Weighted Sampling based Large-scale Enclosing Subgraphs Embedding for Link Prediction

Ganglin Hu

1Chongqing College of Electronic Engineering

November 22, 2023

Abstract

Link prediction is a fundamental problem for graphs, which can reveal the potential relationships between users. Graph embedding can easily encode graph structural relations, and heterogeneous attribute features in a continuous vector space, which is effective in link prediction. However, graph embedding methods for large-scale graphs suffer high computation and space costs, and sampling enclosing subgraphs is a practical yet efficient way to obtain the most features at the least cost. Nevertheless, the existing sampling techniques may lose essential features when the random sampling number of nodes is not large, as node features are assumed to follow a uniform distribution. In this paper, we propose a novel enclosing subgraph embedding model named Weighted Sampling Enclosing-subgraph Embedding (WSEE) to resolve this issue, which maximumly preserves the structural and attribute features of enclosing subgraphs with less sampling. More specifically, we first extract the feature importance of each node in an enclosing subgraph and then take the node importance as node weight. Then, random walks node sequences are obtained by multiple weighted random walks from a target pair of nodes, generating a weighted sampling enclosing subgraph. By leveraging the weighted sampling enclosing subgraph, WSEE can scale to larger graphs with much less overhead while maintaining some essential information of the original graph. Experiments on real-world datasets demonstrate that our model can scale to larger graphs with acceptable overhead while link prediction performance is unaffected.
Weighted Sampling based Large-scale Enclosing Subgraphs Embedding for Link Prediction

Ganglin Hu
Chongqing College of Electronic Engineering, 401331, University City, Chongqing, China
huganglin88@outlook.com

Abstract. Link prediction is a fundamental problem for graphs, which can reveal the potential relationships between users. Graph embedding can easily encode graph structural relations, and heterogeneous attribute features in a continuous vector space, which is effective in link prediction. However, graph embedding methods for large-scale graphs suffer high computation and space costs, and sampling enclosing subgraphs is a practical yet efficient way to obtain the most features at the least cost. Nevertheless, the existing sampling techniques may lose essential features when the random sampling number of nodes is not large, as node features are assumed to follow a uniform distribution. In this paper, we propose a novel enclosing subgraph embedding model named Weighted Sampling Enclosing-subgraph Embedding (WSEE) to resolve this issue, which maximumly preserves the structural and attribute features of enclosing subgraphs with less sampling. More specifically, we first extract the feature importance of each node in an enclosing subgraph and then take the node importance as node weight. Then, random walks node sequences are obtained by multiple weighted random walks from a target pair of nodes, generating a weighted sampling enclosing subgraph. By leveraging the weighted sampling enclosing subgraph, WSEE can scale to larger graphs with much less overhead while maintaining some essential information of the original graph. Experiments on real-world datasets demonstrate that our model can scale to larger graphs with acceptable overhead while link prediction performance is unaffected.

Keywords: Large-scale graphs · enclosing subgraphs · weighted sampling · enclosing subgraphs embedding · link prediction

1 Introduction

Graph embedding (GE) aims to map a graph into a low-dimensional vector space, which includes node embedding, edge embedding, subgraph embedding and whole graph embedding. Graph embedding has been proved extremely useful in link prediction [10, 12]. Link prediction quantifies the likelihood of an interaction between a pair of nodes (e.g., proteins interface prediction [4], social and item recommendations [25], etc.). A number of link prediction models [3, 18, 19, 23] have been proposed by leveraging their structure and attribute information and other heterogeneous information of nodes in graphs. In recent years, GNNs [27, 39] have been a widely accepted method for learning graph node embedding to tackle link prediction problems. The early GNNs focused on shallow encoders have been proposed using random walk-based methods, which are faster and more effective, including DeepWalk [24], Node2Vec [12], LINE [31], and SDNE [36]. More recently, deep learning and attention mechanisms have been used to generate graph embeddings [33, 34]. They extract structure, text, topic and other heterogeneous feature information more effectively and extend the expressive power of GNNs for all downstream tasks of link prediction, node classification, and graph classification. Subgraph-based embedding methods extract the enclosing subgraph features of around the two target nodes (shown in Fig. 1) and generate node embeddings of the enclosing subgraph, which improve GNNs and provide state-of-the-art solutions [10]. In a scale-free network (e.g., social networks and citation networks), it is intuitive that all nodes naturally prefer to interact with similar and powerful nodes. It will naturally form congregational communities of various topics.

Fig. 1. 1-hop and 2-hops enclosing subgraph instances. (a) the original graph. (b) the 1-hop enclosing subgraph instance. (c) the 2-hop enclosing subgraph instance. \( \nu, \mu \) two target nodes.
with a few highly connected central nodes (e.g., celebrities), which hold together numerous small nodes. For the central nodes, the enclosing subgraph size grows exponentially with the increase of hop $k$. Even the central nodes possess very large enclosing subgraphs, even for a small hop $k$. However, the lack of scalability of the enclosing subgraph embedding models prevents them from being applied to large-scale graphs. The main reason is the high computation and space overheads in extracting, preprocessing, and learning (large-scale) enclosing subgraphs for any pair of target nodes.

To tackle the above-identified scalability issue, we make the following contributions:

(1) We demonstrate the importance of sampling enclosing subgraphs in large-scale graphs.

(2) We propose an enclosing subgraph embedding model $WSEE$ to solve the computational and memory bottleneck of large-scale enclosing subgraphs by sparse enclosing subgraphs with minimum feature losses. Within the model, we also design an algorithm $WRWS$ to maximally extract the enclosing subgraph's topological features and attribute features and generate initial node embeddings.

(3) We conduct extensive experiments in nine real-world graphs. Experimental results prove the effectiveness and efficiency of the proposed $WSEE$ model and $WRWS$ algorithm.

The rest of the paper is organized as follows. Section 2 discusses several related works. We provide some definitions and problem formulation in Section 3. Section 4 presents in detail our proposed $WSEE$ model and $WRWS$ algorithm. We then show experimental results in Section 5 before concluding the paper in Section 6.

2 Related Work

In recent years, graph neural networks (GNN) [6, 18, 20, 25, 27, 33, 39] have been the popular and powerful [15] graph node embedding model. However, link prediction remains a challenging problem for these models. A common approach for link prediction is to obtain the embeddings of two nodes and then find a likelihood function to predict the existence probability of an edge between a pair of target nodes by learning the complex relationships between target nodes in a graph.

Recently, there have been two primary categories of link prediction techniques based on GNN as follows:

(1) Graph autoencoder-based (GAE) methods [29, 30, 32].

(2) Subgraph neural graph network methods [10, 26, 38].

GAE are powerful for predicting missing or future links in a graph. It can naturally incorporate node features to learn a latent representation of the graph and then reconstruct it. The link prediction task is performed on this reconstructed graph. Nevertheless, GAE learns node representations from a global graph without sufficiently considering the local relationships between nodes and their neighbouring nodes. Therefore, it overlooks the local structural features of the graph. Subgraph neural network methods focus on learning node representations of small neighbourhoods (subgraphs) in the graph, thereby better capturing local structural features. Subgraph neural networks combine GNN and sampling techniques [8, 11, 20, 26] to obtain subgraphs and compensate for the local structural information that GAE may neglect.

SEAL [10] is a typical model of subgraph neural network methods. As real-world graphs gradually scale up, subgraph neural networks have become a practical approach to dealing with large-scale graphs.

When the scale of the subgraph remains large, subgraph methods, as an effective way to reduce the graph size, can be used to further sparse the subgraph [26]. These subgraph methods mainly utilize random sampling techniques to reduce the dimension of subgraphs, which assume that all nodes are equal and independent. However, this assumption does not correspond to reality, especially in the scale-free graphs. Therefore, this may lose the vital information nodes or preserve some insignificant nodes, so the embedding effect is not ideal after dimensionality reduction. In this paper, we use a large-scale graph node scoring algorithm to preprocess graph nodes in advance and sample them purposefully, preserving the original graph’s features to the maximum extent. Then, the sampling graph is further processed by GNN.

3 Problem Definition

In this section, we present the necessary definitions and formulate the link prediction problem in large-scale enclosing subgraphs.

**Definition 1 (Graph).** A graph can be represented graphically: $G = (V, E, X)$, where $V = \{v_0, v_1, \ldots, v_{|V|-1}\}$ represents the set of nodes, and $|V|$ is the total number of nodes in $G$. $E \subseteq V \times V$ is the set of edges between the nodes. $X$ is a set of attributes of nodes in $G$.

**Definition 2 (Adjacent node and node degree).** An adjacent-node set of node $v \in V$ is defined as $N_v = \{v' | \forall v', (v, v') \in E\}$. $deg(v)$ is the number of nodes in the adjacent-node set of node $v$, called the degree of node $v$.

**Definition 3 (Enclosing subgraph).** Given a graph $G$ and two node sets $V^k = \{v' | \forall v', d(v', \mu) \leq k\}$, $V^k_v = \{v' | \forall v', d(v', v) \leq k\}$ where $d(\cdot, \cdot)$ is a hop distance function of a path between starting node and end node. The $k$-hops enclosing subgraph around target nodes $(\mu \vee \nu)$ is defined as $G_{\mu \vee \nu}^k = (V^{k}_{\mu \vee \nu}, E^{k}_{\mu \vee \nu}, \tilde{W}^{k}_{\mu \vee \nu})$, where $(\mu \vee \nu)$ or $\mu \nu$ means starting at $\mu$ or $\nu$; $V^{k}_{\mu \vee \nu} = \{\tilde{v} | \forall \tilde{v}, \tilde{v} \in (V^k_{\mu} \cup V^k_{\nu})\}$ is the node set of $G_{\mu \vee \nu}^k$ and $|V^{k}_{\mu \vee \nu}|$ is the total number of nodes in $G_{\mu \vee \nu}^k$. $E^{k}_{\mu \vee \nu}$ is the set
of edges linked by the node set \( V_{\mu} \) in \( G_{\mu}^k \). \( \tilde{W} = (w_1, w_2, \ldots, w_{|V_{\mu}|}) \) is a weighted vector consisting of the weights of all nodes in the graph \( G_{\mu}^k \).

**Definition 4 (Sampling-path).** Given a \( k \)-hops enclosing subgraph \( G_{\mu,\nu}^k \), a sampling-path \( p_{\mu,\nu}^k \) is a node and edge sequence with depth-first visiting \( k \)-steps at the target nodes \( (\mu \cup \nu) \), whose \( (\mu \cup \nu) \) means the path sampling starting at node \( \mu \) or \( \nu \). \( p_{\mu,\nu}^k \) is the form of \((\mu \cup \nu) \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_k} v_k\). All sampling-paths sampled from the \( G_{\mu,\nu}^k \) represented as \( P_{\mu,\nu}^k = \{p_{\mu,\nu}^{k,0}, p_{\mu,\nu}^{k,1}, \ldots, p_{\mu,\nu}^{k,\alpha}, p_{\mu,\nu}^{k,\beta}, \ldots, p_{\mu,\nu}^{k,\gamma}\} \), where \( \alpha, \beta, \gamma \) are the quantity of sampling-paths starting at target nodes \( \mu \) and \( \nu \) separately and \( \varpi = \alpha + \beta + \gamma \).

**Definition 5 (Sampling enclosing subgraph).** Given a \( k \)-hops enclosing subgraph \( G_{\mu,\nu}^k \), a sampling enclosing subgraph \( G_{\mu,\nu}^k = (V_{\mu,\nu}^k, E_{\mu,\nu}^k) \) is formed by all sampling-paths \( P_{\mu,\nu}^k \), where \( V_{\mu,\nu}^k \subseteq V_{\mu}^k, E_{\mu,\nu}^k \subseteq E_{\mu,\nu}^k \) and \( V_{\mu}^k \cup E_{\mu,\nu}^k = P_{\mu,\nu}^k \).

**Definition 6 (Node double radius).** The double radius of node \( v' \) refers to the distance pair between node \( v' \) and the two target nodes \( \mu \) and \( \nu \), which is formally expressed as: \( \text{doubleRadius}(v') = (d(v', \mu), d(v', \nu)) \).

**Link prediction problem.** Given an attributed graph \( G = (V, E, X) \), our goal is to find a likelihood function \( f \) to give the existence probability of an edge between a pair of target nodes \((\mu, \nu) \notin E \) by learning node features, such that structure, attribute and the double-radius label of nodes to two target nodes (refer to Def. 6, Eq. 5). Two target nodes, \( \mu \) and \( \nu \), with more similar graph-structure features, node-attribute features, and the node double-radius label features around the target nodes have a higher existence probability of an edge between \( \mu \) and \( \nu \). Two target nodes, \( \mu \) and \( \nu \), with more edges of \((u' \in V_{\mu}^k, v' \in V_{\nu}^k) \in E \) existence and more nodes of \( V_{\mu}^k \cap V_{\nu}^k \) existence, have a higher existence probability of an edge between \( \mu \) and \( \nu \).

### 4 Our Approach

In this section, we first introduce the framework of the proposed model and then introduces each component step by step.

#### 4.1 Framework of the Proposed Model

The framework of the proposed model \textit{WSEE} (Weighted Sampling Enclosing-subgraph Embedding for link prediction) is shown in Fig. 2. \textit{WSEE} consists of two components: Generation algorithm of Weighted Random Walks-based Sampling enclosing subgraph embeddings (WRWS) and enclosing subgraphs feature learning algorithm from Subgraphs labels, Embeddings and Attributes for Link prediction (SEAL). Initially, an algorithm of \textit{WeightRank} is proposed to capture the importance of each node in an enclosing subgraph \( G_{\mu,\nu}^k \), which is taken as each node weight (Section 4.2). Then, we calculate the sampling quantity and transition probability (Section 4.3) based on weights of nodes in the enclosing subgraph \( G_{\mu,\nu}^k \). The first component \textit{WRWS} extracts sampling enclosing subgraphs and generates the initial node embeddings (Section 4.4). The second component \textit{SEAL} catches features of sampling enclosing subgraphs based on enclosing subgraphs labels, initial node embeddings and attributes of nodes for link prediction (Section 4.5).

![Fig. 2. Overview of the framework of model WSEE.](image)

#### 4.2 WeightRank

In order to obtain the weighted vector \( W_{\mu,\nu}^k \) (refer to Def. 3), we update the \textit{PageRank} [14, 22] algorithm named \textit{WeightRank}. The algorithm of \textit{WeightRank} generates the importance of each node in \( G_{\mu,\nu}^k \). We take the importance of each node as node weights, and all the node importances of the \( G_{\mu,\nu}^k \) compose the weighted vector \( W_{\mu,\nu}^k = (w_1, w_2, \ldots, w_{|V_{\mu,\nu}^k|}) \), where \( w_i (0 \leq i < |V_{\mu,\nu}^k|) \) represents a node weight, and \( |V_{\mu,\nu}^k| \) is the number of nodes in \( G_{\mu,\nu}^k \). Before iterative calculation of \( W_{\mu,\nu}^k \), \textit{WeightRank} needs to prepare an adjacent matrix of the enclosing subgraph \( G_{\mu,\nu}^k \).

**Adjacent Matrix.** In order to obtain the adjacent matrix \( A \) of \( G_{\mu,\nu}^k \), each element \( A(v_j, v_i) \) of the \( A \) is defined as follows:

\[
A(v_j, v_i) = \begin{cases} 
1, & (v_j, v_i) \in E_{\mu,\nu}^k \\
0, & (v_j, v_i) \notin E_{\mu,\nu}^k 
\end{cases}
\]
where $v_i, v_j \in V^k$, are any two nodes in graph $G^k_{\mu \nu}$, $v_i = v_j$ means there is a self-connected edge. From Equation 1, $A(v_j, v_i) = 0$ denotes that nodes $v_i$ and $v_j$ don’t directly connected and $A(v_j, v_i) = 1$ means that nodes $v_i$ and $v_j$ are neighbours. So the adjacent matrix $A$ reflects the neighbour’s relationship among nodes. The matrix $A$ is symmetric because the edges $E^k_{\mu \nu}$ in the graph $G^k_{\mu \nu}$ are all undirected.

**Node Weight.** The vector $\bar{\epsilon}$ where $w = (w_1, \ldots, w_n)$ in this way, the node weight $w$ because the edges $E$ are neighbours. So the adjacent matrix $A$ reflects the neighbour’s relationship among nodes.

As shown in Equation 2, each iteration has two computation processes. The first computation process is to calculate the weight $w_i'$ of node $v_i$ for the current iteration, shown as follows:

$$w_i' = \sum_{v_k \in N_i} \frac{w_k}{\deg(v_k)}, (v_i \in V)$$

where $N_i$ is the adjacent-node set of node $v_i$, $v_k \in N_i$, $\deg(v_k)$ is the degree of node $v_k$, and $w_k$ is the weight of node $v_k$ at previous iteration.

After finishing the first iteration computation process, the next computation process is to normalize the node weight for the current iteration. Each iteration is a 1-order aggregate operation from the weights of adjacent nodes. That is to say, the $w_i$ of $v_i$ is replaced by aggregating $\sum_{v_j} deg(v_j)$.

As shown in Equation 2, each iteration has two computation processes. The first computation process is to calculate the weight $w_i'$ of node $v_i$ for the current iteration, shown as follows:

$$w_i = \frac{w_i'}{|V|-1} \sum_{j=0}^{i} (w_j)^2, \quad (0 \leq i < |V|).$$

In this way, the node weight $w_i$ obtained from Equation 2 will eventually converge.

### 4.3 Sampling Trees, Sampling Quantity and Transition Probability

**Sampling Trees.** Before discussing the sampling quantity, we first define the concepts of sampling trees. A sampling tree can be constructed as follows, an example shown in Fig. 3.

1. First, we define a sampling tree as $Tree(\mu \lor \nu)$ whose root node ($\mu$ or $\nu$) is denoted as $Root(\mu \lor \nu)$, $h$ is the height of $Tree(\mu \lor \nu)$.
2. Given a target nodes ($\mu$ or $\nu$), one of the target nodes ($\mu$ or $\nu$) is chosen as a parent node $v_{par}$ and set a grandpa node $v_{gr} = null$.
3. We take the neighbours $N_{v_{par}}$ of the node $v_{par}$ as the child nodes of $v_{par}$ except for the grandpa node $v_{gr}$, where the grandpa node is the parent of the node $v_{par}$.
4. $h$ is the current height of the sampling tree $Tree(\mu \lor \nu)$.

If $h < k$ (refer to Def. 3),

then $v_{gr} = v_{par}$ (save $v_{par}$ as grandpa node $v_{gr}$);
$v_{par} = N_{v_{par}}$ (child nodes $N_{v_{par}}$, the old parent node $v_{par}$ are taken as the new parent nodes);
Go back to Step 3 to process for another iteration.

Otherwise, the iteration terminates.

**Sampling Quantities.** The quantity of sampling paths (abbr. sampling quantity) denotes the number of paths from the root node ($\mu \lor \nu$) to the leaf node in a sampling tree $Tree(\mu \lor \nu)$ (refer to Def. 4 and Section 4.3-Sampling Trees). The sampling quantity $\varpi_{\mu \lor \nu}$ starting at target nodes $\mu$ or $\nu$ can be expressed as follows:

$$\varpi(\mu \lor \nu) = \left[ k \cdot w_{\mu \nu} \right], \quad (3)$$
where $\mu \lor \nu$ and $\mu\nu$ represent the starting node of the sampling path is $\mu$ or $\nu$, $w_{\mu\nu}$ indicates the root-node weights of a $Tree(\mu \lor \nu)$, $\lambda$ is an amplification coefficient, $k$ is a hop distance for sampling a $k$-hops enclosing subgraph and $\lceil \cdot \rceil$ means upwards to the nearest integer.

The reason why we use the multiple of $w_{\mu\nu}$ as the sampling quantity is that the weight of a sampling path starting node $(\mu \lor \nu)$ is obtained by multiple-iteration aggregating neighbour-node weights in a sampling tree $Tree(\mu \lor \nu)$ according to Eq. 2, so the weight of $(\mu \lor \nu)$ can reflect the importance of $Tree(\mu \lor \nu)$. The sampling quantity is also positively correlated with the size of the $k$-hops sampling enclosing subgraph, so we acquire the sampling quantity by the weight $\lambda \cdot k$ multiple of node $(\mu \lor \nu)$. Taking Fig. 4 as an example, we compute the sampling quantity of each sampling enclosing subgraph as follows:

$$
\{w_{\mu}, w_{\nu}\} = \{\lceil \lambda \cdot k \cdot w_{\mu} \rceil, \lceil \lambda \cdot k \cdot w_{\nu} \rceil\} = \{\lceil \lambda \cdot 0.490 \rceil, \lceil \lambda \cdot 0.328 \rceil\}.
$$

If $\lambda = 9$, the sampling quantity of each sampling tree is calculated as:

- when $k=1$ then 
  $$
  \{w_{\mu}, w_{\nu}\} = \{9 \cdot 1 \cdot 0.490, 9 \cdot 1 \cdot 0.328\} = \{5, 3\}. \quad w = w_{\mu} + w_{\nu} = 8.
  $$
- when $k=2$ then 
  $$
  \{w_{\mu}, w_{\nu}\} = \{9 \cdot 2 \cdot 0.490, 9 \cdot 2 \cdot 0.328\} = \{9, 6\}. \quad w = w_{\mu} + w_{\nu} = 15.
  $$

**Transition Probability of Random Walks.** A transition probability matrix $\mathcal{A}$ consists of the transition probabilities between every two nodes of an enclosing subgraph $G^k_{\mu\nu}$. When random walks travel to a node $v_i$, the transition probability to $v_j$ is defined as follows:

$$
p(v_j|v_i) = \begin{cases} 
\frac{w_{ij}}{\sum_{v_k \in N_{v_i}} w_{ik}}, & (v_j, v_i) \in E^k_{\mu\nu} \\
0, & (v_j, v_i) \notin E^k_{\mu\nu}
\end{cases}
$$

where $N_{v_i}$ (refer to Def. 2) denotes the neighbours of node $v_i$, and $w_{vj}$ (refer to Eq. 2) is the weight of node $v_j \in N_{v_i}$.

Starting at the target node $\mu$, for example, when random walks travel to a node $v_1$, as shown in Fig. 3(a) $Tree(\mu)$ and Fig. 4(a), the neighbours of node $v_1$ are $N_{v_1} = \{\nu, v_7, v_8\}$ except for the grandpa node $\mu$ (refer to Section 4.3-Sampling Trees for the definition of grandpa node), and the probabilities of transition to the subsequent nodes are computed as follows:

- $p(v_2|v_1) = 0.183$, $p(v_3|v_1) = 0.183$,
- $p(v_7|v_1) = 0.064$, $p(v_8|v_1) = 0.064$,
- $p(v_9|v_1) = 0.183$, $p(v_1|v_1) = 0.277$.

**4.4 WRWS**

Given the sampling quantity and the transition probability, the algorithm of the Weighted Random Walks-based Sampling enclosing subgraphs embedding generation (WRWS) is presented in Algorithm 1.

Algorithm 1 has four parameters $G = (V, E, X)$, $\omega = w_{\mu} + w_{\nu}$, $\mathcal{A}$ and $k$. $G = (V, E, X)$ is a large-scale graph. $\omega = w_{\mu} + w_{\nu}$ is the sum of sampling quantities starting at target node $\mu$ and $\nu$. $\mathcal{A}$ is a transition probability matrix among nodes of enclosing subgraphs. $k$ is a hyperparameter of a $k$-hops enclosing subgraph, which indicates the order of neighbour of a node to target node $\mu$ or $\nu$. Shown in Algorithm 1, the algorithm of WRWS outputs embedding_dict (line 19 of Algorithm 1), which is an initial weighted embedding-set of nodes and is input to SEAL (shown in Section 4.5). The embedding_dict is generated by the model Node2Vec [12] (line 19), which is a graph embedding model based on random walks for a large-scale graph. Line 3 calculates the sampling quantity according to Eq. 3, i.e., the number of sampled paths. Line 4 is used to compute a transition probability matrix $\mathcal{A}$ according to Eq. 4. The first loop (line 5) selects one of the target nodes $\mu$ or $\nu$, which we sample paths starting at. We save the selected target node in $v'$, and its
our experiments because we only sample a set of positive links. The sampling quantity is \( \varpi \).

where we assign \( \nu \) as the right node of each sampling edge according to \( \mathcal{A} \). In line 11, we select a neighbour node of node \( v_i \) as the right node of each sampling edge \( \{v_i, v_j\} \) according to \( \mathcal{A} \). In lines 12 to 15, we append the sampling edge \( \{v_i, v_j\} \) to \( \mathcal{P} \), set the old left node as grandpa node (line 14) and set the right node \( v_j \) of the current sampling edge as the new left node of next sampling edge. In line 19, the sampling path set \( \mathcal{P} \) is fed to Node2Vec to get the initial node embedding \( \text{embedding} \_\text{dict} \), which will be given to SEAL.

Taking \( G \) (shown in Fig. 4) as an example, the path set \( \mathcal{P} \) obtained by Algorithm 1 can form the sampling enclosing subgraph \( G^{1,8} \) (\( k=1 \)) or \( G^{1,15} \) (\( k=2 \)). The node \( v_9 \) might not be sampled because of the low transition probability to \( v_9 \) when \( k=1 \). For the same reason, the nodes \( v_8 \) and \( v_{11} \) might also not be sampled when \( k=2 \).

**4.5 SEAL**

SEAL [10] is a variant of GNNs [27], which is used to jointly learn from three types of features, including the enclosing subgraph, node embeddings and node attributes. The initial node embedding of the sampling enclosing subgraph generated by the WRWS (shown in Algorithm 1) is input into SEAL. Then SEAL learns node embeddings of an enclosing subgraph for link prediction. SEAL takes \( (A, X) \) as input, where \( A \) is the adjacency matrix of the sampling enclosing subgraph and \( X \) is a node information matrix. SEAL learns node embeddings for link prediction in three steps: (1) node information matrix generating, (2) node embedding learning by GNNs, (3) link prediction of the enclosing subgraph.

**Node Information Matrix Generating.** A node information matrix \( X \) is a node’s feature vector, which is composed of node labels, initial node embeddings and node attributes.

Node labels are calculated by the algorithm of Double-Radius Node Labeling (DRNL), which labels every node of \( G^{k,\infty}_{\mu\nu} \) by the function \( f_1 : V \mapsto \mathbb{N} \) as follows.

\[
f_1(v') = 1 + \min \{d(v', \mu), d(v', \nu)\} + (d/2)[(d/2) + (d\%2) - 1],
\]

where we assign \( f_1(\mu)=1 \) and \( f_1(\nu)=1 \). \( d(v', \mu) \) and \( d(v', \nu) \) (refer to Def. 3) are the hop distances of \( v' \) to \( \mu \) and \( \nu \), respectively. \( \min[\cdot, \cdot] \) means to return the smallest of the two values. \( d = d(v', \mu) + d(v', \nu) \) and \( \lfloor \cdot \rfloor \) denotes upwards to the nearest integer. \( d/2 \) and \( d\%2 \) are the integer and decimal parts of \( d \) divided by \( 2 \), respectively. After acquiring the node labels of \( G^{k,\infty}_{\mu\nu} \), we acquire label vectors of nodes by using their one-hot encoding vectors.

The initial node embeddings, generated by WRWS (refer to Algorithm 1), have bad generalization performance in our experiments because we only sample a set of positive links \( E^P_{\mu\nu} \subseteq E^k_{\mu\nu} \) in the enclosing subgraph \( G^{k}_{\mu\nu} = (V^k_{\mu\nu}, E^k_{\mu\nu}) \).
In order to improve the performance of link prediction, we further sample a set of negative links \( (E^N_{\mu\nu} \not\subseteq E^K_{\mu\nu} \text{ and } V^N_{\mu\nu} \subseteq V^K_{\mu\nu}) \) (refer to Def. 5). We input the \( E^P_{\mu\nu} \cup E^N_{\mu\nu} \) to the Node2Vec [12] and generate the initial node embeddings.

**Link Prediction of Enclosing Subgraph.** We concatenate the node label vector, initial node embedding and attribute vector of each node in the node information matrix \( X \). Taking \( (A, X) \) as input and the GNN can simultaneously learn from all three types of features, where \( A \) is the adjacency matrix of \( G^K_{\mu\nu} \). This model learning from three types of features of labels, embeddings and attributes of nodes is SEAL, which is used for link prediction of target nodes \( (\mu, \nu) \).

## 5 Experiments

In this section, we describe our datasets, and baseline models and present the experimental results to demonstrate the performance of the WSEE model in link prediction tasks.

### 5.1 Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Name</th>
<th>Nodes</th>
<th>Edges</th>
<th>Avg.Deg.</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>USAir</td>
<td></td>
<td>332</td>
<td>2126</td>
<td>12.81</td>
<td>-</td>
</tr>
<tr>
<td>Celegans</td>
<td></td>
<td>297</td>
<td>2148</td>
<td>14.46</td>
<td>-</td>
</tr>
<tr>
<td>Router</td>
<td></td>
<td>5022</td>
<td>6285</td>
<td>2.49</td>
<td>-</td>
</tr>
<tr>
<td>Power</td>
<td></td>
<td>4941</td>
<td>6594</td>
<td>2.67</td>
<td>-</td>
</tr>
<tr>
<td>Yeast</td>
<td></td>
<td>2375</td>
<td>11693</td>
<td>9.85</td>
<td>-</td>
</tr>
<tr>
<td>Ecoli</td>
<td></td>
<td>1805</td>
<td>14660</td>
<td>16.24</td>
<td>-</td>
</tr>
<tr>
<td>PB</td>
<td></td>
<td>1222</td>
<td>16714</td>
<td>27.36</td>
<td>-</td>
</tr>
<tr>
<td>Cora</td>
<td></td>
<td>2708</td>
<td>5429</td>
<td>4</td>
<td>1433</td>
</tr>
<tr>
<td>CiteSeer</td>
<td></td>
<td>3327</td>
<td>4732</td>
<td>2.84</td>
<td>3703</td>
</tr>
</tbody>
</table>

In Table 1, we consider the following real-world graph datasets. USAir [35] is a network of American airlines. Celegans [28] is the neural network of Caenorhabditis elegans. Router [5] is an Internet at the router level. Power [28] is an electrical grid of the western United States. Yeast [7] is a network of protein interactions. Ecoli [9] is a pairwise reaction network of metabolites in Escherichia Coli. PB [1] is a network of American political blogs. Cora [21] is a classification citation network of research papers constructed by McCallum et al. CiteSeer [17], similar to Cora, is a network of citing and cited information between papers.

### 5.2 Baselines

To validate the performance of our approach, we employ eight state-of-the-art network embedding methods as baselines to compare with our WSEE model: Common Neighbors (CN), Adamic-Adar (A-A), GCN, GraphSAGE, Graph Isomorphism Network (GIN), Matrix Factorization (MF), Node2vec and SEAL. CN calculates and gets the common one-hop neighbours of both target nodes and further speculates the potential relationship between the two nodes. A-A [2] is devised by Lada A. Adamic and Eytan Adar to characterize individual information on Internet in order to analyze social network structure. The representative application of A-A is the mining of correlations between groups of people. GCN [16] is a scalable method for semi-supervised learning on graph-structured data which learns hidden layer representations by encoding both local graph structure and features of nodes. GraphSAGE [20] fetches structural and textual features of large graphs to generate low-dimensional node embeddings efficiently. GIN [15] is a theoretical framework for analyzing the expressive power of GNNs to capture different graph structures. MF [37] characterizes both items and users by factor vectors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation, which analyzes patterns of user interest in products to provide personalized recommendations. Node2vec [12] maps nodes to a low-dimensional space by learning continuous node feature representations in networks, which maximizes the likelihood of preserving network neighbourhoods of nodes. SEAL [10] learns a heuristic function mapping the subgraph patterns to link existence by extracting a local subgraph around each target link from a given network.
Table 2. Comparison with the AUC values.

<table>
<thead>
<tr>
<th>Model</th>
<th>Dataset</th>
<th>USAir</th>
<th>Celegans</th>
<th>Router</th>
<th>Power</th>
<th>Yeast</th>
<th>Ecoli</th>
<th>PB</th>
<th>Cora</th>
<th>CiteSeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN</td>
<td></td>
<td>93.1</td>
<td>83.5</td>
<td>55.6</td>
<td>58.1</td>
<td>88.7</td>
<td>92.8</td>
<td>91.4</td>
<td>71.4</td>
<td>65.8</td>
</tr>
<tr>
<td>A-A</td>
<td></td>
<td>94.4</td>
<td>85.3</td>
<td>55.5</td>
<td>58.2</td>
<td>88.8</td>
<td>94.6</td>
<td>91.7</td>
<td>71.5</td>
<td>66.1</td>
</tr>
<tr>
<td>GCN</td>
<td></td>
<td>88.5</td>
<td>81.6</td>
<td>84.1</td>
<td>68.1</td>
<td>90.8</td>
<td>90.9</td>
<td>91.1</td>
<td>89.4</td>
<td>86.7</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td></td>
<td>85.6</td>
<td>74.8</td>
<td>68.9</td>
<td>65.7</td>
<td>88.1</td>
<td>87.9</td>
<td>86.8</td>
<td>85.9</td>
<td>84.5</td>
</tr>
<tr>
<td>GIN</td>
<td></td>
<td>89.1</td>
<td>74.2</td>
<td>75.8</td>
<td>58.3</td>
<td>83.4</td>
<td>89.4</td>
<td>90.3</td>
<td>71.9</td>
<td>71.8</td>
</tr>
<tr>
<td>MF</td>
<td></td>
<td>90.1</td>
<td>75.8</td>
<td>70.6</td>
<td>52.2</td>
<td>86.8</td>
<td>91.1</td>
<td>91.8</td>
<td>60.9</td>
<td>61.9</td>
</tr>
<tr>
<td>Node2vec</td>
<td></td>
<td>86.3</td>
<td>75.0</td>
<td>63.5</td>
<td>72.2</td>
<td>90.9</td>
<td>91.0</td>
<td>84.8</td>
<td>78.8</td>
<td>75.3</td>
</tr>
<tr>
<td>SEAL</td>
<td></td>
<td>97.1</td>
<td>90.6</td>
<td>95.9</td>
<td>84.3</td>
<td>97.4</td>
<td>97.8</td>
<td>95.0</td>
<td>90.6</td>
<td>88.4</td>
</tr>
<tr>
<td>WSEE</td>
<td></td>
<td>96.6</td>
<td>88.5</td>
<td>94.5</td>
<td>83.8</td>
<td>97.5</td>
<td>97.4</td>
<td>94.7</td>
<td>90.9</td>
<td>87.7</td>
</tr>
</tbody>
</table>

5.3 Evaluation Metrics and Parameter Settings

In Table 1, we consider nine real-world graph datasets: USAir, Celegans, NS, Router, Power, Yeast, Ecoli, PB, Cora and CiteSeer. We randomly split the existing edges from each dataset into an 85% training set, a 10% testing set, and a 5% validation set. We randomly generate the same number of negative samples (nonexistent edges) of positive samples (i.e., 1:1 ratio for positive and negative samples). We take a standard evaluation metric, AUC scores (area under the ROC curve) [13], as evaluation metrics to measure the link prediction performance. AUC represents the probability that nodes in a random unobserved link are more similar than those in a random nonexistent link. In the iterative calculation of node weight, we set the exit condition $\epsilon = \frac{1}{10^k}$. When calculating the sampling quantity, we set the hop distance as $k = 3$ and the amplification coefficient as $\lambda = 9$. We run all the models six times and report the averaged performance in Table 2.

5.4 Experimental Results

For experiments, we evaluate the effectiveness and efficiency of our WSEE on nine real-world graph datasets for the link prediction task. Table 2 reports the average AUC for nine datasets and nine link prediction models, in which the larger the value of AUC, the better the link prediction performance. In all datasets, enclosing-subgraph-based models SEAL and WSEE are superior to other baseline models. In addition, WSEE gives very comparable results to SEAL or even better than SEAL in datasets of Yeast and Cora. In order to meet the needs of large-scale graph applications, we achieve this performance on the basis of using the algorithm WRWS by order of magnitudes less resource consumption. In general, in Table 2, it can be seen that the AUC values of WSEE are equivalent to SEAL and higher than other baseline models, which indicates that the method of weighted sampling enclosing subgraphs embedding can properly reserve the features of an enclosing-subgraph.

5.5 Parameter sensitivity analysis

We further perform parameter sensitivity analysis in this section, and the results are summarized in Fig. 5 and Fig. 6. They illustrate how the hop distance $k$ and the amplification coefficient $\lambda$ affect the performance of link prediction and the consumption of computation time and CUDA's memory. The change of AUC, computation time and CUDA's memory of all datasets are qualitatively similar, where the increase of the AUC, computation time and CUDA's memory along with the values of $k$ and $\lambda$. When the hop distance $k$ is less than 3, the 2-hops or 1-hops, the link prediction effectiveness of target node pairs is not ideal enough and the consumption of computation time and CUDA's memory are relatively small. The reason is that the enclosing subgraph around target nodes is not large enough. When $k$ is higher than 5, the enclosing subgraph will also be large-scale with higher computation and memory costs during training, but the performance improvement is not apparent, shown in Fig. 5 and Fig. 6. When the amplification coefficient $\lambda$ is less than 7, the performance is not good enough because of the poor sampling quantities. For the same reason, if the amplification coefficient $\lambda$ is higher than 9, the scale of sampling enclosing subgraphs will gradually increase, which will cost large computational and memory resources with the small increments of AUC values. So, we need to get a balance between the sampling quantity and AUC, which helps to improve AUC values using appropriate sampling, computation and memory costs.

6 Conclusion and Future Work

This paper has presented a practical enclosing subgraph embedding model WSEE, which can solve large-scale graphs’ computing and memory bottlenecks. WSEE utilizes weighted sampling to get a sparse enclosing subgraph without losing the important topological and attribute features of the enclosing subgraph. In WSEE, we obtain sampling enclosing
subgraphs by weighted random walks and generate the initial node embeddings for the sampling enclosing subgraphs. To get the initial node embeddings, we have designed a WRWS algorithm to generate sampling enclosing subgraphs.

Fig. 5. Results of parameter sensitivity analysis (Along with the increase of amplification coefficient $\lambda$, the AUC is analyzed for each hop $k$).
Fig. 6. Results of parameter sensitivity analysis (Along with the increase of amplification coefficient $\lambda$, the time and cuda is analyzed for each hop $k$).

The WRWS can ensure that the importance structural features of nodes are effectively extracted when the sample size is not saturated. Eventually, the SEAL model conducts link prediction for target node pairs by integrating the features of subgraphs labels, initial node embeddings and attributes. Experimental results in nine real-world graphs have shown the efficiency and effectiveness of weighted sampling enclosing subgraphs embedding in link prediction. In the future, we plan to investigate how the WSEE model works in heterogeneous graphs.

Acknowledgements. This paper is funded by the National Natural Science Foundation of China (62032019, 61732019, 61672435), and the Capacity Development Grant of Southwest University (SWU116007).

References
