EPO: AN EFFICIENT EDGE PROXIMITY OPERATOR FOR EDGE REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph representation learning aims to encode high-dimensional and sparse graph data into low-dimensional dense representations. Despite substantial progress in node representation learning, edge-wise representation learning remains relatively underexplored, and directly adapting node-wise methods often fails to adequately capture edge-specific topology and attributes. To address this limitation, we propose EPO, an efficient and effective framework that jointly models structural proximity and edge features. We first establish a connection between edge-based Personalized PageRank and node-based Personalized PageRank on the corresponding subdivision graph, providing a spanning-forest interpretation of high-order edge proximity. Based on this connection, EPO employs a parallel extension of Wilson's algorithm to efficiently approximate a sparse edge proximity operator with theoretical guarantees. Using this operator, we further develop two models: EPO-F learns edge representations from a matrix-factorization perspective, whereas EPO-P directly propagates edge features to capture high-order structural information. Extensive experiments on six real-world datasets show that our models achieve the best or second-best performance in edge classification while maintaining competitive computational efficiency. Moreover, EPO scales effectively to graphs containing more than 15 million edges, demonstrating its applicability to large-scale edge-wise graph learning.

1 INTRODUCTION

Graph representation learning (GRL) aims to encode graph elements, such as nodes and edges, into low-dimensional vector representations that preserve both structural and feature information. The learned representations can be seamlessly integrated into downstream machine learning models, facilitating a variety of graph analytics tasks, including node classification, edge classification, link prediction, and graph reconstruction (Hamilton et al., 2017; Wang et al., 2024; Tsitsulin et al., 2020).

While node-wise GRL is well established, edge-wise GRL remains relatively underexplored. As fundamental components of graphs, edges play a critical role in various edge-centric network analysis tasks, such as detecting fraudulent transactions in financial networks (Minakawa et al., 2022) and analyzing relationships in social networks. Recent work has further demonstrated that architectures maintaining edge embeddings can be strictly more expressive than their node-only counterparts, both theoretically and empirically, particularly in graphs with high-degree hubs (Rohatgi et al., 2025). Unlike traditional node-focused tasks (e.g., node classification), edge-based tasks often exhibit greater complexity but can similarly benefit from embedding-based approaches in a continuous low-dimensional space (Wang et al., 2020a). Therefore, developing efficient, effective, and scalable edge-wise GRL methods is essential for improving performance across a wide range of real-world applications.

Existing edge-wise graph representation learning methods can be broadly categorized into three lines of research. One line aggregates incident node embeddings, which often fails to capture edge-specific information (Minakawa et al., 2022). Another line transforms edges into nodes via line graphs or similar strategies, but this incurs high computational overhead and distorts global structural information. The third line directly embeds edges, as exemplified by recent works such as AttrE2Vec (Bielak et al., 2022), Edge2Vec (Wang et al., 2020a), GNN-based approaches (Kim et al., 2019; Gong & Cheng, 2019), and TER (Wang et al., 2023). Among these, TER is the state-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

of-the-art for learning edge representations solely from graph topology. It introduces a proximity matrix $\mathbf{\Pi}$ based on edge-based Personalized PageRank (PPR) and leverages randomized SVD (Halko et al., 2011) to efficiently approximate $\mathbf{\Pi}$ and extract topological edge representations. However, it cannot apply non-linear transformations (e.g., logarithm or softmax) to the proximity matrix, limiting its expressive power and inductive capability. Prior work has emphasized the need for scalable factorization techniques that allow for non-linear operations on proximity matrices (Yin & Wei, 2019). Therefore, developing alternative approaches to overcome these limitations remains a critical challenge.

To address this challenge, we propose an efficient edge proximity operator, namely EPO, for edge representation learning. The main contributions of this paper are summarized as follows:

- We establish a connection between edge-based PPR and node-based PPR on the corresponding subdivision graph, enabling the explicit computation of the edge proximity matrix.
- We provide a spanning-forest interpretation of edge-based PPR. Building on this, we derive an unbiased estimator and develop a linear-time approximation algorithm with theoretical guarantees based on weighted forest sampling.
- We then develop two EPO-based edge representation learning models, EPO-F and EPO-P. EPO-F adopts a matrix factorization mechanism to extract structural edge embeddings from the proximity matrix, while EPO-P directly performs feature propagation over EPO to obtain edge representations.
- Extensive experiments on six real-world datasets demonstrate that our method consistently outperforms state-of-the-art baselines. Moreover, the proposed framework exhibits superior scalability and computational efficiency, effectively balancing accuracy and runtime performance.

2 Preliminaries

2.1 Graph and Related Matrices

Let $G = (V, E, \boldsymbol{X}_E)$ be an undirected edge-attributed graph, where $V = \{v_1, \ldots, v_n\}$ is the node set, $E = \{e_1, \ldots, e_m\}$ is the edge set, and $\boldsymbol{X}_E \in \mathbb{R}^{m \times d}$ is the edge feature matrix. Each edge $e = (v_i, v_j)$ connects two nodes. The adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ has entries $\boldsymbol{A}_{i,j} = 1$ if $(v_i, v_j) \in E$, and 0 otherwise. The degree matrix $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{A}\mathbf{1})$, the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$.

The incidence matrix $\boldsymbol{E} \in \mathbb{R}^{n \times m}$ is defined such that for each edge $e = (v_i, v_j) \in E$, the entries $\boldsymbol{E}_{v_i,e} = \boldsymbol{E}_{v_j,e} = 1$, and all others are 0. Let $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{R} \in \mathbb{R}^{n \times m}$ denote the row-normalized versions of $\boldsymbol{E}^\top$ and $\boldsymbol{E}$, respectively. For each edge $e = (v_i, v_j)$, $\boldsymbol{Q}_{e,v_i} = \boldsymbol{Q}_{e,v_j} = 0.5$. For each node v, $\boldsymbol{R}_{v,e} = 1/|E_v|$ for all $e \in E_v$, where E_v is the set of edges incident to v.

2.2 Node-Based Personalized PageRank

We first introduce node-based Personalized PageRank (Node-PPR) (Haveliwala, 2002; Jeh & Widom, 2003). Let $\boldsymbol{P} \in \mathbb{R}^{n \times n}$ denote the row-stochastic transition matrix of the graph. For a source node $s \in V$, its Node-PPR vector $\boldsymbol{\pi}_s \in \mathbb{R}^{1 \times n}$ is defined as the solution to $\boldsymbol{\pi}_s = \alpha \boldsymbol{e}_s^\top + (1-\alpha)\boldsymbol{\pi}_s \boldsymbol{P}$, where $\alpha \in (0, 1)$ is a decay factor. The solution to this equation is $\pi_s = \alpha \boldsymbol{e}_s^\top (\boldsymbol{I} - (1-\alpha)\boldsymbol{P})^{-1}$.

The entry $[\boldsymbol{\pi}_s]_t$ measures the proximity from the source node s to node t. It also admits a random-walk interpretation: starting from s, the walker terminates at its current node with probability α and moves to a neighbor according to $\boldsymbol{P}$ with probability $1 - \alpha$. Under this interpretation, $[\boldsymbol{\pi}_s]_t$ is the probability that the walk eventually terminates at node t. By stacking the Node-PPR vectors of all source nodes, we obtain the Node-PPR matrix $\mathbf{\Pi} \in \mathbb{R}^{n \times n}$, whose s-th row is $\boldsymbol{\pi}_s$. Equivalently, $\mathbf{\Pi} = \alpha \left(\boldsymbol{I} - (1-\alpha)\boldsymbol{P}\right)^{-1} = \sum_{k=0}^{\infty} \alpha(1-\alpha)^k \boldsymbol{P}^k$. Thus, $\mathbf{\Pi}_{s,t} = [\boldsymbol{\pi}_s]_t$ represents the Node-PPR from node s to node t.

3 Proposed Model

In this section, we first introduce the edge proximity operator and then present the EPO framework for edge representation learning. Figure 1 illustrates the overall architecture of the proposed model.

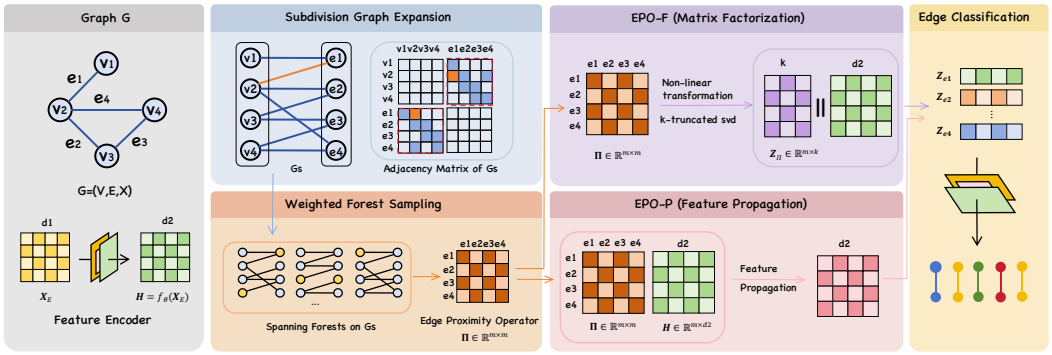


Figure 1: The overall framework of EPO.

3.1 Edge Proximity Operator

Beyond nodes, PPR can also be defined over edges in a graph. Specifically, the matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ represent transition probabilities, where $\boldsymbol{Q}_{e_i,v}$ (resp. $\boldsymbol{R}_{v,e_j}$) denotes the probability of transitioning from edge e_i (resp. node v) to node v (resp. edge e_j) in a single-step random walk. The edge transition matrix, given by

$$\boldsymbol{\Gamma} = \boldsymbol{Q}\boldsymbol{R}, \tag{1}$$

captures the probability of transitioning from edge e_i to edge e_j via an intermediate node $v \in V$.

Then, we introduce edge-based Personalized PageRank (Edge-PPR). For a source edge ξ, the Edge-PPR vector $\pi_\xi \in \mathbb{R}^{1\times m}$ assigns proximity scores $\pi(\xi, e_i)$ to each edge $e_i \in E$, and satisfies the equation: $\pi_\xi = \alpha \boldsymbol{e}_\xi^\top + (1-\alpha)\pi_\xi \boldsymbol{\Gamma}$, with the solution: $\pi_\xi = \alpha \boldsymbol{e}_\xi^\top (\boldsymbol{I} - (1-\alpha)\boldsymbol{\Gamma})^{-1}$. The Edge-PPR matrix, denoted $\boldsymbol{\Pi}$, is an $m \times m$ matrix with entries $\boldsymbol{\Pi}_{e_i,e_j} = [\pi_{e_i}]_{e_j}$, encoding the proximity between edges. $\boldsymbol{\Pi}_{e_i,e_j}$ denotes the probability that an α-random walk starting from edge e_i stops at edge e_j, which is consistent with the definition of high-order similarity in (Wang et al., 2023).

Lemma 3.1 *Both $\boldsymbol{\Gamma}$ and $\boldsymbol{\Pi}$ are symmetric and doubly stochastic.*

In this paper, we focus on the Edge-PPR matrix $\boldsymbol{\Pi}$, which can be regarded as a high-order proximity operator characterizing structural relationships among edges. We use $\boldsymbol{\Pi}$ to denote both the Node-PPR and Edge-PPR matrices when the context is clear.

3.2 The EPO Framework

Based on the Edge-PPR matrix introduced above, we further develop the Edge Proximity Operator (EPO) framework for edge representation learning. The key idea is to use $\boldsymbol{\Pi}$ as a high-order edge proximity operator that captures structural relations among edges beyond local edge adjacency.

Given an edge feature matrix $\boldsymbol{X}_E \in \mathbb{R}^{m\times d}$, where the i-th row $\boldsymbol{X}_{e_i}$ denotes the initial feature of edge e_i, we first encode the input edge features as $\boldsymbol{H} = f_\theta(\boldsymbol{X}_E)$, where $f_\theta(\cdot)$ is implemented as a feed-forward neural network. Based on the encoded edge features $\boldsymbol{H}$, EPO can be instantiated in two ways. The first variant, named **EPO-F**, **f**actorizes $\boldsymbol{\Pi}$ to obtain structural edge embeddings and combines them with the encoded edge features. The second variant, named **EPO-P**, directly uses $\boldsymbol{\Pi}$ as a **p**ropagation operator to aggregate the encoded features of high-proximity edges.

EPO-F: Matrix Factorization with EPO. The first variant aims to extract low-dimensional structural embeddings from the edge proximity operator. Since $\boldsymbol{\Pi}$ encodes high-order proximity between edges, factorizing $\boldsymbol{\Pi}$ provides a natural way to obtain structural edge representations. Specifically, we perform truncated singular value decomposition on $\boldsymbol{\Pi}$ as $\boldsymbol{\Pi} \approx \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{U}_k^\top$, where $\boldsymbol{U}_k \in \mathbb{R}^{m\times k}$ contains the top-k singular vectors and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{k\times k}$ is the diagonal matrix containing the corresponding singular values. We then define the edge embedding as $\boldsymbol{Z}_\Pi = \boldsymbol{U}_k \boldsymbol{\Sigma}_k^{1/2}$. In this way, edges with high Edge-PPR proximity are expected to have similar representations in the latent space. The final edge representation produced by EPO-F is obtained by concatenating the structural embedding with the encoded edge features, i.e., $\boldsymbol{Z}_F = [\boldsymbol{Z}_\Pi || \boldsymbol{H}]$.

EPO-P: Feature Propagation with EPO. The second variant directly treats $\mathbf{\Pi}$ as an edge-level feature propagation operator. Instead of propagating the raw edge features, EPO-P aggregates information from high-proximity edges through $\mathbf{\Pi}\boldsymbol{H}$. For each edge e_i, the propagated representation is given by $(\mathbf{\Pi}\boldsymbol{H})_{e_i} = \sum_{e_j \in E} \mathbf{\Pi}_{e_i,e_j} \boldsymbol{H}_{e_j}$. Thus, each edge aggregates latent feature information from other edges according to their Edge-PPR proximity. The final edge representation produced by EPO-P is therefore defined as $\boldsymbol{Z}_P = \mathbf{\Pi}\boldsymbol{H}$.

Although the model can effectively learn edge representations, efficiency remains a critical concern. Directly solving the above formulation is still challenging, despite the availability of robust matrix factorization and dimensionality reduction techniques (Ou et al., 2016; Yang et al., 2020; Yin & Wei, 2019; Zhang et al., 2021; 2018). The main bottleneck stems from the high computational and storage costs of constructing the dense Edge-PPR matrix. Specifically, naively computing $\boldsymbol{P}$ by Equation 1 requires $O(m^2)$ time in the worst case. Evaluating Edge-PPR for all edge pairs further incurs $O(m^2)$ time and memory, while decomposing the resulting dense $m \times m$ matrix costs $O(m^3)$ time. These costs make exact computation impractical for large-scale graphs.

The state-of-the-art method TER (Wang et al., 2023) improves scalability through a randomized-SVD-based two-stage framework (Halko et al., 2011), computing edge embeddings $\boldsymbol{Z}$ without explicitly constructing the Edge-PPR matrix in $O(mk^2)$ time and $O(mk)$ space, where k is the representation dimension. However, since the proximity matrix is never explicitly formed, TER cannot directly access or apply element-wise non-linear transformations, such as logarithm or softmax, to its entries, limiting its expressiveness and flexibility (Zhou et al., 2017; Qiu et al., 2018; Yin & Wei, 2019).

4 Fast Computation of the Proximity Operator

In this section, we propose an efficient algorithm based on subdivision graph expansion and forest sampling to approximate entries of the proximity matrix $\mathbf{\Pi}$ in linear time. The proposed method provides theoretical guarantees while avoiding the explicit construction of the dense proximity matrix.

4.1 Reducing Edge-PPR to Node-PPR via Subdivision Graph

The Edge-PPR matrix $\mathbf{\Pi}$ is an $m \times m$ row-stochastic matrix, which can be naturally interpreted as the transition matrix of a random walk over edges. However, for large-scale graphs, explicitly computing and storing $\mathbf{\Pi}$ is prohibitively expensive, requiring up to $O(m^3)$ time and $O(m^2)$ space. To overcome this limitation, we show that computing PPR values between edges in the original graph can be reduced to computing PPR values between nodes in an associated subdivision graph.

To this end, we first introduce the concept of the subdivision graph. Given a graph $G = (V, E)$ with n nodes and m edges, its subdivision graph G_S is obtained by replacing each edge in G with a new intermediate node. Specifically, for each edge in G, we insert one new node to represent this edge and connect it to the two endpoints of the original edge. Such an expansion incurs a time complexity of $O(m)$.

We refer to these newly inserted nodes as edge-nodes and denote their set by V_S^e. Therefore, G_S has $n + m$ nodes and $2m$ edges (Cvetkovic, 1975). Since every edge in G_S connects an original node to an edge-node, G_S is a bipartite graph. Its adjacency matrix and transition matrix can be written as

$$\boldsymbol{A}_S = \begin{pmatrix} 0 & \boldsymbol{E} \\ \boldsymbol{E}^\top & 0 \end{pmatrix}, \boldsymbol{P}_S = \begin{pmatrix} 0 & \boldsymbol{R} \\ \boldsymbol{Q} & 0 \end{pmatrix}. \tag{2}$$

As illustrated in Figure 1, the original graph $G = (V, E)$ consists of 4 nodes and 4 edges. The corresponding subdivision graph G_S contains 8 nodes and 8 edges. We next demonstrate that there is a direct relationship between the Edge-PPR in the original graph and the Node-PPR computed on the subdivision graph. Specifically, we have the following result.

Theorem 4.1 *Given a graph $G = (V, E)$ and its subdivision graph G_S, let $e_1, e_2 \in E$ be two edges in G, and let the corresponding edge-nodes in G_S also be denoted by e_1 and e_2 for clarity. Let $\mathbf{\Pi}$ denote the Edge-PPR matrix on G with teleport probability α, and let $\mathbf{\Pi}'$ denote the Node-PPR matrix on G_S with teleport probability α^*, where $\alpha^* = 1 - \sqrt{1-\alpha}$. Then, we have*

$$\mathbf{\Pi}_{e_1,e_2} = \frac{\alpha}{\alpha^*} \mathbf{\Pi}'_{e_1,e_2}.$$

Thus, we conclude that the edge-PPR on the original graph is related to the node-PPR on the subdivision graph by a constant scaling factor. It is obvious that the time complexity of this process is $O(m)$.

4.2 Spanning Forest Interpretation

Building on the above derivation, we reduce the problem to computing the node-level PPR matrix $\mathbf{\Pi}'$ on the subdivision graph G_s. We further show that the forest sampling technique can construct element-wise unbiased estimators for $\mathbf{\Pi}'$ with theoretical error guarantees.

We begin by introducing some key concepts. A spanning tree of a graph is a minimal connected subgraph that includes all nodes. In a tree T with a designated root node j, every node $i \in T$ is considered rooted at j. A spanning forest is a subgraph of the original graph that contains no cycles. Each connected component of a spanning forest has a unique root, forming a spanning rooted forest, whose set of root nodes is referred to as the root set. Let $\mathcal{F}$ denote the collection of all spanning rooted forests of G. For any $F \in \mathcal{F}$, we define $\rho(F)$ as the set of its root nodes. Each node u belongs to a unique component of F, and we denote its corresponding root by $r_F(u)$. The subset $\mathcal{F}_{u\to v}$ consists of forests in which u and v belong to the same component, with v serving as the root: $\mathcal{F}_{u\to v} = \{F \in \mathcal{F} \mid r_F(u) = v\}$. The weight of a spanning forest F, denoted by $w(F)$, is the product of the weights of its edges. For a set of spanning forests $\mathcal{F}$, the total weight is the sum of the weights of all forests in the set: $w(\mathcal{F}) = \sum_{F\in\mathcal{F}} w(F)$.

Recall that a combinatorial interpretation of Node-PPR using spanning rooted forests was established in (Chung & Zhao, 2010). Leveraging the connection in Theorem 4.1, we now extend this perspective to Edge-PPR, presenting its corresponding interpretation in terms of spanning rooted forests, as stated in the following theorem.

Theorem 4.2 *Let $G = (V, E)$ be a graph with edges $e_1, e_2 \in E$, and let G_S be its subdivision graph with a node set V_S, where each original edge corresponds to a node in G_S, i.e., $\{e_1, e_2\} \subset V_S$. The Edge-PPR between e_1 and e_2 is derived as*

$$\mathbf{\Pi}_{e_1,e_2} = \frac{\alpha}{\alpha^*} \frac{\sum_{F\in\mathcal{F}_{e_1\to e_2}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\sum_{F\in\mathcal{F}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i},$$

where $\mathcal{F}$ is the set of all spanning rooted forests in G_S, and $\mathcal{F}_{e_1\to e_2}$ is the subset of forests in which e_1 and e_2 are in the same connected component, with e_2 as the root. Additionally, d_i denotes the degree of node i in G_S. Here, $\alpha^* = 1 - \sqrt{1-\alpha}$.

Based on Theorem 4.2, estimating $\mathbf{\Pi}_{e_1,e_2}$ amounts to sampling a rooted spanning forest F from the distribution

$$p(F) = \frac{\prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\sum_{F'\in\mathcal{F}} \prod_{i\in\rho(F')} \frac{\alpha^*}{1-\alpha^*} d_i},$$

and checking whether F belongs to $\mathcal{F}_{e_1\to e_2}$. We next show that such forests can be sampled exactly using the weighted Wilson algorithm.

4.3 Algorithm Design and Analysis

Wilson's sampling algorithm (Wilson, 1996) and its extension (Avena & Gaudillière, 2018) provide efficient methods for uniformly sampling spanning trees and rooted spanning forests via loop-erased random walks (Lawler, 1980). With appropriate modifications to the transition probabilities, the algorithm can further be adapted to sample spanning forests according to a prescribed weight distribution.

To sample forests according to the distribution $p(F)$ in Theorem 4.2, we first construct a weighted directed graph from the subdivision graph G_S. Let $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{w})$, where $\tilde{V} = V_S$. For each undirected edge $\{u, v\}$ in G_S, we introduce two directed edges $u \to v$ and $v \to u$ in $\tilde{G}$, with weights $\tilde{w}_{uv} = \frac{1-\alpha^*}{\alpha^* d_u}$, and $\tilde{w}_{vu} = \frac{1-\alpha^*}{\alpha^* d_v}$, respectively. For any rooted spanning in-forest F, each non-root node has exactly one outgoing edge. Therefore, $\prod_{e\in F} \tilde{w}_e = \prod_{i\notin\rho(F)} \frac{1-\alpha^*}{\alpha^* d_i} \propto \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i$.

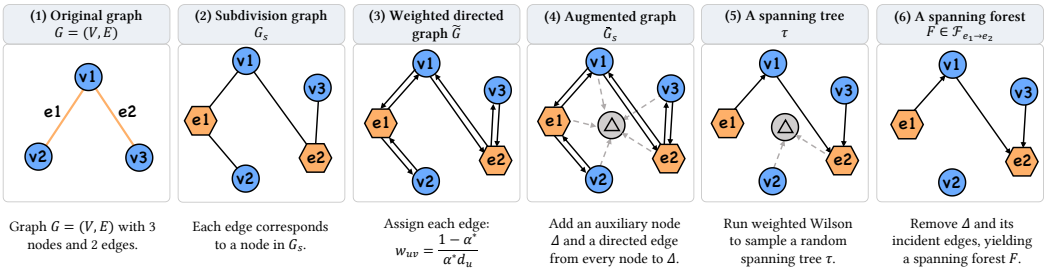


Figure 2: The process of subdivision graph expansion and weighted forest sampling.

Hence, the target distribution $p(F)$ in Theorem 4.2 is equivalent to the weighted rooted-forest distribution induced by $\tilde{G}$.

Next, we construct an augmented graph $\bar{G}_s$ by adding an auxiliary node Δ to $\tilde{G}$ and connecting every node to Δ with a unit-weight edge. We then run Wilson's algorithm on $\bar{G}_s$ using the weighted random walk whose transition probability from u to an adjacent node v is proportional to the corresponding edge weight $\tilde{w}_{uv}$. The weighted Wilson algorithm generates a spanning tree T according to the distribution $\mathbb{P}(T = T_k) \propto \prod_{e \in T_k} \tilde{w}_e$.

Each spanning tree T of $\bar{G}_s$ uniquely induces a rooted spanning forest F of G_s after removing Δ and its incident edges, where the nodes adjacent to Δ form the root set $\rho(F)$. Let $T(F)$ denote the augmented spanning tree corresponding to F. Since all edges incident to Δ have unit weight, we have:

$$\mathbb{P}(F = F_k) = P(T = T_k) \propto \prod_{e \in T_k} \tilde{w}_e \propto \prod_{i \in \rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i.$$

Therefore, the induced forest distribution is exactly $p(F)$, and the weighted Wilson algorithm allows us to sample directly from the desired rooted-forest distribution.

Given a rooted spanning forest F sampled according to the distribution p, we define the estimator for $\mathbf{\Pi}_{e_1,e_2}$ as $\widehat{\mathbf{\Pi}}^{(F)}_{e_1,e_2} = \frac{\alpha}{\alpha^*} \mathbb{I}_{\{r_F(e_1)=e_2\}}$, where $\mathbb{I}_{\{r_F(e_1)=e_2\}}$ is 1 when e_2 is the root of e_1, and 0 otherwise. The estimator is unbiased since

$$\mathbb{E}\left[\widehat{\mathbf{\Pi}}^{(F)}_{e_1,e_2}\right] = \frac{\alpha}{\alpha^*} \mathbb{P}_{F \sim p}\left\{r_F(e_1) = e_2\right\} = \mathbf{\Pi}_{e_1,e_2}.$$

Using l independent samples, we approximate $\mathbf{\Pi}_{e_1,e_2}$ via the empirical mean: $\widetilde{\mathbf{\Pi}}_{e_1,e_2} = \frac{1}{l} \sum_{k=1}^{l} \widetilde{\mathbf{\Pi}}^{(F_k)}_{e_1,e_2}$. We then use Hoeffding's inequality (Hoeffding, 1994) to determine the required sample size l, and establish an (ϵ, δ)-approximation of $\mathbf{\Pi}_{e1,e2}$ in Theorem 4.3.

Theorem 4.3 *For any $\epsilon, \delta \in (0,1)$, if $l = \left\lceil \frac{\alpha^2}{2\epsilon^2(\alpha^*)^2} \ln \frac{2}{\delta} \right\rceil$, where $\alpha^* = 1 - \sqrt{1-\alpha}$, then with probability at least $1-\delta$, $\mathbf{\Pi}_{e_1,e_2} - \epsilon \leq \widetilde{\mathbf{\Pi}}_{e_1,e_2} \leq \mathbf{\Pi}_{e_1,e_2} + \epsilon$.*

4.4 Implementation Details

Sparse Matrix Factorization. Prior studies (Zhou et al., 2017; Qiu et al., 2018) have shown that skip-gram-based methods implicitly factorize the entry-wise logarithm of certain proximity matrices. Accordingly, logarithmic transformations have been widely used in proximity-based matrix factorization methods (Yin & Wei, 2019; Zhang et al., 2021), as they compress the dynamic range of proximity values and can improve the quality of the resulting representations. Following this practice, EPO-F applies an entry-wise logarithmic transformation to the explicitly approximated Edge-PPR matrix $\widetilde{\mathbf{\Pi}}$ before factorization. We also scale all nonzero entries by a factor of $\frac{\alpha^* l}{\alpha}$ to ensure that all entries exceed 1.

The approximated Edge-PPR matrix is naturally sparse. In particular, given l sampled rooted spanning forests, each row of $\widetilde{\mathbf{\Pi}}$ contains at most l nonzero entries, where $l \ll m$. Consequently,

$nnz(\widetilde{\mathbf{\Pi}}) \leq ml$, allowing the transformed proximity matrix to be stored and processed efficiently in sparse form. We then perform truncated singular value decomposition on the sparse matrix $\widetilde{\mathbf{\Pi}}$. When retaining the top-k singular components, the factorization requires $O(nnz(\widetilde{\mathbf{\Pi}}) + mk^2)$ time (Halko et al., 2011), where k denotes the target embedding dimension.

Parallel Sampling. A key advantage of Algorithm 1 over conventional numerical methods for computing $\widetilde{\mathbf{\Pi}}$ is that its sampling procedure can be naturally parallelized. Since the sampled spanning forests are mutually independent, different samples can be assigned to separate threads without requiring synchronization during the sampling stage. Based on this property, we develop a parallel implementation using the Python Numba library. Numba employs just-in-time compilation to generate native machine code and releases the Python Global Interpreter Lock during parallel execution, thereby enabling effective multi-threaded sampling on multi-core CPUs. The independently obtained estimates are finally aggregated to construct the approximate Edge-PPR matrix.

Lightweight Training. Our models are designed to be lightweight and efficient. Instead of relying on deeply stacked neural architectures, EPO performs the structural computation in a preprocessing stage on the CPU. After obtaining the edge proximity matrix, EPO-F and EPO-P respectively employ matrix factorization and feature propagation to replace repeated multi-layer message passing, thereby reducing the computational and memory overhead during training. Consequently, both variants require only a simple MLP on the GPU.

5 Experiments

In this section, we evaluate the effectiveness and efficiency of EPO on edge classification using real-world graph datasets and compare its performance with state-of-the-art baselines. Our code is available at `https://anonymous.4open.science/r/EPO-FC35`.

5.1 Experimental Settings

Datasets and Baselines. We conduct experiments on six real-world datasets, whose statistics are provided in Table 3 in the appendix. We compare our model with thirteen representative baselines, including both node-wise and edge-wise GRL methods. More detailed descriptions and implementation details are provided in the appendix.

Metrics. We report AUC and Macro-F1 on the test sets to account for class imbalance, using the model that achieves the highest AUC on the validation set.

Hyperparameter Settings. We set the decay factor α to 0.5, the forest sampling size l to 1,000, and the truncated SVD dimension k to 256 for all datasets. For the parallel sampling, we set the number of threads to 12. A detailed sensitivity analysis of these hyperparameters is presented in Section 5.4. During training, the learned edge representations are fed into an MLP classifier with one 256-dimensional hidden layer, ReLU activation, batch normalization, and a dropout rate of 0.5. We train the classifier for up to 300 epochs using the Adam optimizer with a learning rate of 0.001 and binary cross-entropy with logits as the loss function.

5.2 Edge Classification Performance

The classification performance of all competing methods across the six datasets is summarized in Table 1, where '-' denotes methods encounter out-of-memory errors.

Compared with node-wise representation learning methods, including GCN, GraphSAGE, DGI, GAT, and GATv2, both EPO-F and EPO-P achieve consistently higher Macro-F1 scores and AUC values across the six datasets, with particularly large gains on MAG, Reddit, MIND, and DBLP. This suggests that constructing representations directly at the edge level is more suitable for edge classification than aggregating node embeddings into edge features.

The proposed methods also outperform conventional edge-wise baselines, including Edge2vec, AttrE2Vec, and Edgeformer, on most datasets and evaluation metrics. Against the strongest competing methods, TER+AER and TopoEdge, EPO-F and EPO-P retain clear advantages in the majority of cases. Specifically, on MOOC, EPO-F improves the best TER+AER result from 0.5803 to 0.6137 in

Table 1: Edge classification performance on six datasets. The best and second-best results are highlighted.

Method	MOOC		Yelpchi		MAG		Reddit		MIND		DBLP	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
GCN	0.5063	0.6249	0.5045	0.8023	0.0677	0.5308	0.0895	0.4551	0.3573	0.5000	0.1614	0.4950
GraphSAGE	0.4976	0.5314	0.4831	0.7904	0.1258	0.5254	0.3695	0.5036	0.3409	0.4953	0.1702	0.4971
DGI	0.4728	0.5208	0.4469	0.7851	0.0989	0.4501	0.0651	0.4126	0.2377	0.4571	0.1358	0.4321
GAT	0.4812	0.4288	0.4516	0.7624	0.1243	0.4832	0.5103	0.4987	0.3561	0.4828	0.1551	0.4862
GATv2	0.4890	0.4353	0.4516	0.7701	0.1259	0.4957	0.5154	0.4997	0.3572	0.4879	0.1603	0.4895
PSNE	0.5082	0.5921	0.4209	0.6325	0.2093	0.8321	0.5017	0.5991	0.4109	0.6283	0.2231	0.5498
node2vec	0.4970	0.4757	0.4713	0.6025	0.0875	0.7481	0.4748	0.5052	0.2324	0.5008	0.0831	0.5132
Edge2vec	-	-	-	-	0.0744	0.4947	-	-	-	-	-	-
AttrE2Vec	0.5123	0.6959	0.6126	0.7820	0.3299	0.9181	0.5445	0.7052	0.5221	0.7569	0.2468	0.6139
Edgeformer	0.5003	0.6392	0.6375	0.7502	0.3120	0.9008	0.5266	0.6861	0.5078	0.7182	0.2104	0.5312
TER+AER(G)	0.5803	0.8433	0.6525	0.8231	0.9076	0.9909	0.7912	0.9382	0.8965	0.9754	0.4526	0.9064
TER+AER(P)	0.5795	0.8449	0.6517	0.8252	0.8645	0.9921	0.7890	0.9356	0.8999	0.9763	0.4675	0.9125
TopoEdge(GCN)	0.5627	0.8210	0.6108	0.7982	0.8266	0.9573	0.8205	0.9470	0.8725	0.9362	0.4507	0.9086
EPO-F$_{\text{w/o log}}$	0.5825	0.8388	0.6437	0.8301	0.9122	0.9863	0.7840	0.9284	0.8953	0.9776	0.4780	0.9052
EPO-F	0.6137	0.8633	0.6520	0.8426	0.9745	0.9989	0.8104	0.9466	0.9064	0.9847	0.4892	0.9176
EPO-P	0.6119	0.8309	0.6441	0.8340	0.9065	0.9869	0.8120	0.9481	0.9130	0.9780	0.4910	0.9191

Macro-F1 and from 0.8449 to 0.8633 in AUC. On MAG, EPO-F achieves 0.9745 in Macro-F1 and 0.9989 in AUC, substantially exceeding both TER+AER variants and TopoEdge. On Reddit, EPO-P obtains the highest Macro-F1 and AUC, reaching 0.8120 and 0.9481, respectively. On DBLP, EPO-P further improves the best TER+AER results from 0.4675 to 0.4910 in Macro-F1 and from 0.9125 to 0.9191 in AUC. Although EPO-F and EPO-P do not rank first on every individual metric, they achieve the best or second-best results on most metrics, indicating a favorable overall performance across different datasets.

5.3 Efficiency Evaluation

Table 2 reports the running times of EPO and the competing methods across six datasets. While node-wise GNN methods are relatively efficient on small datasets, their computational costs increase substantially on larger graphs such as MIND and DBLP. In contrast, EPO exhibits favorable scalability and remains considerably more efficient than most baselines. For example, on DBLP, EPO-F requires only 6,447 seconds, achieving a $6.9\times$ speedup over GCN and a $62.3\times$ speedup over AttrE2Vec. EPO also achieves running times comparable to or lower than TER+AER while delivering better classification performance. In particular, EPO-F is consistently faster than TER+AER(G) across all six datasets, and EPO-P remains computationally competitive with TER+AER(P).

The efficiency of EPO can be primarily attributed to its parallel forest-sampling algorithm. Since individual spanning-forest samples are generated independently, the sampling process can be executed concurrently, substantially reducing the cost of constructing edge proximity representations. Moreover, EPO employs a lightweight representation learning and classification pipeline without computationally expensive iterative message passing. These properties allow EPO to achieve an advantageous trade-off between predictive performance and computational efficiency, particularly when applied to large-scale graphs.

5.4 Parameter Analysis

We analyze the impact of three key parameters in EPO: the decay factor α, the sampling size l, and the embedding dimension k. Since EPO-P does not perform matrix factorization, the analysis of the SVD dimension k applies only to EPO-F. The AUC results across all datasets are shown in Figure 3.

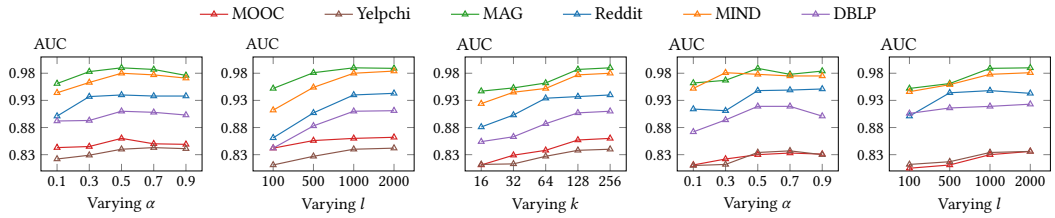
Decay Factor α. The decay factor α controls the contribution of high-order structural proximity to the learned edge representations. As shown in Figure 3, both EPO-F and EPO-P achieve consistently strong performance on nearly all datasets when α lies between 0.3 and 0.7, indicating that the two models are relatively robust to its selection within this range. Their performance generally peaks at $\alpha = 0.5$, which is consistent with the setting adopted in prior work (Wang et al., 2023).

Table 2: Total running time (seconds) of EPO and competitors on six datasets.

Type	Method	MOOC	Yelpchi	MAG	Reddit	MIND	DBLP
Node-Wise GRL Methods	GCN	65.37	50.82	76.14	108.63	1.59k	44.52k
	GraphSAGE	69.21	56.74	89.46	182.35	1.86k	50.91k
	DGI	121.68	101.29	132.57	227.91	14.33k	100.95k
	GAT	70.43	63.16	90.78	190.52	1.96k	53.80k
	GATv2	82.65	71.34	95.27	194.83	2.43k	59.23k
	PSNE	293.19	241.76	391.42	712.58	9.23k	31.10k
	node2vec	761.54	158.23	1.20k	1.34k	23.02k	98.95k
Edge-Wise GRL Methods	AttrE2Vec	1.44k	566.71	1.21k	10.52k	142.91k	401.33k
	Edgeformer	280.36	174.84	890.25	4.21k	52.02k	121.02k
	TER+AER(G)	167.62	42.18	75.47	534.73	815.31	8.40k
	TER+AER(P)	163.28	46.69	73.15	542.86	801.54	8.34k
	TopoEdge(GCN)	243.77	47.32	66.81	609.24	2.47k	61.08k
Ours	EPO-F	156.41	35.73	63.26	400.68	802.17	6.45k
	EPO-P	192.59	22.34	79.82	431.45	872.63	7.02k

Sampling Size l. According to Figure 3, increasing l from 100 to 1,000 substantially improves AUC across all datasets, indicating that additional sampled forests provide more accurate proximity estimates and improve the quality of the learned edge representations. Although performance generally continues to increase at $l = 2,000$, the improvement becomes marginal, suggesting that excessive sampling yields limited additional benefits while incurring greater computational overhead. Therefore, we set $l = 1,000$ as the default to achieve a favorable balance between effectiveness and efficiency.

Embedding Dimension k. We vary the embedding dimension k from 16 to 256. As shown in Figure 3, increasing k consistently improves AUC across all evaluated datasets. This result indicates that higher-dimensional embedding spaces can preserve richer structural and attribute information, thereby producing more expressive edge representations. Based on these observations, we set $k = 256$ as the default embedding dimension.



(a) Varying α, l, and k in EPO-F

(b) Varying α and l in EPO-P

Figure 3: Parameter sensitivity analysis of EPO.

6 Conclusion

This paper presents EPO, an efficient and effective framework for edge-wise representation learning that jointly captures graph topology and edge attributes. EPO employs parallel weighted forest sampling on the subdivision graph to efficiently construct a sparse edge proximity operator with theoretical guarantees. Based on this operator, we develop two complementary variants: EPO-F extracts compact topological representations through an entry-wise nonlinear transformation and truncated SVD, whereas EPO-P directly propagates encoded edge features through the proximity operator. Extensive experiments on six real-world datasets demonstrate that both variants achieve superior edge-classification performance and competitive computational efficiency. In particular, EPO scales effectively to graphs containing more than 15 million edges, benefiting from its parallel sampling procedure and lightweight training pipeline.

In future work, we plan to extend EPO to dynamic and heterogeneous graphs to capture more complex edge interactions. We will also investigate its applicability to a broader range of edge-level tasks, such as temporal edge classification and anomaly detection.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing, wording refinement, and partial code generation. All AI-assisted code was carefully reviewed, verified, and tested for correctness by the authors. We reviewed all AI-assisted content and take full responsibility for the final content of this work, including any text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses publicly available datasets and does not involve human subjects or the collection of sensitive personal information.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide complete proofs for all lemmas and theorems in the appendix. The model architecture and algorithmic details are described in detail in Section 3 and the appendix, respectively. For the experimental evaluation, detailed descriptions of the datasets and baseline methods are provided in the appendix, while the hyperparameter settings are specified in Section 5.1. Our source code is publicly available at `https://anonymous.4open.science/r/EPO-FC35`.

REFERENCES

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 37–48, 2013.

Luca Avena and Alexandre Gaudillière. Two applications of random spanning forests. *Journal of Theoretical Probability*, 31(4):1975–2004, 2018.

Sambaran Bandyopadhyay, Anirban Biswas, M Narasimha Murty, and Ramasuri Narayanam. Beyond node embedding: a direct unsupervised edge representation framework for homogeneous networks. *arXiv preprint arXiv:1912.05140*, 2019.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14, 2001.

Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Attre2vec: Unsupervised attributed edge representation learning. *Information Sciences*, 592:82–96, 2022.

Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks? In *ICLR*, may 2022.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 891–900, 2015.

Seth Chaiken. A combinatorial proof of the all minors matrix tree theorem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):319–329, 1982.

Pavel Chebotarev and Elena Shamis. The matrix-forest theorem and measuring relations in small social groups. *arXiv preprint math/0602070*, 2006.

Xueqi Cheng, Yu Wang, Yunchao Liu, Yuying Zhao, Charu C. Aggarwal, and Tyler Derr. Edge classification on graphs: New directions in topological imbalance. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pp. 392–400. ACM, 2025.

Fan Chung and Wenbo Zhao. Pagerank and random walks on graphs. In *Fete of combinatorics and computer science*, pp. 43–62. Springer, 2010.

Dragoš Cvetkovic. Spectra of graphs formed by some unary operations. *Publ. Inst. Math.(Beograd)*, 19(33):37–41, 1975.

Noah E Friedkin and Eugene C Johnsen. Social influence and opinions. *Journal of mathematical sociology*, 15(3-4):193–206, 1990.

Liyu Gong and Qiang Cheng. Exploiting edge features for graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9211–9219, 2019.

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

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2011.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Taher H Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th international conference on World Wide Web*, pp. 517–526, 2002.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

Glen Jeh and Jennifer Widom. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279, 2003.

Xiaodong Jiang, Pengsheng Ji, and Sheng Li. Censnet: Convolution with edge-node switching in graph neural networks. In *IJCAI*, pp. 2656–2662, 2019.

Bowen Jin, Yu Zhang, Yu Meng, and Jiawei Han. Edgeformers: Graph-empowered transformers for representation learning on textual-edge networks. In *11th International Conference on Learning Representations, ICLR 2023*, 2023.

Yujia Jin, Qi Bao, and Zhongzhi Zhang. Forest distance closeness centrality in disconnected graphs. In *2019 IEEE International Conference on Data Mining (ICDM)*, pp. 339–348. IEEE, 2019.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11–20, 2019.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR 2017*, sep 2017.

Srijan Kumar, William L Hamilton, Jure Leskovec, and Dan Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 world wide web conference*, pp. 933–943, 2018.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1269–1278, 2019.

Gregory F. Lawler. A self-avoiding random walk. *Duke Mathematical Journal*, 47:655–693, 1980.

Suxue Li, Haixia Zhang, Dalei Wu, Chuanting Zhang, and Dongfeng Yuan. Edge representation learning for community detection in large scale information networks. In *Mobility Analytics for Spatio-Temporal and Social Data: First International Workshop, MATES 2017, Munich, Germany, September 1, 2017, Revised Selected Papers 1*, pp. 54–72. Springer, 2018.

Meihao Liao, Rong-Hua Li, Qiangqiang Dai, and Guoren Wang. Efficient personalized pagerank computation: A spanning forests sampling based approach. In *Proceedings of the 2022 International Conference on Management of Data*, pp. 2048–2061, 2022.

Longlong Lin, Yunfeng Yu, Zihao Wang, Zeli Wang, Yuying Zhao, Jin Zhao, and Tao Jia. Psne: Efficient spectral sparsification algorithms for scaling network embedding. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1420–1429, 2024.

Ilya Makarov, Ksenia Korovina, and Dmitrii Kiselev. Jonnee: Joint network nodes and edges embedding. *IEEE Access*, 9:144646–144659, 2021.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR*, 2013.

Naoto Minakawa, Kiyoshi Izumi, Hiroki Sakaji, and Hitomi Sano. Graph representation learning of banking transaction network with edge weight-enhanced attention and textual information. In *Companion Proceedings of the Web Conference 2022*, pp. 630–637, 2022.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 459–467, 2018.

Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 985–994. ACM, 2015.

Dhruv Rohatgi, Tanya Marwah, Zachary Chase Lipton, Jianfeng Lu, Ankur Moitra, and Andrej Risteski. Towards characterizing the value of edge embeddings in graph neural networks. In *Forty-second International Conference on Machine Learning*, 2025.

Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June Hsu, and Kuansan Wang. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th international conference on world wide web*, pp. 243–246, 2015.

Haoxin Sun and Zhongzhi Zhang. Opinion optimization in directed social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4623–4632, 2023.

Haoxin Sun and Zhongzhi Zhang. Efficient computation for diagonal of forest matrix via variance-reduced forest sampling. In *Proceedings of the ACM on Web Conference 2024*, pp. 792–802, 2024.

Yubo Sun, Haoxin Sun, and Zhongzhi Zhang. Scalable algorithms for forest-based centrality on large graphs. In *Proceedings of the ACM Web Conference 2025*, 2025.

Jie Tang, Duo Zhang, and Limin Yao. Social network extraction of academic researchers. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pp. 292–301. IEEE, 2007.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. Verse: Versatile graph embeddings from similarity measures. In *WWW*, pp. 539–548, 2018.

Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Ivan Oseledets, and Emmanuel Müller. Frede: anytime graph embeddings. *arXiv preprint arXiv:2006.04746*, 2020.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

Changping Wang, Chaokun Wang, Zheng Wang, Xiaojun Ye, and Philip S. Yu. Edge2vec: Edge-based social network embedding. *ACM Trans. Knowl. Discov. Data*, 14(4), May 2020a. ISSN 1556-4681.

Hewen Wang, Renchi Yang, Keke Huang, and Xiaokui Xiao. Efficient and effective edge-wise graph representation learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2326–2336, 2023.

Hewen Wang, Renchi Yang, and Xiaokui Xiao. Effective edge-wise representation learning in edge-attributed bipartite graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3081–3091, 2024.

Pengyang Wang, Jiaping Gui, Zhengzhang Chen, Junghwan Rhee, Haifeng Chen, and Yanjie Fu. A generic edge-empowered graph convolutional network via node-edge mutual enhancement. In *Proceedings of the web conference 2020*, pp. 2144–2154, 2020b.

David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 296–303, 1996.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 3597–3606, 2020.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *arXiv*, 2019.

Renchi Yang, Jieming Shi, Xiaokui Xiao, Yin Yang, and Sourav S Bhowmick. Homogeneous network embedding for massive graphs via reweighted personalized pagerank. *Proceedings of the VLDB Endowment*, 13(5):670–683, 2020.

Yulei Yang and Dongsheng Li. Nenn: Incorporate node and edge features in graph neural networks. In *Asian conference on machine learning*, pp. 593–608. PMLR, 2020.

Yuan Yin and Zhewei Wei. Scalable graph embeddings via sparse transpose proximities. In *KDD*, pp. 1429–1437, 2019.

Xingyi Zhang, Kun Xie, Sibo Wang, and Zengfeng Huang. Learning based proximity matrix factorization for node embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 2243–2253, 2021.

Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2778–2786, 2018.

Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Xiaotian Zhou, Haoxin Sun, Wanyue Xu, Wei Li, and Zhongzhi Zhang. Friedkin-johnsen model for opinion dynamics on signed graphs. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

A NOTATIONS

We use lowercase letters like a, b, c for scalars in $\mathbb{R}$, uppercase letters like A, B, C for sets, bold lowercase letters like $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ for vectors, and bold uppercase letters like $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ for matrices. Let $\boldsymbol{e}_i$ be the standard basis vector with 1 in the i-th entry, and $\mathbf{1}$ the all-ones vector. Let $\boldsymbol{I}$ be the identity matrix, and $\boldsymbol{a}^\top$, $\boldsymbol{A}^\top$ denote transposes. Matrix elements are denoted by $\boldsymbol{A}_{i,j}$, and row vectors by $\boldsymbol{A}_i$. For a set A, let $|A|$ be its cardinality.

B RELATED WORK

In this section, we briefly review the existing work related to ours.

B.1 NODE-WISE GRL

Node-wise graph representation learning (GRL) focuses on generating embeddings for individual nodes within a graph. Existing approaches can be categorized into three main types, with random walk-based methods being the first. These methods, inspired by the Skip-Gram model (Mikolov et al., 2013), aim to capture node representations by preserving co-occurrence patterns in sampled walks. The primary distinction among these approaches lies in how they generate positive samples through different random walk strategies. For example, DeepWalk (Perozzi et al., 2014) employs truncated random walks, while Node2vec (Grover & Leskovec, 2016) introduces variations such as higher-order walks and adaptive search strategies that balance depth-first and breadth-first traversal. Additionally, APP (Zhou et al., 2017) and VERSE (Tsitsulin et al., 2018) refine the sampling process with α-random walks. A common challenge shared by these methods is the high computational cost.

The second category of methods consists of matrix factorization-based approaches, which aim to learn node representations by decomposing a proximity matrix that encodes graph structure. These methods begin by constructing a matrix that reflects node connectivity or similarity, followed by dimensionality reduction to obtain lower-dimensional embeddings. In fact, DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016) implicitly perform matrix factorization to capture high-order proximities among nodes. Early techniques, such as Laplacian Eigenmaps (Belkin & Niyogi, 2001) and Graph Factorization (Ahmed et al., 2013), preserved graph topology in a lower-dimensional space. GraRep (Cao et al., 2015) extended this approach by incorporating higher-order relationships through singular value decomposition (SVD). However, the primary challenge lies in scalability, as computing the full proximity matrix is computationally expensive, making these methods impractical for large graphs. To address this, several methods based on Personalized PageRank (PPR) have been developed. HOPE (Ou et al., 2016) introduced sparse matrix techniques to accelerate factorization, thereby reducing computational overhead. AROPE (Zhang et al., 2018) built upon this by enabling the flexible preservation of relationships at different levels. Other methods aim to improve accuracy or efficiency, such as NRP (Yang et al., 2020), which adjusts PPR's weighting mechanism, and FREDE (Tsitsulin et al., 2020), which employs matrix sketching to approximate proximities without constructing the full matrix. While many of these methods avoid computing large matrices, they do not support non-linear operations on the proximity matrix, limiting their inductive strength. STRAP (Yin & Wei, 2019) addresses this limitation by computing the sparse proximity matrix and imposing a sparsity constraint, enabling the use of non-linear operations while performing efficient matrix factorization to obtain the embedding vectors. As a result, STRAP successfully combines scalability and non-linearity. Lemane (Zhang et al., 2021) introduces a learnable decay factor in PPR, providing greater flexibility at the cost of higher computational overhead. PSNE (Lin et al., 2024) further enhances efficiency by speeding up PPR matrix computation and incorporating multiple perspectives to improve embedding quality. Overall, matrix factorization-based methods offer a solid theoretical foundation and effective solutions for graph representation learning.

Deep learning has emerged as an alternative approach for generating embedding vectors, with various methods developed based on different architectures. A prominent category within deep learning is Graph Neural Networks (GNNs), which model graph-structured data by aggregating local neighborhood information. Notable GNN models include GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), SGC (Wu et al., 2019), DGI (Veličković et al., 2019), GAT (Veličković et al., 2018), and GATv2 (Brody et al., 2022), all of which utilize convolutional mechanisms to learn contextual node representations. GNNs have demonstrated strong performance across a variety of tasks, particularly when feature information is available.

B.2 EDGE-WISE GRL

Edge-wise graph representation learning involves generating embeddings directly for the edges of a graph. A common strategy in this area treats edges as nodes and derives edge embeddings by concatenating or aggregating the embeddings of the incident nodes (Jiang et al., 2019; Bandyopadhyay et al., 2019; Makarov et al., 2021; Minakawa et al., 2022; Wang et al., 2020b; Yang & Li, 2020). However, such approaches fail to account for the inherent characteristics of edges, often resulting in suboptimal representations or prohibitive computational costs for large graphs.

Alternatively, some methods generate edge embeddings directly from the edge's perspective. For instance, Li et al. (2018) employ random walks to sample sequences of edges, which are subse-

quently used to produce edge embeddings for community detection through clustering algorithms. Edge2Vec (Wang et al., 2020a) uses deep auto-encoders and skip-gram models to generate edge embeddings that capture both local and global structural information, particularly in the context of biomedical knowledge discovery. However, the use of deep auto-encoders leads to high computational costs, particularly for large graphs. AttrE2Vec (Bielak et al., 2022) generates random walks from a node and uses aggregation functions (e.g., averaging, exponential weighting, and GRUs) to combine node and edge features, obtaining their representations. These representations are refined using auto-encoders and self-attention networks with feature reconstruction and graph structural losses for unsupervised edge embedding learning. However, this method requires multiple random walks per node, resulting in high computational costs for large graphs.

More recently, Wang et al. (2023) proposed TER and AER, which efficiently capture high-order edge proximity through matrix factorization and aggregate edge attributes, respectively. EAGLE (Wang et al., 2024) extends edge-wise representation learning to edge-attributed bipartite graphs using dual-view factorized feature propagation. TopoEdge (Cheng et al., 2025) addresses topological imbalance in edge classification through topology-aware reweighting and wedge-based mixup.

B.3 Forest Matrix, Wilson's Algorithm, and Their Applications

The forest matrix has found broad applications in network science. In the domain of centrality measures, Jin et al. (2019) introduced forest closeness centrality, which is expressed through the diagonal entries of the forest matrix. Subsequent research extended this framework by proposing forest node centrality and forest edge centrality, along with approximation algorithms (Sun et al., 2025). Meanwhile, the forest matrix serves as the fundamental matrix in the Friedkin-Johnsen model (Friedkin & Johnsen, 1990) of opinion dynamics (Zhou et al., 2024). Wilson's algorithm (Wilson, 1996) and its extension (Avena & Gaudillière, 2018) provided a method for uniformly sampling spanning trees and forests using loop-erased random walk (Lawler, 1980). Wilson's algorithm is widely used to estimate various graph-related quantities, such as node-based Personalized PageRank (Liao et al., 2022).

C Proofs

C.1 Proof of Lemma 3.1

Proof. For any two edges $e_i, e_j \in E$, the (e_i, e_j)-th entry of $\mathbf{\Gamma}$ is given by

$$\mathbf{\Gamma}_{e_i,e_j} = \sum_{v \in V} \boldsymbol{Q}_{e_i,v} \boldsymbol{R}_{v,e_j}.$$

By the definitions of $\boldsymbol{Q}$ and $\boldsymbol{R}$, a node v contributes to the summation only if it is incident to both e_i and e_j. Thus, $\mathbf{\Gamma}_{e_i,e_j} = \sum_{v \in e_i \cap e_j} 1/(2|E_v|)$, which immediately implies $\mathbf{\Gamma}_{e_i,e_j} = \mathbf{\Gamma}_{e_j,e_i}$ for all $e_i, e_j \in E$. Hence, $\mathbf{\Gamma}$ is symmetric.

Moreover, since

$$\sum_{e_i \in E} \mathbf{\Gamma}_{e_i,e_j} = \sum_{e_i \in E} \sum_{v \in V} \boldsymbol{Q}_{e_i,v} \boldsymbol{R}_{v,e_j} = 1,$$

each column of $\mathbf{\Gamma}$ sums to one. Together with symmetry, this also implies that each row sums to one. Therefore, $\mathbf{\Gamma}$ is doubly stochastic.

Furthermore, by the Neumann series,

$$\mathbf{\Pi} = \alpha(\boldsymbol{I} - (1-\alpha)\mathbf{\Gamma})^{-1} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \mathbf{\Gamma}^k.$$

Since $\mathbf{\Gamma}^k$ is doubly stochastic for any $k \geq 0$, we have $\mathbf{\Pi}\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^\top \mathbf{\Pi} = \mathbf{1}^\top$. Therefore, $\mathbf{\Pi}$ is also doubly stochastic. □

C.2 Proof of Theorem 4.1

Proof. On the original graph G, where we focus on the Edge-PPR, α represents the probability that the random walk halts at the current edge in G at each step. Meanwhile, α^* represents the probability that the random walk halts at the current node in G_S at each step. Since $\alpha^* = 1 - \sqrt{1-\alpha}$, we have $(1-\alpha^*)^2 = 1-\alpha$. Here, $(1-\alpha^*)^2$ is the probability that the walk makes two consecutive transitions without teleporting, which corresponds to one edge-to-edge transition in the original graph.

The transition matrix $\boldsymbol{P}_S$ alternates between the original nodes of G and the edge-nodes introduced in G_S, reflecting the bipartite structure of the subdivision graph. Since $\boldsymbol{QR} = \boldsymbol{\Gamma}$, for any positive integer k, we have

$$\boldsymbol{P}_S^{2k} = \begin{pmatrix} (\boldsymbol{RQ})^k & 0 \\ 0 & (\boldsymbol{QR})^k \end{pmatrix} = \begin{pmatrix} (\boldsymbol{RQ})^k & 0 \\ 0 & \boldsymbol{\Gamma}^k \end{pmatrix}.$$

This implies that after an even number of steps, a random walk starting at an edge-node remains within the edge-node set, governed by $\boldsymbol{\Gamma}^k$. Since $\boldsymbol{P}_S^t$ with odd t transitions between the two sets, only even powers contribute to transitions within the edge-node set. Consequently, the Node-PPR entry corresponding to edge-nodes in G_S is given by

$$\boldsymbol{\Pi}'_{e_1,e_2} = \sum_{k\geq 0}^{\infty} \alpha^*(1-\alpha^*)^{2k}\boldsymbol{\Gamma}^k_{e_1,e_2} = \alpha^* \sum_{k\geq 0}^{\infty}(1-\alpha)^k \boldsymbol{\Gamma}^k_{e_1,e_2}.$$

Thus, we obtain

$$\boldsymbol{\Pi}_{e_1,e_2} = \frac{\alpha}{\alpha^*}\left(\alpha^* \sum_{k\geq 0}^{\infty}(1-\alpha)^k \boldsymbol{\Gamma}^k_{e_1,e_2}\right) = \frac{\alpha}{\alpha^*}\boldsymbol{\Pi}'_{e_1,e_2}.$$

This completes the proof. □

C.3 Proof of Theorem 4.2

Proof. We first define $\boldsymbol{D}_S$ as the degree matrix and $\boldsymbol{L}_S$ as the Laplacian of the subdivision graph G_S. By Theorem 4.1, we have

$$\begin{aligned}
\boldsymbol{\Pi}_{e_1,e_2} &= \frac{\alpha}{\alpha^*}\boldsymbol{\Pi}'_{e_1,e_2} = \frac{\alpha}{\alpha^*}\left(\sum_{t=0}^{\infty}\alpha^*(1-\alpha^*)^t \boldsymbol{P}_S^t\right)_{e_1,e_2} \\
&= \alpha\left(\boldsymbol{I} - (1-\alpha^*)\boldsymbol{P}_S\right)^{-1}_{e_1,e_2} = \alpha\left(\boldsymbol{I} - (1-\alpha^*)\boldsymbol{D}_S^{-1}\boldsymbol{A}_S\right)^{-1}_{e_1,e_2} \\
&= \alpha\left(\boldsymbol{I} - (1-\alpha^*)\boldsymbol{D}_S^{-1}(\boldsymbol{D}_S - \boldsymbol{L}_S)\right)^{-1}_{e_1,e_2} \\
&= \frac{\alpha}{\alpha^*}\left(\boldsymbol{I} + \frac{1-\alpha^*}{\alpha^*}\boldsymbol{D}_S^{-1}\boldsymbol{L}_S\right)^{-1}_{e_1,e_2}.
\end{aligned}$$

For convenience, we define

$$\boldsymbol{M} = \boldsymbol{I} + \frac{1-\alpha^*}{\alpha^*}\boldsymbol{D}_S^{-1}\boldsymbol{L}_S.$$

From the definition of the matrix inverse, we have

$$\boldsymbol{M}^{-1}_{e_1,e_2} = \frac{(-1)^{e_1+e_2}\det(\boldsymbol{M}_{-e_2,-e_1})}{\det(\boldsymbol{M})}. \tag{3}$$

Let $\tilde{G} = (\tilde{V}, \tilde{E}, \tilde{w})$ be a new weighted directed graph derived from the subdivision graph G_S, where $\tilde{w} : \tilde{E} \to \mathbb{R}^+$ is the edge weight function, with w_e being the weight for edge e. Each edge $e = (u, v)$ in G_S is replaced by two directed edges in $\tilde{G}$: one from u to v, assigned a weight of $\frac{1-\alpha^*}{\alpha^* d_u}$, and another from v to u, assigned a weight of $\frac{1-\alpha^*}{\alpha^* d_v}$. Here, d_u and d_v denote the degrees of u and v in G_S, respectively. It follows that $\frac{1-\alpha^*}{\alpha^*}\boldsymbol{D}_S^{-1}\boldsymbol{L}_S$ is the Laplacian matrix of the new graph $\tilde{G}$, and $\boldsymbol{M}^{-1}$ represents the forest matrix of $\tilde{G}$ (Chebotarev & Shamis, 2006).

By the matrix forest theorem (Chaiken, 1982; Sun & Zhang, 2024), the determinants $\det(\boldsymbol{M})$ and $\det(\boldsymbol{M}_{-e_1,-e_2})$ can be related to the weights of spanning rooted forests in $\tilde{G}$. Notably, there is a one-to-one correspondence between the spanning rooted forests of G_S and those of $\tilde{G}$, from which it can be derived that

$$\det(\boldsymbol{M}) = \sum_{F\in\mathcal{F}} \frac{\prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\prod_{i\in V} \frac{\alpha^*}{1-\alpha^*} d_i}$$

and

$$\det\left(\boldsymbol{M}_{-e_2,-e_1}\right) = (-1)^{e_1+e_2} \sum_{F\in\mathcal{F}_{e_1\to e_2}} \frac{\prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\prod_{i\in V} \frac{\alpha^*}{1-\alpha^*} d_i}.$$

Substituting these expressions into Equation 3, we obtain

$$\boldsymbol{M}^{-1}_{e_1,e_2} = \frac{\sum_{F\in\mathcal{F}_{e_1\to e_2}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\sum_{F\in\mathcal{F}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}.$$

Finally, we have

$$\boldsymbol{\Pi}_{e_1,e_2} = \frac{\alpha}{\alpha^*}\boldsymbol{M}^{-1}_{e_1,e_2} = \frac{\alpha}{\alpha^*}\frac{\sum_{F\in\mathcal{F}_{e_1\to e_2}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}{\sum_{F\in\mathcal{F}} \prod_{i\in\rho(F)} \frac{\alpha^*}{1-\alpha^*} d_i}.$$

This completes the proof. □

C.4 Hoeffding's inequality

Lemma C.1 *Let $x_1, x_2, \ldots, x_l$ be l independent random variables satisfying $a \le x_i \le b$ for all i. Let $x = \frac{1}{l}\sum_{i=1}^{l} x_i$. Then, for any $\epsilon > 0$,*

$$\mathbb{P}\{|x - \mathbb{E}[x]| \ge \epsilon\} \le 2\exp\left(-\frac{2l\epsilon^2}{(b-a)^2}\right).$$

C.5 Proof of Theorem 4.3

Proof. According to the definition of $\widetilde{\boldsymbol{\Pi}}^{(F_k)}_{e_1,e_2}$, we have $0 \le \widetilde{\boldsymbol{\Pi}}^{(F_k)}_{e_1,e_2} \le \frac{\alpha}{\alpha^*}$. Applying Hoeffding's inequality to the empirical mean $\widetilde{\boldsymbol{\Pi}}_{e_1,e_2} = \frac{1}{l}\sum_{k=1}^{l} \widetilde{\boldsymbol{\Pi}}^{(F_k)}_{e_1,e_2}$, we obtain

$$\mathbb{P}\left(\left|\widetilde{\boldsymbol{\Pi}}_{e_1,e_2} - \boldsymbol{\Pi}_{e_1,e_2}\right| \ge \epsilon\right) \le 2\exp\left(-\frac{2l\epsilon^2(\alpha^*)^2}{\alpha^2}\right).$$

To ensure the right-hand side is at most δ, we solve for l:

$$2\exp\left(-\frac{2l\epsilon^2(\alpha^*)^2}{\alpha^2}\right) \le \delta.$$

Taking logarithms and rearranging gives

$$l \ge \frac{\alpha^2}{2\epsilon^2(\alpha^*)^2}\ln\frac{2}{\delta}.$$

Since l must be an integer, we take the ceiling, completing the proof. □

D Algorithm Detail

Description. Algorithm 1 details our complete approximation algorithm. We first construct the subdivision graph G_s from the input graph G (Line 2). We then sample rooted spanning forests according to the target distribution (Lines 4–14) and approximate the Edge-PPR matrix using the proposed estimator (Lines 15–18).

Algorithm 1: ApproxEdgePPR(G, ϵ, δ)

Input : Graph G, error parameters ϵ, δ.

Output : Edge-PPR matrix $\widetilde{\mathbf{\Pi}}$.

Initialize: $\widetilde{\mathbf{\Pi}} \leftarrow 0$, $l \leftarrow \left\lceil \frac{\alpha^2}{2\epsilon^2(\alpha^*)^2} \ln \frac{2}{\delta} \right\rceil$

Construct the subdivision graph G_S with $n+m$ nodes and $2m$ edges according to Equation 2

for $k = 1, 2, \cdots, l$ **do**

  $\mathrm{InForest}_u \leftarrow$ false, $\mathrm{next}_u \leftarrow 0$, $\mathrm{root}_u \leftarrow 0$ for $u \in V \cup V_S^e$

  foreach $u \in V_s$ **do**

    $i \leftarrow u$

    while $\mathrm{InForest}_u =$ false **do**

      if $\textsc{Rand}(0,1) < \alpha^*$ **then**

        $\mathrm{InForest}_u \leftarrow$ true, $\mathrm{root}_u \leftarrow u$

      else

        $\mathrm{next}_u \leftarrow$ a randomly selected neighbor of u

        $u \leftarrow \mathrm{next}_u$

    $v \leftarrow \mathrm{root}_u$, $u \leftarrow i$

    while $\mathrm{InForest}_u =$ false **do**

      $\mathrm{InForest}_u =$ true, $\mathrm{root}_u \leftarrow v$, $u \leftarrow \mathrm{next}_u$

  foreach $u \in V_S^e$ **do**

    $v \leftarrow \mathrm{root}_u$

    if $v \in V_S^e$ **then**

      $\widetilde{\mathbf{\Pi}}_{u,v} \leftarrow \widetilde{\mathbf{\Pi}}_{u,v} + \frac{\alpha}{\alpha^* l}$

return $\widetilde{\mathbf{\Pi}}$

Time Complexity. Next, we analyze the time complexity of Algorithm 1. Constructing the subdivision graph requires $O(m)$ time. We then sample rooted spanning forests from the weighted directed graph $\tilde{G}$, which contains $n+m$ nodes. Following an analysis similar to that in (Sun & Zhang, 2023), each forest can be generated in

$$O(T_{\text{sample}}) = O(w_{\max}(m+n)) = O\left(\frac{1-\alpha^*}{\alpha^* d_{\min}}(m+n)\right)$$

time. Therefore, with l independent samples, the overall time complexity of Algorithm 1 is $O(m + T_{\text{sample}} l)$.

E Supplement Details for Experiments

E.1 Dataset

In this subsection, we provide the details of the datasets used in our experiments.

Table 3: Datasets used in experiments.

Dataset	Node	Edge	Feature	Label
MOOC	7,144	437,845	4	2
Yelpchi	38,264	67,395	25	2
MAG	40,000	141,725	256	10
Reddit	67,180	901,056	86	2
MIND	242,940	2,065,006	256	10
DBLP	2,907,929	15,467,003	256	10

MOOC (Kumar et al., 2019) is collected from an online course platform, where edges correspond to user actions on course contents and edge labels indicate whether a user drops out after a specific action. YelpChi (Rayana & Akoglu, 2015) is a review network in which users interact with restaurants and hotels through reviews; edge labels denote whether a review is genuine or suspicious. MAG (Sinha et al., 2015) and DBLP (Tang et al., 2007) are academic networks, where nodes represent scholars and edges denote coauthorship relations induced by papers, with edge labels corresponding to the research fields of the associated papers. Reddit (Kumar et al., 2018) is a hyperlink network among online communities, where edges represent posts linking two communities and the edge labels characterize the sentiment of the source community toward the target community. MIND (Wu et al., 2020) is constructed from user behavior logs on Microsoft News, where edges capture user–news interactions and edge labels correspond to news categories. These datasets exhibit substantial variation in graph size, ranging from thousands to hundreds of thousands of nodes and from tens of thousands to millions of edges, allowing us to evaluate the effectiveness and scalability of the compared methods under diverse settings. Each dataset is randomly split into 80% training, 10% cross-validation, and 10% test set.

E.2 Experimental Environment

All experiments were conducted on a Linux server with a 36-core 2.10GHz Intel(R) Xeon(R) Platinum 8352V CPU, 256GB of RAM, and a NVIDIA RTX 4090 GPU with 24GB memory. Each experiment is repeated five times.

E.3 Baselines

We compare our model with thirteen representative baselines, grouped into two categories:

- Node-Wise GRL Methods: GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), DGI (Veličković et al., 2019), GATv2 (Brody et al., 2022), PSNE (Lin et al., 2024), node2vec (Grover & Leskovec, 2016), and TopoEdge (Cheng et al., 2025). Following EAGLE (Wang et al., 2024), for methods requiring node features, we initialize each node by averaging the attributes of its incident edges. After learning the endpoint representations, we construct the representation of edge $e = (u, v)$ as $\boldsymbol{Z}_e = [\boldsymbol{H}_u \| \boldsymbol{H}_v \| \boldsymbol{X}_e]$.
- Edge-Wise GRL Methods: Edge2Vec (Wang et al., 2020a), AttrE2Vec (Bielak et al., 2022), EdgeFormer (Jin et al., 2023), TER and AER (Wang et al., 2023). These methods directly generate edge representations from graph structure and edge attributes. For TER+AER, we evaluate both the geometric and Poisson weighting schemes.

For GCN, GraphSAGE, GAT, and GATv2, we use the standard implementations provided by the DGL library. Each model consists of two GNN layers followed by a linear layer, with ReLU activations and a dropout rate of 0.5. We train the models for up to 300 epochs using the Adam optimizer with a learning rate of 0.001. For the remaining baselines, we use the official source code released by their authors and follow the recommended parameter settings. The resulting edge representations are subsequently fed into an MLP classifier.

E.4 Standard Deviation

In this subsection, we supplement the main experimental results by reporting the standard deviations of the edge classification performance over five independent runs. These results provide additional evidence on the stability and robustness of the evaluated methods across different datasets.

E.5 Ablation Study

In this subsection, we conduct ablation studies to evaluate the contribution of each component in EPO. Specifically, $\text{EPO-F}_{\text{w/o log}}$ denotes the variant of EPO-F without the entry-wise logarithmic transformation. For the efficiency analysis, $\text{EPO}_{\text{w/o sample}}$ replaces the subdivision graph expansion and forest sampling with direct matrix inversion, whereas $\text{EPO}_{\text{w/o parallel}}$ performs forest sampling sequentially without parallelization. These variants allow us to examine the effectiveness of the nonlinear transformation and quantify the efficiency gains brought by forest sampling and its parallel implementation.

Table 4: Standard deviations of edge classification performance on six datasets.

Method	MOOC		Yelpchi		MAG		Reddit		MIND		DBLP	
	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
GCN	0.0061	0.0084	0.0048	0.0056	0.0032	0.0071	0.0054	0.0068	0.0043	0.0051	0.0038	0.0062
GraphSAGE	0.0058	0.0072	0.0064	0.0049	0.0041	0.0065	0.0063	0.0057	0.0052	0.0048	0.0040	0.0054
DGI	0.0072	0.0065	0.0055	0.0061	0.0037	0.0080	0.0042	0.0074	0.0058	0.0063	0.0035	0.0070
GAT	0.0066	0.0078	0.0053	0.0059	0.0040	0.0072	0.0057	0.0060	0.0045	0.0056	0.0039	0.0061
GATv2	0.0062	0.0071	0.0051	0.0053	0.0038	0.0067	0.0055	0.0058	0.0042	0.0052	0.0036	0.0057
PSNE	0.0057	0.0069	0.0062	0.0070	0.0048	0.0046	0.0059	0.0064	0.0049	0.0058	0.0044	0.0066
node2vec	0.0060	0.0076	0.0057	0.0068	0.0035	0.0054	0.0061	0.0060	0.0047	0.0056	0.0032	0.0063
Edge2vec	-	-	-	-	0.0031	0.0069	-	-	-	-	-	-
AttrE2Vec	0.0054	0.0063	0.0047	0.0052	0.0043	0.0035	0.0051	0.0056	0.0045	0.0049	0.0040	0.0058
Edgeformer	0.0058	0.0066	0.0045	0.0057	0.0046	0.0039	0.0053	0.0059	0.0048	0.0054	0.0042	0.0064
TER+AER(G)	0.0048	0.0042	0.0041	0.0048	0.0030	0.0015	0.0043	0.0028	0.0032	0.0019	0.0044	0.0032
TER+AER(P)	0.0046	0.0040	0.0040	0.0045	0.0035	0.0013	0.0041	0.0030	0.0030	0.0018	0.0042	0.0030
TopoEdge(GCN)	0.0050	0.0046	0.0048	0.0050	0.0038	0.0026	0.0039	0.0025	0.0034	0.0027	0.0045	0.0031
EPO-$F_{\text{w/o log}}$	0.0044	0.0041	0.0042	0.0043	0.0038	0.0028	0.0043	0.0051	0.0031	0.0027	0.0040	0.0043
EPO-F	0.0038	0.0035	0.0037	0.0036	0.0039	0.0005	0.0035	0.0024	0.0047	0.0013	0.0037	0.0038
EPO-P	0.0059	0.0050	0.0047	0.0046	0.0049	0.0017	0.0034	0.0023	0.0066	0.0056	0.0041	0.0047

Table 5: Running time (seconds) and peak memory usage (GB) of EPO and its variants on three large datasets.

Variant	Reddit		MIND		DBLP	
	Time	Mem.	Time	Mem.	Time	Mem.
$\text{EPO}_{\text{w/o sample}}$	67,034	72.59	-	-	-	-
$\text{EPO}_{\text{w/o parallel}}$	438	4.02	964	18.06	7092	66.52
EPO	205	6.92	518	14.23	4025	79.13

According to Table 1, removing the logarithmic transformation consistently reduces AUC by 0.007-0.025 across the six datasets. This degradation demonstrates the effectiveness of the nonlinear transformation. Specifically, the logarithmic operation mitigates the dominance of large entries in the proximity matrix while enhancing the contribution of relatively weak proximity signals, thereby producing more informative and discriminative edge representations.

Since EPO is designed to accelerate the construction of the proximity matrix $\mathbf{\Pi}$, we further report the running time and peak memory usage during this stage. As shown in Table 5, EPO achieves a speedup of more than two orders of magnitude over direct matrix inversion on large graphs while requiring substantially less memory. This improvement primarily results from the forest sampling algorithm, whose time complexity is linear in the graph size for a fixed number of samples and which avoids explicitly constructing and inverting a large dense matrix. Moreover, since individual forest samples are mutually independent, they can be generated in parallel, further accelerating proximity estimation. These results confirm that both forest sampling and its parallel implementation are essential to the scalability of EPO.