57º SBPO
SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
Gramado • RS
DE 5 A 9 DE OUTUBRO DE 2025

TREINAMENTO DE MODELOS SUPERVISIONADOS COM INSTÂNCIAS ESTRUTURAIS PARA O CORTE MÁXIMO

Jailon William B. Oliveira da Silva, Carlos Vinicio D. Araújo, Pablo Luiz Braga Soares
Laboratório de Pesquisa & Desenvolvimento do NEMO, Universidade Federal do Ceará
Avenida Felipe Santiago – Nº 411, 62.900-420, Russas/CE, Brasil
{williambrunocc, carlosvinicio}@alu.ufc.br, pablo.soarea@ufc.br

RESUMO

Este estudo aplica algoritmos de Aprendizado de Máquina supervisionado ao Problema do Corte Máximo, utilizando instâncias da *Biq Mac Library*. Modelos como Naive Bayes, kNN, Regressão Logística e Redes Neurais Feedforward foram treinados em grafos de 20 a 500 vértices, pré-processados com PCA e normalização. A Regressão Logística destacou-se com 96.34% de aproximação aos valores de corte do solver Biq Mac, enquanto, no geral, os modelos mostraram ser eficaz em grafos com coeficiente de aglomeração elevados e centralidade de intermediação homogênea. A análise evidenciou que a seleção de instâncias com características estruturais compatíveis (ex: densidade > 0.4) é crucial, sendo a redução de dimensionalidade via PCA crítica para grafos esparsos. Os resultados destacam o potencial de modelos especializados em subconjuntos estruturais de grafos, oferecendo bases para aplicações em otimização combinatória de larga escala.

PALAVRAS CHAVE. Problema do Corte Máximo, Aprendizado de Máquina Supervisionado, Análise de Instâncias.

Tópicos: EST&AM – PO Analytics em Estatística e Aprendizado de Máquina, AS&DS – Análise e Ciência de Dados

ABSTRACT

This study applies supervised Machine Learning algorithms to the Max-Cut Problem, using instances from the Biq Mac Library. Models such as Naive Bayes, kNN, Logistic Regression, and Feedforward Neural Networks were trained on graphs ranging from 20 to 500 vertices, preprocessed with PCA and normalization. Logistic Regression stood out with a 96.34% approximation to the cut values obtained by the Biq Mac solver, while overall, the models proved effective on graphs with high clustering coefficients and homogeneous betweenness centrality. The analysis revealed that selecting instances with compatible structural features (e.g., density > 0.4) is crucial, with dimensionality reduction via PCA being critical for sparse graphs. The results highlight the potential of specialized models for structural graph subsets, providing a foundation for large-scale combinatorial optimization applications.

KEYWORDS. Max-Cut problem. Machine learning. Supervised learning.

Paper topics: EST&AM – PO Statistical and Machine Learning Analytics, AS&DS –Data Analysis and Science

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

1. Introdução

Na Teoria dos Grafos, diversos problemas são classificados como NP-completos ou NP-difíceis, como o Isomorfismo de Subgrafos, o Caixeiro Viajante e a Cobertura de Vértices [Garey e Johnson, 1979]. Entre eles, destaca-se o Problema do Corte Máximo, que consiste em encontrar, em um grafo ponderado $G = (V, E)$, o subconjunto de vértices $S \subseteq V$ que maximiza o peso do corte, definido como a soma dos pesos das arestas com extremidades em S e $V \setminus S$ [Boros e Hammer, 1991].

Dada a complexidade combinatória desses problemas e sua relevância prática, surgiram algoritmos classificados em exatos e aproximados. Para problemas NP-completos ou NP-difíceis, o espaço de soluções cresce exponencialmente com o tamanho da entrada, tornando os métodos exatos computacionalmente inviáveis. Métodos aproximados, por sua vez, visam soluções próximas da ótima, sem garantir otimalidade [Blum e Roli, 2003].

O Aprendizado de Máquina (AM) tem avançado significativamente, com aplicações em diversas áreas, como sequenciamento genômico [Libbrecht e Noble, 2015], diagnóstico de doenças [Oliveira et al., 2023] e predição de evasão escolar [de Almeida Teodoro e Kappel, 2020]. No contexto de grafos, AM já foi aplicado à coloração de grafos [Goudet et al., 2022], ao Caixeiro Viajante [Miki et al., 2018] e à Cobertura de Vértices [Gianinazzi et al., 2021].

O objetivo dessa pesquisa é investigar o uso de AM na resolução do Problema do Corte Máximo, um problema NP-difícil [Karp, 1972], com aplicações em redes sociais [Agrawal et al., 2003], agrupamento de dados [Otterbach et al., 2017], segmentação de imagens [de Sousa et al., 2013] e projeto de circuitos VLSI [Liers et al., 2011]. Dito isso, diferentes abordagens baseadas em AM têm sido propostas na literatura. Por exemplo, Gu e Yang [2018] utiliza redes de ponteiros com atenção e aprendizado supervisionado, enquanto Gu e Yang [2020] combina aprendizado supervisionado e por reforço. Já Yao et al. [2019] emprega Redes Neurais em Grafos, explorando a estrutura inerente dos dados.

Este trabalho diferencia-se ao utilizar instâncias de pequena escala para treinar modelos supervisionados, viabilizando o aprendizado em tempo razoável. O objetivo é aplicar os modelos treinados em instâncias de maior porte, visando soluções eficientes e escaláveis para o Problema do Corte Máximo. Além de realizar uma análise das instâncias da literatura, categorizando-as em grupos de acordo com características que sirvam como base de treinamento para os modelos de aprendizado de máquina.

O restante deste trabalho está estruturado da seguinte forma: a Seção 2 apresenta os conceitos teóricos necessários para a compreensão da pesquisa, com ênfase no Problema do Corte Máximo e nas técnicas de Aprendizado de Máquina utilizadas; a Seção 3 descreve a metodologia adotada para o desenvolvimento da solução proposta; a Seção 4 expõe e analisa os resultados obtidos; por fim, a Seção 5 discute as conclusões alcançadas e sugere possíveis direções para trabalhos futuros.

2. Fundamentação Teórica

Este tópico aborda os fundamentos teóricos necessários para a compreensão deste trabalho. A Seção 2.1 aborda características estruturais dos grafos importante para o entendimento da pesquisa. Ja Seção 2.2, é apresentado o Problema do Corte Máximo, com ênfase em sua definição formal, natureza de otimização e complexidade computacional. Ná na Seção 2.3, são introduzidos os principais conceitos relacionados ao Aprendizado de Máquina, destacando as tarefas preditivas e descritivas, bem como os tipos de aprendizado.

Por fim, na Seção 2.4, são apresentados os principais algoritmos de aprendizado supervisionado utilizados neste trabalho. Em adicional, esta última também apresenta a Análise de Componentes Principais (PCA), que foi utilizada no pre-processamento dos dados.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

2.1. Conceitos introdutórios da teoria dos grafos

O grau de um vértice pode ser utilizado para determinar sua importâncias em relação as conexões dentro do grafo, por meio de medidas de centralidade. Por sua vez, a centralidade de grau de um vértice é determinada pela quantidade de arestas que os tocam diretamente, refletindo sua relevância em termos de conexões dentro do grafo. Já a centralidade de intermediação, está interessada na quantidade de caminhos mínimos que passam pelo vértice, ou seja, em intermediar conexões, medindo sua importância em relação ao fluxo de conexões dentro do grafo [Latora et al., 2017].

O termo subgrafo é utilizado para referir-se a grafos contidos em outros. Em outras palavras, dado um grafo $G = (V, E)$ e um grafo $G' = (V', E')$, G' será um subgrafo de G se $V' \subseteq V$ e $E' \subseteq E$, podendo ser expressado como $G' \subseteq G$ [Diestel, 2010]. A quantidade de vértices e arestas de um grafo pode ser expressa por $|V|$ e $|E|$, respectivamente. Desta forma, um grafo (ou subgrafo) simples é denso se o valor de $|E|$ estiver próximo de $|V|^2$, pois implica que quase todas as possíveis arestas já estão presentes. Caso contrário, se o grafo possuir poucas arestas, é classificado como esparso [Jonathan L. Gross, 2013].

Por último, a fim de identificar formações de grupos locais, isto é, de vértices e arestas dentro do grafo, é utilizado coeficiente de agrupamento. Para tal, é utilizado a quantidade de triângulos, que é subgrafos completos de três vértices, que são formados em torno de um determinado vértice, em relação ao número total de triângulos que poderiam ser formados [Barabási, 2016].

2.2. Problema do Corte Máximo

Dado um grafo não direcionado $G = (V,\ E)$. Definimos como corte o ato de separar um conjunto de vértices em dois subconjuntos disjuntos, denotados S e $\overline{S}$, tal que $S \cup \overline{S} = V$ e $S \cap S = \emptyset$. O tamanho do corte é determinado pelo número de arestas que cruzam as partições, ou seja, que partem de um subconjunto e chegam no outro. Em outras palavras, as arestas de corte são as arestas uv, tal que $uv \in E$ e $u \in S$ enquanto $v \in \overline{S}$, ou alternativamente, $u \in \overline{S}$ e $v \in S$. Por outro lado, as arestas que conectam vértices pertencentes a um mesmo subconjunto, seja S ou $\overline{S}$, são conhecidas como arestas não cortadas [Sipser, 2012].

O problema de Corte Máximo é um problema de otimização combinatória NP-difícil [Karp, 1972], que busca encontrar um subconjunto $S \subseteq V$ em um grafo $G = (V, E)$ que maximiza o peso do corte. O corte é a separação do conjunto V em dois subconjuntos disjuntos S e $\overline{S}$, e o peso do corte é a soma das arestas cujos extremos estão em S e $\overline{S}$, onde $S \cup \overline{S} = V$ e $S \cap \overline{S} = \emptyset$ [Sipser, 2012; Boros e Hammer, 1991]. Adicionalmente, problemas de otimização combinatório não buscam apenas uma solução viável, isto é, envolve o uso eficiente de recursos disponíveis, a otimização do tempo para realizar tarefas, a maximização de lucros e minimização de prejuízos conforme discutido por [Miyazawa e de Souza, 2015].

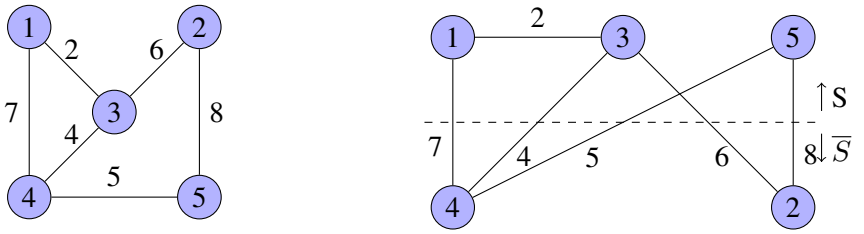
A Figura 1 ilustra uma instância do Problema do Corte Máximo. Na Figura 1 (a), temos um grafo $G = (V, E)$ com $|V| = 5$ vértices e $|E| = 6$ arestas ponderadas. A Figura 1 (b) apresenta a solução ótima, com partições $S = \{1, 3, 5\}$ e $\overline{S} = \{4, 2\}$. As arestas de corte são $\{\{1, 4\}, \{2, 3\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$, totalizando peso 30. A única aresta não cortada é $\{1, 3\}$.

2.3. Aprendizado de Máquina

O Aprendizado de Máquina (AM) é a área que permite que ferramentas operem de forma mais autônoma, reduzindo a intervenção humana. Isso possibilita que computadores aprendam sem serem explicitamente programados, ou seja, que aprendam a partir de dados [Géron, 2019].

Esse aprendizado ocorre por meio do aprendizado indutivo, que generaliza padrões a partir de exemplos. Um conceito é aprendido por inferência indutiva aplicada sobre esses exemplos,

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br



(a) Grafo não direcionado com arestas ponderadas

(b) Solução ótima do Corte Máximo

Figura 1: Exemplo de uma instância do Problema do Corte Máximo

sendo esse aprendizado pode ser dividido em Aprendizado Supervisionado e Não Supervisionado. O algoritmo que realiza essa aprendizagem é denominado indutor [Monard e Baranauskas, 2003].

Assim, o AM envolve a criação e uso de modelos treinados a partir de dados. O algoritmo analisa os dados para identificar padrões e construir uma relação matemática ou probabilística entre variáveis, chamada de modelo. No aprendizado supervisionado, os dados possuem rótulos com as respostas esperadas. Já no não supervisionado, tais respostas não estão presentes [Grus, 2016].

2.4. Algoritmos de aprendizado de máquina

O k-Nearest Neighbors (kNN) é um modelo não paramétrico baseado em similaridade, que classifica registros pela votação majoritária entre os k vizinhos mais próximos, utilizando funções de distância [Xiong e Yao, 2021; Aggarwal, 2014]. O Naive Bayes, que tem como base o teorema de Bayes, assume independência entre atributos para estimar probabilidades condicionais e a fim de atribuir classes [Parsian, 2015; Rish et al., 2001].

A Regressão Logística, embora nomeada como regressão, é um classificador binário que modela probabilidades via função sigmoide, estabelecendo uma relação linear entre os atributos de entrada e a variável de saída. Seus pesos são ajustados para maximizar a verossimilhança [Raschka, 2015; Aggarwal, 2014]. Já as Redes Neurais Artificiais (RNAs) são inspiradas no processamento cerebral, com neurônios interconectados que processam entradas via combinações lineares e funções de ativação. Arquiteturas *feedforward* propagam informações unidirecionalmente, passando pela camada de entrada, seguido da camada oculta até a saída, ajustando pesos via *backpropagation* [McCulloch e Pitts, 1943; Faceli et al., 2011].

Por fim, é importante citar a Análise de Componentes Principais (PCA), que é uma técnica de aprendizado de máquina não supervisionado utilizada para a redução de dimensionalidade e extração de informações relevantes de um conjunto de dados complexos, sendo aplicada em diversos campos, como neurociência e computação gráfica [Shlens, 2014].

3. Metodologia

A Seção 3.1 detalha o procedimento de aquisição dos dados destinados ao treinamento e avaliação do modelo supervisionado. Esses dados serão adquiridos por meio de instâncias da *Biq Mac Library*, juntamente com suas respectivas respostas, utilizando um *solver* baseado em método exato. Já a Seção 3.2 detalha a etapa de pré-processamento dos dados, onde é realizada a criação da base de dados e o tratamento dos dados para serem utilizados no treinamento do modelo. Por fim, a Seção 3.3 dá foco em como será a estrutura do treinamento do modelo, especificando a abordagem escolhida para o treinamento, enquanto a Seção 3.4 especifica ferramentas que serão utilizadas.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

3.1. Coleta de dados

Para esta pesquisa, foram selecionadas 330 instâncias da *Biq Mac Library* [Wiegele, 2007], uma fonte reconhecida na área do Problema do Corte Máximo. A biblioteca oferece uma coleção diversificada de grafos com tamanhos entre 20 e 500 vértices, contemplando diferentes características estruturais, como número de triângulos, densidade, coeficiente de aglomeração, centralidade e intermediação.

Para realizar a aquisição dos dados necessários para o treinamento do modelo supervisionado, é essencial não apenas as instâncias em si, mas também as respostas correspondentes, ou seja, as partições dos vértices. Portanto, essas instâncias foram submetidas ao *solver Biq Mac*, um algoritmo exato que se baseia no método *Branch & Bound* e utiliza Programação Semi-Definida (SDP) [Rendl et al., 2010].

O Biq Mac é projetado especificamente para problemas de Corte Máximo e Programação Quadrática 0-1, que são, em geral, problemas de otimização não convexos. Embora seja um solver exato, ele não garante encontrar o valor ótimo em tempo hábil para instâncias de maior porte (tipicamente acima de algumas centenas de vértices), que se tornam computacionalmente intratáveis.

3.2. Pré-processamento

O pré-processamento foi fundamental para viabilizar o uso de aprendizado de máquina. O principal desafio era como representar grafos de tamanhos variados (de 20 a 500 vértices) com um vetor de características de dimensão fixa. A abordagem adotada foi a seguinte: para cada vértice v_i em um grafo, seu vetor de conexões (a linha i da matriz de adjacência) foi extraído. Como esses vetores tinham comprimentos diferentes, eles foram padronizados através de *padding* com zeros até atingirem o comprimento do maior grafo da base (500). Isso resultou em um conjunto de dados onde cada vértice era representado por um vetor de 500 dimensões.

Sobre este conjunto de dados de alta dimensionalidade, aplicou-se a Análise de Componentes Principais (PCA). O objetivo foi reduzir a dimensionalidade das conexões para um número fixo de componentes, capturando a maior parte da variância. Testes indicaram que 20 componentes ofereciam um bom equilíbrio entre a preservação da informação e a eficiência computacional, sendo este o valor utilizado. Assim, cada vértice de qualquer grafo passou a ser representado por 20 características oriundas do PCA, além de outras métricas extraídas.

Os dados também foram normalizados para uniformizar a escala dos atributos. Além das conexões representadas pelo PCA, foram extraídas características adicionais para cada vértice: seu grau, a média, o menor e o maior peso das arestas conectadas, e o potencial total (soma dos pesos de suas arestas). Essas métricas complementam a representação estrutural, enriquecendo o conjunto de atributos.

Uma análise da partição ótima fornecida pelo solver Biq Mac revelou que os conjuntos S e $\overline{S}$ não eram necessariamente balanceados. Embora técnicas de reamostragem como SMOTE ou RandomUnderSampler pudessem ser aplicadas para equilibrar as classes, optou-se por não utilizá-las nesta fase para preservar a estrutura original da solução ótima. A investigação do impacto do balanceamento de classes fica como uma sugestão para trabalhos futuros.

3.3. Treinamento dos modelos

Inicialmente, os modelos foram treinados com uma divisão aleatória das 330 instâncias da *Biq Mac Library*, em dois grupos de 165 instâncias cada (experimento 1). Cada grupo foi alternadamente utilizado para treinamento e teste, garantindo que a instância alvo não estivesse presente na base de treinamento. Entretanto, essa abordagem resultou em desempenho insatisfatório para a maioria dos modelos, exceto o kNN, que apresentou resultados moderados em 57 instâncias.

Para investigar essa disparidade, realizou-se uma análise exploratória das 57 instâncias com melhor desempenho. O algoritmo *k-means* foi utilizado com o suporte dos métodos da silhueta e do cotovelo para definir a quantidade de *clusters* ideais.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

O k-means foi escolhido por sua eficiência computacional e simplicidade, sendo adequado para uma análise exploratória inicial. Embora outros algoritmos como o k-medoids sejam mais robustos a outliers, o k-means foi suficiente para o objetivo de identificar a principal característica estrutural que segmentava as 57 instâncias alvo das 330. O método do cotovelo sugeriu k=3 e o da silhueta, k=2. Ambos os valores foram testados, e a clusterização com k=2 forneceu a separação mais clara e consistente das instâncias de interesse, alinhando-se com a análise visual dos dados. Essa etapa identificou que as instâncias alvo possuíam o coeficiente de aglomeração elevado.

Com base nessa descoberta, adotou-se uma segunda estratégia (experimento 2): agrupar as instâncias que possuíam o coeficiente de aglomeração elevado e realizar os treinamentos selecionando 56 instâncias e o teste com apenas uma. Assim, uma determinada instância obtinha o valor de corte com base no treinamento dos modelos com os outros 56 grafos.

A abordagem de aprendizado de máquina transforma o problema do Corte Máximo em uma tarefa de classificação binária para cada vértice. O modelo treinado recebe as características de um vértice v_i e retorna a probabilidade de ele pertencer à partição S. Com um limiar de 0.5, cada vértice é classificado como pertencente a S (classe 1) ou a $\overline{S}$ (classe 0). Após a classificação de todos os vértices, a solução para o Corte Máximo é formada. O valor do corte é então calculado somando-se os pesos de todas as arestas que conectam um vértice em S a um vértice em $\overline{S}$. Esta partição é sempre viável, e sua qualidade é medida pela proximidade ao valor de corte obtido pelo solver.

Com a base preparada, diferentes algoritmos de aprendizado supervisionado foram aplicados: Naive Bayes; *k-Nearest Neighbors* (kNN); Regressão Logística; e Redes Neurais Artificiais (*feedforward*). Para o kNN, o valor de k foi determinado através de uma busca em grade (grid search) com validação cruzada, onde o valor $k = 5$ apresentou o melhor desempenho de forma consistente. Todos os modelos foram treinados para mapear as características dos vértices às suas partições, visando maximizar o corte.

3.4. Ferramentas utilizadas

Para a implementação, foi utilizado o Python como linguagem de programação, fazendo uso da IDE *Visual Studio Code* com a extensão *Jupyter*. Essa extensão permite ao programador separar o código em células dentro do ambiente de desenvolvimento, possibilitando a execução de um trecho do código de forma individual.

Entre as bibliotecas, destacam-se o Pandas e o *NumPy* para manipulação e processamento de dados, o *Scikit-learn* para implementação de algoritmos de aprendizado de máquina, ajuste de hiperparâmetros e avaliação de modelos, e o *Networkx* para análise de grafos, incluindo cálculos de métricas como centralidade de grau e excentricidade.

4. Análise dos resultados

Esta seção apresenta os resultados obtidos através desta pesquisa. O desempenho dos modelos foi avaliado utilizando a métrica “Proximidade da Solução”, definida como a razão percentual entre o valor de corte alcançado pelo modelo de aprendizado de máquina e o valor de referência do solver Biq Mac.

Esta métrica quantifica quão perto a solução do modelo chegou da solução de alta qualidade fornecida pelo solver. A Seção 4.1 apresenta os resultados do experimento 1. A Seção 4.2 demonstra a análise das 57 instâncias que tiveram resultados moderados no modelo kNN. Por fim, a Seção 4.3 aborda o experimento 2, que performou significativamente superior.

4.1. Treinamento dos modelos com várias instâncias aleatórias

Ao realizar o treinamento dos modelos supervisionados com as instâncias da *Biq Mac Library*, as bases de dados foram divididas aleatoriamente em dois grupos de tamanhos iguais. Todos os modelos foram treinados com a mesma base de dados, utilizando um grupo para treinamento e

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

o outro para previsão das instâncias não presentes no conjunto de origem. Essa estratégia permitiu avaliar a capacidade de generalização dos modelos em dados não vistos durante o treinamento, seguindo a abordagem de concatenar instâncias selecionadas e posteriormente particionar aquelas não aprendidas.

Devido às diferenças nas propriedades dos grafos, como centralidade de grau e coeficiente de aglomeração, observou-se uma grande variação entre os conjuntos de dados. Isso resultou em correlações distintas entre as variáveis das bases de dados de diferentes instâncias, tornando o processo de generalização dos modelos um desafio. Essas variações dificultaram a capacidade dos modelos de identificar padrões consistentes e aplicáveis a grafos com características estruturais variadas.

Os resultados obtidos após o treinamento e teste indicaram que, das 330 instâncias testadas, os modelos não apresentaram um desempenho satisfatório geral, com apenas 58 instâncias exibindo valores significativamente mais elevados do que as demais, ainda que relativamente baixos.

É importante destacar que o kNN foi o modelo que apresentou o melhor desempenho geral. Isso sugere que o aprendizado baseado em instâncias, que armazena as informações de cada amostra, foi eficaz em detectar padrões de instâncias que se assemelham aos grafos já aprendidos pelo modelo. No geral, os resultados obtidos se alinha com a expectativa de que a modelagem seria mais eficiente para um subconjunto de grafos que compartilham propriedades estruturais específicas, reforçando a importância de considerar a similaridade estrutural entre as instâncias nesse processo de treinamento.

Assim como em bases de dados tradicionais é importante remover valores extremos e realizar o pré-processamento dos dados para selecionar aqueles mais semelhantes, a fim de melhorar a qualidade do treinamento, esse mesmo princípio pode ser aplicado às instâncias do Problema do Corte Máximo. Para otimizar o aprendizado dos modelos, é necessário selecionar apenas as instâncias que apresentam similaridades estruturais, aumentando, assim, o valor de corte obtido pelo modelo.

4.2. Análise das instâncias que proporcionaram melhor desempenho aos modelos

A análise focou nas características estruturais das 58 instâncias-alvo, avaliando suas estatísticas básicas e aplicando os métodos do cotovelo (*Elbow Method*) e da silhueta (*Silhouette method*) para determinar a quantidade ideal de grupos.

O coeficiente de silhueta mede a coesão de um ponto dentro de seu cluster em comparação com outros clusters; o número de clusters que maximiza a pontuação média de silhueta é considerado ótimo. O algoritmo k-means foi utilizado para realizar a separação, com o objetivo de identificar a característica chave que contribuiu para o bom desempenho do modelo nessas instâncias.

A Figura 2, demonstra a estimativa de densidade kernel (KDE) para a característica de densidade das instâncias-alvo, em comparação com toda a coleção de instâncias, sendo que estas últimas variam de 0.01 até 1.1, com uma média de 0.49 e desvio padrão de 0.37. Dito isso, o modelo desenvolvido não lida bem com grafos esparsos, o que pode ser pelo uso do PCA nas colunas que representam as conexões e seus respectivos pesos. Em outras palavras, como os grafos são muito esparsos, apresentando poucas conexões entre os vértices, ou seja, contendo muitos valores 0, a matriz resultante possui baixa variabilidade. Isso dificulta a eficácia do PCA na redução de dimensionalidade, pois há pouca informação nas arestas com peso em comparação às arestas que não existem.

Como a centralidade de intermediação, que avalia a importância dos vértices em intermediar o fluxo de conexões, mostrou valores extremos e consistentes em todas as 58 instâncias, com um desvio padrão próximo de 0, ela pode ser um dos principais fatores que explicam o melhor desempenho do modelo nessas instâncias-alvo. Com isso em mente, foram utilizados os métodos

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

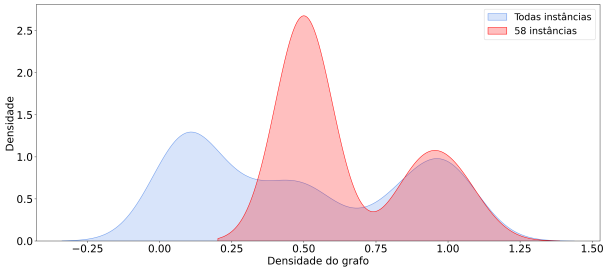


Figura 2: KDE da característica de densidade.

do cotovelo (indicou 3) e silhueta (indicou 2) para determinar a quantidade ideal de grupos, além da aplicação do k-means para particionar todas as 330 instâncias, levando em consideração apenas a centralidade de intermediação.

Ao realizar a partição das instâncias com base na centralidade de intermediação, seja separando em 3 ou 2 grupos, observou-se que as 58 instâncias foram sempre agrupadas no mesmo conjunto, juntamente com outras 10 instâncias. Esse fato sugere que a centralidade de intermediação é uma das características-chave para o bom desempenho do modelo. No entanto, a presença de outras instâncias nesse grupo, onde o modelo não teve um desempenho satisfatório, embora em pequena quantidade, indica que essa característica, por si só, não determina o sucesso do modelo. Isso sugere que outros fatores podem estar influenciando o desempenho e precisam ser considerados.

A característica que registra a média do coeficiente de aglomeração para cada vértice apresenta uma mediana de 0.5, com um desvio padrão muito baixo de 0.09. O grafo com a menor média possui um valor de 0.24, enquanto o grafo com a maior média tem 0.51. Adicionalmente, as medianas das características que registraram os valores mínimos e máximos do coeficiente de aglomeração são 0.46 e 0.54, respectivamente. No geral, esses dados sugerem uma quantidade moderada de agrupamentos locais.

A Figura 3 apresenta a estimativa de densidade kernel (KDE) para as colunas de média, máximo e mínimo do coeficiente de aglomeração observados nas instâncias, ilustrando a distribuição dos valores dentro do intervalo observado em cada uma dessas colunas. A 3 (a) mostra a distribuição para todas as instâncias, enquanto a 3 (b) foca no conjunto das 58 instâncias-alvo. Na 3 (a), observa-se a presença de duas populações distintas: uma classe majoritária com valores muito próximos de zero, incluindo até valores negativos, e uma classe minoritária composta exclusivamente por valores positivos. A 3 (b) demonstra precisamente essa classe minoritária, que corresponde às instâncias-alvo observadas, reforçando a separação entre essas duas populações.

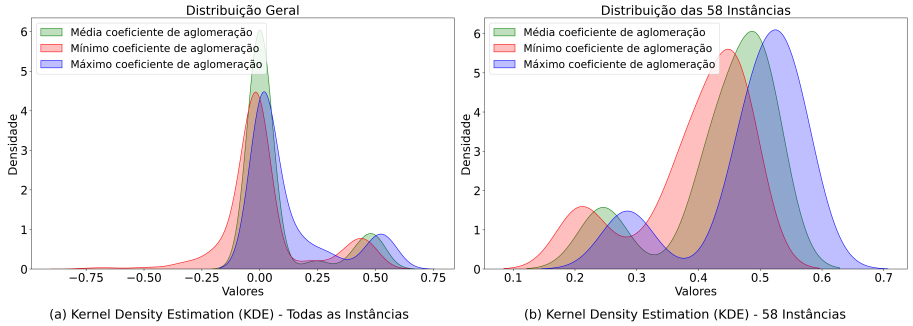


Figura 3: KDE da característica de Coeficiente de aglomeração.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

Além disso, ao aplicar o algoritmo k-means, todas as 58 instâncias foram corretamente separadas das demais, reforçando a ideia de que o coeficiente de aglomeração é uma característica chave para diferenciar as instâncias-alvo. Isso sugere que para que o modelo consiga ter um bom desempenho ao realizar o particionamento dos vértices para o Problema do Corte máximo, de novas instâncias, é necessário que elas possuem valores de coeficiente de aglomeração positivos e que não estejam muito próximo de zero.

4.3. Treinamento dos modelos com instâncias de maior coeficiente de aglomeração

A fim de confirmar a hipótese levantada na seção 4.1, de que a qualidade do treinamento dos modelos se sobressaem ao utilizar instâncias no treinamento que compartilham características estruturais semelhantes, foi realizado um novo experimento focando o treinamento e teste na população de instâncias minoritária e discutidas na seção 4.2.

Com o agrupamento de instâncias baseadas no coeficiente de aglomeração, os modelos passaram a se especializar em grafos que compartilham essa característica estrutural, especificamente as propriedades desse subconjunto de instâncias. Para testar essa abordagem, foi realizada uma validação cruzada, onde 57 instâncias foram utilizadas como conjunto de treinamento, enquanto uma servia como teste, até que todas as instâncias fossem avaliadas.

Esse processo revelou desempenhos variados entre os modelos nas 57 instâncias, quanto à proximidade com os valores de corte do solver Biq Mac. A Regressão Logística se destacou com média de 96.34% e desvio padrão de 1.49%, indicando alta consistência. Naive Bayes, RNA e kNN obtiveram médias de 88.73%, 87.28% e 86.87%, com desvios de 6.8%, 3.3% e 3.7%, respectivamente. Desta forma, os resultados indicam que os modelos generalizam melhor quando treinados em grafos com estruturas similares, confirmando a hipótese levantada, em especial, com coeficiente de aglomeração altos.

Os resultados detalhados deste experimento estão disponíveis nas Tabelas 1, 2 e 3, onde o valor de corte obtido pelos modelo supervisionados foi comparado com o valor fornecido pelo *solver Biq Mac*. O desempenho dos modelos foi significativamente superior nas instâncias com coeficiente de aglomeração mais elevado, reforçando a importância de considerar essa propriedade específica dos grafos ao treinar o modelo.

Tabela 1: Resultados para o Treinamento especializado - Parte I

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
gka1b	5744	4678.0	4330.0	5353.0	4652.0
gka2b	12451	9514.0	9757.0	12291.0	10320.0
gka3b	22115	16861.0	19459.0	21702.0	18761.0
gka4b	34857	29743.0	29856.0	33809.0	31412.0
gka5b	49942	45134.0	45281.0	47822.0	44976.0
gka6b	68189	58950.0	55934.0	67188.0	60696.0
gka7b	87428	65660.0	77946.0	84902.0	77778.0
gka8b	109969	95436.0	100895.0	108287.0	102557.0
gka9b	135757	97029.0	122643.0	130581.0	124552.0
gka10b	209946	118596.0	194859.0	206584.0	196701.0

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

Tabela 2: Resultados para o Treinamento especializado - Parte II

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
g05_60.1	532	492.0	464.0	504.0	460.0
g05_60.2	529	460.0	433.0	514.0	452.0
g05_60.3	538	474.0	450.0	518.0	440.0
g05_60.4	527	489.0	455.0	497.0	461.0
g05_60.5	533	481.0	470.0	526.0	433.0
g05_60.6	531	467.0	455.0	507.0	454.0
g05_60.7	535	483.0	422.0	515.0	427.0
g05_60.8	530	496.0	465.0	494.0	462.0
g05_60.9	533	511.0	421.0	519.0	477.0
g05_80.0	929	833.0	812.0	885.0	809.0
g05_80.1	941	875.0	764.0	884.0	801.0
g05_80.2	934	851.0	805.0	890.0	832.0
g05_80.3	923	854.0	801.0	891.0	810.0
g05_80.4	932	840.0	788.0	880.0	793.0
g05_80.5	926	856.0	789.0	882.0	751.0
g05_80.6	929	855.0	807.0	899.0	820.0
g05_80.7	929	889.0	769.0	894.0	837.0
g05_80.8	925	862.0	814.0	901.0	795.0
g05_80.9	923	831.0	813.0	894.0	777.0
g05_100.0	1430	1300.0	1236.0	1351.0	1226.0
g05_100.1	1425	1352.0	1289.0	1399.0	1277.0
g05_100.2	1432	1357.0	1245.0	1393.0	1275.0
g05_100.3	1424	1345.0	1233.0	1368.0	1253.0
g05_100.4	1440	1309.0	1270.0	1375.0	1286.0
g05_100.5	1436	1319.0	1276.0	1392.0	1288.0
g05_100.6	1434	1354.0	1239.0	1378.0	1282.0
g05_100.7	1431	1326.0	1257.0	1390.0	1242.0
g05_100.8	1432	1280.0	1281.0	1346.0	1293.0
g05_100.9	1430	1312.0	1265.0	1351.0	1234.0

Tabela 3: Resultados para o Treinamento especializado - Parte II

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
pw05_100.0	8190	7619.0	7018.0	7976.0	6998.0
pw05_100.1	8045	7149.0	6999.0	7834.0	6836.0
pw05_100.2	8039	7494.0	6934.0	7703.0	6654.0
pw05_100.3	8139	7568.0	6748.0	7730.0	7124.0
pw05_100.4	8125	6803.0	6998.0	7676.0	6658.0
pw05_100.5	8169	6488.0	7033.0	7827.0	7166.0
pw05_100.6	8217	6752.0	7223.0	7751.0	6924.0
pw05_100.7	8249	7590.0	7014.0	7754.0	6937.0
pw05_100.8	8199	7377.0	6893.0	7989.0	7168.0
pw09_100.0	13585	12331.0	12323.0	13184.0	11597.0

Continua na próxima página

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

Continuação da Tabela 3

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
pw09_100.1	13417	12009.0	12588.0	12875.0	11659.0
pw09_100.2	13461	12520.0	12296.0	13274.0	12565.0
pw09_100.3	13656	12169.0	12217.0	13172.0	12428.0
pw09_100.4	13514	11773.0	12134.0	13250.0	12451.0
pw09_100.5	13574	11742.0	11883.0	13347.0	12044.0
pw09_100.6	13640	12440.0	12551.0	13352.0	12512.0
pw09_100.7	13501	12429.0	12172.0	13045.0	12524.0
pw09_100.8	13593	11743.0	12450.0	13021.0	11706.0
pw09_100.9	13658	13027.0	12555.0	13424.0	12351.0

5. Conclusão

Neste trabalho, foram exploradas aplicações de algoritmos de aprendizado de máquina supervisionado para criar modelos treinados a partir de instâncias de grafos de tamanhos pequenos e intermediários, visando observar os resultados em instâncias maiores do Problema do Corte Máximo. Os modelos foram treinados com instâncias previamente resolvidas pelo Solver Biq Mac e utilizados para realizar a partição dos vértices das instâncias não observadas no treinamento.

A análise revelou uma relação significativa entre as propriedades estruturais dos grafos e o desempenho dos modelos supervisionados na solução do problema. Instâncias com maior coeficiente de aglomeração, centralidade de grau elevada e centralidade de intermediação próxima de zero mostraram desempenho superior, sugerindo que essas características estruturais são fundamentais para um treinamento eficiente dos modelos.

O uso de uma base de dados de treinamento composta por instâncias selecionadas aleatoriamente demonstrou que, assim como em bases de dados tradicionais, a ausência de grandes disparidades nas características estruturais dos dados é crucial. A seleção de instâncias com coeficiente de aglomeração positivo e distante de zero foi determinante no treinamento, gerando resultados próximos aos fornecidos pelo Solver Biq Mac.

Os resultados indicam que, com a escolha adequada das amostras de treinamento, o modelo tem potencial para melhorar seu desempenho no futuro, ampliando suas aplicações no Problema do Corte Máximo.

Referências

Aggarwal, C. C. (2014). *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC, 1 edition. ISBN 978-1-4665-8675-8,978-1-4665-8674-1.

Agrawal, R., Rajagopalan, S., Srikant, R., e Xu, Y. (2003). Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th international conference on World Wide Web*, p. 529–535.

Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

Blum, C. e Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM computing surveys (CSUR)*, 35(3):268–308.

Boros, E. e Hammer, P. L. (1991). The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33(3):151–180.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

de Almeida Teodoro, L. e Kappel, M. A. A. (2020). Aplicação de técnicas de aprendizado de máquina para predição de risco de evasão escolar em instituições públicas de ensino superior no brasil. *Revista Brasileira de Informática na Educação*, 28:838–863.

de Sousa, S., Haxhimusa, Y., e Kropatsch, W. G. (2013). Estimation of distribution algorithm for the max-cut problem. In *Graph-Based Representations in Pattern Recognition: 9th IAPR-TC-15 International Workshop, GbRPR 2013, Vienna, Austria, May 15-17, 2013. Proceedings 9*, p. 244–253. Springer.

Diestel, R. (2010). *Graph Theory*. Graduate Texts in Mathematics, 173. Springer, 4th edition. ISBN 3642142788,9783642142789.

Faceli, K., Lorena, A. C., Gama, J., Almeida, T. A. d., e Carvalho, A. C. P. d. L. F. d. (2011). Inteligência artificial: uma abordagem de aprendizado de máquina. p. 378.

Garey, M. R. e Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman, first edition edition. ISBN 0716710455,9780716710455.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., Sebastopol, CA, second edition. ISBN 9781492032649.

Gianinazzi, L., Fries, M., Dryden, N., Ben-Nun, T., Besta, M., e Hoefler, T. (2021). Learning combinatorial node labeling algorithms. *arXiv preprint arXiv:2106.03594*.

Goudet, O., Grelier, C., e Hao, J.-K. (2022). A deep learning guided memetic framework for graph coloring problems. *Knowledge-Based Systems*, 258:109986.

Grus, J. (2016). *Data Science do Zero*. Alta Books, Rio de Janeiro, 1 edition. ISBN 978-85-508-0387-6.

Gu, S. e Yang, Y. (2018). A pointer network based deep learning algorithm for the max-cut problem. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25*, p. 238–248. Springer.

Gu, S. e Yang, Y. (2020). A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies. *Mathematics*, 8(2):298.

Jonathan L. Gross, P. Z., Jay Yellen (2013). *Handbook of Graph Theory, Second Edition*. Discrete Mathematics and Its Applications. Chapman and Hall/CRC, 2 edition. ISBN 1439880182,9781439880180.

Karp, R. M. (1972). Reducibility among combinatorial problems. complexity of computer computations (the ibm research symposia series).

Latora, V., Nicosia, V., e Russo, G. (2017). *Complex Networks: Principles, Methods and Applications*. Cambridge University Press, Cambridge.

Libbrecht, M. W. e Noble, W. S. (2015). Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br

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

McCulloch, W. S. e Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

Miki, S., Yamamoto, D., e Ebara, H. (2018). Applying deep learning and reinforcement learning to traveling salesman problem. In *2018 international conference on computing, electronics & communications engineering (ICCECE)*, p. 65–70. IEEE.

Miyazawa, F. K. e de Souza, C. C. (2015). Introdução à otimização combinatória. *Sociedade Brasileira de Computação*.

Monard, M. C. e Baranauskas, J. A. (2003). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.

Oliveira, M. C., Ferreira, L. V. B., e de Oliveira Barreiros, M. (2023). Classificação de doenças cardiovasculares utilizando aprendizado de máquina. *Multidebates*, 7(1):38–44.

Otterbach, J. S., Manenti, R., Alidoust, N., Bestwick, A., Block, M., Bloom, B., Caldwell, S., Didier, N., Fried, E. S., Hong, S., et al. (2017). Unsupervised machine learning on a hybrid quantum computer. *arXiv preprint arXiv:1712.05771*.

Parsian, M. (2015). *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark*. O'Reilly Media. ISBN 1491906189,9781491906187.

Raschka, S. (2015). *Python Machine Learning*. Packt Publishing, Birmingham - Mumbai. ISBN 978-1-78355-513-0.

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

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, p. 41–46.

Shlens, J. (2014). A tutorial on principal component analysis. *arXiv preprint arXiv:1404.1100*.

Sipser, M. (2012). *Introduction to the Theory of Computation*. Course Technology, 3 edition. ISBN 113318779X,9781133187790.

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

Xiong, L. e Yao, Y. (2021). Study on an adaptive thermal comfort model with k-nearest-neighbors (knn) algorithm. *Building and Environment*, 202:108026.

Yao, W., Bandeira, A. S., e Villar, S. (2019). Experimental performance of graph neural networks on random instances of max-cut. In *Wavelets and Sparsity XVIII*, volume 11138, p. 242–251. SPIE.

DOI: 10.59254/sbpo-2025-212174
https://proceedings.science/p/212174?lang=pt-br