Structural Information Enhanced Graph Representation for Link Prediction

Lei Shi[*1], **Bin Hu**[*1], **Deng Zhao**[*1], **Jianshan He**[2], **Zhiqiang Zhang**[1], **Jun Zhou**[1]
[1]Machine Intelligence Department, Ant Group,
[2]Consumer Finance Technology Department, Ant Group
{sl316606, axel.hb, zhaodeng.zd, yebai.hjs}@antgroup.com, {lingyao.zzq, jun.zhoujun}@antfin.com

Abstract

Link prediction is a fundamental task of graph machine learning, and Graph Neural Network (GNN) based methods have become the mainstream approach due to their good performance. However, the typical practice learns node representations through neighborhood aggregation, lacking awareness of the structural relationships between target nodes. Recently, some methods have attempted to address this issue by node labeling tricks. However, they still rely on the node-centric neighborhood message passing of GNNs, which we believe involves two limitations in terms of information perception and transmission for link prediction. First, it cannot perceive long-range structural information due to the restricted receptive fields. Second, there may be information loss of node-centric model on link-centric task. In addition, we empirically find that the neighbor node features could introduce noise for link prediction. To address these issues, we propose a structural information enhanced link prediction framework, which involves removing the neighbor node features while fitting neighborhood graph structures more focused through GNN. Furthermore, we introduce Binary Structural Transformer (BST) to encode the structural relationships between target nodes, complementing the deficiency of GNN. Our approach achieves remarkable results on multiple popular benchmarks, including ranking first on ogbl-ppa, ogbl-citation2 and Pubmed.

Introduction

Link prediction is the task of predicting missing or potential edges according to existing nodes and edges in a graph, the typical applications of which include knowledge completion, recommendation, and more. For example, it can complete missing colleague relationships, and city of birth for a person, or can predict users' interest in certain products in a recommendation system. There are various methods for link prediction, among which Graph Neural Network (GNN) based methods have become the mainstream ones due to their good performance and efficiency.

GNNs (Kipf and Welling 2016a; Hamilton, Ying, and Leskovec 2017; Velikovi et al. 2017) have achieved great results in a lot of fields. They are centered on a certain node, and aggregate features of neighbor nodes and edges to the central node through the Message Passing (MP) (Gilmer et al. 2017), which are good at learning node representations. When applied to link prediction, the typical practice is to use the Autoencoder framework on Graph (GAE (Kipf and Welling 2016b)). This involves encoding individual node representations with GNNs, aggregating representations of the target nodes, and decoding them into link probability.

However, such an approach may miss some important features for link prediction. A simple example is illustrated in Fig. 1-(a), where nodes h_1 and h_2 are isomorphic. Without node features, GNN learns the same node representation, resulting in the same link representation of pair (h_1, t) and pair (h_2, t). In fact, the results should not be the same. Node h_2 and t share a common neighbor, indicating a higher probability of a link. We believe that GAEs fail fundamentally because they focus on capturing neighbor information centered on nodes during the encoding stage, without perceiving the structural relationships of target node pairs.

Recently, some works have attempted to address this issue by node labeling tricks. For example, ID-GNN (You et al. 2021) colors target nodes to distinguish them from neighbor nodes, and SEAL (Zhang and Chen 2018; Zhang et al. 2021) calculates DRNL (Double-Radius Node Labeling) of all nodes in an enclosing subgraph. However, these approaches only help perceive path depth, while lacking perception of path 'width', such as node degree or the number of paths. Moreover, they are still limited by the MP paradigm of GNNs, leading to shortcomings in terms of information perception and transmission for link prediction. On one hand, the way of neighbor node labeling converging to target nodes and then aggregating into relationship representations is implicit and convoluted, potentially risking information loss. On the other hand, the receptive field of GNNs may not be sufficient enough for link prediction.

GNNs are node-centric models, whose receptive field is the neighborhood of a central node. However, some target nodes in link prediction may be far apart, and GNNs struggle to capture the long-range information between them. The right part of Fig. 1 shows a simple example, where neither ID-GNN nor SEAL with the most commonly used 3-layer GNNs can distinguish between the two link structures shown in (b) and (c). When computing the representation of the head nodes h using GNNs, the computational graphs (including nodes' DRNL) in both structures are the same, as

[*]These authors contributed equally.

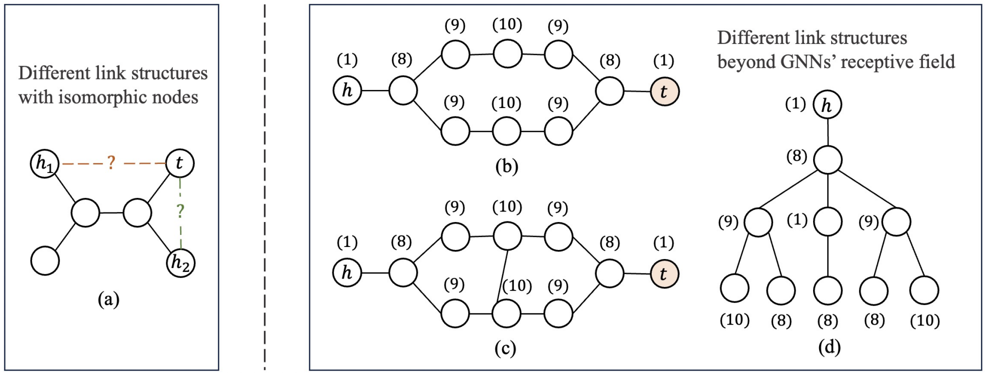

Figure 1: Link prediction cases where GNNs fail. In (a), node h_1 and h_2 are isomorphic, but link (h_1, t) and (h_2, t) are not. (b) and (c) show two different structures, and (d) shows the common computational graph of the head nodes h. The numbers above the nodes are encoded with DRNL described by equation (2), as in SEAL, and the tail nodes t are colored as in ID-GNN.

shown in (d), and the color of tail nodes t are not aggregated. The same applies when computing the representation of the tail nodes t. Therefore, the two link structures cannot be distinguished by 3-layer GNNs, even with labeling tricks. To enhance the long-range information capturing ability of GNNs, the number of GNN layers needs to be increased to expand the receptive field, which, however, may lead to problems such as oversmoothing and overfitting (Li, Han, and Wu 2018; Rong et al. 2020).

In this paper, we propose an additional attention module to compensate for the lack of GNN in capturing pairwise structures and directly learn relationship representations between target nodes. Self-attention is a powerful mechanism to learn pairwise relationships and shines brightly in Transformers (Vaswani et al. 2017), but it lacks the ability to perceive structures of elements. In addition, the computation is performed pairwise between all elements, resulting in a quadratic computational complexity and a large number of parameters leading to the problem of data hungry (Hassani et al. 2021). In order to capture the pairwise structural features of target nodes, we introduce the topological encoding to encode pairwise heuristics such as shortest path distance (SPD), Adamic-Adar, etc., and fuse them into attention. Besides, our encoding target is not the global information, but rather the pairwise structures of target nodes missed by GNNs. Therefore, we restrict attention computation to the target nodes, avoiding expensive computations and the risk of introducing noise from other nodes. As a result, the proposed attention module focuses on the target nodes and effectively encodes structural features, which we refer to as Binary Structural Attention (BSA).

On the other hand, GNNs aggregate rich information from the neighborhood, mainly including graph structure information and the features of neighbors. However, a few works have raised doubts about neighbors in graph learning, which prompt us to question the necessity of neighbor features in link prediction. Thus, we attempted to remove neighbor node features. Taking SEAL (with NGNN on ogbl-ppa and ogbl-citation2) as an example in Table 1, we observed the improvement without neighbor node features (experimental

Model	ogbl-ppa	ogbl-citation2	ogbl-vessel
	Hits@100 (%)	MRR (%)	AUC (%)
w NF	59.71±2.45	88.91±0.22	80.50±0.21
w/o NF	**60.05±4.85**	**89.18±0.11**	**84.70±0.70**

Table 1: Resuts of SEAL (with/without neighbor features) on OGB link prediction datasets.

settings are the same as the proposed method, described in the Experiments section and Appendix). This phenomenon contradicts intuition yet seems to be explainable: in link prediction, neighbor node features may be task-irrelevant and prone to noise. For example, in predicting a spouse relationship, the key information could be that two individuals have a common child, but the child's interests, phone number, and other features may not be important.

Based on the above, we propose a Structural Information Enhanced Graph representation framework for link prediction (SIEG). On one hand, we use a GNN to collect the neighborhood information of target nodes, removing neighbor node features to make the model more focused on structural features. On the other hand, the BSA is introduced to encode the pairwise features of the target nodes, especially the long-range structural features, to complement the deficiency of GNN. Our method shows remarkable performance on six popular benchmarks, including ranking first on ogbl-ppa, ogbl-citation2 and Pubmed.

Related Works

Link Prediction. There are four main paradigms of existing works on link prediction: walk-based methods, shallow Knowledge Graph Embedding methods (shallow-KGEs), path-based methods, and GNN-based methods. Walk-based methods take the random walk sequence from a node as the representation of it, whose representatives are DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2Vec (Grover and Leskovec 2016). Shallow-KGEs focus more on the triple relationship of head node, edge, and tail node, including TransE (Bordes et al. 2013), DistMult (Yang et al. 2014), and ubsequent works (Trouillon et al. 2016; Sun et al. 2019; Zhang et al. 2020b) improving score functions. However, these methods lack inductive properties and parameter sharing mechanisms, making them difficult to apply to large and scalable graphs. Path-based methods focus on connectivity between target nodes, including heuristic methods such as SPD and Katz (Katz 1953), rule-based approaches like NeuralLP (Yang, Yang, and Cohen 2017) and DRUM (Sadeghian et al. 2019), path representation methods such as Path-RNN (Neelakantan, Roth, and Mccallum 2015) and others (Wang, Ren, and Leskovec 2020; Zhu et al. 2021; Kong, Chen, and Zhang 2023). Among them, the heuristic methods are simple, interpretable, and inductive, making them suitable as structural features in our BSA.

Due to good performance and efficiency, GNN-based methods have become the mainstream approach for link prediction. Prevalent methods (Kipf and Welling 2016b; Vashishth et al. 2019; Ahn and Kim 2021) adopt an Autoencoder framework, where GNN serves as the encoder for node representations and edges are decoded as a function over node pairs. Another stream of frameworks, including SEAL (Zhang and Chen 2018), IGMC (Zhang and Chen 2020), and GraIL (Teru, Denis, and Hamilton 2020), encodes the enclosing subgraph around each node pair for link prediction. However, a good node-level representation does not necessarily lead to a good link-level representation (Zhang et al. 2021). Methods (You et al. 2021; Zhang et al. 2021) bridges the gap between node-level and edge-level GNN expressiveness with labeling tricks. However, they are still constrained by the MP paradigm of GNNs, unable to capture comprehensive long-range structures directly.

Graph Transformer. Transformer has achieved great success in Natural Language Processing (Vaswani et al. 2017) and Computer Vision (Dosovitskiy et al. 2021), and its application to graph has also emerged. The Transformer faces two main challenges when applied to graph data: position/structure encoding and computational complexity. Firstly, the core module of Transformer, Self-attention, lacks the ability to perceive the structural information of targets. To address this, some methods have introduced additional encoding schemes, such as Laplacian encoding (Dwivedi and Bresson 2020), distance correlation encoding (Ying et al. 2021; Mialon et al. 2021), and structure-aware encoding (Chen, O'Bray, and Borgwardt 2022; Park et al. 2022), while some works propose specialized networks to learn positional encoding (Kreuzer et al. 2021). In order to capture the pairwise structural features of target nodes in BSA, we extend the topological encoding (Park et al. 2022) to encode pairwise heuristics and fuse them into attention.

In addition, Self-attention calculates the similarity between all nodes pairwise, resulting in a quadratic computational complexity. To alleviate this problem, methods (Dwivedi and Bresson 2020; Zhang et al. 2020a) limit the number of nodes involved in computation through sampling. However, the sampling strategies are still node-centric rather than link-centric. For BSA, the encoding target is specifically the pairwise structures of target nodes in link prediction. Therefore, we restrict attention computation to the target nodes, avoiding the quadratic computations.

Doubt Neighbors. A few works have raised doubts about neighbors in graph learning. Feng et al. (Feng et al. 2021) suggests that some of the neighbors in GNN aggregation may be unreliable and misleading. While GraphENS (Park, Song, and Yang 2022) have found that models tend to overfit to neighbor sets of minor class in class-imbalanced node classification. However, there is no research questioning the necessity of neighbor node features in link prediction, to the best of our knowledge.

Methodology

Problem Definition. Assuming a graph is denoted as $G = (V, E, A)$, where V, E are the set of nodes and the set of edges, respectively, and A is the adjacency matrix of the graph. Node feature matrix is denoted as $X \in \mathbb{R}^{|V|\times D}$, where D is the dimension of node features. The goal of link prediction is to predict whether there exists an edge $e_{i,j}$ between two given nodes (v_i, v_j), according to the graph.

Overall Architecture. As it is shown in Fig. 2, the proposed link prediction framework consists of four main parts: subgraph sampling, structural information enhanced GNN for neighborhood structure encoding, Binary Structural Transformer (BST) for pairwise structure encoding, and link decoding.

Subgraph Sampling

Given a target node pair (v_i, v_j), we sample a k-hop enclosing subgraph (Zhang and Chen 2017) around it. This subgraph is composed of the union of the two k-hop subgraphs around each target node, with edges between the two subgraphs also being incorporated. For instance, in Fig. 2, an additional edge $e_{1,5}$ beyond 1-hop is included in the 1-hop enclosing subgraph of target node pair (h, t). Enclosing subgraphs provide better connectivity and more informative input for link prediction than normal subgraphs. It has been proved in WLNM (Zhang and Chen 2017) that not only first and second-order heuristic features can be directly calculated in enclosing subgraph, but even high-order heuristics are also feasible with a small number of hops.

Neighborhood Structure Encoding

To encode the neighborhood structures and learn the representations of each target node, any GNN can be used. GNNs aggregate neighborhood information through MP centered around nodes, and the typical form (Kipf and Welling 2016a) is shown as

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)}). \tag{1}$$

Where $\tilde{A} := \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ denotes the normalized adjacency matrix. $\hat{A} := A + I$, where I is identity matrix. $\hat{D_{ii}} := \sum_j \hat{A_{ij}}$. $H^{(l)}$ is the node embedding matrix and $W^{(l)}$ is the linear parameter matrix in the l-th layer. σ represents the ReLU activation function. Typically, similarity representations can be obtained by aggregating the respective embeddings of the target nodes. In this work, we adopt

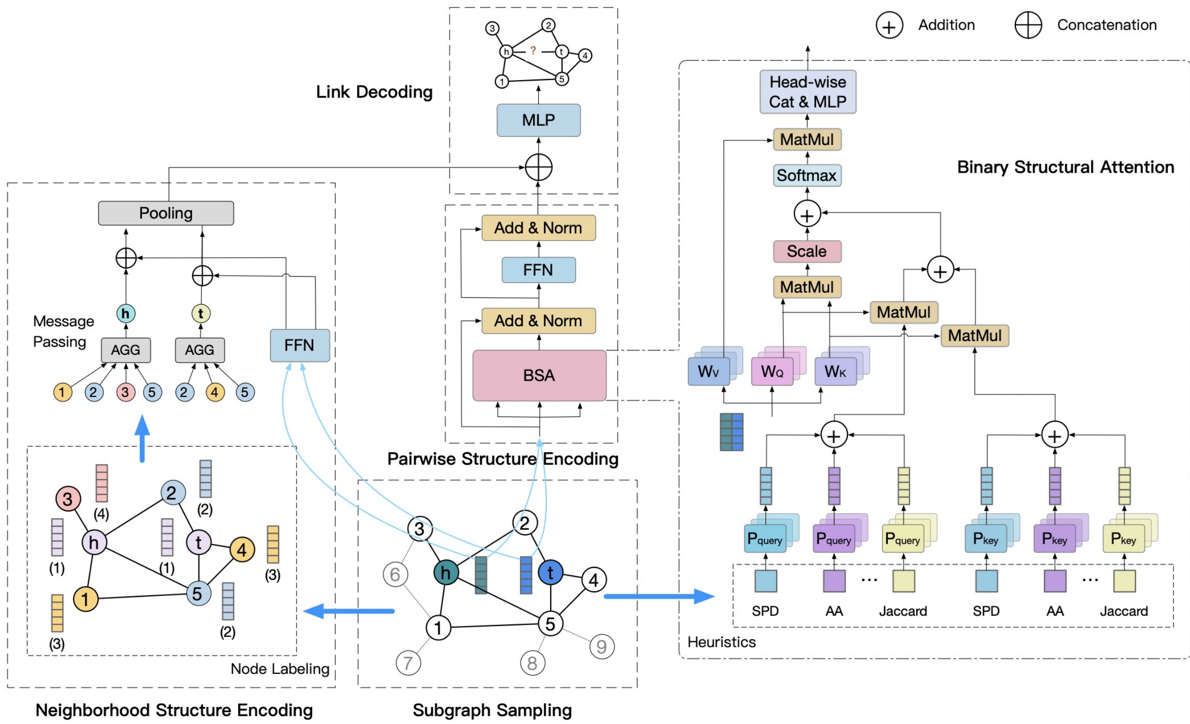

Figure 2: Overall model architecture of the proposed SIEG.

the SortPooling (Zhang et al. 2018) to form the similarity representation.

To enhance the fitting to structural information, we remove the neighbor node features to avoid potential noise, making the model focus more on the neighborhood structures of target nodes. Specifically, the information inputted to the GNN does not include any node features, but instead adopts the structural features encoded by DRNL (Zhang and Chen 2018). DRNL encodes the distances between each node within the subgraph and the target nodes, the specific form of which is shown as

$$f_l(i) = \begin{cases} 1 + \min(d_x, d_y) + (d/2)[(d/2) \\ \quad +(d\%2) - 1], \text{ if } d_x \cdot d_y \neq 0, \\ \qquad 1, \qquad \text{otherwise.} \end{cases} \tag{2}$$

Here, d_x and d_y are the distances from a node to the two taget nodes, respectively. $d := d_x + d_y$. $(d/2)$ and $(d\%2)$ correspond to the integer quotient and remainder of d divided by 2, respectively. On the other hand, the node featues of target nodes is encoded with the additional Feed-forward Network (FFN), and then concatenated with the structural representations encoded by GNN before pooling.

Pairwise Structure Encoding

As we analyzed in the Introduction section, even with labeling tricks, the GNN framework has two limitations for link prediction: the absence of long-range structural information due to restricted receptive fields and the conflict between node-centric models and link-centric tasks. To address these shortcomings, we propose the BST to capture pairwise structures and feature similarities between target nodes exclusively, representing relations directly and supplementing the limitations of GNNs.

The calculation of BSA, which forms the core of BST, is illustrated on the right side of Fig. 2. The process involves two inputs: the features of the target nodes and the structural features of the target node pair. The latter are calculated using heuristic methods such as shortest path distance (SPD), number of shortest paths (SPN), common neighbor (CN), Jaccard index, Adamic-Adar index (AA), and so on. For instance, the Jaccard and AA are computed as

$$S_{\text{Jaccard}}[u, v] = \frac{|\mathrm{N}(u) \cap \mathrm{N}(v)|}{|\mathrm{N}(u) \cup \mathrm{N}(v)|}, \tag{3}$$

$$S_{\text{AA}}[u, v] = \sum_{z \in \mathrm{N}(u) \cap \mathrm{N}(v)} \frac{1}{\log(d_z)}, \tag{4}$$

where u and v denote the target nodes. $\mathrm{N}(\cdot)$ denotes the set of neighbor nodes, and $|\cdot|$ denotes the number of elements in the set. $\cap$ and $\cup$ denote intersection and union of sets, respectively. d_z denotes the degree of the common neighbor z. In addition to describing the connectivity path of the node pair, these two heuristics also introduce number of edges of target nodes and degree of the common neighbor, respectively, for normalization. Furthermore, we adopt structural lookup embedding in conjunction with node feature embeddings to obtain the structural encoding, thereby enhancing the representation of structure information:

$$s_{ij} = q_i \cdot \mathcal{P}^{query}_{\psi(i,j)} + k_j \cdot \mathcal{P}^{key}_{\psi(i,j)}, \tag{5}$$

$$\text{where} \quad q_i = h_i W^Q, \quad k_j = h_j W^K. \tag{6}$$

$$\mathcal{P}^{query}_{\psi(i,j)} = \mathcal{P}^{query}_{\mathrm{SPD}(i,j)} + \mathcal{P}^{query}_{\mathrm{Jaccard}(i,j)} + \mathcal{P}^{query}_{\mathrm{AA}(i,j)} + \cdots \tag{7}$$

$$\mathcal{P}^{key}_{\psi(i,j)} = \mathcal{P}^{key}_{\mathrm{SPD}(i,j)} + \mathcal{P}^{key}_{\mathrm{Jaccard}(i,j)} + \mathcal{P}^{key}_{\mathrm{AA}(i,j)} + \cdots \tag{8}$$

Here, $\mathcal{P}^{query}_{\mathrm{SPD}(i,j)}$ and $\mathcal{P}^{key}_{\mathrm{SPD}(i,j)}$ denote two structural vectors obtained by using SPD as an index to lookup two separate parameterized tables, $\mathcal{P}^{query}$ and $\mathcal{P}^{key}$. Structural vectors obtained from other heuristics follow the same process. Next, we aggregate all the query structural vectors as $\mathcal{P}^{query}_{\psi(i,j)}$, and all the key structural vectors as $\mathcal{P}^{key}_{\psi(i,j)}$. Then, we multiply the two structural vectors with q_i and k_j respectively, and add them to obtain the structural encoding, denoted as b_{ij}. Here, q_i and k_j are linear transformations of the target node features, with W^Q and W^K representing the transformation matrices. This effectively fuses the rich heuristic structures with node features, and characterizes the structural correlation of target nodes. Finally, the structural encoding b_{ij} is incorporated into attention coefficient a_{ij}:

$$a_{ij} = \frac{q_i \cdot k_j^\top + b_{ij}}{\sqrt{d}} \tag{9}$$

Where d is the dimension of q_i. To obtain the result of a single-head BSA, we multiply the softmax-normalized attention coefficient with the Value embedding, and we concatenate multiple heads to form the multi-head BSA:

$$\mathrm{Attn}_k(h_i, h_j) = \mathrm{softmax}(a_{ij}) \cdot v_j \tag{10}$$

$$\mathrm{Attn}(h_i, h_j) = \|_{k=1}^{n} \mathrm{Attn}_k(h_i, h_j) W^O \tag{11}$$

Where the Value embedding, $v_j := h_j W^V$, is the linear transformation of h_j through W^V. The symbol $\|$ represents the concatenate operation, and W^O is a linear transformation matrix for multi-head fusion.

Lastly, the BST block comprises of the BSA, Layer Normalization (LN), Residual Connection (He et al. 2015), and FFN components, as depicted in Fig. 2. Similar to the regular Transformer, the pairwise structure encoding module, BST, is obtained by stacking multiple BST blocks.

The proposed BST offers several advantages over the regular Transformer. Firstly, the potential noise and overfitting risk introduced by nodes away from the targets are avoided. Secondly, more task-relevant structure information is collected. Finally, computational complexity is greatly reduced.

Link Decoding

Generally, neighborhood embedding and pairwise embedding can be fused with any specifically designed technique to achieve a good link prediction result. As the focus of our study is not on fusion techniques, we simply concatenate the embeddings to form the relation representation and decode it into a link score with Multi-layer Perceptron (MLP). This process is formulated as

$$score_{ij} = \mathrm{MLP}(h^{\mathrm{GNN}}_{ij} \parallel h^{\mathrm{BST}}_{ij}). \tag{12}$$

Where h^{GNN}_{ij} is the similarity embedding of target nodes encoded by GNN, while h^{BST}_{ij} is the relation embedding encoded by BST, and $\|$ denotes the concatenation operation.

Module	SPD	Attention
FCA	$\mathcal{O}(N(N+E)\log N)$	$\mathcal{O}(N^2D + ND^2)$
BSA	$\mathcal{O}((N+E)\log N)$	$\mathcal{O}(D^2)$

Table 2: Complexity of SPD and Attention in FCA and BSA.

Computational Complexity

In comparison to FCA, BSA achieves a significant reduction in computational complexity. FCA entails two main computational costs: SPD searching and Attention calculation, with complexities of $\mathcal{O}(N(N+E)\log N)$ (Dijkstra 1959) and $\mathcal{O}(N^2D + ND^2)$, respectively. Here, N and E denote the number of nodes and edges in the subgraph, and D denotes the dimension of node features. In contrast, the computational complexity of BSA is much lower than that of FCA, as indicated in Table 2, with only $\mathcal{O}((N+E)\log N)$ for SPD and $\mathcal{O}(D^2)$ for Attention.

The overall computational cost of the Transformer is proportional to the sum of the computational cost of SPD and Attention. Consequently, the acceleration ratio from fully-connected Transformer (FCT) to BST ranges from N to N^2, which can be approximated to N or N^2 when $N \ll D$ or $N \gg D$ respectively. BST effectively addresses the computationally expensive nature of the Transformer in link prediction, enabling SIEG to be applied on larger-scale datasets.

Expressive Power

The Weisfeiler-Lehman test (Weisfeiler and Leman 1968) is a widely used graph isomorphism test, and it has been proven (Xu et al. 2018) that the expressive power of Message Passing GNNs (MPGNNs), such as GCN, GraphSAGE, and GAT, is upper-bounded by the 1-dimensional Weisfeiler-Lehman (1-WL) algorithm. Specifically, the 1-WL algorithm cannot distinguish nodes with the same subtrees but different substructures. Consequently, the GAEs based on MPGNNs cannot distinguish link structures involving these target nodes. In contrast, the proposed SIEG is able to effectively distinguish such structures by providing distinct representations, even beyond receptive fields of GNNs. This implies that SIEG surpasses the 1-WL test and overcomes the message passing limitation of GNNs. The detailed proof is presented in the Appendix.

Experiments

Experimental Settings

Datasets and Tasks. In order to prove the effectiveness of the proposed method, we conduct link prediction tasks on six popular datasets: three Open Graph Benchmark (OGB) datasets (Hu et al. 2020) including ogbl-ppa, ogbl-citation2, ogbl-vessel, and three classic attributed graph datasets (categorized as Classic in this paper) including Pubmed (Namata et al. 2012), Cora (Mccallum et al. 2000), Citeseer (Giles, Bollacker, and Lawrence 1998). The details of the datasets, statistics, specific prediction tasks, and evaluation metrics are presented in the Appendix.

Method	ogbl-ppa	ogbl-citation2	ogbl-vessel
	Hits@100(%)	MRR(%)	AUC(%)
CN	27.65±0.00	51.47±0.00	48.49±0.00
AA	32.45±0.00	51.89±0.00	48.49±0.00
RA	49.33±0.00	-	-
Node2vec	22.26±0.83	61.41±0.11	49.54±0.57
MF	32.29±0.94	51.86±4.43	49.97±0.05
MLP	50.62±0.35	-	47.94±1.33
GCN	18.67±1.32	84.74±0.21	43.49±9.61
GraphSAGE	16.55±2.40	82.60±0.36	49.89±6.78
SEAL	48.80±3.16	87.67±0.32	80.50±0.21
NGNN	59.71±2.45	88.91±0.22	-
SUREL	54.32±0.44	88.83±0.18	84.96±0.68
Neural CN	61.19±0.85	-	-
S3GRL	42.42±0.18	88.14±0.08	80.56±0.06
AGDN	41.23±1.59	85.49±0.29	-
SIEG w/ NF	61.88±4.33	89.57±0.10	96.77±0.01
SIEG w/o NF	**63.22±1.74**	**89.87±0.18**	**96.87±0.01**

Table 3: Test performance on three OGB benchmarks.

Method	Pubmed	Cora	Citeseer
AA	64.26±0.40	71.48±0.69	65.86±0.80
CN	64.26±0.40	71.40±0.69	65.84±0.81
PPR	75.80±0.35	82.87±1.01	74.35±1.51
Katz	75.7±0.0	83.4±0.0	76.8±0.0
GCN	92.72±0.64	89.14±1.20	87.89±1.48
GraphSAGE	81.60±1.22	85.96±2.04	84.05±1.72
GIN	82.49±2.89	68.74±2.74	69.63±2.77
GAT	91.79±0.02	90.27±0.23	90.84±0.35
KernelGCN	94.5±0.03	93.1±0.06	90.9±0.08
SEAL	97.57±0.05	90.82±1.97	88.49±1.22
PS2	97.60±0.10	92.31±0.31	90.29±0.36
VGAE	94.4±0.02	91.4±0.01	90.8±0.02
sGraphite	94.8±0.03	93.7±0.13	94.1±0.10
S-VGAE	96.0±0.1	**94.1±0.1**	**94.7±0.2**
ARGE	96.8±0.001	92.4±0.003	91.9±0.003
Linear VAE	95.91±0.13	92.55±0.97	91.60±0.90
SIEG w/ NF	98.50±0.09	93.04±0.54	90.85±0.74
SIEG w/o NF	**98.74±0.04**	93.92±0.05	92.10±0.44

Table 4: Test performance (AUC, %) on three Classic datasets.

Baselines. As the six datasets are widely popular, abundant baseline methods have been evaluated on them. On the OGB benchmarks, we classify the main baselines into three categories: heuristic methods including common neighbor (CN) (Liben-Nowell and Kleinberg 2007), Adamic-Adar (AA) (Lada et al. 2003), resource allocation (RA) (Zhou and Zhang 2009); non-GNN learning methods including Node2vec (Grover and Leskovec 2016), Matrix Factorization (MF) (Koren, Bell, and Volinsky 2009), MLP; GNN-based methods including GCN (Kipf and Welling 2016a), GraphSAGE (Hamilton, Ying, and Leskovec 2017), SEAL (Zhang et al. 2021), NGNN (Song et al. 2021), AGDN (Sun et al. 2022), S3GRL (Louis, Jacob, and Salehi-Abari 2023), Neural CN (Wang, Yang, and Zhang 2023), SUREL (Yin et al. 2023). On the Classic datasets, we classify the main baselines into three categories (the citations are the experimental results sources): heuristic methods including CN (Louis, Jacob, and Salehi-Abari 2023), AA (Louis, Jacob, and Salehi-Abari 2023), Personalized PageRank (PPR) (Louis, Jacob, and Salehi-Abari 2023), Katz (Zhu et al. 2021); GNN-based methods including GCN (Louis, Jacob, and Salehi-Abari 2023), GraphSAGE (Louis, Jacob, and Salehi-Abari 2023), GIN (Louis, Jacob, and Salehi-Abari 2023), GAT (Wang and Vinel 2021), KernelGCN (Tian et al. 2019), SEAL (Tan et al. 2023), PS2 (Tan et al. 2023); GAEs including VGAE (Kipf and Welling 2016b), sGraphite (Di et al. 2019), S-VGAE (Davidson et al. 2018), ARGE (Pan et al. 2018), linear VAE (Salha, Vazirgiannis, and Hennequin 2020).

Implementation Details. The training and test settings, hyperparameters about architecture and data processing, and other implementation details are presented in the Appendix. Code[1] is available.

[1]https://github.com/anonymous20221001/SIEG_OGB

Experimental Results

Comparison with Baselines. Table 3 reports the performance of the proposed method, SIEG with and without neighbor features (w/ and w/o NF), and baselines on three OGB benchmarks. All results of the baselines are taken from the OGB leaderboards[2]. Table 4 presents the performance of SIEG and baselines on the Classic datasets, with results of the baselines being taken from the corresponding papers. The proposed method achieves competitive results for all datasets, including ranking first on ogbl-ppa, ogbl-citation2, and Pubmed. (Note that the first place on ogbl-vessel, GAV (Wittmann et al. 2023), is specifically designed for flow-driven spatial networks rather than general graphs and is therefore not included as a baseline.) Compared to smaller or sparser graphs, the advantage of SIEG is more pronounced on larger or denser ones (statistics for each dataset are provided in the Appendix). One possible explanation for this is that larger graphs may contain more long-range target node pairs, while denser graphs could introduce more noise from neighbor features, which SIEG is designed to address. Notably, SIEG w/o NF performs better than SIEG w/ NF on all datasets, and we discuss this, along with other module comparisons, in the Ablation Study paragraph below.

Ablation Study. To investigate the effects of different modules and features in SIEG, we first created three variants based on SIEG w/o NF by adding neighbor features, removing BST, and removing GNN, respectively. Second, we replaced BST with FCT in SIEG w/o NF, resulting in the SIEG (FCT) w/o NF, and then created three variants based on SIEG (FCT) w/o NF by adding neighbor features, removing GNN, and removing FCT (resulting in the same variant as SIEG w/o BST), respectively. The test results of the mod-

[2]https://ogb.stanford.edu/docs/leader_linkprop/

Category	Model	ogbl-ppa	ogbl-citation2	ogbl-vessel	Pubmed	Cora	Citeseer
		Hits@100 (%)	MRR (%)	AUC (%)	AUC (%)	AUC (%)	AUC (%)
SIEG	w/o NF	**63.22±1.74**	**89.87±0.18**	**96.87±0.01**	**98.74±0.04**	**93.92±0.05**	**92.10±0.44**
	w/ NF	61.88±4.33	89.57±0.10	96.77±0.01	98.50±0.09	93.04±0.54	90.85±0.74
	w/o BST	60.05±4.85	89.18±0.11	84.70±0.70	97.94±0.08	93.24±0.43	88.22±1.18
	w/o GNN	33.46±2.58	85.86±0.04	94.88±0.01	96.32±0.23	91.26±0.47	85.57±1.29
SIEG (FCT)	w/o NF	-	-	-	97.35±0.31	93.22±0.33	86.94±1.98
	w/ NF	-	-	-	96.55±0.35	92.30±0.46	84.69±0.90
	w/o GNN	-	-	-	95.56±0.13	90.35±0.60	84.03±0.16
	w/o FCT	60.05±4.85	89.18±0.11	84.70±0.70	97.94±0.08	93.24±0.43	88.22±1.18

Table 5: Performance of SIEG and its variants.

els on six datasets are presented on Table 5. However, models with FCT can hardly run on OGB datasets due to the quadratic complexity.

Primarily, we analyze intra-category results based on SIEG w/o NF. Adding neighbor features (SIEG w/ NF) leads to worse performance, consistent with the results of GNN alone in Table 1. This confirms our suspicion that neighbor features are not effective features for link prediction or introduce noise. Besides, removing BST (SIEG w/o BST) leads to a drop in performance, indicating that BST learns important features missed by GNN, particularly pairwise structural features. In SIEG (FCT) category, adding neighbor features and removing GNN based on SIEG (FCT) w/o NF lead to drops, while removing FCT leads to a rise. This suggests that FCT negatively impacts link prediction on top of GNN, despite capturing pairwise structures, like BST. What causes this damage?

Here, we analyze the cross-category results by comparing each model of SIEG with the corresponding model of SIEG (FCT). We find that replacing BST with FCT results in a drop in all models, which supports the conclusion analyzed in the Methodology section. FCT introduces a large amount information away from the targets, which may lead to potential noise or overfitting risks.

Computational Efficiency. We present the Time consumption results of SIEG and SIEG (FCT) on a small dataset (Cora) and a medium-size dataset (ogbl-citation2) in Fig. 3. As theoretically analyzed in the Methodology section, the computational cost ratio between FCT and BST ranges from N to N^2. Experimentally, the results on Cora show that SIEG w/o NF runs 10 times faster than SIEG (FCT) w/o NF, and SIEG w/o GNN runs 16 times faster than SIEG (FCT) w/o GNN, demonstrating the significant efficiency advantage of BST over FCT. When dealing with medium-sized ogbl-citation2, models with FCT can hardly run.

Case Study. We use a simple example to illustrate how SIEG leverages pairwise structural features for link prediction. On Cora test set, SIEG assigns a high score of 8.22 to a positive publication pair with target node IDs (1111, 1273), and predictes a citation relationship between them. Examination of the pairwise structural features of this pair reveals a CN score of 4 and an AA score of 2.21, indicating the presence of 4 common citations who are not widely cited,

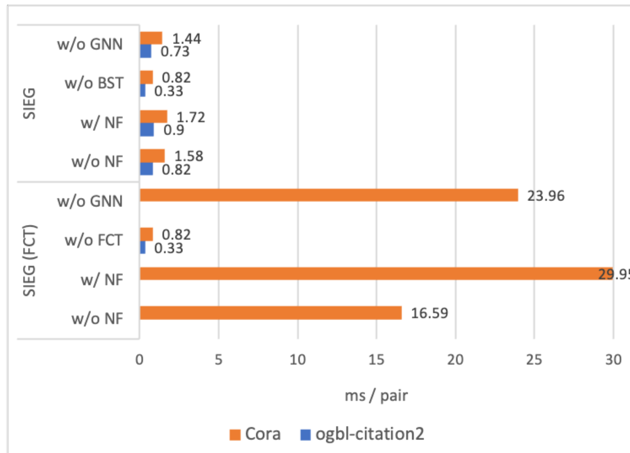

Figure 3: Time consumption of SIEG and SIEG (FCT) on Cora and ogbl-citation2.

which should be an important reason for SIEG to infer their citation relationship.

Conclusions

In this paper, we theoretically analyze GNNs' two inherent defects for link prediction in terms of information perception and transmission, and experimentally discover the harmful effects of neighbor node features. To address these issues, we propose a structural information enhanced link prediction framework, which involves removing the neighbor node features while fitting neighborhood structures more focused through GNN, and introducing BST to encode the structural relationships between target nodes, complementing the defects of GNN. The proposed method achieves remarkable results on six popular benchmarks, including ranking first on ogbl-ppa, ogbl-citation2 and Pubmed. Lastly, note that link prediction is a special case of multi-node representation learning, which also includes triplet (Liu, Ma, and Li 2021), motif (Besta et al. 2021), and subgraph (Alsentzer et al. 2020) tasks, among others. Theoretically, our method is also applicable to these tasks. In future work, we will practically extend SIEG to multiple tasks and develop it into a general multi-node representation learning framework.

References

Ahn, S. J.; and Kim, M. H. 2021. Variational Graph Normalized Auto-Encoders.

Alsentzer, E.; Finlayson, S. G.; Li, M. M.; and Zitnik, M. 2020. Subgraph Neural Networks.

Besta, M.; Grob, R.; Miglioli, C.; Bernold, N.; Kwasniewski, G.; Gjini, G.; Kanakagiri, R.; Ashkboos, S.; Gianinazzi, L.; and Dryden, N. 2021. Motif Prediction with Graph Neural Networks.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Neural Information Processing Systems*.

Chen, D.; O'Bray, L.; and Borgwardt, K. 2022. Structure-Aware Transformer for Graph Representation Learning.

Davidson, T. R.; Falorsi, L.; Cao, N. D.; Kipf, T.; and Tomczak, J. M. 2018. Hyperspherical Variational Auto-Encoders.

Di, X.; Yu, P.; Bu, R.; and Sun, M. 2019. Mutual Information Maximization in Graph Neural Networks.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; and Gelly, S. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs.

Feng, F.; Huang, W.; He, X.; Xin, X.; Wang, Q.; and Chua, T. S. 2021. Should Graph Convolution Trust Neighbors? A Simple Causal Inference Method. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Giles, C. L.; Bollacker, K. D.; and Lawrence, S. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, 89–98.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. *ACM*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs.

Hassani, A.; Walton, S.; Shah, N.; Abuduweili, A.; Li, J.; and Shi, H. 2021. Escaping the Big Data Paradigm with Compact Transformers.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Deep Residual Learning for Image Recognition. arXiv:1512.03385.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Katz, L. 1953. A new status index derived from sociometric analysis. *Psychometrika*, 18: 39–43.

Kipf, T.; and Welling, M. 2016a. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv: Learning*.

Kipf, T. N.; and Welling, M. 2016b. Variational Graph Auto-Encoders.

Kong, L.; Chen, Y.; and Zhang, M. 2023. Geodesic Graph Neural Network for Efficient Graph Representation Learning. arXiv:2210.02636.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8): 30–37.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *Neural Information Processing Systems*.

Lada; A; Adamic; ; ; Eytan; and Adar. 2003. Friends and neighbors on the Web. *Social Networks*.

Li, Q.; Han, Z.; and Wu, X. M. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning.

Liben-Nowell, D.; and Kleinberg, J. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7): 1019–1031.

Liu, Y.; Ma, J.; and Li, P. 2021. Neural Higher-order Pattern (Motif) Prediction in Temporal Networks.

Louis, P.; Jacob, S. A.; and Salehi-Abari, A. 2023. Simplifying Subgraph Representation Learning for Scalable Link Prediction. arXiv:2301.12562.

Mccallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2): 127–163.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. GraphiT: Encoding Graph Structure in Transformers.

Namata, G.; London, B.; Getoor, L.; and Namatag, B. H. and BLONDON and GETOOR and BERT@CS.UMD.EDU. 2012. Query-driven Active Surveying for Collective Classification. In *Mining and Learning with Graphs*.

Neelakantan, A.; Roth, B.; and Mccallum, A. 2015. Compositional Vector Space Models for Knowledge Base Completion. *Computer ence*, 1–16.

Pan, S.; Hu, R.; Long, G.; Jing, J.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding.

Park, J.; Song, J.; and Yang, E. 2022. GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. In *International Conference on Learning Representations*.

Park, W.; Chang, W.-G.; Lee, D.; Kim, J.; and seung-won hwang. 2022. GRPE: Relative Positional Encoding for Graph Transformer. In *ICLR2022 Machine Learning for Drug Discovery*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online Learning of Social Representations. *ACM*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *International Conference on Learning Representations*.

Sadeghian, A.; Armandpour, M.; Ding, P.; and Wang, D. Z. 2019. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs.

Salha, G.; Vazirgiannis, M.; and Hennequin, R. 2020. Simple and Effective Graph Autoencoders with One-Hop Linear Models.

Song, X.; Ma, R.; Li, J.; Zhang, M.; and Wipf, D. P. 2021. Network In Graph Neural Network. *arXiv preprint arXiv:2111.11638*.

Sun, C.; Hu, J.; Gu, H.; Chen, J.; and Yang, M. 2022. Adaptive Graph Diffusion Networks. arXiv:2012.15024.

Sun, Z.; Deng, Z. H.; Nie, J. Y.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space.

Tan, Q.; Zhang, X.; Liu, N.; Zha, D.; Li, L.; Chen, R.; Choi, S.-H.; and Hu, X. 2023. Bring your own view: Graph neural networks for link prediction with personalized subgraph selection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 625–633.

Teru, K.; Denis, E.; and Hamilton, W. 2020. Inductive Relation Prediction by Subgraph Reasoning. In *International Conference on Machine Learning*.

Tian, Y.; Zhao, L.; Peng, X.; and Metaxas, D. N. 2019. Rethinking Kernel Methods for Node Representation Learning on Graphs.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, R.; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. *JMLR.org*.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2019. Composition-based Multi-Relational Graph Convolutional Networks.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Velikovi, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2017. Graph Attention Networks.

Wang, H.; Ren, H.; and Leskovec, J. 2020. Relational Message Passing for Knowledge Graph Completion.

Wang, X.; and Vinel, A. 2021. Benchmarking Graph Neural Networks on Link Prediction.

Wang, X.; Yang, H.; and Zhang, M. 2023. Neural Common Neighbor with Completion for Link Prediction. *arXiv preprint arXiv:2302.00890*.

Weisfeiler, B. Y.; and Leman, A. A. 1968. A reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction (in Russian).

Wittmann, B.; Paetzold, J. C.; Prabhakar, C.; Rueckert, D.; and Menze, B. 2023. Link Prediction for Flow-Driven Spatial Networks. arXiv:2303.14501.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How Powerful are Graph Neural Networks. In *International Conference on Learning Representations*.

Yang, B.; Yih, W. T.; He, X.; Gao, J.; and Deng, L. 2014. Embedding Entities and Relations for Learning and Inference in Knowledge Bases.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning.

Yin, H.; Zhang, M.; Wang, J.; and Li, P. 2023. SUREL+: Moving from Walks to Sets for Scalable Subgraph-based Graph Representation Learning. arXiv:2303.03379.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do Transformers Really Perform Bad for Graph Representation? *arXiv: Learning*.

You, J.; Gomes-Selman, J.; Ying, R.; and Leskovec, J. 2021. Identity-aware Graph Neural Networks.

Zhang, J.; Zhang, H.; Xia, C.; and Sun, L. 2020a. Graph-Bert: Only Attention is Needed for Learning Graph Representations.

Zhang, M.; and Chen, Y. 2017. Weisfeiler-Lehman Neural Machine for Link Prediction.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhang, M.; and Chen, Y. 2020. Inductive Matrix Completion Based on Graph Neural Networks. In *International Conference on Learning Representations*.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Zhang, M.; Li, P.; Xia, Y.; Wang, K.; and Jin, L. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. In *Neural Information Processing Systems*.

Zhang, Z.; Cai, J.; Zhang, Y.; and Wang, J. 2020b. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(3): 3065–3072.

Zhou, T.; and Zhang, L. L.-C. 2009. Predicting missing links via local information. *The European Physical Journal B*.

Zhu, Z.; Zhang, Z.; Xhonneux, L.-P.; and Tang, J. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. *Advances in Neural Information Processing Systems*, 34: 29476–29490.