# Vector-Based Approach to the Max-Cut Problem: Conversion to Clustering and Validation on Structured Graphs

**Jailon W. B. Oliveira da Silva**[1], **Carlos V. Dantas Araújo**[1], **Pablo L. Braga Soares**[1]

[1]Laboratório de Pesquisa & Desenvolvimento do NEMO
Universidade Federal do Ceará – Campus Russas
Avenida Felipe Santiago – Nº 411 62.900-624, Russas/CE, Brasil

{williambrunocc, carlosvinicio}@alu.ufc.br, pablo.soares@ufc.br

***Abstract.*** *This paper proposes an unsupervised approach for the Maximum Cut Problem (Max-Cut) through conversion to a clustering task. It combines three vertex representations (Node2Vec, Spectral, and local attributes) with KMeans and hierarchical algorithms. Validation on 57 instances from the Big Mac Library (clustering coefficient* $> 0.2$*) shows superior performance of local attributes representation with KMeans (91.4% similarity, 41 instances* $> 90\%$*), followed by Spectral (82%) and Node2Vec (72%). Results indicate effectiveness in highly clustered graphs, with limitations in sparse structures (coefficient* $< 0.2$*). The approach replaces combinatorial methods with direct vector analysis, offering a scalable solution for medium-scale Max-Cut.*

***Resumo.*** *Este trabalho propõe uma abordagem não supervisionada para o Problema do Corte Máximo (Max-Cut) via conversão em tarefa de agrupamento. Combina três representações de vértices (Node2Vec, Spectral e atributos locais) com algoritmos KMeans e hierárquico. Validação em 57 instâncias da Big Mac Library (coeficiente de aglomeração* $> 0.2$*) mostra desempenho superior da representação por atributos locais com KMeans (91.4% de similaridade, 41 instâncias* $> 90\%$*), seguida por Spectral (82%) e Node2Vec (72%). Resultados indicam eficácia em grafos com alta aglomeração, com limitações em estruturas esparsas (coeficiente* $< 0.2$*). A abordagem substitui métodos combinatórios por análise vetorial direta, oferecendo solução escalável para Max-Cut em média escala.*

## 1. Introdução

O Problema do Corte Máximo (Max-Cut), que consiste em particionar os vértices de um grafo ponderado em dois subconjuntos disjuntos para maximizar a soma dos pesos das arestas entre eles, é um desafio central em otimização combinatória. Classificado como NP-difícil [Garey and Johnson 1979], o Max-Cut possui aplicações críticas que vão desde o design de circuitos VLSI [Barahona 1982] até a otimização de redes quânticas [Farhi et al. 2014]. A complexidade inerente ao problema inviabiliza soluções exatas para grafos de grande escala, onde métodos tradicionais, como a programação semidefinida, demandam recursos computacionais proibitivos [Rendl et al. 2010a].

Abordagens heurísticas convencionais enfrentam dois desafios principais: 1) **Sensibilidade a ruídos estruturais**: Em grafos com baixo coeficiente de aglomeração, a

falta de uma estrutura comunitária clara pode levar a soluções instáveis [Wiegele 2007]. 2) **Ineficiência em grafos de grande escala**: Muitas técnicas baseadas em relaxações matemáticas tornam-se computacionalmente inviáveis à medida que o número de vértices e arestas aumenta.

Neste contexto, este trabalho investiga uma abordagem alternativa: a conversão do Max-Cut em uma tarefa de aprendizado não supervisionado. A intuição central é que, se os vértices puderem ser representados em um espaço vetorial onde a distância reflete a conveniência de separá-los em partições distintas, algoritmos de agrupamento como o KMeans poderiam encontrar um corte de alto valor. A hipótese é que, ao separar os vértices em dois clusters, estamos implicitamente definindo os dois conjuntos disjuntos do Max-Cut. Esta pesquisa inova ao explorar sistematicamente três paradigmas de representação gráfica para este fim:

- **Node2Vec**: Preserva relações de vizinhança através de passeios aleatórios, tradicionalmente usado para detectar comunidades (homofilia) [Grover and Leskovec 2016].
- **Agrupamento Espectral**: Captura a conectividade global via decomposição do Laplaciano do grafo, também fundamentalmente ligado à detecção de clusters coesos [Von Luxburg 2007].
- **Atributos Locais**: Utiliza características diretas de cada vértice (grau, estatísticas de pesos), oferecendo uma representação de baixo custo computacional.

A abordagem proposta mitiga os desafios mencionados da seguinte forma: a sensibilidade a ruídos é tratada pela filtragem de instâncias com coeficiente de aglomeração abaixo de um limiar, garantindo uma estrutura mínima para análise. A ineficiência é contornada ao substituir cálculos combinatórios complexos pela análise vetorial direta, especialmente na abordagem de atributos locais, que possui custo computacional linear. A contribuição central deste trabalho é, portanto, a avaliação sistemática dessa conversão, revelando quais tipos de representação são mais adequados para a tarefa e descobrindo que, para grafos com alta coesão estrutural, atributos locais simples superam *embeddings* complexos, oferecendo uma solução escalável e eficaz para o Max-Cut em média escala.

## 2. Fundamentação Teórica

Este tópico está estruturado da seguinte forma: a Seção 2.1 apresenta o problema do Corte Máximo e suas aplicações, a Seção 2.2 descreve o método node2vec para representação de vértices, a Seção 2.3 aborda o Clustering Espectral baseado em decomposição Laplaciana, a Seção 2.4 introduz conceitos de aprendizado não supervisionado em grafos, as Seções 2.5 e 2.6 discutem algoritmos de agrupamento por partição e hierárquico respectivamente, e a Seção 2.7 enumera características locais fundamentais para análise estrutural de grafos.

### 2.1. Problema do Corte Máximo

Dado um grafo não direcionado $G = (V, E)$ com pesos não negativos $w_{ij} \geq 0$ para cada aresta $(i, j) \in E$, o Problema do Corte Máximo (Max-Cut) busca uma partição do conjunto de vértices $V$ em dois subconjuntos disjuntos, $S$ e seu complemento $\bar{S} = V \setminus S$. O objetivo é maximizar o peso total do corte, que é a soma dos pesos das arestas

que conectam um vértice em $S$ a um vértice em $\bar{S}$. Formalmente, o problema pode ser expresso como:

$$\text{maximizar} \quad w(S, \bar{S}) = \sum_{i \in S, j \in \bar{S}, (i,j) \in E} w_{ij} \tag{1}$$

Classificado como NP-difícil [Garey and Johnson 1979], o Max-Cut tem sido extensivamente estudado, com aplicações notáveis em design de circuitos VLSI [Liers et al. 2011] e física estatística. A formulação de Goemans e Williamson [Goemans and Williamson 1995], baseada em programação semidefinida, oferece a melhor garantia de aproximação teórica conhecida, mas seu custo computacional limita sua aplicação prática em grafos de grande escala.

### 2.2. Node2vec

O node2vec [Grover and Leskovec 2016] é um método de representação vetorial de vértices que mapeia relações topológicas para espaços euclidianos de baixa dimensão. Utilizando passeios aleatórios balanceados entre busca em largura (BFS) e profundidade (DFS), preserva propriedades estruturais via modelo Skip-gram. Dado um vértice $u$, gera vizinhos $N_S(u)$ via passeios de comprimento $l$, maximizando:

$$\max_f \sum_{u \in V} \log \left( \Pr(N_S(u) | f(u)) \right) \tag{2}$$

onde $f : V \rightarrow R^d$ é a função de *embedding* que associa cada vértice a um vetor $d$-dimensional, e $\Pr(N_S(u)|f(u))$ modela a probabilidade de observar a vizinhança $N_S(u)$ dado o vetor $f(u)$. Essa representação captura homofilia e equivalência estrutural [Hamilton et al. 2017].

### 2.3. Clustering Espectral

O Clustering Espectral particiona grafos baseado na decomposição espectral da matriz Laplaciana [Von Luxburg 2007]. Para um grafo com matriz de adjacências $A$, define-se a Laplaciana $L = D - A$, onde $D$ é a matriz diagonal de graus com elementos $D_{ii} = \sum_j A_{ij}$. Os autovetores de $L$ capturam propriedades de conectividade global:

$$L\mathbf{v} = \lambda \mathbf{v} \tag{3}$$

onde $\lambda$ são autovalores e $\mathbf{v}$ os respectivos autovetores. O segundo menor autovetor (Fiedler vector) fornece uma ordenação para particionamento hierárquico [Fiedler 1973].

### 2.4. Aprendizado Não Supervisionado

Técnicas não supervisionadas identificam padrões intrínsecos em dados não rotulados [Ghahramani 2004]. A função objetivo geral minimiza divergência entre observações e representações latentes:

$$\min_\theta \sum_i \mathcal{D}(x_i, g_\theta(x_i)) \tag{4}$$

onde $x_i$ são observações, $g_\theta$ é o modelo paramétrico, e $\mathcal{D}$ é uma medida de dissimilaridade. Em grafos, permitem descoberta de estruturas sem rótulos prévios [Tsitsulin et al. 2023].

### 2.5. KMeans

O algoritmo KMeans [Lloyd 1982] particiona $n$ observações em $k$ clusters minimizando a variância intra-cluster. Dados vetores $\{\mathbf{x}_i\}_{i=1}^n$ em $R^p$, resolve iterativamente:

$$\min_{\{C_j\}_{j=1}^k} \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \tag{5}$$

onde $C_j$ é o conjunto de pontos no cluster $j$, $\boldsymbol{\mu}_j = |C_j|^{-1} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$ é o centróide, e $\|\cdot\|$ denota norma euclidiana. Sua eficiência computacional ($O(nk)$ por iteração) o torna adequado para grandes conjuntos de dados.

### 2.6. Agrupamento Hierárquico

Métodos hierárquicos constroem dendrogramas representando relações aninhadas entre clusters [Ward Jr 1963]. Abordagens *agglomerative* iniciam com cada ponto como cluster e fundem pares progressivamente. A medida de Ward minimiza a variância total ao fundir clusters:

$$\Delta(A, B) = \frac{\|\boldsymbol{\mu}_A - \boldsymbol{\mu}_B\|^2}{1/|A| + 1/|B|} \tag{6}$$

onde $\boldsymbol{\mu}_A$ e $\boldsymbol{\mu}_B$ são centróides dos clusters $A$ e $B$, respectivamente. Esta flexibilidade permite capturar hierarquias complexas [Schaeffer 2007].

### 2.7. Conceitos Necessários de Características dos Grafos

Quatro características locais são fundamentais para análise estrutural:

1. **Grau do Vértice**: $\deg(v) = |\{u : (v, u) \in E\}|$, número de arestas incidentes, indicando conectividade local [Newman 2018].
2. **Média dos Pesos**: $\frac{1}{\deg(v)} \sum_{u \in \mathcal{N}(v)} w_{vu}$, média aritmética dos pesos incidentes, refletindo intensidade média das conexões [Barrat et al. 2004].
3. **Menor/Maior Peso**: $\min_{u \in \mathcal{N}(v)} w_{vu}$ e $\max_{u \in \mathcal{N}(v)} w_{vu}$, valores extremos nas arestas incidentes, capturando variabilidade de conectividade [Opsahl et al. 2010].
4. **Somatório dos Pesos**: $\sum_{u \in \mathcal{N}(v)} w_{vu}$, soma dos pesos incidentes, **medida representativa** da centralidade de força [Freeman 1978].

## 3. Metodologia

Este tópico está estruturado da seguinte forma: a Seção 3.1 detalha o processo de seleção e filtragem das instâncias da Biq Mac Library, a Seção 3.2 descreve as três abordagens de representação vetorial dos grafos (Node2Vec, Spectral Embedding e matriz de adjacência enriquecida), a Seção 3.3 apresenta os algoritmos de agrupamento k-Means e Hierárquico aplicados ao problema de bipartição, a Seção 3.4 explica o processo de otimização de hiperparâmetros via GridSearch, e por fim são listadas as ferramentas computacionais utilizadas na implementação.

### 3.1. Coleta dos Dados

A base de dados utilizada neste trabalho é composta por um conjunto inicial de 330 instâncias da *Biq Mac Library* [Wiegele 2007], que abrange grafos com tamanhos variando de 20 a 500 vértices. As soluções de referência (valores de corte ótimos ou quase-ótimos) foram obtidas através do solver *Biq Mac* [Rendl et al. 2010b].

Durante os experimentos preliminares, observou-se que instâncias com baixo coeficiente de aglomeração médio resultavam em desempenho instável e próximo ao aleatório para os algoritmos de agrupamento. Para garantir a robustez da análise, foi adotado um critério de filtragem: apenas instâncias com coeficiente de aglomeração médio superior a 0.2 foram selecionadas. O limiar de 0.2 foi determinado empiricamente, representando um ponto de equilíbrio entre manter um número suficiente de instâncias (resultando em 57) e assegurar uma estrutura de vizinhança coesa, onde a hipótese de agrupamento é mais plausível.

### 3.2. Representação dos Grafos

Para que os algoritmos de aprendizado de máquina pudessem ser aplicados, os grafos foram convertidos em representações vetoriais, onde cada vértice corresponde a um vetor de características. Três abordagens foram investigadas:

1. **Node2Vec**: Mapeia vértices para um espaço vetorial de baixa dimensão usando caminhadas aleatórias, capturando a estrutura de vizinhança [Grover and Leskovec 2016].
2. **Spectral Embedding**: Utiliza os autovetores da matriz Laplaciana do grafo para gerar uma representação que captura a conectividade global [Von Luxburg 2007].
3. **Atributos Locais**: Representa cada vértice por um vetor contendo suas características estruturais diretas: grau, média, mínimo, máximo e soma dos pesos das arestas incidentes.

Para as duas primeiras abordagens, a dimensionalidade do *embedding* é um hiperparâmetro fixo, garantindo consistência dimensional entre diferentes instâncias. No entanto, a abordagem de atributos locais, que se mostrou a mais promissora, gera um vetor para cada vértice baseado na sua vizinhança direta. Como o número de vizinhos varia, a representação direta resultaria em matrizes de características com dimensões diferentes para cada grafo, tornando inviável a aplicação de um modelo único. Para contornar essa questão, a abordagem foi simplificada para usar apenas as cinco estatísticas agregadas mencionadas, resultando em um vetor de dimensão fixa para cada vértice e eliminando a necessidade de tratamento dimensional complexo. As representações resultantes foram normalizadas via *StandardScaler* para uniformizar escalas.

### 3.3. Agrupamento Não Supervisionado

Com os vetores de representação obtidos de cada uma das 57 instâncias, foram aplicados dois algoritmos de agrupamento não supervisionado com o objetivo de particionar os vértice dos grafos no subconjunto S e seu complementar, onde o valor de corte obtido pelo modelo será dado pelas arestas de corte, que são as arestas com pontos terminais em ambos conjuntos. Já a abordagem de agrupamento são:

- **k-Means**: Método particional que divide os vértices em dois clusters ($k = 2$), correspondendo às partições $S$ e $V \setminus S$, utilizando distância euclidiana e inicialização *k-means++*. O objetivo é minimizar a variância intra-cluster, produzindo divisões coesas.
- **Agrupamento Hierárquico**: Estratégia baseada em ligação completa (*complete linkage*) com métrica de distância euclidiana. A árvore hierárquica gerada (dendrograma) é cortada de modo a formar duas partições. A escolha de $k = 2$ reflete a natureza binária do Problema do Corte Máximo, que visa bipartir o grafo de modo a maximizar o somatório dos pesos entre as partições.

### 3.4. Ajuste dos Parâmetros

O processo de ajuste de hiperparâmetros empregou a ferramenta *GridSearch*, que realiza buscas sistemáticas em malhas paramétricas para identificar configurações que maximizem uma métrica específica. Neste estudo, o critério de seleção foi o valor de corte resultante, garantindo que os *embeddings* capturassem estruturas gráficas relevantes para o Problema do Corte Máximo. Através de testes exaustivos, avaliaram-se múltiplas combinações paramétricas para cada técnica de representação.

Para o método *Node2Vec*, a análise concentrou-se em cinco dimensões parametrizadas: a dimensionalidade do espaço de *embedding*, o número de caminhadas por nó, o comprimento dessas caminhadas, e o comportamento dos parâmetros $p$ (controle de retorno ao vértice anterior) e $q$ (regulação da exploração entre *clusters*). Cada parâmetro foi submetido a diferentes configurações para determinar seu impacto na qualidade da partição.

Paralelamente, para o *Spectral Embedding*, investigaram-se três aspectos fundamentais: a dimensionalidade do subespaço projetivo definida pelo número de autovetores $k$, o tipo de matriz Laplaciana utilizada (normalizada contra não-normalizada), e a seleção do método de extração de autovalores (SM - Smallest Magnitude contra LM - Largest Magnitude). Esta abordagem permitiu identificar configurações que preservam adequadamente as propriedades de conectividade do grafo.

A seleção final das configurações ótimas baseou-se estritamente na capacidade de cada combinação parametrizada em maximizar o valor do corte, estabelecendo relações diretas entre a sintonia fina dos hiperparâmetros e a eficácia na representação topológica.

### 3.5. Ferramentas e Bibliotecas Utilizadas

Para a implementação e análise experimental, foram empregadas várias ferramentas e bibliotecas da linguagem *Python 3*:

- **Node2Vec**: empregada para gerar embeddings dos nós de um grafo. Essa ferramenta transforma os grafos em representações vetoriais que podem ser usadas para tarefas de aprendizado supervisionado ou não supervisionado.
- **scipy.sparse.linalg**: utilizadas para cálculo de autovalores e autovetores em grafos, especialmente para o método espectral de análise de grafos.
- **Pandas e NumPy**: utilizadas para o processamento e manipulação de dados, fundamentais para o gerenciamento eficiente dos conjuntos de dados e para a execução de operações matemáticas e estatísticas.

- **matplotlib.pyplot e seaborn**: empregadas para a criação de visualizações gráficas dos dados e dos resultados, facilitando a análise e interpretação visual dos resultados.
- **Jupyter Notebooks**: todo o desenvolvimento do código foi realizado em notebooks Jupyter, proporcionando um ambiente interativo para programação, visualização e documentação dos resultados.

## 4. Experimentos e Análise dos Resultados

Esta seção está organizada da seguinte forma: a Seção 4.1 apresenta os valores de corte obtidos experimentalmente para as 57 instâncias, como observado nas Tabelas 1, 2 e 3, comparando o desempenho dos métodos Node2Vec, Spectral Embedding e representação por características contra o solver Biq Mac, enquanto a Seção 4.2 detalha o processo de otimização dos hiperparâmetros para ambas as técnicas de representação gráfica utilizando a abordagem GridSearch.

### 4.1. Valores de Cortes Obtidos nos Experimentos

Durante a análise, pode-se observar que os modelos se saíram melhor em instâncias com um coeficiente de aglomeração por vértice superior a 0.2. Nesses casos, os algoritmos conseguiram generalizar melhor, realizando boas partições. Entretanto, nas instâncias com coeficiente de aglomeração inferior a 0.2, os modelos tendiam a particionar os vértices de forma aleatória, sem identificar padrões significativos. Os resultados completos das 57 instâncias testadas são apresentados nas Tabelas 1, 2 e 3.

O **coeficiente de aglomeração** local para um vértice $v$ mede a densidade de conexões na sua vizinhança, sendo definido como a proporção de arestas existentes entre seus vizinhos em relação ao número máximo de arestas possíveis entre eles. O coeficiente médio do grafo, utilizado como nosso critério de filtragem, é a média desses valores para todos os vértices. Um valor alto indica a presença de comunidades densas (clusters), um cenário onde a partição se torna mais significativa.

Para avaliar o desempenho de forma geral, calculou-se a **razão de aproximação** para cada instância, definida como o valor do corte obtido pelo modelo dividido pelo valor do corte do solver ($C_{\text{modelo}}/C_{\text{solver}}$). A média desta razão entre todas as instâncias é usada como métrica de desempenho agregado, onde valores mais próximos de 1.0 indicam melhor desempenho.

O coeficiente de aglomeração é uma medida que indica a tendência de um vértice estar rodeado por vértices fortemente conectados, ou seja, o quão fortemente conectado está a vizinhança de um vértice. Esse valor é útil, neste caso, para realizar o agrupamento de vértices, visto que grafos com maior coeficiente de aglomeração tendem a ter estruturas de vizinhança mais harmônica, facilitando a identificação de *clusters* e, consequentemente, a realização de cortes mais precisos.

De modo geral, os dados demonstram uma variação considerável entre as instâncias, sugerindo que os dados possuem uma estrutura complexa, o que pode impactar diretamente o desempenho dos métodos de agrupamento. Essa variação pode estar associada a padrões ocultos nas instâncias ou à sensibilidade dos algoritmos às características estruturais do grafo.

**Tabela 1. Resultado dos Cortes das 58 Instâncias - Parte I**

| Instância | Solver | Node2Vec | | Spectral | | Características | |
|---|---|---|---|---|---|---|---|
| | | k-Means | Hierárquico | k-Means | Hierárquico | k-Means | Hierárquico |
| pw05_100.0 | 8190 | 6317.2 | 6098.7 | 6308 | 6691 | 5174 | 6473 |
| pw05_100.1 | 8045 | 6648.2 | 5856.5 | 5838 | 6551 | 4886 | 6574 |
| pw05_100.2 | 8039 | 6614.4 | 6263.1 | 5350 | 6344 | 4603 | 6463 |
| pw05_100.3 | 8139 | 6328.9 | 5446.9 | 6990 | 6340 | 5121 | 6467 |
| pw05_100.4 | 8125 | 6274.5 | 6304.2 | 5801 | 5874 | 4928 | 6620 |
| pw05_100.5 | 8169 | 6819.6 | 5874.3 | 6865 | 4407 | 5182 | 6191 |
| pw05_100.6 | 8217 | 6893.9 | 6491.4 | 6994 | 6987 | 5535 | 6666 |
| pw05_100.7 | 8249 | 6710.0 | 6670.2 | 7135 | 6631 | 5346 | 6474 |
| pw05_100.8 | 8199 | 6191.3 | 6408.1 | 7081 | 7005 | 5254 | 6217 |
| pw09_100.0 | 13585 | 10599.7 | 11262.5 | 12259 | 12186 | 9561 | 10561 |
| pw09_100.1 | 13417 | 9659.2 | 9230.9 | 12485 | 11696 | 9612 | 10329 |
| pw09_100.2 | 13461 | 8275.3 | 6636.9 | 11183 | 9836 | 9543 | 10468 |
| pw09_100.3 | 13656 | 9903.9 | 9503.9 | 12457 | 12306 | 9842 | 11172 |
| pw09_100.4 | 13514 | 11245.3 | 7810.1 | 12218 | 12319 | 10467 | 11376 |
| pw09_100.5 | 13574 | 11788.0 | 8183.2 | 12264 | 11581 | 10498 | 11383 |
| pw09_100.6 | 13640 | 11233.3 | 6246.5 | 11867 | 11413 | 10366 | 11486 |
| pw09_100.7 | 13501 | 10953.0 | 7968.1 | 12065 | 12305 | 10391 | 11646 |
| pw09_100.8 | 13593 | 9337.8 | 10466.8 | 12134 | 10935 | 10215 | 11388 |
| pw09_100.9 | 13658 | 10411.4 | 8455.2 | 12405 | 12390 | 10257 | 11683 |

**Tabela 2. Resultado dos Cortes das 58 Instâncias - Parte II**

| Instância | Solver | Node2Vec | | Spectral | | Características | |
|---|---|---|---|---|---|---|---|
| | | k-Means | Hierárquico | k-Means | Hierárquico | k-Means | Hierárquico |
| gka1b | 5744 | 4092.4 | 3526.8 | 3916 | 3886 | 4511 | 4462 |
| gka2b | 12451 | 4801.5 | 4657.0 | 9296 | 9595 | 9024 | 8967 |
| gka3b | 22115 | 5941.4 | 7962.7 | 17278 | 16690 | 16894 | 17167 |
| gka4b | 34857 | 24864.6 | 16626.3 | 28313 | 26581 | 27202 | 26861 |
| gka6b | 68189 | 11525.7 | 31112.8 | 56512 | 47281 | 54187 | 53803 |
| gka7b | 87428 | 36250.5 | 48407.1 | 55170 | 72651 | 67289 | 68412 |
| gka8b | 109969 | 17280.6 | 35256.7 | 87441 | 82456 | 79301 | 81400 |
| gka9b | 135757 | 10691.4 | 70662.2 | 119804 | 120331 | 117119 | 117504 |
| gka10b | 209946 | 49217.0 | 117645.9 | 182314 | 173544 | 171800 | 169395 |

Ao comparar os métodos de agrupamento, observou-se que a representação dos grafos do *Spectral Embedding* apresentou um desempenho superior ao *Node2Vec*, demonstrando uma maior capacidade de capturar padrões estruturais nos dados. A fim de analisar os resultados de forma geral, foi calculado a razão de aproximação média dos valores de corte fornecidos com o solver Biq Mac, e foi tirado uma média dessa taxa. Essa análise indicou que o método Spectral obteve uma razão de aproximação média de 82% para o k-Means e 78% para o hierárquico, enquanto o *Node2Vec* apresentou 72% e 69%, respectivamente.

Esses valores obtidos são promissores, indicando que esses métodos representação de grafo, se lapidados, a fim de reduzir essa oscilação mencionada anteriormente, podem produzir ótimos resultados, seja utilizando o método k-Means ou Hierarquico para o particionamento dos vértices. Assim, esses resultados sugerem que a representação espectral preserva melhor as estruturas inerentes ao grafo, resultando em agrupamentos mais alinhados ao esperado.

**Tabela 3. Resultado dos Cortes das 58 Instâncias - Parte III**

| Instância | Solver | Node2Vec | | Spectral | | Características | |
|---|---|---|---|---|---|---|---|
| | | k-Means | Hierárquico | k-Means | Hierárquico | k-Means | Hierárquico |
| g05_100.0 | 1430 | 1213.4 | 1192.1 | 1101 | 1188 | 1187 | 1176 |
| g05_100.1 | 1425 | 1191.9 | 1186.2 | 1213 | 1172 | 1154 | 1149 |
| g05_100.2 | 1432 | 1207.5 | 1211.4 | 1214 | 1272 | 1181 | 1183 |
| g05_100.3 | 1424 | 1183.1 | 1101.0 | 1257 | 1033 | 1165 | 1131 |
| g05_100.4 | 1440 | 1187.2 | 1160.9 | 1195 | 1180 | 1168 | 1176 |
| g05_100.5 | 1436 | 1204.9 | 1108.5 | 1202 | 931 | 1184 | 1161 |
| g05_100.6 | 1434 | 1172.6 | 1063.6 | 1208 | 1245 | 1123 | 1140 |
| g05_100.7 | 1431 | 1178.7 | 1147.6 | 1246 | 1221 | 1173 | 1179 |
| g05_100.8 | 1432 | 1223.9 | 1059.9 | 1247 | 1226 | 1182 | 1186 |
| g05_100.9 | 1430 | 1140.9 | 1209.7 | 1208 | 1238 | 1164 | 1173 |
| g05_60.1 | 532 | 431.8 | 402.4 | 407 | 358 | 418 | 403 |
| g05_60.2 | 529 | 429.6 | 395.9 | 381 | 423 | 404 | 399 |
| g05_60.3 | 538 | 426.5 | 392.5 | 452 | 326 | 417 | 408 |
| g05_60.4 | 527 | 428.9 | 368.9 | 461 | 394 | 421 | 413 |
| g05_60.5 | 533 | 402.9 | 382.1 | 360 | 350 | 391 | 389 |
| g05_60.6 | 531 | 426.8 | 382.1 | 448 | 429 | 413 | 411 |
| g05_60.7 | 535 | 416.8 | 390.5 | 424 | 337 | 401 | 402 |
| g05_60.8 | 530 | 441.5 | 414.9 | 403 | 347 | 408 | 405 |
| g05_60.9 | 533 | 418.4 | 390.2 | 425 | 434 | 417 | 418 |
| g05_80.0 | 929 | 744.3 | 733.3 | 788 | 786 | 755 | 764 |
| g05_80.1 | 941 | 761.6 | 750.5 | 797 | 753 | 762 | 772 |
| g05_80.2 | 934 | 771.8 | 715.3 | 823 | 718 | 767 | 765 |
| g05_80.3 | 923 | 746.5 | 708.4 | 721 | 731 | 741 | 739 |
| g05_80.4 | 932 | 744.4 | 681.6 | 735 | 758 | 728 | 726 |
| g05_80.5 | 926 | 712.6 | 683.3 | 586 | 402 | 667 | 654 |
| g05_80.6 | 929 | 698.1 | 721.2 | 779 | 791 | 734 | 732 |
| g05_80.7 | 929 | 758.3 | 723.1 | 754 | 724 | 746 | 749 |
| g05_80.8 | 925 | 750.4 | 722.7 | 744 | 731 | 739 | 738 |
| g05_80.9 | 933 | 766.3 | 741.9 | 775 | 753 | 752 | 756 |

Já a abordagem baseada nas características do grafo, ao contrário dos dois métodos anteriores, apresentou um desempenho mais consistente, com pouca variação nos resultados para cada instância. Ao aplicar o *k-means*, foi obtida uma taxa média de 91.4% de similaridade com os valores do solver, com um desvio padrão de 0,05. A menor taxa de aproximação foi de 75%, obtida na instância pw05_100.3, enquanto a maior foi de 97.9%, na instância g05_80.1. Essa abordagem obteve um total de 41 instâncias com uma taxa superior a 90%.

A superioridade da abordagem baseada em atributos locais, especialmente em grafos com alto coeficiente de aglomeração, possui uma justificativa teórica. Um alto coeficiente de aglomeração implica que a vizinhança de um vértice é densamente conectada. Para maximizar o corte, uma estratégia eficaz é isolar um vértice (ou um pequeno grupo de vértices) de sua vizinhança coesa. Atributos como o grau e a soma dos pesos das arestas servem como *proxies* diretos da "força" ou "centralidade" de um vértice em seu contexto local. O algoritmo KMeans, ao agrupar vértices com base na similaridade desses atributos, consegue separar eficientemente os "hubs" locais de seus vizinhos, uma ação que se alinha diretamente com o objetivo de criar um corte de alto valor ao quebrar essas conexões densas. Em contraste, Node2Vec e Spectral Clustering foram projetados para encontrar comunidades (problema análogo ao Min-Cut), agrupando vértices que estão

fortemente conectados, o que explica seu desempenho inferior na tarefa de Max-Cut.

Por outro lado, o método hierárquico teve mais oscilações, embora tenha mostrado uma consistência maior em comparação com o Node2Vec e o *Spectral*. Para o *k-means*, o método hierárquico obteve uma média de 83%, com desvio padrão de 0.14. O maior e o menor resultado foram registrados nas instâncias gka2b e g05_80.3, com taxas de 37% e 97.5%, respectivamente.

### 4.2. Ajuste dos Hiperparâmetros

Para realizar o ajuste dos hiperparâmetros, foi utilizada a ferramenta *GridSearch*, que testa diversas combinações de parâmetros para encontrar a configuração que maximiza um valor de avaliação específico. Neste caso, o critério escolhido foi o valor de corte, garantindo que os embeddings capturassem com maior fidelidade a organização do grafo que melhor se adaptaria ao Problema do Corte Máximo. Dessa forma, os melhores hiperparâmetros obtidos para o Node2vec foram:

- Dimensão do embedding: controla o número de dimensões da representação vetorial dos vértices. Os valores testados foram 32, 64, 128 e 256, sendo 256 o que apresentou melhor desempenho.
- Número de caminhadas por nó: define quantas vezes cada nó do grafo será visitado durante a geração dos caminhos aleatórios. Foram testados 5, 10, 20 e 30, e o melhor resultado foi obtido com 5 caminhadas.
- Comprimento das caminhadas: representa a quantidade de passos em cada caminhada aleatória. Os valores testados foram 50, 100, 200 e 300, e a melhor configuração foi 200 passos.
- Parâmetro $p$ (retorno ao nó anterior): regula a probabilidade de voltar ao nó anterior durante a caminhada. Os valores testados foram 0.01, 0.1, 0.5, 1 e 2, sendo 0.1 o mais adequado.
- Parâmetro $q$ (exploração do grafo): determina a tendência da caminhada em permanecer no mesmo cluster ou explorar novos vértices. Foram avaliados os valores 0.01, 0.1, 0.5, 1 e 2 e 0.01 apresentou melhor resultado.

Da mesma forma, os hiperparâmetros do *spectral embeddings* que demonstraram melhor desempenho no agrupamento do subconjunto $S$, responsável por retorna o valor de corte, foram selecionados com base na performance observada durante os experimentos:

- Número de autovetores $k$: representa a dimensionalidade do espaço onde os vértices serão projetados. Foram testados 4, 8, 16 e 32, sendo 8 o valor que gerou representações mais informativas.
- Tipo de Laplaciana: define a matriz utilizada no cálculo dos autovalores. Foram testadas laplaciana normalizada e não normalizada, com a versão normalizada apresentando melhor desempenho.
- Método de cálculo dos autovalores: determina quais autovalores são extraídos da matriz laplaciana. Foram testados os métodos SM (Smallest Magnitude) e LM (Largest Magnitude), sendo SM o mais eficiente na captura das estruturas de conectividade.

## 5. Conclusão

Este trabalho explorou uma metodologia inovadora para o Problema do Corte Máximo, convertendo-o em uma tarefa de agrupamento não supervisionado. A principal contribuição da pesquisa é a descoberta de que, para grafos com estrutura de vizinhança coesa (coeficiente de aglomeração $> 0.2$), **atributos locais simples superam representações de grafos complexas**. A abordagem que combina características locais (grau e estatísticas de pesos) com o algoritmo KMeans demonstrou um desempenho consistente, alcançando uma razão de aproximação média de **91.4%** em relação às melhores soluções conhecidas em 57 instâncias da *Big Mac Library*, com 41 delas superando 90%. Este resultado é significativamente superior aos obtidos com *embeddings* como Spectral (82%) e Node2Vec (72%).

Este achado é relevante por duas razões: primeiro, ele desafia a noção de que *embeddings* mais complexos são universalmente melhores, mostrando que para o Max-Cut, a informação local é mais pertinente do que a estrutura comunitária global que Node2Vec e Spectral são projetados para capturar. Segundo, ele oferece uma **solução altamente escalável e eficiente**. A extração de atributos locais possui custo computacional linear, posicionando esta abordagem como uma alternativa viável a métodos de alta complexidade, como a programação semidefinida, para grafos de média e grande escala. A filtragem por coeficiente de aglomeração também se mostrou um passo metodológico crucial para garantir a estabilidade das soluções.

A principal limitação reside na dependência da coesão estrutural do grafo, com desempenho degradado em estruturas esparsas (coeficiente $< 0.2$). Como trabalhos futuros, propõe-se:

1. **Análise Estatística:** Realizar testes de significância (e.g., teste de Wilcoxon) para validar formalmente as diferenças de desempenho observadas.
2. **Integração com *Graph Neural Networks* (GNNs):** Utilizar GNNs para aprender representações de vértices que combinem atributos locais com informações hierárquicas da vizinhança, potencialmente otimizadas para a tarefa de Max-Cut.
3. **Acoplamento a Meta-heurísticas:** Usar as partições geradas pelo agrupamento como soluções iniciais (sementes) para algoritmos de refinamento, como a busca local.

## Referências


Barahona, F. (1982). On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241.

Barrat, A., Barthélemy, M., Pastor-Satorras, R., and Vespignani, A. (2004). The architecture of complex weighted networks. *PNAS*, 101(11):3747–3752.

Farhi, E., Goldstone, J., and Gutmann, S. (2014). A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305.

Freeman, L. C. (1978). Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. Freeman.

Ghahramani, Z. (2004). Unsupervised learning. *Advanced lectures on machine learning*, pages 72–112.

Goemans, M. X. and Williamson, D. P. (1995). Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

Hamilton, W. L., Ying, R., and Leskovec, J. (2017). Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74.

Liers, F., Nieberg, T., and Pardella, G. (2011). Via minimization in vlsi chip design-application of a planar max-cut algorithm.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Newman, M. (2018). *Networks*. Oxford university press.

Opsahl, T., Agneessens, F., and Skvoretz, J. (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social networks*, 32(3):245–251.

Rendl, F., Rinaldi, G., and Wiegele, A. (2010a). Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121:307–335.

Rendl, F., Rinaldi, G., and Wiegele, A. (2010b). Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Programming*, 121(2):307.

Schaeffer, S. E. (2007). Graph clustering. *Computer science review*, 1(1):27–64.

Tsitsulin, A., Palowitch, J., Perozzi, B., and Müller, E. (2023). Graph clustering with graph neural networks. *JMLR*, 24:1–21.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, 17:395–416.

Ward Jr, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244.

Wiegele, A. (2007). Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. *Preprint*, 51.