Explorando GNNs Sensíveis a Arestas para Previsão de Carga em uma Rede Backbone

Wagner Almeida[1], **Fábio Ramos**[1], **Alex V. Borges**[2],
José Augusto M. Nacif[1], **Ricardo F. dos Santos**[1]

[1]Departamento de Informática – Universidade Federal de Viçosa (UFV)
36570-900 – Viçosa – MG – Brazil

[2]Departamento de Ciência Computação – Universidade Federal de Juiz de Fora (UFJF)
36036-900– Juiz de Fora – MG – Brazil

{wagnerdjunior,fabio.ramos,jnacif,ricardo}@ufv.br, alex.borges@ufjf.br

Abstract. *Graph Neural Networks (GNNs) are specific tools to apply machine learning on various types of complex data structured as graphs. Most GNNs, however, usually focus on node or whole graph representation, overlooking edge features and edge structural relationships. In this paper, we present an attention-based edge-aware GNN model to predict node loads in a backbone network. The proposed model can process implicit and explicit edge features and node features, contributing to improved data representation. Our model outperforms the results of network node load predictions obtained by state-of-the-art non-edge-aware GNN models. The framework we developed is publicly availabe for reproduction and different testage.*

Resumo. *Redes neurais de grafos (GNNs) são ferramentas para aplicação de aprendizado de máquina a vários tipos de dados complexos estruturados em grafos. A maioria das GNNs, no entanto, é focada em representar nós ou grafos inteiros, deixando de lado informações que possam estar contidas em atributos de arestas. Neste trabalho, apresentamos um modelo de GNN sensível a arestas com mecanismos de atenção aplicado à previsão de carga em nós de uma rede backbone. O modelo proposto é capaz de processar atributos implícitos e explícitos de arestas juntamente aos atributos de nós, contribuindo para aprimorar a representação dos dados. Nos testes realizados para previsão de carga, nosso modelo superou os resultados obtidos pelo estado da arte dos modelos de GNNs não sensíveis às arestas. A ferramenta que desenvolvemos para testes está disponível publicamente.*

1. Introdução

Aprendizado de máquina tem sido uma ferramenta valiosa para extrair e analisar dados de uma grande variedade de problemas. Técnicas tradicionais de aprendizado de máquina operam sobre dados bem estruturados e ordenados, como texto, som e imagens. No entanto, diversos conjuntos de dados, que contém informações do mundo real, não podem ser representados de maneira estruturada. [Barabási et al. 2016] apresentam o conceito de redes complexas, que podem representar dados de diversas áreas, como química, biologia, economia, neurociência, redes financeiras, redes sociais e também redes de computadores. Para representar esses dados, é necessário uma estrutura mais complexa, como grafos [Veličković 2023].

Grafos são estruturas não euclidianas e de alta dimensionalidade, capazes de representar vários conjuntos de dados complexos do mundo real. O grande volume, complexidade e relevância desses dados faz com que o uso de aprendizado de máquina seja um caminho natural para análise, extração de informações e geração de modelos preditivos. No entanto, a aplicação de técnicas tradicionais de aprendizado de máquina em grafos ainda é um desafio já que elas não estão equipadas para lidar com estruturas tão complexas de maneira eficiente [Zhou et al. 2020]. Para esse fim, métodos específicos são necessários, e embora vários tenham sido propostos [Perozzi et al. 2014, Tang et al. 2015, Grover and Leskovec 2016], as redes neurais de grafos (*Graph Neural Networks* - GNNs) surgiram como o estado da arte para aplicação de aprendizado de máquina a dados estruturados em grafos [Xu et al. 2018, Tsitsulin et al. 2023, Bessadok et al. 2021, Capanema et al. 2022].

GNNs usam esquemas de passagem de mensagens para gerar representações de grafos em dimensionalidade reduzida. Esses esquemas transformam, agregam e propagam informações sobre vizinhanças entre nós e podem gerar representações precisas da topologia de um grafo. O fato de agregarem informações sobre vizinhanças faz com que as redes neurais de grafos sejam eficazes em capturar e representar nuances de relacionamentos entre os nós de uma rede complexa [Zhou et al. 2020].

Uma limitação das redes neurais de grafos tradicionais é que, em geral, elas consideram apenas atributos dos nós ou de grafos completos para gerar representações e modelos. No entanto, diversos conjuntos de dados podem conter informações ricas nos atributos e relacionamentos de suas arestas. Quando são consideradas, geralmente as representações de arestas são geradas a partir da agregação da representação dos nós que elas conectam [Gao et al. 2019]. Embora possa ser efetivo em alguns casos, muita informação pode ser potencialmente perdida.

Nesse cenário, observa-se um crescimento significativo de pesquisas e aplicações de redes neurais de grafos sensíveis a arestas em variados domínios do conhecimento. De fato, resultados mostram que incluir informações extraídas de arestas pode levar a ganhos tanto em dados de testes comparativos [Bandyopadhyay et al. 2019, Bielak et al. 2020, Jiang et al. 2019, Gong and Cheng 2019, Yang and Li 2020, Jo et al. 2021], quanto em aplicações do mundo real [Kim et al. 2019, Gao et al. 2019, Wang et al. 2018, Mirhoseini et al. 2021].

Aplicações do mundo real como redes de computadores, em especial as gerenciadas por provedores de serviços, são ótimas candidatas para serem analisadas usando GNNs. Assim, apresentamos uma aplicação de previsão de carga dos nós de uma rede de *backbone* que faz uso de uma GNN com mecanismo sensível a arestas baseado em atenção [Veličković et al. 2018] e comparamos os resultados com duas soluções tradicionais e bem estabelecidas de GNNs: *Graph Convolutional Networks* [Kipf and Welling 2016] e GraphSAGE [Hamilton et al. 2017] que não utilizam os atributos das arestas.

Neste trabalho apresentamos também algumas aplicações diversas que usam modelos de GNNs sensíveis a arestas para resoluções de problemas do mundo real. Os resultados dessas aplicações motivaram o desenvolvimento do modelo de GNN proposto e sua aplicação para o problema específico da previsão de carga em nós de uma rede *backbone*.

Em comparação com duas soluções tradicionais e amplamente usadas de GNNs, nosso modelo foi capaz de prever com maior precisão a carga dos nós da rede, apresentando valores de coeficiente de determinação (R^2) cerca de 4% maiores, em média.

Este trabalho traz duas contribuições principais: (i) Identificamos o potencial de desempenho de GNNs sensíveis a arestas e propusemos uma abordagem aplicada ao problema de previsão de carga em um uma rede *backbone* usando dados reais de tráfego; (ii) Durante a pesquisa tivemos dificuldade em encontrar ferramentas de teste e reprodutibilidade dos modelos de GNNs aplicados a redes de computadores. Dessa forma, disponibilizamos nossa ferramenta[1] para reprodução dos resultados deste artigo e realização de testes de diferentes redes neurais de grafos aplicadas em um contexto de dados reais. O restante deste trabalho está organizado da seguinte forma: Na Seção 2 são apresentados outros trabalhos que usam alguma forma de integração de atributos de arestas em GNNs para solução de problemas práticos. A Seção 3 apresenta o conjunto de dados analisado e o modelo proposto de rede neural sensível a arestas. Na Seção 4 são apresentados os resultados comparativos da aplicação do modelo. Concluímos na Seção 5 com os aprendizados obtidos e possíveis opções de pesquisas futuras.

2. Trabalhos Relacionados

Esta seção apresenta trabalhos publicados que fazem uso de atributos de arestas como mecanismo de aprimoramento de técnicas de aprendizado de máquina aplicadas à resolução de problemas do mundo real. Também são mostrados diferentes trabalhos que usam GNNs para resolver problemas de redes de computadores, incluíndo a previsão de carga.

Muitos sistemas complexos representados por grafos possuem estruturas homogêneas, com nós e arestas de um único tipo. Grafos de conhecimento [da Silva et al. 2019], no entanto, podem possuir estruturas heterogêneas, com diversos tipos de dados compondo nós e arestas. [Gao et al. 2019] propõem Edge2Vec, um modelo de aprendizado de máquina sensível a arestas que visa a representar nós de grafos de conhecimento que sejam topológica e semanticamente parecidos em espaços similares de dimensionalidade reduzida.

Nuvens de pontos são coleções de pontos espalhados em um espaço 2D ou 3D que representam algum objeto. Essas estruturas também podem ser representadas por grafos. Neste contexto, o EdgeConv [Wang et al. 2018] foi apresentado como uma alternativa que, ao invés de criar representações com base nas posições dos pontos (nós), utiliza os atributos de arestas para descrever o relacionamento entre os pontos. Apesar de nem sempre conseguirem superar o estado da arte [Qi et al. 2016] em processamento de nuvens de pontos, os resultados sugerem que as informações estruturais e relacionais obtidas das representações de arestas podem ser igualmente ou até mais valiosas do que as coordenadas dos pontos.

Few-shot Learning [Snell et al. 2017] é definido como o problema de classificação de dados quando apenas uma pequena parte das amostras disponíveis em cada categoria possui rótulos. Uma abordagem usando redes neurais sensíveis a arestas foi proposta por [Kim et al. 2019]. O modelo, chamado *edge-labeling GNN* (EGNN), é uma rede neural em camadas. Cada camada é composta por um bloco de atualização de atributos e

[1] https://github.com/wagneraljr/EdgeAwareGNN

outro de atualização de atributos de arestas. Os resultados obtidos são comparados com modelos de referência e mostram que o uso de uma GNN sensível a arestas foi capaz de superar algoritmos do estado da arte de *Few-shot Learning*.

O posicionamento e conexão de blocos de circuitos integrados é uma tarefa essencial no desenvolvimento de hardware. Três principais categorias de métodos de posicionamento e conexão de blocos são os mais comumente usados para automação dessa tarefa: métodos baseados em partições [Breuer 1977], métodos estocásticos [Kirkpatrick et al. 1983] e solucionadores analíticos [Luo and Pan 2008]. No entanto, o esforço humano de engenheiros ainda produz resultados com melhor acurácia, com a desvantagem de que esse trabalho não automatizado pode levar meses para ser concluído. [Mirhoseini et al. 2021] propõem um método inovador que utiliza uma GNN sensível a arestas e é capaz de gerar uma solução com qualidade comparável àquelas produzidas por humanos em menos de seis horas.

No contexto de redes de computadores, [Hope 2020] explora a possibilidade de combinar uma política de roteamento baseada em arquitetura de GNN com um complexo sistema de aprendizado profundo para obter desempenho próximo ao ótimo para o problema de minimizar a congestão de links. Esse trabalho é uma evolução de [Valadarsky et al. 2017] que usa MLP para essa mesma tarefa. O uso de GNN provê mais consistência e resiliência ao modelo, além de permitir que um modelo treinado em uma rede possa ser aplicado à diversas outras. Apesar de usar o mesmo conjunto de dados deste trabalho, o potencial dos atributos de arestas não é explorado.

[Peng et al. 2024] compara o desempenho de GNNs focadas em nós(Vertex-GNNs) e GNNs focadas em arestas (Edge-GNNs) na tarefa de aprender políticas de alocação de recursos em redes de comunicação sem fio considerando três problemas principais: Escalonamento de *links*, controle de energia e precodificação. O trabalho apresenta simulações comparativas e os resultados demonstram que, para essa tarefa, as Edge-GNNs são capazes de alcançar pelo menos o mesmo desempenho que os Vertex-GNNs, mas com tempo de treinamento consideravelmente menor.

De maneira similar a este trabalho, [Tao et al. 2023] buscam realizar a tarefa de previsão de tráfego e para isso propõem o modelo CAGNN(*Cross Aggregate GNN*), que divide o processamento de nós e arestas em dois blocos idênticos de GNNs que têm suas representações somadas. Esse trabalho usa o mesmo conjunto de dados que nós usamos, no entanto não são feitas comparações de desempenho com outros modelos de GNNs. Uma vez que a implementação original não está publicamente disponível, não foi possível comparar o desempenho de métricas diretamente, mas nosso modelo obteve valores de RMSE e MAE inferiores aos divulgados.

3. Modelo Proposto

Nesta seção apresentamos o modelo proposto de GNN sensível às arestas. Como aplicação prática, realizamos a tarefa de previsão de carga em nós de uma rede de *backbone* de um provedor de serviços. Detalhamos o conjunto de dados, demonstramos a estrutura e os mecanismos utilizados no modelo e os processos de treinamento e otimização de hiperparâmetros.

3.1. Conjunto de Dados

Para exemplificar o uso de redes neurais sensíveis a arestas, e demonstrar suas vantagens em relação a redes que não consideram atributos de arestas, buscamos trabalhar com um conjunto de dados que representa uma rede de computadores do mundo real.

A Rede Abilene era uma rede *backbone* norte-americana de alto desempenho que funcionava como parte do projeto Internet2 [Teitelbaum et al. 1999]. Esse projeto foi uma iniciativa colaborativa entre universidades, indústria e governo com objetivo de desenvolver tecnologias avançadas de rede e aplicações para pesquisas e educação. Criada em 1999 e encerrada em 2007, a rede Abilene era composta por 11 nós e 14 *links*, conectando majoritariamente instituições de ensino e pesquisa. Durante todo seu funcionamento, ela teve sua capacidade regularmente aprimorada, chegando a operar na velocidade de até 10 Gigabits por segundo em sua fase final.

[Fang et al. 2007] publicaram um trabalho relativo a tomografia de rede juntamente com os dados de estrutura e tráfego da rede. A partir de então, essa rede vem sendo comumente utilizada como parâmetro para testes de algoritmos e soluções de otimização. A Figura 1 mostra a estrutura da rede Abilene. Embora as tecnologias e infraestruturas de rede tenham evoluído significativamente desde a concepção da rede Abilene, seus dados históricos são comumente usados para desenvolver ou testar modelos de aprimoramento de desempenho ou roteamento em redes de computadores [Hope 2020, Tao et al. 2023, Qiu et al. 2023, Yang et al. 2023, Zheng et al. 2024].

O conjunto de dados usado neste trabalho contém seis meses de dados de tráfego entre os nós da rede Abilene, medidos a cada cinco minutos. Os dados são dispostos em matrizes de tráfego $N \times N$, onde N é o número de nós da rede. Uma vez que o objetivo deste trabalho é estimar a carga nos nós, o tráfego de entrada e saída de cada nó na matriz de tráfego foi somado e normalizado. A estrutura da rede contém arestas com oito atributos cada, representados por valores reais. Além disso, o modelo foi enriquecido com o cálculo de atributos implícitos das arestas: centralidade de intermediação, grau das arestas e coeficiente de agrupamento das arestas.

O objetivo principal do modelo é prever a carga dos nós da rede Abilene em um determinado intervalo de tempo de cinco minutos. O modelo foi treinado com medições de dois diferentes períodos temporais: todo o tráfego medido no dia anterior ao da previsão e todo o tráfego medido exatamente uma semana antes do alvo da previsão.

3.2. AttEdgeAwareGNN

A seguir apresentamos o modelo de rede neural de grafos sensível a arestas proposto neste trabalho. A arquitetura do modelo é composta por várias camadas, incluindo camadas convolucionais para processamento de características dos nós e uma camada de atenção para as arestas. Essa estrutura híbrida permite que o modelo capture tanto as informações locais dos nós quanto os atributos e as relações definidas pelas arestas. O modelo utiliza camadas de redes neurais de grafos tradicionais para processar os atributos de nós e arestas, porém introduzimos um mecanismo de atenção para atribuir maior peso às arestas mais relevantes da rede. A Figura 2 detalha o fluxo do modelo AttEdgeAwareGNN.

A partir do grafo gerado para representar a estrutura da rede Abilene são extraídos atributos de nós e arestas. Esses atributos são então processados por dois fluxos de camadas de redes neurais separadas. Após o processamento inicial, há uma etapa de agregação

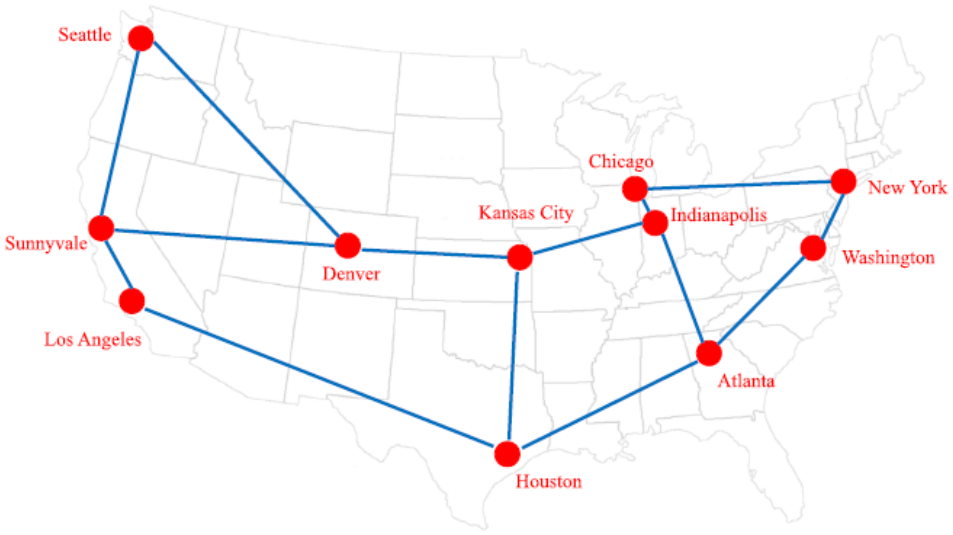


Figura 1. Representação da rede Abilene.

de representações de nós e de arestas ponderadas e uma etapa de concatenação que leva às representações finais dos nós usadas na previsão de carga. As seções a seguir detalham essas etapas.

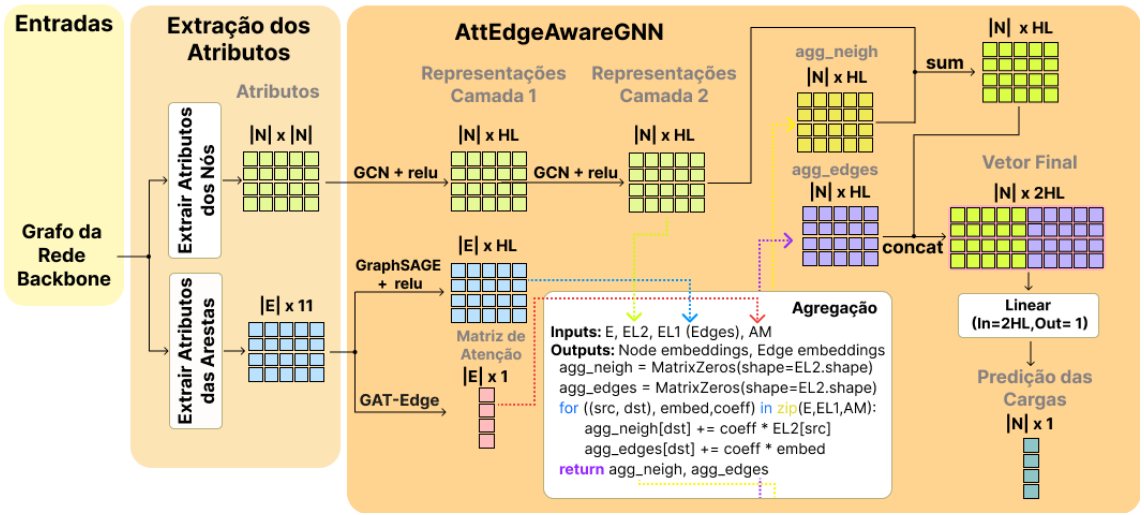


Figura 2. Fluxo do Modelo AttEdgeAwareGNN.

3.2.1. Atributos dos Nós

Inicialmente, os atributos originais x_i de um nó i, representados por codificação *one-hot*, são processados por duas camadas de redes convolucionais de grafos (GCNs) [Hamilton et al. 2017]: $\mathbf{x}_i^{(1)} = \mathrm{ReLU}(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1)$ e $\mathbf{x}_i^{(2)} = \mathrm{ReLU}(\mathbf{W}_2\mathbf{x}_i^{(1)} + \mathbf{b}_2)$, onde $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$ são as matrizes de pesos e os *biases* das duas camadas de convolução, respectivamente. O uso da função de ativação ReLU (*Rectified Linear Unit*)

aplicada a camadas de convolução de redes neurais introduz não-linearidade e ajuda a mitigar o problema do gradiente desvanecente [Hanin 2018]. Também é usado um *dropout* após a primeira camada para regularização: $\mathbf{x}_i^{(1)} = \text{Dropout}(\mathbf{x}_i^{(1)})$.

A convolução em grafos fundamenta-se no princípio de agregação de mensagens. Nesse processo, cada nó atualiza seu estado (atributos do nó) ao agregar informações provenientes dos nós vizinhos e das arestas conectadas a ele. Essa agregação é geralmente uma combinação linear dos atributos dos vizinhos, seguida por uma função de ativação não-linear, no caso, a ReLU.

3.2.2. Atributos das Arestas

As arestas do conjunto de dados possuem atributos explícitos, representados por oito valores reais. O conjunto de dados analisado não especifica exatamente o significado desses atributos. Além disso, foram calculados centralidade de intermediação, grau das arestas e coeficiente de agrupamento das arestas como atributos implícitos. Observamos que, quanto mais atributos à disposição do modelo, melhores os resultados. Isso indica que grafos que tenham mais atributos explícitos de arestas podem ser melhor interpretados pelo modelo proposto.

Inicialmente, foi necessário reduzir o espaço latente dos atributos de arestas. O processamento de atributos foi feito com uma camada baseada na rede neural GraphSAGE [Kipf and Welling 2016]. O GraphSAGE (*Graph Sample and AggreGatE*) é uma técnica que, diferentemente da GCN, seleciona uma amostra fixa de vizinhos para cada nó antes de realizar a agregação. Isso pode reduzir significativamente a carga computacional em grafos grandes. Além disso, o GraphSAGE possui maior flexibilidade na escolha do método de agregação e, embora não suporte explicitamente o processamento de atributos de arestas, sua implementação pode ser mais facilmente adaptada para esse fim.

Formalmente, sejam $\mathbf{e}_{ij}$ os atributos originais das arestas conectando os nós i e j, a transformação dos atributos das arestas é dada por: $\mathbf{e}'_{ij} = \text{ReLU}(\mathbf{W}_{aresta}\mathbf{e}_{ij} + \mathbf{b}_{aresta})$, onde $\mathbf{W}_{aresta}$ e $\mathbf{b}_{aresta}$ são a matriz de pesos e o vetor de *bias* da camada de convolução para aresta.

3.2.3. Mecanismo de Atenção

Os atributos originais das arestas são processados por um mecanismo de atenção baseados nas *Graph Attention Networks* (GATs) [Veličković et al. 2018]. Originalmente projetadas para processar atributos de nós, aqui, adaptamos esse mecanismo para processar os atributos das arestas. Pelo mecanismo proposto, os atributos originais explícitos e implícitos das arestas são passados a uma camada de rede neural linear que reduz a dimensionalidade original dos atributos para um espaço latente. Sejam e_i os atributos da aresta i, a transformação linear é dada por $\mathbf{e}'_i = \mathbf{W}_e\mathbf{e}_i + \mathbf{b}_e$, onde $\mathbf{W_e}$ é a matriz de pesos e $\mathbf{b_e}$ é o vetor de *bias* da camada linear. O coeficiente de atenção para a aresta i é então calculado como $a_i = \sum(\mathbf{e}'_i \odot \mathbf{e}'_i)$, onde $\odot$ é o produto elemento a elemento e a soma é realizada sobre todas as dimensões dos atributos. Uma função de ativação LeakyReLU, $a'_i = \text{LeakyReLU}(a_i)$ é aplicada para introduzir não-linearidade ao modelo. Por fim os co-

eficientes de atenção são normalizados usando uma função *softmax*, $\alpha_i = \frac{\exp(a'_i)}{\sum_j \exp(a'_j)}$. Esta etapa resulta em um coeficiente único para cada aresta, representando sua importância relativa no grafo.

Os coeficientes obtidos pela função de atenção são usados na etapa final do modelo de GNN, quando os atributos de nós e arestas são agregados, ponderando as características tanto das vizinhanças locais dos nós quanto de todas as arestas do grafo de acordo com os coeficientes de atenção. Formalmente, para cada nó i, os atributos das suas arestas e dos nós vizinhos são agregados de forma ponderada usando os coeficientes de atenção: $\mathbf{e}_i^{agr} = \sum_{j \in N(i)} \alpha_{ji} \mathbf{e}'_{ji}$, $\mathbf{h}_i^{agr} = \sum_{j \in N(i)} \alpha_{ji} \mathbf{x}_j^{(2)}$, onde $\mathbf{N}_{(i)}$ são os nós vizinhos de i, α_{ji} são os coeficientes de atenção e $\mathbf{e}'_{ji}$ são as representações dos atributos das arestas já transformados.

O uso desse mecanismo de atenção permite que os atributos das arestas ajudem a fornecer representações mais ricas e contextualizada de cada nó. As representações dos nós já enriquecidas pelos mecanismos de ponderação e agregação são concatenadas às representações obtidas inicialmente pela GCN, gerando os valores finais das representações dos nós: $\mathbf{x}_i^{final} = [\mathbf{x}_i^{(2)} + \mathbf{h}_i^{agr}; \mathbf{e}_i^{agr}]$. Então é aplicada uma última camada de rede neural linear para produzir a saída do modelo com a previsão das cargas para cada nó: $\mathbf{y}_i = \mathbf{W}_f \mathbf{x}_i^{final} + \mathbf{b}_f$.

3.3. Treinamento e Otimização de Parâmetros

O treinamento do modelo proposto foi realizado em dois cenários diferentes. No primeiro, usamos as medições obtidas no dia anterior ao dia alvo da previsão. No segundo, usamos as medições de uma semana anterior ao dia alvo. Essas medições foram feitas durante as 24 horas do dia em intervalos de 5 minutos. O objetivo foi prever uma janela de 5 minutos. No processo de treinamento foram usados o otimizador Adam [Kingma and Ba 2014] e um escalonador implementado pela biblioteca StepLR do PyTorch.

Para obtenção dos melhores resultados possíveis do modelo proposto, e dos modelos de comparação, foi realizado um processo de otimização de hiperparâmetros usando a biblioteca do Python Optuna [Akiba et al. 2019]. Foram otimizados a quantidade de épocas de treinamento, o *dropout* e os parâmetros do otimizador e do escalonador.

A Optuna utiliza um processo iterativo, onde a cada iteração um conjunto de hiperparâmetros é selecionado com base em uma estratégia de otimização. Alguns algoritmos usados pela Optuna são Tree-structured Parzen Estimator (TPE), CMA-ES e métodos de busca aleatória. Esses algoritmos levam em consideração os resultados das iterações anteriores para orientar a seleção de hiperparâmetros em iterações subsequentes. Buscam, assim, equilibrar a exploração (*exploration*), testando novas áreas do espaço de hiperparâmetros e o aproveitamento (*exploitation*), refinando hiperparâmetros promissores.

Para cada conjunto de hiperparâmetros selecionado, a função de objetivo é executada, geralmente envolvendo o treinamento e a validação de um modelo de aprendizado de máquina com esses hiperparâmetros. O desempenho do modelo é então avaliado, tipicamente usando uma métrica específica (como precisão, AUC, erro quadrático médio, R^2). Este desempenho é retornado à Optuna, que o utiliza para ajustar sua estratégia de seleção de hiperparâmetros. Ao final das iterações definidas, o conjunto de hiperparâmetros que resultou no melhor desempenho do modelo é retornado.

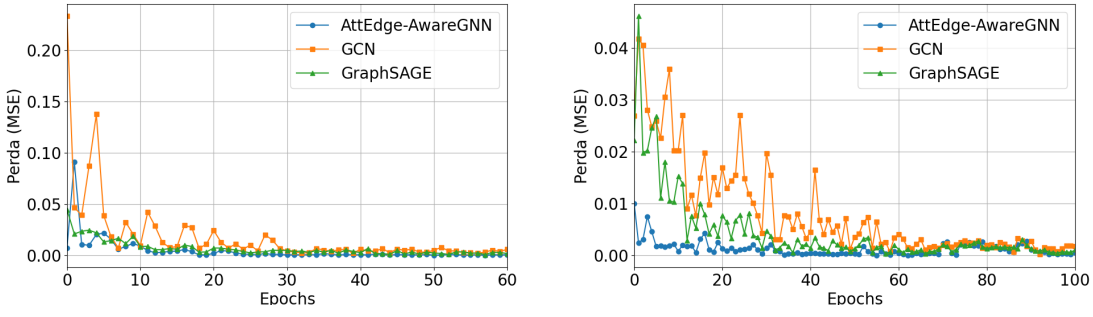
Com o objetivo de manter consistência entre os resultados e permitir a reprodutibilidade dos experimentos, as sementes aleatórias de inicialização dos pesos dos modelos foram fixadas. Mais de 20 sementes diferentes foram testadas para cada modelo, e os resultados aqui apresentados correspondem às melhores otimizações de cada um deles. Nossa ferramenta permite a alteração de sementes, bem como a inclusão de diferentes modelos de GNNs e matrizes de tráfego para testes comparativos.

4. Resultados

Esta seção detalha experimentos realizados com o modelo AttEdgeAwareGNN. Os resultados mostrados a seguir são relativos aos treinos em dois cenários: o primeiro feito com dados de 24 horas de tráfego capturados no dia anterior ao do dia alvo da previsão. No segundo cenário, o treinamento foi realizado com dados capturados também por 24 horas, mas uma semana antes do dia alvo. Nossa ferramenta publicamente disponível permite a reprodução desses experimentos bem como a adaptação para o uso de diferentes modelos de GNNs e conjuntos de dados.

Comparamos os resultados obtidos pelo modelo de rede neural de grafos sensível a arestas com dois modelos de redes neurais de grafos bem estabelecidos: GCN [Hamilton et al. 2017] e GraphSAGE [Kipf and Welling 2016]. Esses modelos também serviram como base para o processamento de atributos dos nós pelo modelo proposto. Os resultados foram obtidos através de vários testes com diferentes inicializações de pesos dos modelos. Aqui compilamos e mostramos o melhor resultado de cada modelo.

As Figuras 3a e 3b mostram a evolução do valor do erro quadrático médio (MSE) durante as épocas de treinamento no primeiro (dados do dia anterior) e segundo (dados da semana anterior) cenários, respectivamente. O MSE é calculado conforme a equação 1, onde y_i representa o i-ésimo valor observado e $\hat{y}_i$ o i-ésimo valor previsto. Neste caso, quanto menor o valor e mais rápida a estabilização do gráfico, melhor. Em ambos os casos nota-se que o modelo proposto converge mais rapidamente e se mantém mais estável, se comparado aos modelos de referência. No segundo cenário, os modelos não sensíveis a arestas apresentaram ainda mais instabilidade, além de todos os modelos demorarem mais épocas para convergir durante o treinamento se comparados ao primeiro cenário. Apesar de convergirem antes, os melhores resultados foram obtidos com treinamentos que duraram entre 100 e 300 épocas, em todos os modelos. Treinamentos por mais de 300 épocas levaram a *overfitting*, perdendo eficácia.

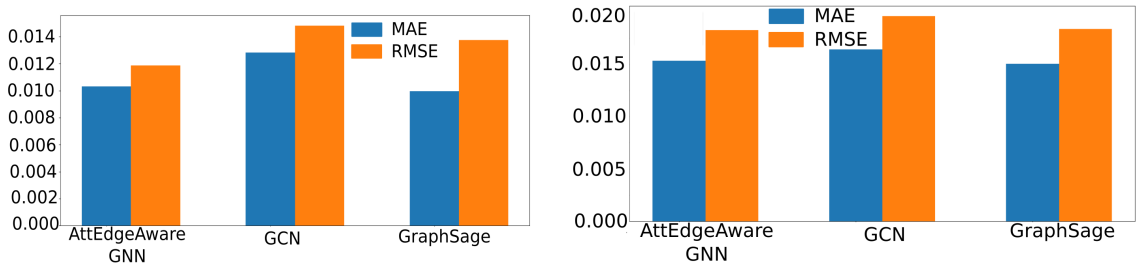


(a) Cenário 1 **(b) Cenário 2**

Figura 3. Curvas de perda durante o tempo.

$$\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n} \tag{1}$$

As Figuras 4a e 4b apresentam o erro médio absoluto (MAE) e a raiz do erro quadrático médio (RMSE), respectivamente calculados pelas equações 2 e 3. Em ambos os casos, quanto menor o valor, melhor. Nestas métricas, os modelos treinados no primeiro cenário também foram superiores. No AttEdgeAwareGNN, ainda que em ambos os casos os valores sejam pequenos, o primeiro cenário apresentou MAE cerca de 48% e RMSE cerca de 54% menores do que o segundo cenário. A mesma tendência foi observada nos outros modelos. É interessante observar que o AttEdgeAwareGNN obteve um RMSE inferior aos modelos de comparação. No entanto, o MAE foi levemente superior ao obtido pelo GraphSAGE. Isso se deve ao fato de que o RMSE aplica uma maior penalização a previsões fora da curva, enquanto o MAE penaliza todos os erros igualmente. No caso, isso demonstra que o modelo proposto foi capaz de capturar melhor os valores fora da curva em ambos os cenários de treinamento.



(a) Cenário 1 **(b) Cenário 2**

Figura 4. Comparação de métricas MAE e RMSE entre diferentes modelos.

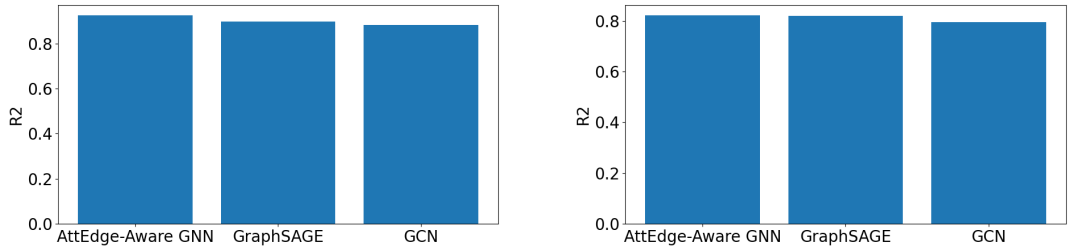
$$\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{n} \tag{2}$$

$$\sum_{i=1}^{n} \sqrt{\frac{(y_i - \hat{y}_i)^2}{n}} \tag{3}$$

A métrica R^2, também conhecida como coeficiente de determinação, é uma métrica estatística relevante na avaliação de modelos de aprendizado de máquina. Sua principal importância reside em fornecer uma medida de quão bem as variáveis independentes do modelo conseguem explicar a variação da variável dependente. Um valor R^2 mais alto indica que o modelo explica uma grande parte da variação, sugerindo um bom ajuste entre o modelo e os dados. O coeficiente de determinação é calculado conforme a equação 4, onde y são os valores observados, $\hat{y}$ os valores previstos e $\bar{y}$ a média dos valores observados.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}. \tag{4}$$

As Figuras 5a e 5b apresentam os valores R^2 para os modelos avaliados. Em ambos os cenários de treinamento, o modelo proposto superou os modelos de referência. No contexto geral, os melhores resultados foram obtidos com treinamento no primeiro cenário. Nesse cenário, o valor R^2 do modelo sensível a arestas foi de 0,9247, comparativamente os melhores valores obtidos pelos modelos GraphSAGE e GCN foram, respectivamente, 0,8988 e 0,8823. Isso demonstra que, embora haja uma tendência cíclica de demanda em redes de computadores, dados mais recentes podem levar à previsão de padrões mais precisos de tráfego.



(a) R^2 - Cenário 1 **(b) R^2 - Cenário 2**

Figura 5. Comparação de valores R^2.

Realizamos a mesma tarefa de previsão de carga para um intervalo específico nos dois cenários usando métodos estatísticos populares. A Tabela 1 mostra o comparativo de resultados de MAE, RMSE e MAPE (erro percentual absoluto médio) obtidos pelas soluções baseadas em GNNs e métodos estatísticos. Para todas as métricas, quanto menor o valor, melhor. Os melhores valores estão destacados em negrito. Em todos os casos, os modelos de GNN superam os modelos tradicionais. Nosso modelo supera ou iguala os melhores resultados na maioria das métricas, especialmente no cenário 1 onde as previsões são mais precisas. Nesse cenário, acreditamos ainda que o valor de MAE levemente maiores do que o GraphSAGE se devem a uma diferença na penalização de pontos fora da curva, o que indica que o modelo sensível a arestas é capaz de lidar melhor com esses casos.

Tabela 1. Comparação de Métricas.

	Cenário 1					
	AttEdgeAware	GraphSAGE	GCN	ARIMA	SARIMA	LSTM
MAE	0,010	**0,009**	0,012	0,025	0,020	0,025
RMSE	**0,011**	0,013	0,014	0,029	0,024	0,030
MAPE	**0,140**	0,197	0,146	1,096	0,932	0,829
	Cenário 2					
	AttEdgeAware	GraphSAGE	GCN	ARIMA	SARIMA	LSTM
MAE	**0,015**	**0,015**	0,016	0,025	0,025	0,029
RMSE	**0,018**	**0,018**	0,019	0,029	0,031	0,034
MAPE	0,177	**0,171**	0,191	1,096	0,885	3,438

5. Conclusões

GNNs são o estado da arte da aprendizado de máquina aplicada à estrutura de grafos. Uma vez que diversos sistemas do mundo real possuem relações complexas que podem ser representadas por grafos, o estudo e aprimoramento dessas soluções vem sendo cada vez mais relevantes. Uma fronteira ainda pouco explorada é a de redes neurais de grafos

sensíveis a arestas. A partir de exemplos de aplicações dessas redes em diversos problemas do mundo real, propusemos nosso próprio modelo de rede neural sensível a arestas denominado AttEdgeAwareGNN e usamos para fazer a previsão de carga de nós em uma rede *backbone*. O modelo possui mecanismos avançados de atenção usados para melhor balancear a importância de cada aresta na representação dos nós e foi capaz de superar modelos de referência já amplamente utilizados. Concluímos que o modelo demonstrou eficácia na realização dessas previsões, superando os modelos comparados e alcançando, em média, coeficientes de determinação R^2 aproximadamente 4% superiores. Além disso, disponibilizamos uma ferramenta para reprodução dos resultados e realização de testes usando diferentes arquiteturas de GNNs e conjuntos de dados. Para o futuro esperamos aprimorar tanto nosso modelo de GNN quanto a ferramenta para testes, melhorando a usabilidade e flexibilidade.

Referências

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD int. conference on knowledge discovery & data mining*, pages 2623–2631.

Bandyopadhyay, S., Biswas, A., Murty, M. N., and Narayanam, R. (2019). Beyond node embedding: A direct unsupervised edge representation framework for homogeneous networks. *CoRR*, abs/1912.05140.

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

Bessadok, A., Mahjoub, M. A., and Rekik, I. (2021). Graph neural networks in network neuroscience. *CoRR*, abs/2106.03535.

Bielak, P., Kajdanowicz, T., and Chawla, N. V. (2020). Attre2vec: Unsupervised attributed edge representation learning. *CoRR*, abs/2012.14727.

Breuer, M. A. (1977). A class of min-cut placement algorithms. In *Proceedings of the 14th Design Automation Conference*, DAC '77, page 284–290. IEEE Press.

Capanema, C., Silva, F., and Loureiro, A. (2022). *Redes Neurais de Grafos no Contexto das Cidades Inteligentes*, pages 135–176.

da Silva, D. N. R., Ziviani, A., and Porto, F. (2019). *Aprendizado de máquina e inferência em Grafos de Conhecimento*, pages 93–122.

Fang, J., Vardi, Y., and Zhang, C.-H. (2007). *An iterative tomogravity algorithm for the estimation of network traffic*, page 12–23. Institute of Mathematical Statistics.

Gao, Z., Fu, G., Ouyang, C., et al. (2019). edge2vec: Representation learning using edge semantics for biomedical knowledge discovery. *BMC Bioinformatics 20*.

Gong, L. and Cheng, Q. (2019). Exploiting edge features in graph neural networks. *CoRR*, abs/1809.02709.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Hanin, B. (2018). Which neural net architectures give rise to exploding and vanishing gradients? In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Hope, O. D. N. (2020). Generalisable data-driven routing using deep rl with gnns. Master's thesis, University of Cambridge, Cambridge.

Jiang, X., Ji, P., and Li, S. (2019). Censnet: Convolution with edge-node switching in graph neural networks. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 2656–2662.

Jo, J., Baek, J., Lee, S., et al. (2021). Edge representation learning with hypergraphs. *CoRR*, abs/2106.15845.

Kim, J., Kim, T., Kim, S., and Yoo, C. D. (2019). Edge-labeling graph neural network for few-shot learning. *CoRR*, abs/1905.01436.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Luo, T. and Pan, D. Z. (2008). Dplace2.0: A stable and efficient analytical placement based on diffusion. In *2008 Asia and South Pacific Design Automation Conference*, pages 346–351.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Tuncer, E., Le, Q., Laudon, J., Ho, R., Carpenter, R., and Dean, J. (2021). A graph placement methodology for fast chip design. *Nature*, 594:207–212.

Peng, Y., Guo, J., and Yang, C. (2024). Learning resource allocation policy: Vertex-gnn or edge-gnn? *IEEE Transactions on Machine Learning in Communications and Networking*, 2:190–209.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining*, pages 701–710. ACM.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593.

Qiu, L., Jin, L., and Chai, L. (2023). Network traffic prediction based on spatio-temporal graph convolutional network. In *2023 42nd Chinese Control Conference (CCC)*, pages 8426–8431. IEEE.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). LINE: Large-Scale Information Network Embedding. In *Proceedings of the 24th Int. Conference on World Wide Web*, pages 1067–1077.

Tao, J., Cao, K., and Liu, T. (2023). Traffic matrix prediction based on cross aggregate gnn. In *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 0234–0239. IEEE.

Teitelbaum, B., Hares, S., Dunn, L., Neilson, R., Narayan, V., and Reichmeyer, F. (1999). Internet2 qbone: building a testbed for differentiated services. *IEEE Network*, 13(5):8–16.

Tsitsulin, A., Palowitch, J., Perozzi, B., and Müller, E. (2023). Graph clustering with graph neural networks.

Valadarsky, A., Schapira, M., Shahaf, D., and Tamar, A. (2017). Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, HotNets-XVI, page 185–191, New York, NY, USA. Association for Computing Machinery.

Veličković, P. (2023). Everything is connected: Graph neural networks. *Current Opinion in Structural Biology*, 79:102538.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2018). Dynamic graph cnn for learning on point clouds.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *CoRR*, abs/1810.00826.

Yang, Y. and Li, D. (2020). Nenn: Incorporate node and edge features in graph neural networks. In *Asian Conference on Machine Learning*, pages 593–608.

Yang, Z., Yang, L. T., Wang, H., Ren, B., and Yang, X. (2023). Bayesian tensor completion for network traffic data prediction. *IEEE Network*, 37(4):74–80.

Zheng, W., Li, Y., Hong, M., Zhao, G., and Fan, X. (2024). Network traffic matrix prediction with incomplete data via masked matrix modeling. *Information Sciences*, 657:119835.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.