/2236626>.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Springer Science and Business Media LLC, v. 323, n. 6088, p. 533–536, out. 1986. Disponível em: <https://doi.org/10.1038/323533a0>.

SABIDUSSI, G. The centrality index of a graph. **Psychometrika**, Springer, v. 31, n. 4, p. 581–603, 1966.

SALHA, G.; HENNEQUIN, R.; VAZIRGIANNIS, M. Simple and effective graph autoencoders with one-hop linear models. In: _____. **Machine Learning and Knowledge Discovery in Databases**. Springer International Publishing, 2021. p. 319–334. ISBN 9783030676582. Disponível em: <http://dx.doi.org/10.1007/978-3-030-67658-2_19>.

SCARSELLI, F. et al. The graph neural network model. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 61–80, 2009.

SCHLICHTKRULL, M. et al. Modeling relational data with graph convolutional networks. In: SPRINGER. **The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15**. [S.l.], 2018. p. 593–607.

SHEN, Z. et al. Adversarial learning based residual variational graph normalized autoencoder for network representation. **Information Sciences**, Elsevier BV, v. 640, p. 119055, set. 2023. ISSN 0020-0255. Disponível em: <http://dx.doi.org/10.1016/j.ins.2023.119055>.

SONG, Z. et al. **Graph-based Semi-supervised Learning: A Comprehensive Review**. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2102.13303>.

_____. Graph-based semi-supervised learning: A comprehensive review. **IEEE Transactions on Neural Networks and Learning Systems**, p. 1–21, 2022.

SOUTO, M. C. P. et al. Técnicas de aprendizado de máquina para problemas de biologia molecular. In: **Congresso da Sociedade Brasileira de Computação**. [S.l.]: SBC, 2003.

SUN, D. et al. Glass: A graph laplacian autoencoder with subspace clustering regularization for graph clustering. **Cognitive Computation**, Springer Science and Business Media LLC, v. 15, n. 3, p. 803–821, jan. 2023. ISSN 1866-9964. Disponível em: <http://dx.doi.org/10.1007/s12559-022-10098-0>.

VASWANI, A. et al. Attention is all you need. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017. v. 30.

VELIčKOVIć, P. et al. Graph attention networks. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2018.

WANG, C. et al. Deep neighbor-aware embedding for node clustering in attributed graphs. **Pattern Recognition**, Elsevier BV, v. 122, p. 108230, fev. 2022. ISSN 0031-3203. Disponível em: <http://dx.doi.org/10.1016/j.patcog.2021.108230>.

WANG, Y. et al. Seeing all from a few: Nodes selection using graph pooling for graph clustering. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 35, n. 5, p. 7231–7237, maio 2024. ISSN 2162-2388. Disponível em: <http://dx.doi.org/10.1109/TNNLS.2022.3210370>.

WU, L. et al. **Graph Neural Networks: Foundations, Frontiers, and Applications**. Singapore: Springer Singapore, 2022. 725 p.

WU, M. et al. Learning graph neural networks with positive and unlabeled nodes. **ACM Trans. Knowl. Discov. Data**, Association for Computing Machinery, New York, NY, USA, v. 15, n. 6, jun 2021. ISSN 1556-4681. Disponível em: <https://doi.org/10.1145/3450316>.

WU, Z. et al. A comprehensive survey on graph neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, v. 32, n. 1, p. 4–24, 2021.

_____. A comprehensive survey on graph neural networks. **IEEE Transactions on Neural Networks and Learning Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 32, n. 1, p. 4–24, jan 2021. Disponível em: <https://doi.org/10.1109%2Ftnnls.2020.2978386>.

XIA, F. et al. Graph learning: A survey. **IEEE Transactions on Artificial Intelligence**, Institute of Electrical and Electronics Engineers (IEEE), v. 2, n. 2, p. 109–127, apr 2021. Disponível em: <https://doi.org/10.1109%2Ftai.2021.3076021>.

XIA, J.; SILVA, M.; DOE, J. Graph-based methods for data dependency modeling. **Journal of Network Science**, v. 15, n. 2, p. 123–137, 2021.

XIA, J.; WANG, L.; RODRIGUES, F. A graph modeling approach to urban vehicle traffic. **IEEE Transactions on Intelligent Transportation Systems**, v. 21, n. 10, p. 4120–4132, 2020.

YANG, Y. et al. Attributed graph embedding with random walk regularization and centrality-based attention. **Mathematics**, MDPI AG, v. 11, n. 8, p. 1830, abr. 2023. ISSN 2227-7390. Disponível em: <http://dx.doi.org/10.3390/math11081830>.

YANG, Z.; COHEN, W. W.; SALAKHUTDINOV, R. **Revisiting Semi-Supervised Learning with Graph Embeddings**. 2016.

YU, B.; YIN, H.; ZHU, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: **Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence**. International Joint Conferences on Artificial Intelligence Organization, 2018. (IJCAI-2018), p. 3634–3640. Disponível em: <http://dx.doi.org/10.24963/ijcai.2018/505>.

ZHANG, W. et al. Personalized web page ranking based graph convolutional network for community detection in attribute networks. **IEEE Access**, v. 11, p. 84270–84282, 2023.

ZHU, X.; GOLDBERG, A. B. **Introduction to Semi-Supervised Learning**. Morgan and Claypool Publishers, 2009. (Synthesis Lectures on Artificial Intelligence and Machine Learning). Disponível em: <http://dx.doi.org/10.2200/S00196ED1V01Y200906AIM006>.